# Guide to the Systems Engineering Body of Knowledge (SEBoK), version 2.2

**Part 3**

# Guide to the Systems Engineering Body of Knowledge, Part 3

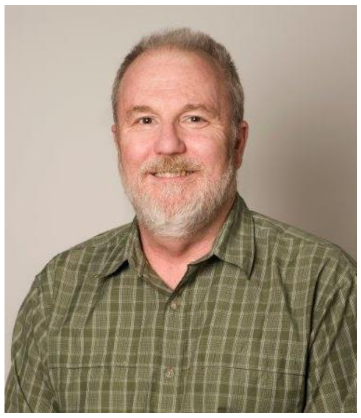
version 2.1

# Contents

## Articles

# Front Matter

## Letter from the Editor



Hi there! Welcome to the October 2019 instantiation of the Systems Engineering Body of Knowledge. Since version 1.0 appeared in September 2012, that means the SEBoK just celebrated its 7th birthday! This release, version 2.1 is also my third release as Editor in chief. This release brings what I hope are some exciting changes for the readers and authors.

The **first change** I hope you notice is that we have added bylines to those articles for which we can track their origins. As of this release, we are recognizing the contribution of lead authors and the additional contributing authors. It is our hope that these contributions will be beneficial to the authors in their professional lives - being able to prove their contributions to this important knowledge base.

The **next obvious change** should be the way glossary bubbles have been updated. They are more readable now, with a grey background and black text.

Other changes include **new articles** on:

- Digital Engineering
- Mission Engineering
- Set Based Design
- MBSE Adoption Trends 2009-2018

Additionally, we have **updated content** on Resilience, Human Systems Integration, and Capability Engineering. Part 1 also received a wire brushing. We have also begun incorporating video. You will find a short video on the Main page. We are also going to begin to look at existing INCOSE YouTube channel content to look for 1-3 minute clips we can strategically place throughout the SEBoK to add value.

There is a big announcement to be made relative to the SEBoK. The IEEE Computer Society has been one of the three stewards of the SEBoK from the beginning. They have had a seat on the Board of Governors, and have provided invaluable counsel. In January 2020, that stewardship will be transferred from the IEEE Computer Society to the IEEE Systems Council. The Editorial Board looks forward to the continued support and participation of IEEE. Thank you IEEE Computer Society, and in particular to Rich Hilliard and Andy Chen.

Regarding the reach of the SEBoK, there were over 29,000 visitors and 68,781 page views during the the month of July 2019. That brings our total page views to over 3.45M since 2012. Top content pages in July: 1) Stakeholder Needs and Requirements, 2) Types of Models, 3) Types of Systems, 4) Systems Requirements, and 5) Reliability, Availability, and Maintainability. Top countries accessing the SEBoK in July:

1. US
2. India
3. Australia
4. United Kingdom
5. Philippines

Looking forward to the next release, it is my hope that those of you that enjoy working with video will think about creating video content now that we have that capability. Please limit your submissions to no more than 3 minute clips, and the preferred format is mp4.

I am still looking for additional authors and folks interested in taking a leadership role as editors to help manage and grow our content for specific areas. It would be nice to add some more content in Part 6: Related Disciplines, and Part 7: SE Implementation Examples. If you would like to author an article for those sections, please reach our to Nicole Hutchinson (emtnicole@gmail.com) or myself (rcloutier@southalabama.edu).

That is it for now ... I hope to see you at the upcoming International Workshop being held in Torrence, CA in January 2020. If you have ideas for the SEBoK, or would like to get involved, be sure to find me there and we can have some coffee and chat. But, do not feel you have to wait until then to get involved - reach out now! Thanks for your ongoing support.

[illegible]

# BKCASE Governance and Editorial Board

## BKCASE Governing Board

The three SEBoK steward organizations – the International Council on Systems Engineering (INCOSE), the Institute of Electrical and Electronics Engineers Computer Society (IEEE-CS), and the Systems Engineering Research Center (SERC) provide the funding and resources needed to sustain and evolve the SEBoK and make it available as a free and open resource to all. The stewards appoint the BKCASE Governing Board to be their primary agents to oversee and guide the SEBoK and its companion BKCASE product, GRCSE.

The BKCASE Governing Board includes:

- **The International Council on Systems Engineering (INCOSE)**
  - Art Pyster (Governing Board Chair), Paul Frenz
- **Systems Engineering Research Center (SERC)**
  - Jon Wade, Cihan Dagli
- **IEEE Computer Society (IEEE CS)**
  - Andy Chen, Rich Hilliard

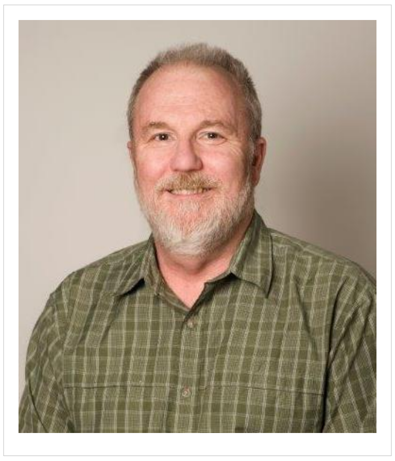
Past INCOSE governors Bill Miller, Kevin Forsberg, David Newbern, David Walden, Courtney Wright, Dave Olwell, Ken Nidiffer, Richard Fairley, Massood Towhidnejad, and John Keppler. The governors would also like to acknowledge John Keppler, IEEE Computer Society, who has been instrumental in helping the Governors to work within the IEEE CS structure.

The stewards appoint the BKCASE Editor in Chief to manage the SEBoK and GRCSE and oversee the Editorial Board.

## Editorial Board

The SEBoK Editorial Board is chaired by an Editor in Chief, supported by a group of Associate Editors.

**SEBoK Editor in Chief**



**Robert J. Cloutier**

*University of South Alabama*

rcloutier@southalabama.edu [1]

Responsible for the appointment of SEBoK Editors and for the strategic direction and overall quality and coherence of the SEBoK.

**SEBoK Managing Editor**

**Nicole Hutchison**

*Stevens Institute of Technology*

nicole.hutchison@stevens.edu [2] or emtnicole@gmail.com [3]

Responsible for the the day-to-day operations of the SEBoK and supports the Editor in Chief.

Each Editor has his/her area(s) of responsibility, or shared responsibility, highlighted in the table below.

**SEBoK Part 1: SEBoK Introduction**

**Lead Editor: Robert J. Cloutier**

*University of South Alabama*

rcloutier@southalabama.edu [1]

Responsible for Part 1

## Graduate Student Support

With SEBoK v. 2.1, the Governing Board has hired a graduate student to support the Editor in Chief and Managing Editor. Madeline Haas, a master's student at George Mason University, is the current graduate student supporting the SEBoK and we gratefully acknowledge her exemplary efforts.

## Interested in Editing?

The Editor in Chief is looking for additional editors to support the evolution of the SEBoK. Editors are responsible for maintaining and updating one to two knowledge areas, including recruiting and working with authors, ensuring the incorporation of community feedback, and maintaining the quality of SEBoK content. We are specifically interested in support for the following knowledge areas:

- System Deployment and Use
- Product and Service Life Management
- Enabling Businesses and Enterprises
- Systems Engineering and Software Engineering
- Procurement and Acquisition
- Systems Engineering and Specialty Engineering

If you are interested in being considered for participation on the Editorial Board, please visit the BKCASE website http://www.bkcase.org/join-us/or contact the BKCASE Staff directly at bkcase.incose.ieeecs@gmail.com [24].

**SEBoK v. 2.1, released 31 October 2019**

## References

[1] mailto:rcloutier@southalabama.edu
[2] mailto:nicole.hutchison@stevens.edu
[3] mailto:emtnicole@gmail.com
[4] mailto:gary.r.smith@airbus.com
[5] mailto:dori@mit.edu
[6] mailto:dhyberts@mitre.org
[7] mailto:Peter@coexploration.net
[8] mailto:dagli@mst.edu
[9] mailto:boehm@usc.edu
[10] mailto:kforsberg@ogrsystems.com
[11] mailto:gparnell@uark.edu
[12] mailto:garry.j.roedler@lmco.com
[13] mailto:prmarbach@gmail.com
[14] mailto:kenneth.zemrowski@incose.org
[15] mailto:jdahmann@mitre.org
[16] mailto:M.J.d.Henshaw@lboro.ac.uk
[17] mailto:james.martin@incose.org
[18] mailto:Rick.Hefner@ngc.com
[19] mailto:Timothy.Ferris@cranfield.ac.uk
[20] mailto:bernardo.delicado@mbda-systems.com
[21] mailto:alice.squires@wsu.edu
[22] mailto:cliftonbaldwin@gmail.com
[23] mailto:dolwell@stmartin.edu
[24] mailto:bkcase.incose.ieeecs@gmail.com

# Acknowledgements and Release History

This article describes the contributors to the current version of the SEBoK. For information on contributors to past versions of the SEBoK, please follow the links under "SEBoK Release History" below. To learn more about the updates to the SEBoK for v. 2.1, please see the Letter from the Editor.

The BKCASE Project began in the fall of 2009. Its aim was to add to the professional practice of systems engineering by creating two closely related products:

- *Guide to the Systems Engineering Body of Knowledge (SEBoK)*
- *Graduate Reference Curriculum for Systems Engineering (GRCSE)*

## BKCASE History, Motivation, and Value

The **Guide to the Systems Engineering Body of Knowledge (SEBoK)** is a living authoritative guide that discusses knowledge relevant to Systems Engineering. It defines how that knowledge should be structured to facilitate understanding, and what reference sources are the most important to the discipline. The curriculum guidance in the **Graduate Reference Curriculum for Systems Engineering (GRCSE)** (Pyster and Olwell et al. 2015) makes reference to sections of the SEBoK to define its core knowledge; it also suggests broader program outcomes and objectives which reflect aspects of the professional practice of systems engineering as discussed across the SEBoK.

Between 2009 and 2012 BKCASE was led by Stevens Institute of Technology and the Naval Postgraduate School in coordination with several professional societies and sponsored by the U.S. Department of Defense (DoD), which provided generous funding. More than 75 authors and many other reviewers and supporters from dozens of companies, universities, and professional societies across 10 countries contributed many thousands of hours writing the SEBoK articles; their organizations provided significant other contributions in-kind.

The SEBoK came into being through recognition that the systems engineering discipline could benefit greatly by having a living authoritative guide closely related to those groups developing guidance on advancing the practice, education, research, work force development, professional certification, standards, etc.

At the beginning of 2013, BKCASE transitioned to a new governance model with shared stewardship between the Systems Engineering Research Center (SERC) [1], the International Council on Systems Engineering (INCOSE) [2], and the Institute of Electrical and Electronics Engineers Computer Society (IEEE-CS) [3]. This governance structure was formalized in a memorandum of understanding between the three stewards that was finalized in spring of 2013. The stewards have reconfirmed their commitment to making the SEBoK available at no cost to all users, a key principle of BKCASE.

As of the end of July 2019, SEBoK articles have had over 3.4M pageviews from 1.7M unique visits. We hope the SEBoK will regularly be used by thousands of systems engineers and others around the world as they undertake technical activities such as eliciting requirements, creating systems architectures, or analysis system test results; and professional development activities such as developing career paths for systems engineers, deciding new curricula for systems engineering university programs, etc.

## Governance

The SEBoK is shaped by the BKCASE Editorial Board and is overseen by the BKCASE Governing Board. A complete list of members for each of these bodies can be found on the BKCASE Governance and Editorial Board page.

## Content and Feature Updates for 2.1

This version of the SEBoK was released 31 October 2019. This is a significant release of the SEBoK which includes new articles, new functionality and minor updates throughout. The SEBoK PDF was also updated (see Download SEBoK PDF).

For more information about this release please refer to Development of SEBoK v. 2.1.

## SEBoK Release History

There have been 21 releases of the SEBoK to date, collected into 13 main releases.

### Main Releases

- Version 2.1 - Current version. This is a significant release with new articles, new functionality, and minor updates throughout.
- Version 2.0 - This was a major release of the SEBoK which included incorporation of multi-media and a number of changes to the functions of the SEBoK.
- Version 1.9.1 - This was a micro release of the SEBoK which included updates to the editorial board, and a number of updates to the wiki software.
- Version 1.9 - A minor update which included updates to the System Resilience article in Part 6: Related Disciplines, as well as a major restructuring of Part 7: Systems Engineering Implementation Examples. A new example has been added around the use of model based systems engineering for the thirty-meter telescope.
- Version 1.8 - A minor update, including an update of the Systems of Systems (SoS) knowledge area in Part 4: Applications of Systems Engineering where a number of articles were updated on the basis of developments in the area as well as on comments from the SoS and SE community. Part 6: Related Disciplines included updates to the Manufacturability and Producibility and Reliability, Availability, and Maintainability articles.
- Version 1.7 - A minor update, including a new Healthcare SE Knowledge Area (KA), expansion of the MBSE area with two new articles, Technical Leadership and Reliability, Availability, and Maintainability and a new case study on the Northwest Hydro System.
- Version 1.6 - A minor update, including a reorganization of Part 1 SEBoK Introduction, a new article on the Transition towards Model Based Systems Engineering and a new article giving an overview of Healthcare Systems Engineering, a restructure of the Systems Engineering and Specialty Engineering KA.
- Version 1.5 - A minor update, including a restructure and extension of the Software Engineering Knowledge Area, two new case studies, and a number of corrections of typographical errors and updates of outdated references throughout the SEBoK.
- Version 1.4 - A minor update, including changes related to ISO/IEC/IEEE 15288:2015 standard, three new case studies and updates to a number of articles.
- Version 1.3 - A minor update, including three new case studies, a new use case, updates to several existing articles, and updates to references.
- Version 1.2 - A minor update, including two new articles and revision of several existing articles.
- Version 1.1 - A minor update that made modest content improvements.
- Version 1.0 - The first version intended for broad use.

Click on the links above to read more information about each release.

## Wiki Team

In January 2011, the authors agreed to move from a document-based SEBoK to a wiki-based SEBoK, and beginning with v. 0.5, the SEBoK has been available at www.sebokwiki.org [4] Making the transition to a wiki provided three benefits:

1. easy worldwide access to the SEBoK;
2. more methods for search and navigation; and
3. a forum for community feedback alongside content that remains stable between versions.

The Managing Editor is responsible for maintenance of the wiki infrastructure as well as technical review of all materials prior to publication. Contact the managing editor at emtnicole@gmail.com [3]

The wiki is currently supported by Ike Hecht from WikiWorks.

**SEBoK v. 2.1, released 31 October 2019**

## References

[1] http://www.sercuarc.org
[2] http://www.incose.org
[3] http://www.computer.org
[4] http://www.sebokwiki.org

# Cite the SEBoK

When **citing the SEBoK in general**, users must cite in the following manner:

> SEBoK Editorial Board. 2019. *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*, v. 2.1, R.J. Cloutier (Editor in Chief). Hoboken, NJ: The Trustees of the Stevens Institute of Technology. Accessed [DATE]. www.sebokwiki.org. BKCASE is managed and maintained by the Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Computer Society.

To **cite a specific article** within the SEBoK, please use:

> SEBoK Authors. Author name(s). "Article Title." in SEBoK Editorial Board. 2019. *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*, v. 2.1 R.J. Cloutier (Editor in Chief). Hoboken, NJ: The Trustees of the Stevens Institute of Technology. Accessed [DATE]. www.sebokwiki.org. BKCASE is managed and maintained by the Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Computer Society.

***Note that each page will include the by line (author names) for the article. If no byline is listed, please use "SEBoK Authors".***

When **using material** from the SEBoK, attribute the work as follows:

> This material is used under a Creative Commons Attribution-NonCommercial ShareAlike 3.0 Unported License from The Trustees of the Stevens Institute of Technology. See Stevens Terms for Publication located in Copyright Information.

**Cite this Page**

This feature is located under "Tools" on the left menu. It provides full information to cite the specific article that you are currently viewing; this information is provided in various common citation styles including APA, MLA, and Chicago.

# Bkcase Wiki:Copyright

**Please read this page which contains information about how and on what terms you may use, copy, share, quote or cite the Systems Engineering Body of Knowledge (SEBoK):**

## Copyright and Licensing

## Attribution

When **using text material from the SEBoK**, users who have accepted one of the Creative Commons Licenses described above terms noted below must attribute the work as follows:

This material is used under a Creative Commons Attribution-NonCommercial ShareAlike 3.0 Unported License from The Trustees of the Stevens Institute of Technology.

When **citing the SEBoK in general**, please refer to the format described on the Cite the SEBoK page.

When **using images, figures, or tables from the SEBoK**, please note the following intellectual property (IP) classifications:

- Materials listed as "SEBoK Original" may be used in accordance with the Creative Commons attribution (above).
- Materials listed as "Public Domain" may be used in accordance with information in the public domain.
- Materials listed as "Used with Permission" are copyrighted and *permission must be sought from the copyright owner* to reuse them.

# Table of Contents

# SEBoK Table of Contents

**Navigating the SEBoK**

The SEBoK may be searched in the same way as a traditional wiki. In addition, navigation links have been added to each page. Please use these links if you would like to navigate the SEBoK sequentially through the table of contents. These links look like:

< Previous Article | Parent Article | Next Article >

- *Previous* - The "Previous" link will take you back one article in the table of contents.
- *Next* - The "Next" link will take you forward one article in the table of contents.
- *Parent* - The "Parent" link takes you up one level in the table of contents. (For topic articles, the "Parent" link will take you to the overarching Knowledge Area (KA). For KA articles, the "Parent" link will take you to the overarching Part.)

---

## Part 2

## Part 4

## Part 5

---

Go to First SEBoK Article >

**SEBoK v. 2.1, released 31 October 2019**

# Part 3: Systems Engineering and Management

## Systems Engineering and Management

***Lead Authors:*** *Bud Lawson, Alan Faisandier,* ***Contributing Authors:*** *Rick Adcock, Dick Fairley, Garry Roedler, Ray Madachy, Deva Henry, Sanford Friedenthal*

Part 3 of the Guide to the SE Body of Knowledge (SEBoK) focuses on the general knowledge of *how* systems are engineered.

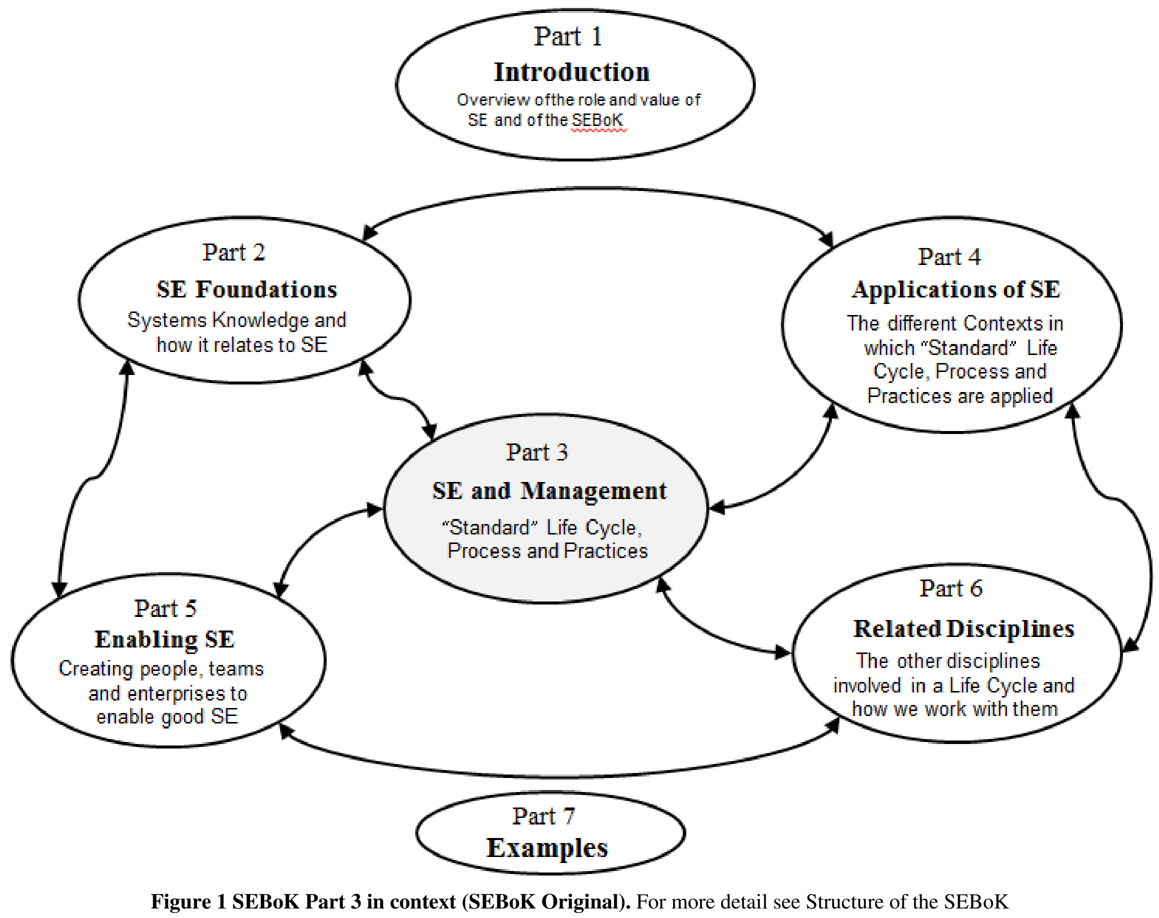

Part 1
Introduction
Overview of the role and value of SE and of the SEBoK
Part 2
SE Foundations
Systems Knowledge and how it relates to SE
Part 4
Applications of SE
The different Contexts in which "Standard" Life Cycle, Process and Practices are applied
Part 3
SE and Management
"Standard" Life Cycle, Process and Practices
Part 5
Enabling SE
Creating people, teams and enterprises to enable good SE
Part 6
Related Disciplines
The other disciplines involved in a Life Cycle and how we work with them
Part 7
Examples

**Figure 1 SEBoK Part 3 in context (SEBoK Original).** For more detail see Structure of the SEBoK

This part builds upon Part 2: Foundations of Systems Engineering, which discusses the need for a systems approach applied to one or more engineered system contexts as a part of managed interventions into complex real world problems. Part 3 provides an overview of the common uses of life cycle models to organize the technical and none technical aspects of SE and discusses Systems Engineering Management activities. Part 3 also discusses the most commonly-used SE technical processes; provides additional references to the common methods, tools, and techniques used in these processes

The commonly recognized definition of systems engineering (SE) used across the SEBoK (INCOSE 2015) defines SE as an interdisciplinary approach which applies across the complete life cycle of an identified System-of-Interest.

The definition states that systems engineering "**integrates all the disciplines and speciality groups into a team effort forming a structured development process that proceeds from concept to production to operation**". Thus, SE is an engineering discipline concerned with all aspects of an engineered systems life, including how we organize to do the engineering, what is produced by that engineering and how the resulting systems are used and sustained to meet stakeholder needs.

Part 3 provides only an overview of how systems are engineered in a generic sense. Part 4 provides more specific information as to how the principles discussed in Part 3 are applied differently in consideration of product systems, service systems, enterprise systems, and systems of systems (SoS) contexts. Part 5 explains how people and organizations may approach utilizing these principles as part of a holistic systems approach. Part 6 contains references to other engineering and management disciplines, which work with the SE processes within a systems life cycle, but do not fall under the umbrella of SE.

Systems engineering is transitioning to a model-based approach, model-based systems engineering (MBSE), like many other engineering disciplines. The aim is to enhance productivity and quality, and to cope with the design of increasingly complex systems. Although, models have always been used by systems engineering to create information about engineered systems, that information has been translated and managed through document based artifacts. In a model-based approach, the information about the system is captured in a shared system model, made up of a set of integrated models appropriate to the life cycle stages. This model is managed and controlled throughout the system life cycle as noted in Part 2 under Representing Systems with Models. This provides the ability to maintain more consistent, precise, and traceable information about the system. The system model provides an authoritative source of information that can be communicated across the development team and other stakeholders, can be used to generate views of the system relevant to particular stakeholders, and be used to generate documentation about the system similar to more traditional systems engineering documentation. The model can also be analyzed to assess the integrity of the system specification and design. A model also captures knowledge in a way that can be more readily reused than traditional document based approaches. In a model-based systems engineering approach, the processes referred to in this and other Parts of the SEBoK remain fundamentally the same, but the artifacts produced are model-based. Some examples of MBSE methods are highlighted in A Survey of Model-Based Systems Engineering (MBSE) Methodologies (Estefan 2008). It is anticipated that as the transition to model-based practices occurs, the SEBoK will be updated to reflect the body of current and emerging practice.

## Knowledge Areas in Part 3

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. Part 3 contains the following knowledge areas:

- Introduction to Life Cycle Processes
- Life Cycle Models
- Concept Definition
- System Definition
- System Realization
- System Deployment and Use
- Systems Engineering Management
- Product and Service Life Management
- Systems Engineering Standards

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

## Value of Ontology Concepts for Systems Engineering

Ontology is the set of entities presupposed by a theory (Collins English Dictionary 2011). Systems engineering, and system development in particular, is based on concepts related to mathematics and proven practices. A SE ontology can be defined considering the following path.

SE provides engineers with an approach based on a set of concepts (i.e., stakeholder, requirement, function, scenario, system element, etc.) and generic processes. Each process is composed of a set of activities and tasks gathered logically around a theme or a purpose. A process describes "what to do" using the applied concepts. The implementation of the activities and tasks is supported by methods and modeling techniques, which are composed themselves of elementary tasks; they describe the "how to do" of SE. The activities and tasks of SE are transformations of generic data using predefined concepts. Those generic data are called entities, classes, or types. Each *entity* is characterized by specific *attributes*, and each attribute may have a different value. All along their execution, the activities and tasks of processes, methods, and modeling techniques exchange instances of generic entities according to logical *relationships*. These relationships allow the engineer to link the entities between themselves (traceability) and to follow a logical sequence of the activities and the global progression (engineering management). Cardinality is associated with every relationship, expressing the minimum and maximum number of entities that are required in order to make the relationship valid. Additional information on this subject may be found in *Engineering Complex Systems with Models and Objects* (Oliver, Kelliher, and Keegan 1997).

The set of SE entities and their relationships form an ontology, which is also referred to as an "engineering meta-model". Such an approach is used and defined in the ISO 10303 standard (ISO 2007). There are many benefits to using an ontology. The ontology allows or forces:

- the use of a standardized vocabulary, with carefully chosen names, which helps to avoid the use of synonyms in the processes, methods, and modeling techniques
- the reconciliation of the vocabulary used in different modeling techniques and methods
- the automatic appearance of the traceability requirements when implemented in databases, SE tools or workbenches, and the quick identification of the impacts of modifications in the engineering data set
- the continual observation of the consistency and completeness of engineering data; etc.

Throughout Part 3, there are discussions of the ontological elements specifically relevant to a given topic.

## Mapping of Topics to ISO/IEC 15288, System Life Cycle Processes

Figure 2, below, shows the relative position of the KA's of the SEBoK with respect to the processes outlined in the ISO/IEC/IEEE 15288 (ISO 2015) standard.

As shown, all of the major processes described in ISO/IEC/IEE 15288:2015 are discussed within the SEBoK.

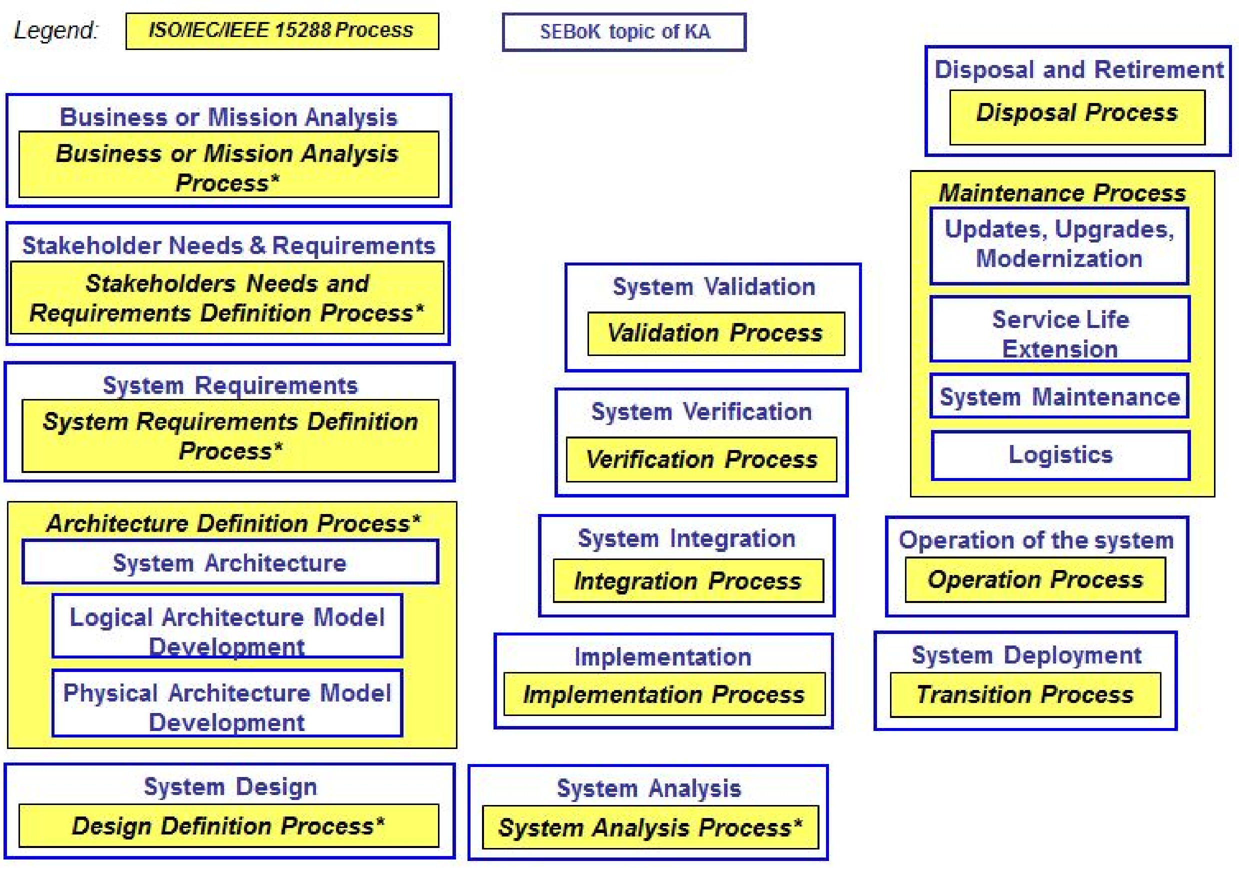


**Figure 2. Mapping of Technical Topics of Knowledge Areas of SEBoK with ISO/IEC/IEEE 15288 Technical Processes.** (SEBoK Original)

The ISO/IEC/IEEE 15288:2015 marked with an * are new or have been renamed and modified in scope for this revision of the standard.

These changes and associated changes to the SEBoK now mean that the two are significantly more closely aligned than before. It should also be noted that the latest update of the INCOSE SE Handbook (INCOSE 2015) is now fully aligned with the 2015 revision of the standard.

Any future evolution of Life Cycle Process knowledge in the SEBoK will be complementary to these standard descriptions of the generic SE process set.

## References

Collins English Dictionary, s.v. "Ontology." 2011.

Estefan, J. 2008. *A Survey of Model-Based Systems Engineering (MBSE) Methodologies*, rev, B. Seattle, WA: International Council on Systems Engineering. INCOSE-TD-2007-003-02. Accessed April 13, 2015 at http://www.omgsysml.org/MBSE_Methodology_Survey_RevB.pdf

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute for Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO. 2007. *Systems Engineering and Design.* Geneva, Switzerland: International Organization for Standardization (ISO). ISO 10303-AP233.

Oliver, D., T. Kelliher, and J. Keegan. 1997. *Engineering Complex Systems with Models and Objects*. New York, NY, USA: McGraw-Hill.

### Primary References

INCOSE. 2015. *Systems Engineering Handbook* - A Guide for System Life Cycle Processes and Activities", version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

ISO/IEC/IEEE. 2015. Systems and Software Engineering -- System Life Cycle Processes. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2015.

### Additional References

Bell Telephone Laboratories. 1982. *Engineering and Operations in the Bell System*. Murray Hill, NJ: Bell Telephone Laboratories.

Fortescue, P.W., J. Stark, and G. Swinerd. 2003. *Spacecraft Systems Engineering*. New York, NY, USA: J. Wiley.

---

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Knowledge Area: Introduction to Life Cycle Processes

## Introduction to Life Cycle Processes

***Lead Authors:*** *Rick Adcock, Sanford Friedenthal*

In this Knowledge Area we introduce key principles of life cycle, life cycle model and life cycle processes. A generic SE paradigm is described; this forms a starting point for discussions of more detailed life cycle knowledge.

### Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Generic Life Cycle Model
- Applying Life Cycle Processes
- Life Cycle Processes and Enterprise Need

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

### Life Cycle Terminology

The term life cycle is one that engineering has borrowed from the natural sciences, it is used to describe both the changes a single organism goes through over it life and how the lives of multiple organisms interact to sustain or evolve a population. We use it in engineering in the same ways to describe the complete life of an instance of a system-of-interest (SoI); and the managed combination of multiple such instances to provide capabilities which deliver stakeholder satisfaction.

A life cycle model identifies the major stages that a specific SoI goes through, from its inception to its retirement. Life cycle models are generally implemented in development projects, and are strongly aligned with management planning and decision making.

### Generic Systems Engineering Paradigm

Figure 1 identifies the overall goals of any SE effort, which are: the understanding of stakeholder value; the selection of a specific need to be addressed; the transformation of that need into a system (the product or service that provides for the need); and the use of that product or service to provide the stakeholder value. This paradigm has been developed according to the principles of the systems approach discussed in Part 2 and is used to establish a basis for the KAs in Part 3 and Part 4 of the SEBoK.

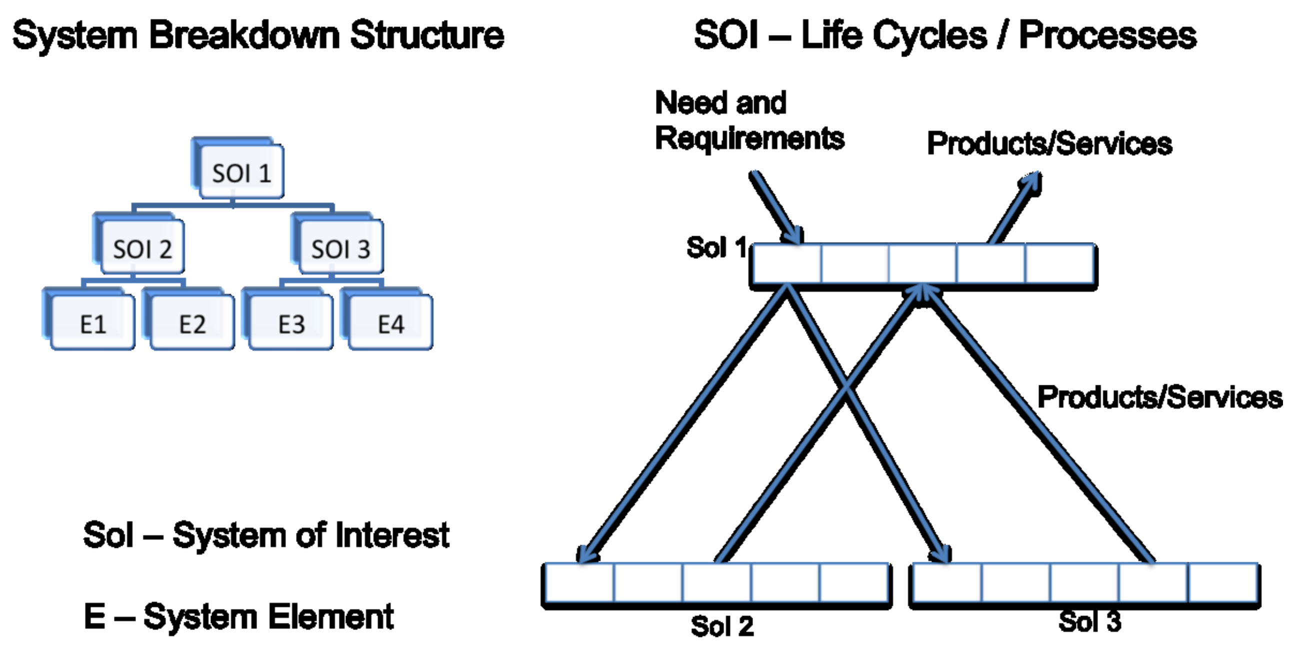


**Figure 1. Generic Systems Engineering Paradigm.** (SEBoK Original)

On the left side of Figure 1, there are SoI's identified in the formation of a system breakdown structure. SoI 1 is broken down into its basic elements, which in this case are systems as well (SoI 2 and SoI 3). These two systems are composed of system elements that are not refined any further.

On the right side of Figure 1, each SoI has a corresponding life cycle model (glossary) which is composed of stages that are populated with processes. The function of these processes is to define the work that is to be performed and the associated artifacts to be produced. In a model-based approach, these artifacts are captured in the system model that represent the SoI's. Note that some of the requirements defined to meet the need are distributed in the early stages of the life cycle for SoI 1, while others are designated to the life cycles of SoI 2 or SoI 3. The decomposition of the system illustrates the fundamental concept of recursion (glossary) as defined in the ISO/IEC/IEEE 15288 standard; with the standard being reapplied for each SoI (ISO 2015). It is important to point out that the requirements may be allocated to different system elements, which may be integrated in different life cycle stages of any of the three SoI's; however, together they form a cohesive system. For example, SoI 1 may be a simple vehicle composed of a chassis, motor and controls, SoI 2 an embedded hardware system, and Sol 3 a software intensive interface and control system.

When performing SE processes in stages, iteration (glossary) between stages is often required (e.g. in successive refinement of the definition of the system or in providing an update or upgrade of an existing system). The work performed in the processes and stages can be performed in a concurrent manner within the life cycle of any of the systems of interest and also among the multiple life cycles.

This paradigm provides a fundamental framework for understanding generic SE (seen in Part 3), as well as for the application of SE to the various types of systems described in Part 4.

## References

### Works Cited

ISO/IEC/IEEE. 2015.*Systems and software engineering - system life cycle processes*.Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC 15288:2015.

### Primary References

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

### Additional References

None.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Generic Life Cycle Model

***Lead Authors:*** *Kevin Forsberg, Richard Turner*, ***Contributing Author:*** *Rick Adcock*

As discussed in the generic life cycle paradigm in Introduction to Life Cycle Processes each system-of-interest (SoI) has an associated life cycle model. The generic life cycle model below applies to a single SoI. SE must generally be synchronised across a number tailored instances of such life cycle models to fully satisfy stakeholder needs. More complex life cycle models which address this are described in Life Cycle Models.

## A Generic System Life Cycle Model

There is no single “one-size-fits-all” system life cycle model that can provide specific guidance for all project situations. Figure 1, adapted from (Lawson 2010, ISO 2015, and ISO 2010), provides a generic life cycle model that forms a starting point for the most common versions of pre-specified, evolutionary, sequential, opportunistic, and concurrent life cycle processes. The model is defined as a set of stages, within which technical and management activities are performed. The stages are terminated by decision gates where the key stakeholders decide whether to proceed into the next stage, to remain in the current stage, or to terminate or re-scope related projects.

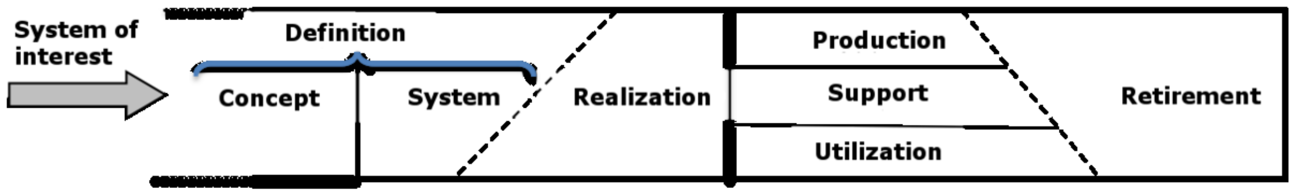


**Figure 1. A Generic Life Cycle Model.** (SEBoK Original)

Elaborated definitions of these stages are provided below, in the glossary, and in various other ways in subsequent articles.

The **Concept Definition** stage begins with a decision by a protagonist (individual or organization) to invest resources in a new or improved engineered system. Inception begins with a set of stakeholders agreeing to the need for change to an engineered system context and exploring whether new or modified SoI can be developed, in which the life cycle benefits are worth the investments in the life cycle costs. Activities include: developing the concept of operations and business case; determining the key stakeholders and their desired capabilities; negotiating the stakeholder requirements among the key stakeholders and selecting the system's non-developmental items (NDIs).

The **System Definition** stage begins when the key stakeholders decide that the business needs and stakeholder requirements are sufficiently well defined to justify committing the resources necessary to define a solution options in sufficient detail to answer the life cycle cost question identified in concept definition and provide a basis of system realization if appropriate. Activities include developing system architectures; defining and agreeing levels of system requirements; developing systems-level life cycle plans and performing system analysis in order to illustrate the compatibility and feasibility of the resulting system definition. The transition into the system realization stage can lead to either single-pass or multiple-pass development.

It should be noted that the concept and system definition activities above describe activities performed by *systems engineers* when performing *systems engineering*. There is a very strong concurrency between proposing a problem situation or opportunity and describing one or more possible system solutions, as discussed in Systems Approach Applied to Engineered Systems. Other related definition activities include: prototyping or actual development of high-risk items to show evidence of system feasibility; collaboration with business analysts or performing mission effectiveness analyses to provide a viable business case for proceeding into realization; and modifications to realized systems to improve their production, support or utilization. These activities will generally happen through the system life cycle to handle system evolution, especially under multiple-pass development. This is discussed in more detail in the Life Cycle Models knowledge area.

The **System Realization** stage begins when the key stakeholders decide that the SoI architecture and feasibility evidence are sufficiently low-risk to justify committing the resources necessary to develop and sustain the initial operational capability (IOC) or the single-pass development of the full operational capability (FOC). Activities include: construction of the developmental elements; integration of these with each other and with the non-developmental item (NDI) elements; verification and validation of the elements and their integration as it proceeds; and preparing for the concurrent production, support, and utilization activities.

The **System Production, Support, and Utilization (PSU)** stages begins when the key stakeholders decide that the SoI life-cycle feasibility and safety are at a sufficiently low-risk level that justifies committing the resources necessary to produce, field, support, and utilize the system over its expected lifetime. The lifetimes of production, support, and utilization are likely to be different. After market support will generally continue after production is complete and users will often continue to use unsupported systems.

**System Production** involves the fabrication of instances or versions of a SoI and of associated after market spare parts. It also includes production quality monitoring and improvement; product or service acceptance activities; and continuous production process improvement. It may include low-rate initial production (LRIP) to mature the production process or to promote the continued preservation of the production capability for future spikes in demand.

**Systems Support** includes various classes of maintenance: corrective (for defects), adaptive (for interoperability with independently evolving co-dependent systems), and perfective (for enhancement of performance, usability, or other key performance parameters). It also includes hot lines and responders for user or emergency support and the provisioning of needed consumables (gas, water, power, etc.). Its boundaries include some gray areas, such as the boundary between small system enhancements and the development of larger complementary new additions, and the boundary between rework/maintenance of earlier fielded increments in incremental or evolutionary development. *Systems Support* usually continues after *System Production* is terminated.

**System Utilization** includes the use of the SoI in its context by operators, administrators, the general public, or systems above it in the system-of-interest hierarchy. It usually continues after *Systems Support* is terminated.

The **System Retirement** stage is often executed incrementally as system versions or elements become obsolete or are no longer economical to support and therefore undergo disposal or recycling of their content. Increasingly affordable considerations make system re-purposing an attractive alternative.

## Applying the Life Cycle Model

Figure 1 shows just the single-step approach for proceeding through the stages of a SoI life cycle. In Life Cycle Models knowledge area we discuss examples of real world enterprises and their drivers, both technical and organizational. These have lead to a number of documented approaches for sequencing the life cycle stages to deal with some of the issues raised. The Life Cycle Models KA summarizes a number of incremental and evolutionary life cycle models, including their main strengths and weaknesses and also discusses criteria for choosing the best-fit approach.

In figure 1 we have listed key technical and management activities critical to successful completion of each stage. This is a useful way to illustrate the goals of each stage and gives an indication of how processes align with these stages. This can be important when considering how to plan for resources, milestones, etc. However, it is important to observe that the execution of process activities are not compartmentalized to particular life cycle stages (Lawson 2010). In Applying Life Cycle Processes we discuss a number of views on the nature of the inter-relationships between process activities within a life cycle model. In general, the technical and management activities are applied in accordance with the principles of concurrency, iteration and recursion described in the generic life cycle paradigm.

## References

### Works Cited

ISO/IEC/IEEE. 2015.*Systems and software engineering - system life cycle processes*.Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC 15288:2015.

ISO/IEC. 2010. Systems and Software Engineering, Part 1: Guide for Life Cycle Management. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 24748-1:2010.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

### Primary References

Forsberg, K., H. Mooz, H. Cotterman. 2005. *Visualizing Project Management*, 3rd Ed. Hoboken, NJ: J. Wiley & Sons.

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

### Additional References

None.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Applying Life Cycle Processes

***Lead Author:*** *Rick Adcock*

The Generic Life Cycle Model describes a set of life cycle stages and their relationships. In defining this we described some of the technical and management activities critical to the success of each stage. While this association of activity to stage is important we must also recognise the through life relationships between these activities to ensure we take a systems approach.

Systems Engineering technical and management activities are defined in a set of life cycle processes. These group together closely related activities and allow us to describe the relationships between them. In this topic we discuss a number of views on the nature of the inter-relationships between process activities within a life cycle model.

In general, the technical and management activities are applied in accordance with the principles of concurrency, iteration and recursion described in the generic systems engineering paradigm. These principles overlap to some extent and can be seen as related views of the same fundamental need to ensure we can take a holistic systems approach, while allowing for some structuring and sequence of our activities. The views presented below should be seen as examples of the ways in which different SE authors present these overlapping ideas.

## Life Cycle Process Terminology

### Process

A process is a series of actions or steps taken in order to achieve a particular end. Processes can be performed by humans or machines transforming inputs into outputs.

In SEBoK processes are interpreted in several ways, including: technical, life cycle, business, or manufacturing flow processes. Many of the Part 3 sections are structured along technical processes (e.g. design, verification); however, Life Cycle Models also describes a number of high level program life cycle sequence which call themselves processes (e.g. rational unified process (RUP), etc.).

Part 4: Applications of Systems Engineering and Part 5: Enabling Systems Engineering utilize processes that are related to services and business enterprise operations.

Systems Engineering life cycle process define technical and management activities performed across one or more stages to provide the information needed to make life cycle decisions; and to enable realization, use and sustainment of a system-of-interest (SoI) across its life cycle model as necessary. This relationship between life cycle model and process activities can be used to describe how SE is applied to different system contexts.

### Requirement

A requirement is something that is needed or wanted, but may not be compulsory in all circumstances. Requirements may refer to product or process characteristics or constraints. Different understandings of requirements are dependent on process state, level of abstraction, and type (e.g. functional, performance, constraint). An individual requirement may also have multiple interpretations over time.

Requirements exist at multiple levels of enterprise or system with multiple levels abstraction. This ranges from the highest level of the enterprise capability or customer need to the lowest level of the system design .Thus, requirements need to be defined at the appropriate level of detail for the level of the entity to which it applies. See the article Life Cycle Processes and Enterprise Needs for further detail on the transformation of needs and requirements from the enterprise to the lowest system element across concept definition and system definition.

### Architecture

An architecture refers to the organizational structure of a system, whereby the system can be defined in different contexts. Architecting is the art or practice of designing the structures. See below for further discussions on the use of levels of Logical and Physical architecture models to define related system and system elements; and support the requirements activities.

Architectures can apply for a system product, enterprise, or service. For example, Part 3 mostly considers product or service related architectures that systems engineers create, but enterprise architecture describes the structure of an organization. Part 5: Enabling Systems Engineering interprets enterprise architecture in a much broader manner than an IT system used across an organization, which is a specific instance of architecture.

Frameworks are closely related to architectures, as they are ways of representing architectures. See the glossary of terms Architecture Framework for definition and examples.

### Other Processes

A number of other life cycle processes are mentioned below, including system analysis, integration, verification, validation, deployment, operation, maintenance and disposal are discussed in detail in the System Realization and System Deployment and Use knowledge areas.

## Life Cycle Process Concurrency

In the Generic Life Cycle Model we have listed key activities critical to successful completion of each stage. This is a useful way to illustrate the goals of each stage and gives an indication of how processes align with these stages. This can be important when considering how to plan for resources, milestones, etc. However, it is important to observe that the execution of process activities are not compartmentalized to particular life cycle stages (Lawson 2010).

Figure 1 shows a simple illustration of the through life nature of technical and management processes. This figure builds directly on the "hump diagram" principles described in Systems Approach Applied to Engineered Systems.

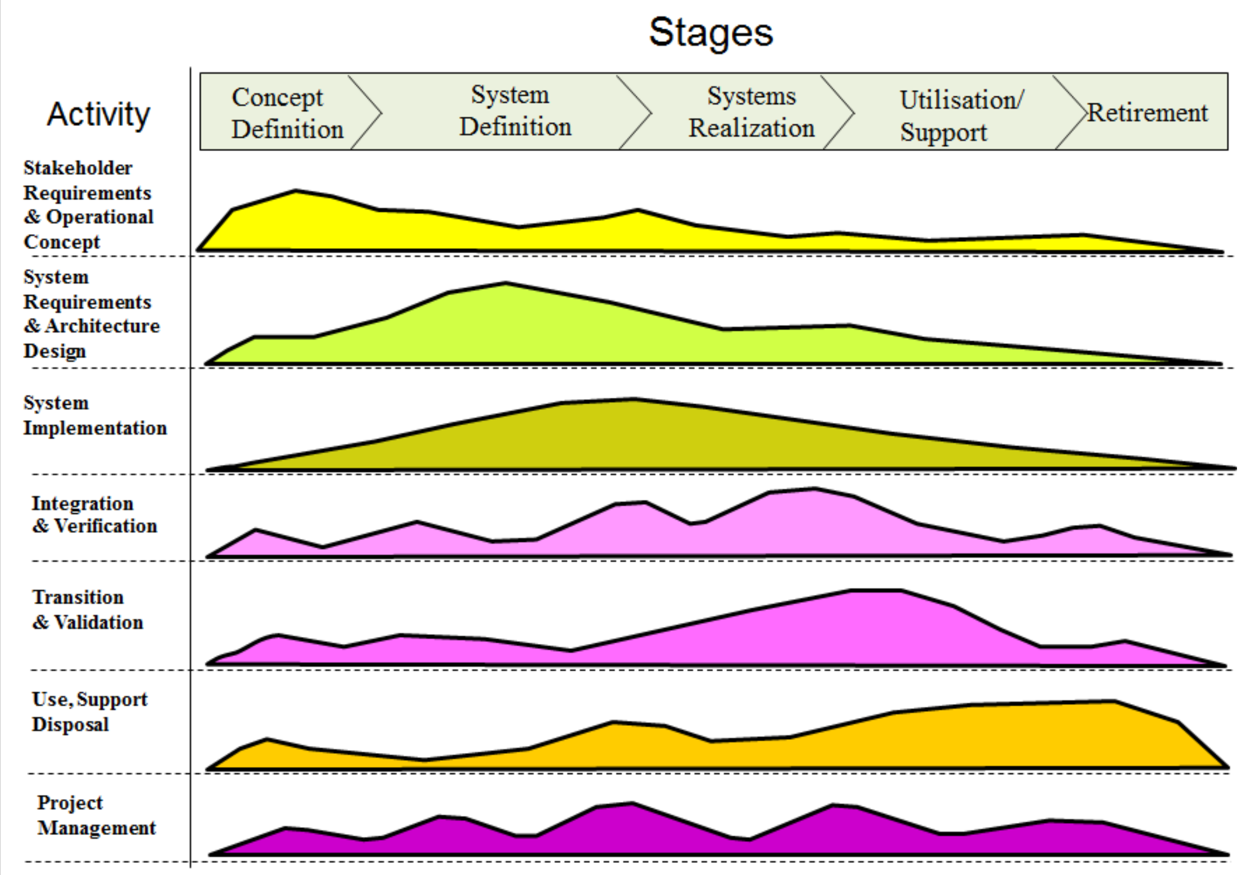


Figure 1: Generic Relationships between life cycle stages and processes (modified from Lawson 2010)

The lines on this diagram represent the amount of activity for each process over the generic life cycle. The peaks (or humps) of activity represent the periods when a process activity becomes the main focus of a stage. The activity before and after these peaks may represent through life issues raised by a process focus, e.g. how will likely maintenance constraints be represented in the system requirements. These considerations help maintain a more holistic perspective in each stage; or they can represent forward planning to ensure the resources needed to complete future activities have been included in estimates and plans, e.g. are all resources need for verification in place or available. Ensuring this hump diagram principle is implemented in a way which is achievable, affordable and appropriate to the situation is a critical driver for all life cycle models.

## Life Cycle Process Iteration

The concept of iteration applies to life cycle stages within a life cycle model, and also applies to processes. Figure 2 below gives an example of iteration in the life cycle processes associated with concept and system definition.

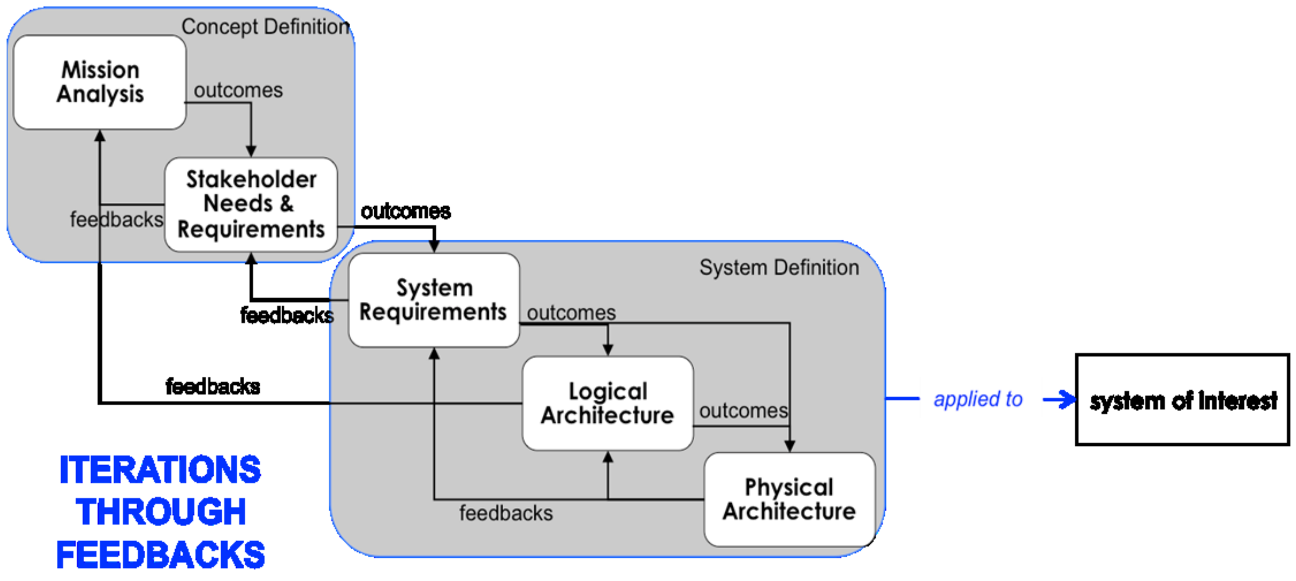


**Figure 2. Example of Iterations of Processes Related to System Definition (Faisandier 2012).** Permission Granted by Sinergy'Com. All other rights are reserved by the copyright owner.

There is generally a close coupling between the exploration of a problem or opportunity and the definition of one or more feasible solutions, see Systems Approach to Engineered Systems. Thus the related processes in this example will normally be applied in an iterative way. The relationships between these process are further discussed in the System Definition KA.

Figure 3 below gives an example of the iteration between the other life cycle processes.

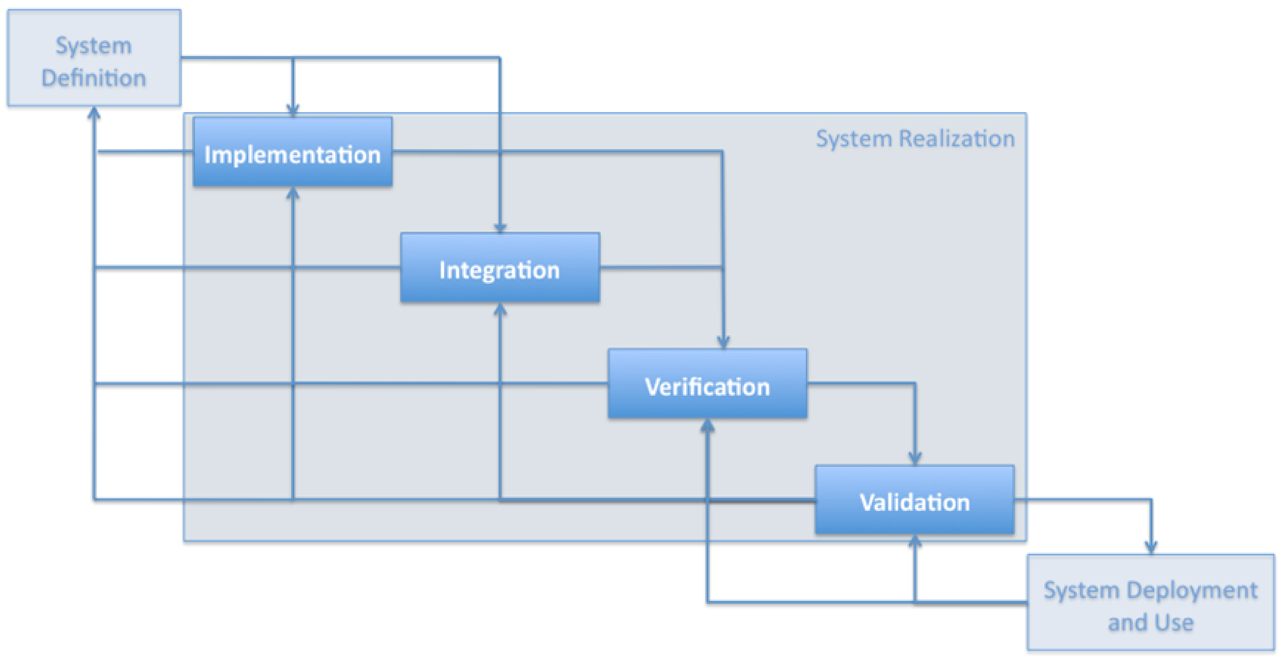


**Figure 3. System Realization.** (SEBoK Original)

The iterations in this example relate to the overlaps in process outcomes shown in Figure 1. They either allow consideration of cross process issues to influence the system definition, e.g. considering likely integration or verification approaches might make us think about failure modes or add data collection or monitoring elements into the system. Or they allow risk management and through life planning activities to identify the need for future activities.

The relationships between these processes are further discussed in system realization and system deployment and use.

## Life Cycle Process Recursion

The comprehensive definition of a SoI is generally achieved using decomposition layers and system elements (glossary). Figure 4 presents a fundamental schema of a system breakdown structure.

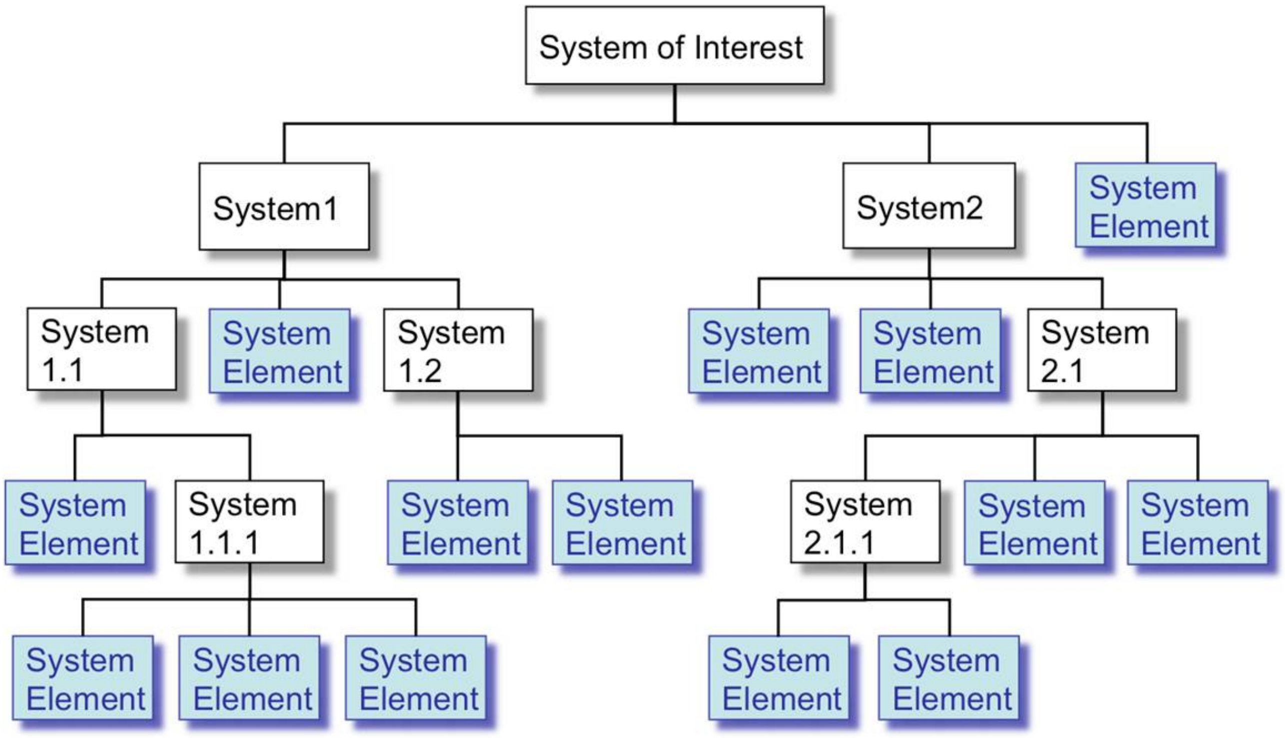


**Figure 4. Hierarchical Decomposition of a System-of-Interest (Faisandier 2012).** Permission Granted by Sinergy'Com. All other rights are reserved by the copyright owner.

In each decomposition layer and for each system, the System Definition processes are applied recursively because the notion of "system" is in itself recursive; the notions of SoI, system, and system element are based on the same concepts (see Part 2). Figure 5 shows an example of the recursion of life cycle processes.

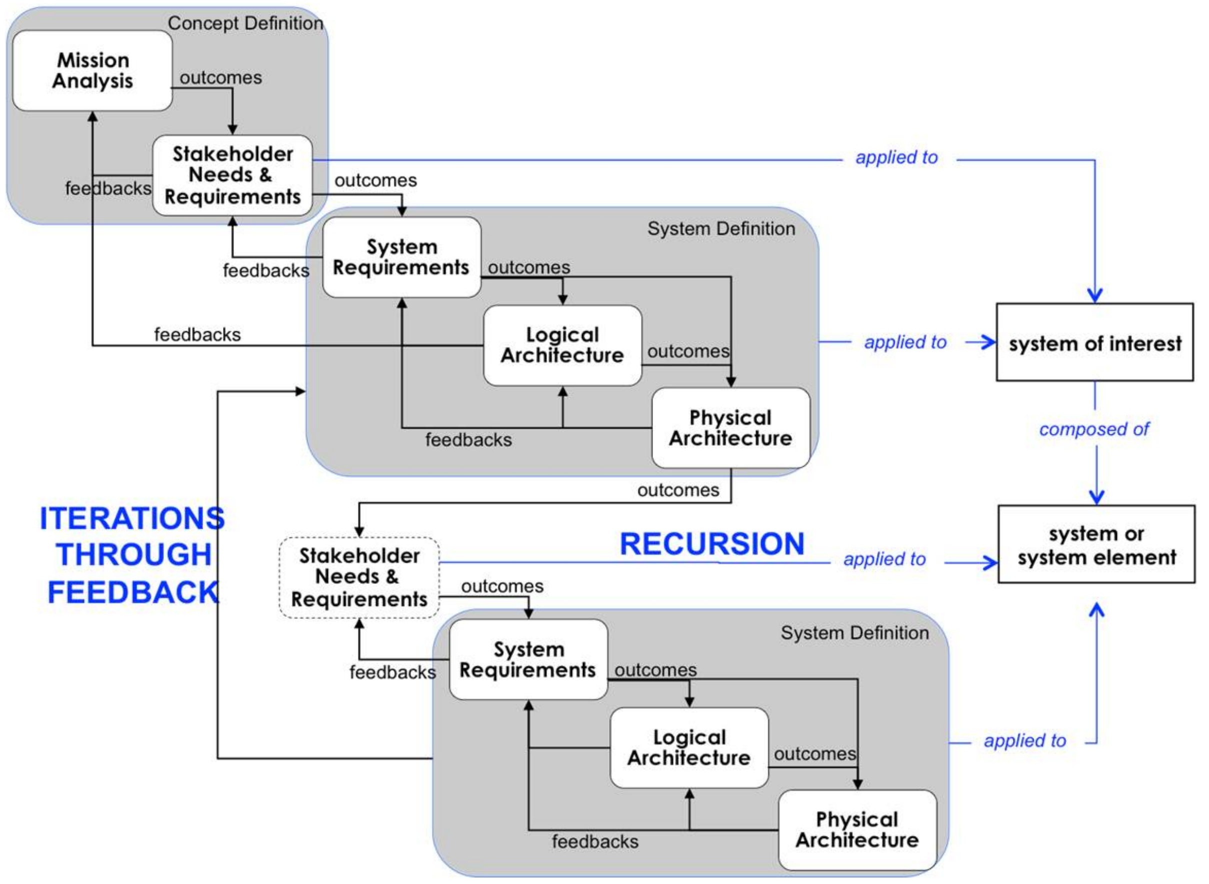


**Figure 5. Recursion of Processes on Layers (Faisandier 2012).** Permission Granted by Sinergy'Com. All other rights are reserved by the copyright owner.

## Systems Approach to Solution Synthesis

The sections above give different perspectives on how SE life cycle processes are related and how this shapes their application. Solution synthesis is described in Part 2 as away of taking a systems approach to creating solution. Synthesis is in general a mixture of top down and bottom up approaches as discussed below.

### Top-Down Approach: From Problem to Solution

In a top-down approach, concept definition activities are focused primarily on understanding the problem, the operational needs/requirements within the problem space, and the conditions that constrain the solution and bound the solution space. The concept definition activities determine the mission context, Mission Analysis, and the needs to be fulfilled in that context by a new or modified system (i.e. the SoI), and addresses stakeholder needs and requirements.

The System Definition activities consider functional, behavioral, temporal, and physical aspects of one or more solutions based on the results of concept definition. System Analysis considers the advantages and disadvantages of the proposed system solutions both in terms of how they satisfy the needs established in concept definition, as well as the relative cost, time scales and other development issues. This may require further refinement of the concept definition to ensure all legacy relationships and stakeholders relevant to a particular solution architecture have been considered in the stakeholder requirements.

The outcomes of this iteration between Concept Definition and System Definition define a required system solution and its associated problem context, which are used for System Realization, System Deployment and Use, and Product and Service Life Management of one or more solution implementations. In this approach problem understanding and solution selection activities are completed in the front-end portion of system development and design and then maintained and refined as necessary throughout the life cycle of any resulting solution systems. Top-down activities can be sequential, iterative, recursive or evolutionary depending upon the life cycle model.

### Bottom-Up Approach: Evolution of the Solution

In some situations, the concept definition activities determine the need to evolve existing capabilities or add new capabilities to an existing system. During the concept definition, the alternatives to address the needs are evaluated. Engineers are then led to reconsider the system definition in order to modify or adapt some structural, functional, behavioral, or temporal properties during the product (glossary) or service (glossary) life cycle for a changing context (glossary) of use or for the purpose of improving existing solutions.

Reverse engineering is often necessary to enable system engineers to (re)characterize the properties of the system-of-interest (SoI) or its elements. This is an important step to ensure that system engineers understand the SoI before beginning modification. For more information on system definition, see the System Definition article.

A bottom-up approach is necessary for analysis purposes, or for (re)using existing elements in the design architecture. Changes in the context of use or a need for improvement can prompt this. In contrast, a top-down approach is generally used to define an initial design solution corresponding to a problem or a set of needs.

### Solution Synthesis

In most real problems a combination of bottom-up and top-down approaches provides the right mixture of innovative solution thinking driven by need and constrained and pragmatic thinking driven by what already exists. This is often referred to as a “middle-out” approach.

As well as being the most pragmatic approach, synthesis has the potential to keep the life cycle focused on whole system issues, while allow the exploration of the focused levels of detail needed to describe realizable solutions, see Synthesising System Solutions.

## References

### Works Cited

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

### Primary References

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

### Additional References

None.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Life Cycle Processes and Enterprise Need

***Lead Author:*** *Rick Adcock*

The Generic Life Cycle Model describes a simple translation from a need to achieve an outcome to a proposal to realize a new or modified engineered system. This then forms the basis for the decision to invest time, money and other resources in the life cycle of that system-of-interest (SoI). A significant proportion of the SE activities involved in this transformation involve varying levels and types of requirement.

## Requirements and Life Cycle

During the concept definition stage of a life cycle, as the enterprise identifies new capabilities that are desired, the business or mission analysis develops a high level set of strategies and need (which may be expressed as mission or business requirements) that reflect the problem space perspective. As the solution space is explored and solution classes are characterized, stakeholder needs are developed and transformed into stakeholder requirements (from a user perspective). After the solution class has been determined and the specific solution is sought, during the system definition stage of a life cycle, the stakeholder requirements are transformed into system requirements (from a solution perspective). As the system definition recursively defines the lower level detail of the solution, requirements are defined with lower levels of abstraction. At the highest level, the ideal requirement is implementation-independent, and therefore not specific to a solution, allowing for a range of possible solutions. At the lowest level, requirement statements may become more specific to the selected solution.

Concept Definition has further descriptions of business or enterprise and stakeholder needs and requirements. It discusses how the new capability for the business or enterprise is defined as part of the understanding of the problem space. It also discusses the development of the stakeholder needs and their transformation into requirements from the user perspective.

System Definition has further descriptions of requirements and their types (e.g. system requirement, stakeholder requirement, derived requirement). It discusses how different process roles/ states, levels, and the nature of requirements apply to the understanding of requirements.

## Transforming Enterprise Needs to Requirements

Needs and requirements can exist at a number of levels and the terminology used to describe these levels will vary between application domains and the enterprise which serve them. This can make it difficult to associate generic SE life cycle processes with them. Ryan (Ryan, 2013) proposes a generic model (see Figure 1) in which an enterprise of some kind forms a focus for translating strategic intentions into a system definition. There is an enterprise view in which enterprise leadership sets the enterprise strategies, concepts and plans; a business management view in which business management derive business needs and constraints as well as formalize their requirements; a business operations view in which stakeholders define their needs and requirements. In the systems view an engineered system is defined, expanding to views at the lower-level of system elements if needed. Note that an engineered system may comprise a number of elements including products, people, and processes. A system architecture can be created to define logical and physical views of how these elements together enable a needed capability for the organization.

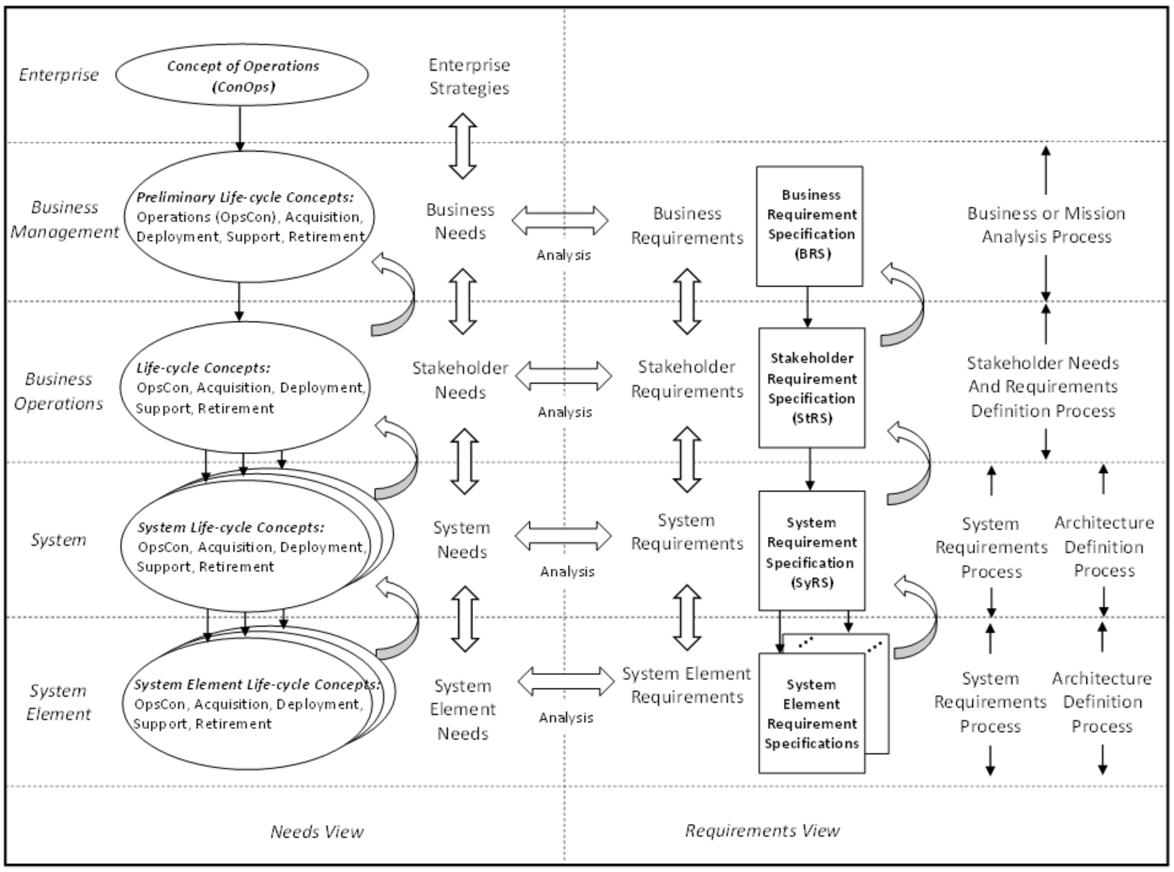

Enterprise
Concept of Operations (ConOps)
Enterprise Strategies
Business Management
Preliminary Life-cycle Concepts: Operations (OpsCon), Acquisition, Deployment, Support, Retirement
Business Needs
Analysis
Business Requirements
Business Requirement Specification (BRS)
Business or Mission Analysis Process
Business Operations
Life-cycle Concepts: OpsCon, Acquisition, Deployment, Support, Retirement
Stakeholder Needs
Stakeholder Requirements
Stakeholder Requirement Specification (StRS)
Stakeholder Needs And Requirements Definition Process
System
System Life-cycle Concepts: OpsCon, Acquisition, Deployment, Support, Retirement
System Needs
System Requirements
System Requirement Specification (SyRS)
System Requirements Process
Architecture Definition Process
System Element
System Element Life-cycle Concepts: OpsCon, Acquisition, Deployment, Support, Retirement
System Element Needs
System Element Requirements
System Element Requirement Specifications
Needs View
Requirements View

**Figure 1. Transformation of needs into requirements (Ryan, 2013)** Permission granted by M. Ryan. All other rights are reserved by the copyright owner.

In the following discussion Ryan further defines a set of general terminology to define levels of need and requirements and associates them with generic organizational roles. For a discussion of how this general description might map onto different application contexts or organisational structures see Part 4 and Part 5 respectively.

The various views in Figure 1 are referred to as layers. At the highest layer, the enterprise has a number of strategies that will guide its future. In the illustration above for example, a system has its genesis in a Concept of Operations (ConOps) or Strategic Business Plan (SBP) that communicates the leadership's intentions with regard to the operation of the organization in terms of existing systems and systems to be developed. At this layer the ConOps, or SBP, defines the enterprise in terms of 'brand' and establishes a mission statement and corresponding goals and objectives, which clearly state the reason for the enterprise and its strategy for moving forward.

The Business or Mission Analysis Process begins with the organization's mission or vision statement, goals and objectives communicated by the ConOps or SBP. Business management uses this guidance to define business needs, largely in the form of a life-cycle concepts, which capture the business management's concepts for acquisition, development, marketing, operations, deployment, support, and retirement. These concepts are then used to define specific needs for that layer.

The business needs contained in the life-cycle concepts are elaborated and formalized into business requirements, which are documented in the Business Requirements Specification (BRS) or Business Requirement Document (BRD). The process by which business needs are transformed into business requirements is called mission analysis or business analysis.

Once business management is satisfied their needs and requirements are reasonably complete, they pass them on to the business operations layer. Here, the Stakeholder Needs and Requirements (SNR) Definition Process uses the ConOps or SBP and concepts contained in the life-cycle concepts as guidance. The Requirements Engineer (RE) or

Business Analyst (BA) leads stakeholders from the business operations layer through a structured process to elicit stakeholder needs—in the form of a refined OpsCon or similar document and other life-cycle concepts (see Figure 1). The RE or BA uses a structured process to elicit specific needs, as documented in user stories, use cases, scenarios, system concepts, or operational concepts. For further discussion of the Concept of Operations and the Operational Concept Document, and their interplay, see ANSI/AIAA G-043-2012e, Guide to the Preparation of Operational Concept Documents.

Stakeholder needs are then transformed into a formal set of Stakeholder Requirements, which are documented in the Stakeholder Requirement Specification (StRS) or Stakeholder Requirement Document (StRD). That transformation is guided by a well-defined, repeatable, rigorous, and documented process of requirements analysis. This requirements analysis may involve the use of functional flow diagrams, timeline analysis, N2 Diagrams, design reference missions, modeling and simulations, movies, pictures, states and modes analysis, fault tree analysis, failure modes and effects analysis, and trade studies. In some cases these requirements analysis methods may make use of views created as part of a high level logical architecture.

At the system layer, in the System Requirements Definition Process, the requirements in the StRS are then transformed by the RE or BA into System Requirements, which are documented in the System Requirement Specification (SyRS) or System Requirement Document (SyRD). As in the previous process, the RE or BA accomplishes the transformation of needs into requirements using the same requirements analysis methods described above to define the requirements. At each layer, the resulting requirements will be documented, agreed-to, baselined, and will be put under configuration management. As above the system requirements analysis may also be linked to appropriate logical and physical architecture, either informally or under shared configuration control. Note that some organizations may prepare individual life-cycle concepts for each of a number of systems that are developed to meet the business needs.

Once a set of requirements has been documented, agreed-to, and baselined at one layer they will flow down to the next layer as shown in Figure 1. At the next layer, the requirements are a result of the transformation process of the needs at that layer as well a result of the decomposition or derivation of the requirements from the previous layer. As such, a number of SyRS or SyRD requirements may be either decomposed from (that is, made explicit by the requirements of) or derived from (that is, implied by the requirements of) the StRS or StRD. The same is true at the subsystem or system element layer, where a number of the subsystem or system element requirements may be either decomposed or derived from the SyRS or SyRD. In all cases, for each layer shown in Figure 1, the set of requirements can be traced back to the requirements at the previous layer from which they were either decomposed or derived. This process continues for the next layer of system elements.

How requirements are expressed differs through these layers, and therefore so do the rules for expressing them. As requirements are developed – and solutions designed – down through the layers of abstraction, we expect statements of requirement to become more and more specific. At the highest level, the ideal requirement is not specific to a particular solution, and permits a range of possible solutions. At the lowest level, statements of requirement will be entirely specific to the selected solution. It is important to note that the form of requirements at one layer may not be appropriate for another layer. For example, at the business management layer, there may be a requirement that all products are “safe”. While this is a poor system requirement, it is appropriate for the Business Management layer. At the next layer, business operations, there will be less ambiguous and more detailed requirements that define safe. These requirements apply across all product lines. At the system layer, more specific safety requirements will be developed for that specific system. These requirements will then be allocated to the system elements at the next lower layer.

## References

### Works Cited

ANSI/AIAA G-043-2012e, Guide to the Preparation of Operational Concept Documents.

Dick, J. and J. Chard, "The Systems Engineering Sandwich: Combining Requirements, Models and Design", INCOSE International Symposium IS2004, July 2004.

Hull, E., K. Jackson, J. Dick, Requirements Engineering, Springer, 2010.

Ryan, M.J., "An Improved Taxonomy for Major Needs and Requirements Artefacts", INCOSE International Symposium IS2013, June 2013.

### Primary References

Hull, E., K. Jackson, J. Dick, Requirements Engineering, Springer, 2010.

Ryan, M.J., "An Improved Taxonomy for Major Needs and Requirements Artefacts", INCOSE International Symposium IS2013, June 2013.

### Additional References

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Knowledge Area: Life Cycle Models

## Life Cycle Models

***Lead Authors:*** *Kevin Forsberg, Rick Adcock*, ***Contributing Author:*** *Alan Faisandier*

The life cycle model is one of the key concepts of systems engineering (SE). A life cycle for a system generally consists of a series of stages regulated by a set of management decisions which confirm that the system is mature enough to leave one stage and enter another.

### Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- System Life Cycle Process Drivers and Choices
- System Life Cycle Process Models: Vee
- System Life Cycle Process Models: Iterative
- Integration of Process and Product Models
- Lean Engineering

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

### Type of Value Added Products/Services

The Generic Life Cycle Model shows just the single-step approach for proceeding through the stages of a system’s life cycle. Adding value (as a product, a service, or both), is a shared purpose among all enterprises, whether public or private, for profit or non-profit. Value is produced by providing and integrating the elements of a system into a product or service according to the system description and transitioning it into productive use. These value considerations will lead to various forms of the generic life cycle management approach in Figure 1. Some examples are as follows (Lawson 2010):

- A manufacturing enterprise produces nuts, bolts, and lock washer products and then sells their products as value added elements to be used by other enterprises; in turn, these enterprises integrate these products into their more encompassing value-added system, such as an aircraft or an automobile. Their requirements will generally be pre-specified by the customer or by industry standards.
- A wholesaling or retailing enterprise offers products to their customers. Its customers (individuals or enterprises) acquire the products and use them as elements in their systems. The enterprise support system will likely evolve opportunistically, as new infrastructure capabilities or demand patterns emerge.
- A commercial service enterprise such as a bank sells a variety of *products* as services to their customers. This includes current accounts, savings accounts, loans, and investment management. These services add value and are incorporated into customer systems of individuals or enterprises. The service enterprise’s support system will also likely evolve opportunistically, as new infrastructure capabilities or demand patterns emerge.
- A governmental service enterprise provides citizens with services that vary widely, but may include services such as health care, highways and roads, pensions, law enforcement, or defense. Where appropriate, these services become infrastructure elements utilized in larger encompassing systems of interest to individuals and/or

enterprises. Major initiatives, such as a next-generation air traffic control system or a metropolitan-area crisis management system (hurricane, typhoon, earthquake, tsunami, flood, fire), will be sufficiently complex enough to follow an evolutionary development and fielding approach. At the elemental level, there will likely be pre-specified single-pass life cycles.

- For aircraft and automotive systems, there would likely be a pre-specified multiple-pass life cycle to capitalize on early capabilities in the first pass, but architected to add further value-adding capabilities in later passes.
- A diversified software development enterprise provides software products that meet stakeholder requirements (needs), thus providing services to product users. It will need to be developed to have capabilities that can be tailored to be utilized in different customers' life-cycle approaches and also with product-line capabilities that can be quickly and easily applied to similar customer system developments. Its business model may also include providing the customer with system life-cycle support and evolution capabilities.

Within these examples, there are systems that remain stable over reasonably long periods of time and those that change rapidly. The diversity represented by these examples and their processes illustrate why there is no one-size-fits-all process that can be used to define a specific systems life cycle. Management and leadership approaches must consider the type of systems involved, their longevity, and the need for rapid adaptation to unforeseen changes, whether in competition, technology, leadership, or mission priorities. In turn, the management and leadership approaches impact the type and number of life cycle models that are deployed as well as the processes that will be used within any particular life cycle.

There are several incremental and evolutionary approaches for sequencing the life cycle stages to deal with some of the issues raised above. The Life Cycle Models knowledge area summarizes a number of incremental and evolutionary life cycle models, including their main strengths and weaknesses and also discusses criteria for choosing the best-fit approach.

## Categories of Life Cycle Model

The Generic System Life Cycle Model in Figure 1 does not explicitly fit all situations. A simple, precedential, follow-on system may need only one phase in the definition stage, while a complex system may need more than two. With build-upon systems (vs. throwaway) prototypes, a good deal of development may occur during the definition stage. System integration, verification, and validation may follow implementation or acquisition of the system elements. With software, particularly test-first and daily builds, integration, verification, and validation are interwoven with element implementation. Additionally, with the upcoming *Third Industrial Revolution* of three-dimensional printing and digital manufacturing (Whadcock 2012), not only initial development but also initial production may be done during the concept stage.

Software is a flexible and malleable medium which facilitates iterative analysis, design, construction, verification, and validation to a greater degree than is usually possible for the purely physical components of a system. Each repetition of an iterative development model adds material (code) to the growing software base, in which the expanded code base is tested, reworked as necessary, and demonstrated to satisfy the requirements for the baseline.

Software can be electronically bought, sold, delivered, and upgraded anywhere in the world within reach of digital communication, making its logistics significantly different and more cost-effective than hardware. It does not wear out and its fixes change its content and behavior, making regression testing more complex than with hardware fixes. Its discrete nature dictates that its testing cannot count on analytic continuity as with hardware. Adding 1 to 32767 in a 15-bit register does not produce 32768, but 0 instead, as experienced in serious situations, such as with the use of the Patriot Missile.

There are a large number of potential life cycle process models. They fall into three major categories:

1. primarily pre-specified and sequential processes (e.g. the single-step waterfall model)

2. primarily evolutionary and concurrent processes (e.g. lean development, the rational unified process, and various forms of the vee and spiral models)
3. primarily interpersonal and emergent processes (e.g. agile development, scrum, extreme programming (XP), the dynamic system development method, and innovation-based processes)

The emergence of integrated, interactive hardware-software systems made pre-specified processes potentially harmful, as the most effective human-system interfaces tended to emerge with its use, leading to further process variations, such as soft SE (Warfield 1976, Checkland 1981) and human-system integration processes (Booher 2003, Pew and Mavor 2007). Until recently, process standards and maturity models have tried to cover every eventuality. They have included extensive processes for acquisition management, source selection, reviews and audits, quality assurance, configuration management, and document management, which in many instances would become overly bureaucratic and inefficient. This led to the introduction of more lean (Ohno 1988; Womack et al. 1990; Oppenheim 2011) and agile (Beck 1999; Anderson 2010) approaches to concurrent hardware-software-human factors approaches such as the concurrent vee models (Forsberg 1991; Forsberg 2005) and Incremental Commitment Spiral Model (Pew and Mavor 2007; Boehm and Lane 2007).

In the next article on System Life Cycle Process Drivers and Choices, these variations on the theme of life cycle models will be identified and presented.

## Systems Engineering Responsibility

Regardless of the life cycle models deployed, the role of the systems engineer encompasses the entire life cycle of the system-of-interest. Systems engineers orchestrate the development and evolution of a solution, from defining requirements through operation and ultimately until system retirement. They ensure that domain experts are properly involved, all advantageous opportunities are pursued, and all significant risks are identified and, when possible, mitigated. The systems engineer works closely with the project manager in tailoring the generic life cycle, including key decision gates, to meet the needs of their specific project.

Systems engineering tasks are usually concentrated at the beginning of the life cycle; however, both commercial and government organizations recognize the need for SE throughout the system's life cycle. Often this ongoing effort is to modify or change a system, product or service after it enters production or is placed in operation. Consequently, SE is an important part of all life cycle stages. During the production, support, and utilization (PSU) stages, for example, SE executes performance analysis, interface monitoring, failure analysis, logistics analysis, tracking, and analysis of proposed changes. All these activities are essential to ongoing support of the system.

All project managers must ensure that the business aspect (cost, schedule, and value) and the technical aspect of the project cycle remain synchronized. Often, the technical aspect drives the project. It is the systems engineers' responsibility to ensure that the technical solutions that are being considered are consistent with the cost and schedule objectives. This can require working with the users and customers to revise objectives to fit within the business bounds. These issues also drive the need for decision gates to be appropriately spaced throughout the project cycle. Although the nature of these decision gates will vary by the major categories above, each will involve in-process validation between the developers and the end users. In-process validation asks the question: *"Will what we are planning or creating satisfy the stakeholders' needs?"* In-process validation begins at the initialization of the project during user needs discovery and continues through daily activities, formal decision gate reviews, final product or solution delivery, operations, and ultimately to system closeout and disposal.

## References

### Works Cited

Anderson, D. 2010. *Kanban.* Sequim, WA: Blue Hole Press.

Beck, K. 1999. *Extreme Programming Explained.* Boston, MA: Addison Wesley.

Boehm, B. and J. Lane. 2007. “Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering.” *CrossTalk*. October 2007: 4-9.

Booher, H. (ed.) 2003. *Handbook of Human Systems Integration*. Hoboken, NJ, USA: Wiley.

Checkland, P. 1999. *Systems Thinking, Systems Practice,* 2nd ed. Hoboken, NJ, USA: Wiley.

Cusumano, M., and D. Yoffie. 1998. *Competing on Internet Time*, New York, NY, USA: The Free Press.

Forsberg, K. and H. Mooz. 1991. "The Relationship of System Engineering to the Project Cycle," *Proceedings of INCOSE*, October 1991.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management*, 3rd ed. Hoboken, NJ: J. Wiley & Sons.

ISO/IEC/IEEE. 2015.*Systems and software engineering - system life cycle processes*.Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC 15288:2015.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

Ohno, T. 1988. *Toyota Production System*. New York, NY: Productivity Press.

Oppenheim, B. 2011. *Lean for Systems Engineering.* Hoboken, NJ: Wiley.

Pew, R. and A. Mavor (eds.). 2007. *Human-System Integration in The System Development Process: A New Look.* Washington, DC, USA: The National Academies Press.

Warfield, J. 1976. *Systems Engineering*. Washington, DC, USA: US Department of Commerce (DoC).

Whadcock, I. 2012. “A third industrial revolution.” *The Economist.* April 21, 2012.

Womack, J.P., D.T. Jones, and D. Roos 1990. *The Machine That Changed the World: The Story of Lean Production.* New York, NY, USA: Rawson Associates.

### Primary References

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Forsberg, K., H. Mooz, H. Cotterman. 2005. *Visualizing Project Management*, 3rd Ed. Hoboken, NJ: J. Wiley & Sons.

INCOSE. 2012. *Systems Engineering Handbook*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-2003-002-03.2.2.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

Pew, R. and A. Mavor (Eds.). 2007. *Human-System Integration in The System Development Process: A New Look.* Washington, DC, USA: The National Academies Press.

## Additional References

Chrissis, M., M. Konrad, and S. Shrum. 2003. *CMMI: Guidelines for Process Integration and Product Improvement.* New York, NY, USA: Addison Wesley.

Larman, C. and B. Vodde. 2009. *Scaling Lean and Agile Development.* New York, NY, USA: Addison Wesley.

The following three books are not referenced in the SEBoK text, nor are they systems engineering "texts"; however, they contain important systems engineering lessons, and readers of this SEBOK are encouraged to read them.

Kinder, G. 1998. *Ship of Gold in the Deep Blue Sea.* New York, NY, USA: Grove Press.

This is an excellent book that follows an idea from inception to its ultimately successful implementation. Although systems engineering is not discussed, it is clearly illustrated in the whole process from early project definition to alternate concept development to phased exploration and “thought experiments” to addressing challenges along the way. It also shows the problem of not anticipating critical problems outside the usual project and engineering scope. It took about five years to locate and recover the 24 tons of gold bars and coins from the sunken ship in the 2,500-meter-deep ocean, but it took ten years to win the legal battle with the lawyers representing insurance companies who claimed ownership based on 130-year-old policies they issued to the gold owners in 1857.

McCullough, D. 1977. *The Path Between the Seas: The Creation of the Panama Canal (1870 – 1914).* New York, NY, USA: Simon & Schuster.

Although “systems engineering” is not mentioned, this book highlights many systems engineering issues and illustrates the need for SE as a discipline. The book also illustrates the danger of applying a previously successful concept (the sea level canal used in Suez a decade earlier) in a similar but different situation. Ferdinand de Lesseps led both the Suez and Panama projects. It illustrates the danger of lacking a fact-based project cycle and meaningful decision gates throughout the project cycle. It also highlights the danger of providing project status without visibility. After five years into the ten-year project investors were told the project was more than 50 percent complete when in fact only 10 percent of the work was complete. The second round of development under Stevens in 1904 focused on “moving dirt” rather than digging a canal, a systems engineering concept key to the completion of the canal. The Path Between the Seas won the National Book Award for history (1978), the Francis Parkman Prize (1978), the Samuel Eliot Morison Award (1978), and the Cornelius Ryan Award (1977).

Shackleton, Sir E.H. 2008. (Originally published in by William Heinemann, London, 1919). *South: The Last Antarctic Expedition of Shackleton and the Endurance.* Guilford, CT, USA: Lyons Press.

This is the amazing story of the last Antarctic expedition of Shackleton and the *Endurance* in 1914 to 1917. The systems engineering lesson is the continuous, daily risk assessment by the captain, expedition leader, and crew as they lay trapped in the arctic ice for 18 months. All 28 crew members survived.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# System Life Cycle Process Drivers and Choices

***Lead Authors:*** *Kevin Forsberg, Rick Adcock*

As discussed in the Generic Life Cycle Model article, there are many organizational factors that can impact which life cycle processes are appropriate for a specific system. Additionally, technical factors will also influence the types of life cycle models appropriate for a given system. For example, system requirements can either be predetermined or they can be changing, depending on the scope and nature of the development for a system. These considerations lead to different life cycle model selections. This article discusses different technical factors which can be considered when selecting a life cycle process model and provides examples, guidance and tools from the literature to support life cycle model selection. The life cycle model selected can impact all other aspects of system design and development. (See the knowledge areas in Part 3 for a description of how the life cycle can impact systems engineering (SE) processes.)

## Fixed-Requirements and Evolutionary Development Processes

Aside from the traditional, pre-specified, sequential, single-step development process, there are several models of incremental and evolutionary development; however, there is no one-size-fits-all approach that is best for all situations. For rapid-fielding situations, an easiest-first, prototyping approach may be most appropriate. For enduring systems, an easiest-first approach may produce an unscalable system, in which the architecture is incapable of achieving high levels of performance, safety, or security. In general, system evolution now requires much higher sustained levels of SE effort, earlier and continuous integration and testing, proactive approaches to address sources of system change, greater levels of concurrent engineering, and achievement reviews based on evidence of feasibility versus plans and system descriptions.

Incremental and evolutionary development methods have been in use since the 1960s (and perhaps earlier). They allow a project to provide an initial capability followed by successive deliveries to reach the desired system-of-interest (SoI). This practice is particularly valuable in cases in which

- rapid exploration and implementation of part of the system is desired;
- requirements are unclear from the beginning, or are rapidly changing;
- funding is constrained;
- the customer wishes to hold the SoI open to the possibility of inserting new technology when it becomes mature; and
- experimentation is required to develop successive versions.

In iterative development, each cycle of the iteration subsumes the system elements of the previous iteration and adds new capabilities to the evolving product to create an expanded version of the software. Iterative development processes can provide a number of advantages, including

- continuous integration, verification, and validation of the evolving product;
- frequent demonstrations of progress;
- early detection of defects;
- early warning of process problems; and
- systematic incorporation of the inevitable rework that occurs in software development.

## Primary Models of Incremental and Evolutionary Development

The primary models of incremental and evolutionary development focus on different competitive and technical challenges. The time phasing of each model is shown in Figure 1 below in terms of the increment (1, 2, 3, …) content with respect to the definition (Df), development (Dv), and production, support, and utilization (PSU) stages in Figure 1 (A Generic System Life Cycle Model) from the Life Cycle Models article.

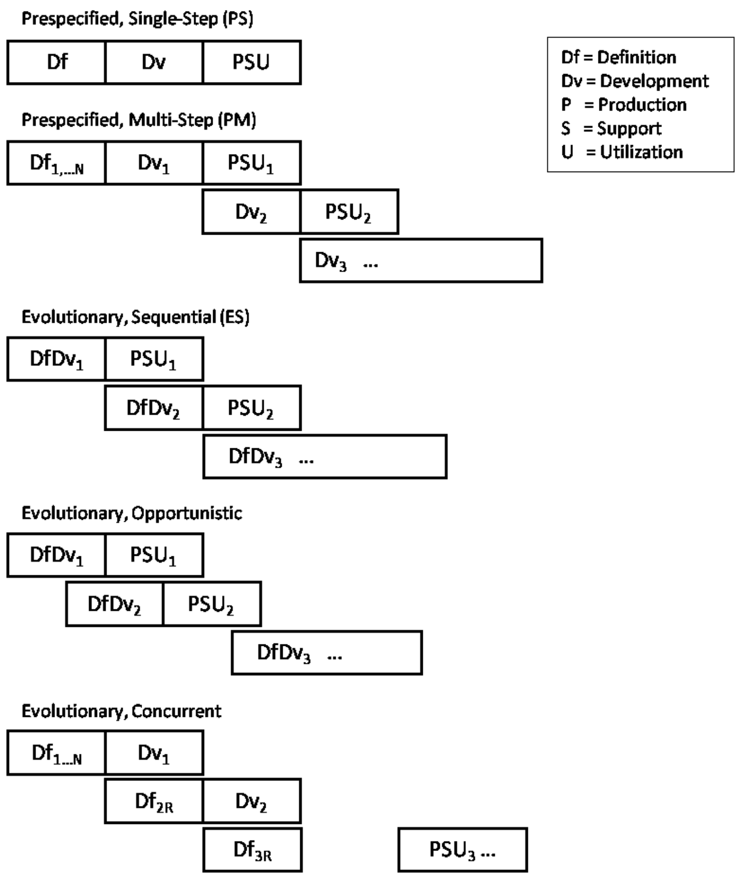


**Figure 1. Primary Models of Incremental and Evolutionary Development.**
(SEBoK Original)

The Figure 1 notations (Df1..N and Dv1..N) indicate that their initial stages produce specifications not just for the first increment, but for the full set of increments. These are assumed to remain stable for the pre-specified sequential model but are expected to involve changes for the evolutionary concurrent model. The latter's notation ( Dv1 and Df2R) in the same time frame, PSU1, Dv2 and Df3R in the same time frame, etc.) indicates that the plans and specifications for the next increment are being re-baselined by a systems engineering team concurrently with the development of the current increment and the PSU of the previous increment. This offloads the work of handling the change traffic from the development team and significantly improves its chances of finishing the current increment on budget and schedule.

In order to select an appropriate life cycle model, it is important to first gain an understanding of the main archetypes and where they are best used. Table 1 summarizes each of the primary models of single-step, incremental and evolutionary development in terms of examples, strengths, and weaknesses, followed by explanatory notes.

### Table 1. Primary Models of Incremental and Evolutionary Development (SEBoK Original).

| Model | Examples | Pros | Cons |
|---|---|---|---|
| **Pre-specified Single-step** | Simple manufactured products: Nuts, bolts, simple sensors | Efficient, easy to verify | Difficulties with rapid change, emerging requirements (complex sensors, human-intensive systems) |
| **Pre-specified Multi-step** | Vehicle platform plus value-adding pre-planned product improvements (PPPIs) | Early initial capability, scalability when stable | Emergent requirements or rapid change, architecture breakers |
| **Evolutionary Sequential** | Small: Agile<br>Larger: Rapid fielding | Adaptability to change, smaller human-intensive systems | Easiest-first, late, costly fixes, systems engineering time gaps, slow for large systems |
| **Evolutionary Opportunistic** | Stable development, Maturing technology | Mature technology upgrades | Emergent requirements or rapid change, SysE time gaps |
| **Evolutionary Concurrent** | Rapid, emergent development, systems of systems | Emergent requirements or rapid change, stable development increments, SysE continuity | Overkill on small or highly stable systems |

The *Pre-specified Single-step* and *Pre-specified Multi-step* models from Table 1 are not evolutionary. Pre-specified multi-step models split the development in order to field an early initial operational capability, followed by several pre-planned product improvements (P3Is). An alternate version splits up the work but does not field the intermediate increments. When requirements are well understood and stable, the pre-specified models enable a strong, predictable process. When requirements are emergent and/or rapidly changing, they often require expensive rework if they lead to undoing architectural commitments.

The *Evolutionary Sequential* model involves an approach in which the initial operational capability for the system is rapidly developed and is upgraded based on operational experience. Pure agile software development fits this model. If something does not turn out as expected and needs to be changed, it will be fixed in thirty days at the time of its next release. Rapid fielding also fits this model for larger or hardware-software systems. Its major strength is to enable quick-response capabilities in the field. For pure agile, the model can fall prey to an easiest-first set of architectural commitments which break when, for example, system developers try to scale up the workload by a factor of ten or to add security as a new feature in a later increment. For rapid fielding, using this model may prove expensive when the quick mash-ups require extensive rework to fix incompatibilities or to accommodate off-nominal usage scenarios, but the rapid results may be worth it.

The *Evolutionary Opportunistic* model can be adopted in cases that involve deferring the next increment until: a sufficiently attractive opportunity presents itself, the desired new technology is mature enough to be added, or until other enablers such as scarce components or key personnel become available. It is also appropriate for synchronizing upgrades of multiple commercial-off-the-shelf (COTS) products. It may be expensive to keep the SE and development teams together while waiting for the enablers, but again, it may be worth it.

The *Evolutionary Concurrent* model involves a team of systems engineers concurrently handling the change traffic and re-baselining the plans and specifications for the next increment, in order to keep the current increment development stabilized. An example and discussion are provided in Table 2, below.

## Incremental and Evolutionary Development Decision Table

The Table 2 provides some criteria for deciding which of the processes associated with the primary classes of incremental and evolutionary development models to use.

**Table 2. Incremental and Evolutionary Development Decision Table. (Boehm and Lane 2010). Reprinted with permission of the Systems Engineering Research Center. All other rights are reserved by the copyright owner.**

| Model | Stable, pre-specifiable requirements? | OK to wait for full system to be developed? | Need to wait for next-increment priorities? | Need to wait for next-increment enablers*? |
|---|---|---|---|---|
| **Pre-specified Single-step** | Yes | Yes | | |
| **Pre-specified Multi-step** | Yes | No | | |
| **Evolutionary Sequential** | No | No | Yes | |
| **Evolutionary Opportunistic** | No | No | No | Yes |
| **Evolutionary Concurrent** | No | No | No | No |

***Example enablers: Technology maturity; External-system capabilities; Needed resources; New opportunities***

The *Pre-specified Single-step* process exemplified by the traditional waterfall or sequential Vee model is appropriate if the product's requirements are pre-specifiable and have a low probability of significant change and if there is no value or chance to deliver a partial product capability. A good example of this would be the hardware for an earth resources monitoring satellite that would be infeasible to modify after it goes into orbit.

The *Pre-specified Multi-step* process splits up the development in order to field an early initial operational capability and several P3I's. It is best if the product's full capabilities can be specified in advance and are at a low probability of significant change. This is useful in cases when waiting for the full system to be developed incurs a loss of important and deliverable incremental mission capabilities. A good example of this would be a well-understood and well-prioritized sequence of software upgrades for the on-board earth resources monitoring satellite.

The *Evolutionary Sequential* process develops an initial operational capability and upgrades it based on operational experience, as exemplified by agile methods. It is most needed in cases when there is a need to obtain operational feedback on an initial capability before defining and developing the next increment's content. A good example of this would be the software upgrades suggested by experiences with the satellite's payload, such as what kind of multi-spectral data collection and analysis capabilities are best for what kind of agriculture under what weather conditions.

The *Evolutionary Opportunistic* process defers the next increment until its new capabilities are available and mature enough to be added. It is best used when the increment does not need to wait for operational feedback, but it may need to wait for next-increment enablers such as technology maturity, external system capabilities, needed resources, or new value-adding opportunities. A good example of this would be the need to wait for agent-based satellite anomaly trend analysis and mission-adaptation software to become predictably stable before incorporating it into a scheduled increment.

The *Evolutionary Concurrent* process, as realized in the incremental commitment spiral model (Pew and Mavor 2007; Boehm and Lane 2007) and shown in Figure 2, has a continuing team of systems engineers handling the change traffic and re-baselining the plans and specifications for the next increment, while also keeping a development team stabilized for on-time, high-assurance delivery of the current increment and employing a concurrent verification and validation (V&V) team to perform continuous defect detection to enable even higher assurance levels. A good example of this would be the satellite's ground-based mission control and data handling software's next-increment re-baselining to adapt to new COTS releases and continuing user requests for data processing upgrades.

The satellite example illustrates the various ways in which the complex systems of the future, different parts of the system, and its software may evolve in a number of ways, once again affirming that there is no one-size-fits-all process for software evolution. However, Table 2 can be quite helpful in determining which processes are the best fits for evolving each part of the system. Additionally, the three-team model in Figure 2 provides a way for projects to develop the challenging software-intensive systems of the future that will need both adaptability to rapid change and high levels of assurance.

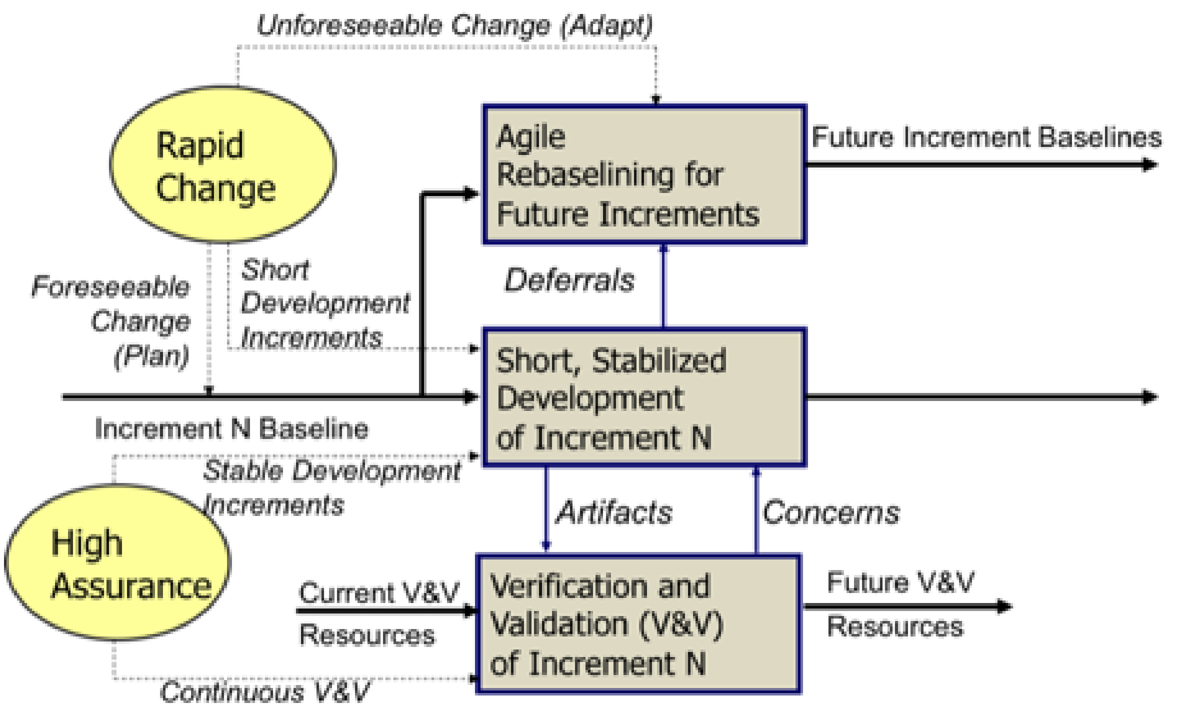

Unforeseeable Change (Adapt)
Rapid Change
Agile Rebaselining for Future Increments
Future Increment Baselines
Foreseeable Change (Plan)
Short Development Increments
Deferrals
Short, Stabilized Development of Increment N
Increment N Baseline
Stable Development Increments
Artifacts
Concerns
High Assurance
Current V&V Resources
Verification and Validation (V&V) of Increment N
Future V&V Resources
Continuous V&V

**Figure 2. Evolutionary-Concurrent Rapid Change Handling and High Assurance (Pew and Mavor 2007, Figure 2-6).** Reprinted with permission from the National Academy of Sciences, Courtesy of National Academies Press, Washington, D.C. All other rights are reserved by the copyright owner.

## References

### Works Cited

Boehm, B. 2006. "Some Future Trends and Implications for Systems and Software Engineering Processes." *Systems Engineering.* 9(1): 1-19.

Boehm, B. and J. Lane. 2007. "Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering." *CrossTalk.* October 2007: 4-9.

Boehm, B. and J. Lane. 2010. *DoD Systems Engineering and Management Implications for Evolutionary Acquisition of Major Defense Systems.* SERC RT-5 report, March 2010. USC-CSSE-2010-500.

Cusumano, M. and D. Yoffee. 1998. *Competing on Internet Time: Lessons from Netscape and Its Battle with Microsoft.* New York, NY, USA: Free Press.

Pew, R. and A. Mavor (eds.). 2007. *Human-System Integration in the System Development Process: A New Look.* Washington DC, USA: The National Academies Press.

### Primary References

Pew, R., and A. Mavor (eds.). 2007. *Human-System Integration in the System Development Process: A New Look.* Washington, DC, USA: The National Academies Press.

### Additional References

None.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# System Life Cycle Process Models: Vee

***Lead Authors:*** *Dick Fairley, Kevin Forsberg*, ***Contributing Author:*** *Ray Madachy*

There are a large number of life cycle process models. As discussed in the System Life Cycle Process Drivers and Choices article, these models fall into three major categories: (1) primarily pre-specified and sequential processes; (2) primarily evolutionary and concurrent processes (e.g., the rational unified process and various forms of the Vee and spiral models); and (3) primarily interpersonal and unconstrained processes (e.g., agile development, Scrum, extreme programming (XP), the dynamic system development method, and innovation-based processes).

This article specifically focuses on the Vee Model as the primary example of pre-specified and sequential processes. In this discussion, it is important to note that the Vee model, and variations of the Vee model, all address the same basic set of systems engineering (SE) activities. The key difference between these models is the way in which they group and represent the aforementioned SE activities.

General implications of using the Vee model for system design and development are discussed below; for a more specific understanding of how this life cycle model impacts systems engineering activities, please see the other knowledge areas (KAs) in Part 3.

## A Primarily Pre-specified and Sequential Process Model: The Vee Model

The sequential version of the Vee Model is shown in Figure 1. Its core involves a sequential progression of plans, specifications, and products that are baselined and put under configuration management. The vertical, two-headed arrow enables projects to perform concurrent opportunity and risk analyses, as well as continuous in-process validation. The Vee Model encompasses the first three life cycle stages listed in the "Generic Life Cycle Stages" table of the INCOSE *Systems Engineering Handbook*: exploratory research, concept, and development (INCOSE 2012).

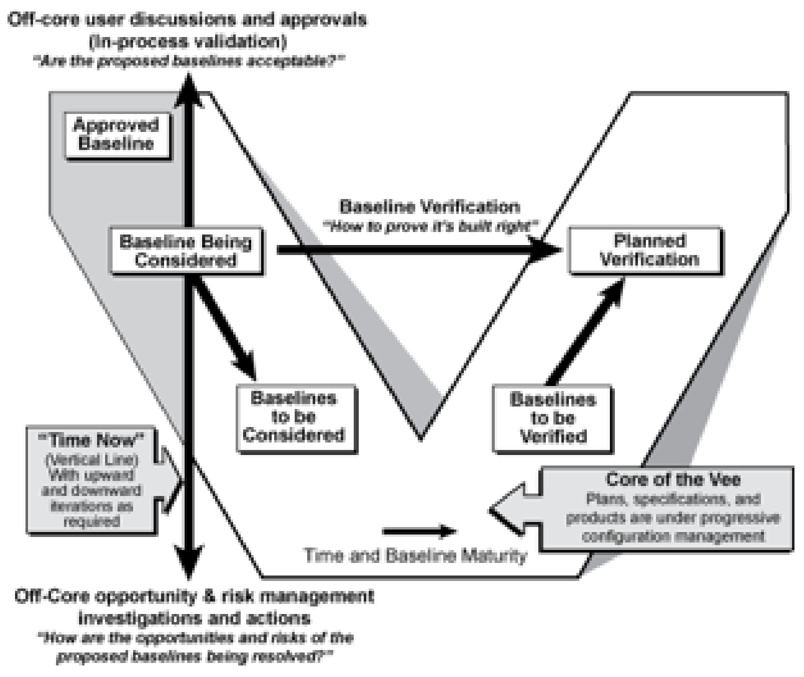

Off-core user discussions and approvals
(In-process validation)
"Are the proposed baselines acceptable?"
Approved Baseline
Baseline Being Considered
Baseline Verification
"How to prove it's built right"
Planned Verification
Baselines to be Considered
Baselines to be Verified
"Time Now" (Vertical Line) With upward and downward iterations as required
Core of the Vee
Plans, specifications, and products are under progressive configuration management
Time and Baseline Maturity
Off-Core opportunity & risk management investigations and actions
"How are the opportunities and risks of the proposed baselines being resolved?"

**Figure 1. Left Side of the Sequential Vee Model (Forsberg, Mooz, and Cotterman 2005).** Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

The Vee Model endorses the INCOSE *Systems Engineering Handbook* (INCOSE 2012) definition of life cycle stages and their purposes or activities, as shown in Figure 2 below.

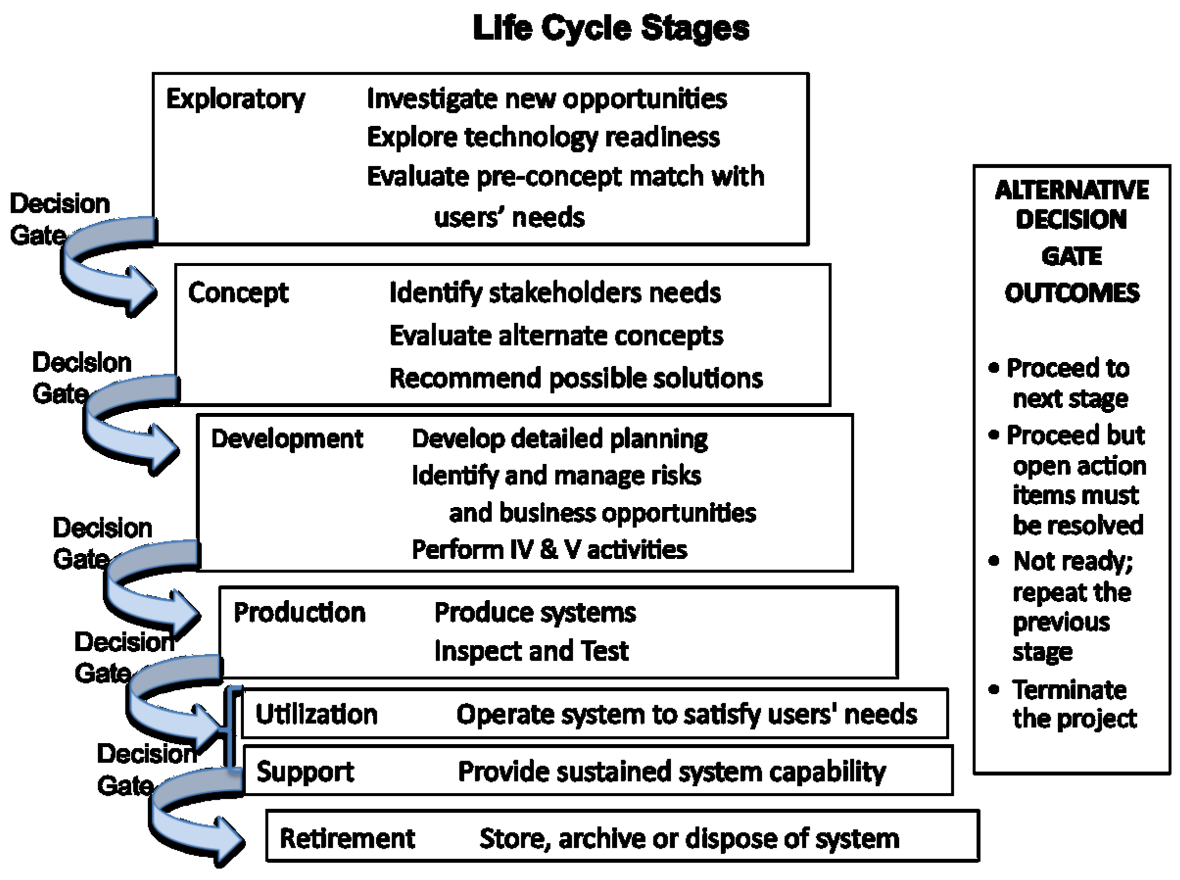

Life Cycle Stages
Exploratory
Investigate new opportunities
Explore technology readiness
Evaluate pre-concept match with users' needs
Decision Gate
Concept
Identify stakeholders needs
Evaluate alternate concepts
Recommend possible solutions
Decision Gate
Development
Develop detailed planning
Identify and manage risks and business opportunities
Perform IV & V activities
Decision Gate
Production
Produce systems
Inspect and Test
Decision Gate
Utilization
Operate system to satisfy users' needs
Support
Provide sustained system capability
Decision Gate
Retirement
Store, archive or dispose of system
ALTERNATIVE DECISION GATE OUTCOMES
• Proceed to next stage
• Proceed but open action items must be resolved
• Not ready; repeat the previous stage
• Terminate the project

**Figure 2. An Example of Stages, Their Purposes and Major Decision Gates.** (SEBoK Original)

The INCOSE *Systems Engineering Handbook* 3.2.2 contains a more detailed version of the Vee diagram (2012, Figures 3-4, p. 27) which incorporates life cycle activities into the more generic Vee model. A similar diagram, developed at the U.S. Defense Acquisition University (DAU), can be seen in Figure 3 below.

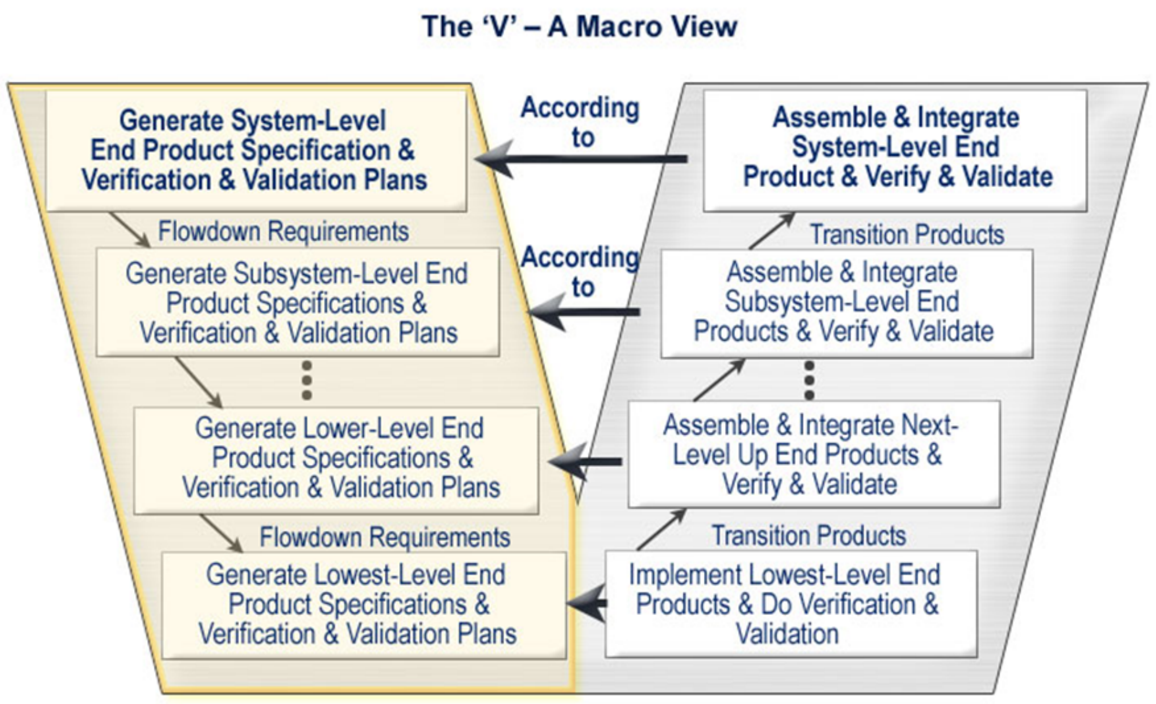

The 'V' – A Macro View
Generate System-Level End Product Specification & Verification & Validation Plans
Flowdown Requirements
Generate Subsystem-Level End Product Specifications & Verification & Validation Plans
Generate Lower-Level End Product Specifications & Verification & Validation Plans
Flowdown Requirements
Generate Lowest-Level End Product Specifications & Verification & Validation Plans
According to
According to
Assemble & Integrate System-Level End Product & Verify & Validate
Transition Products
Assemble & Integrate Subsystem-Level End Products & Verify & Validate
Assemble & Integrate Next-Level Up End Products & Verify & Validate
Transition Products
Implement Lowest-Level End Products & Do Verification & Validation

**Figure 3. The Vee Activity Diagram (Prosnik 2010).** Released by the Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

## Application of the Vee Model

Lawson (Lawson 2010) elaborates on the activities in each life cycle stage and notes that it is useful to consider the structure of a generic life cycle stage model for any type of system-of-interest (SoI) as portrayed in Figure 4. This **(T)** model indicates that one or more definition stages precede a production stage(s) where the implementation (acquisition, provisioning, or development) of two or more system elements has been accomplished.

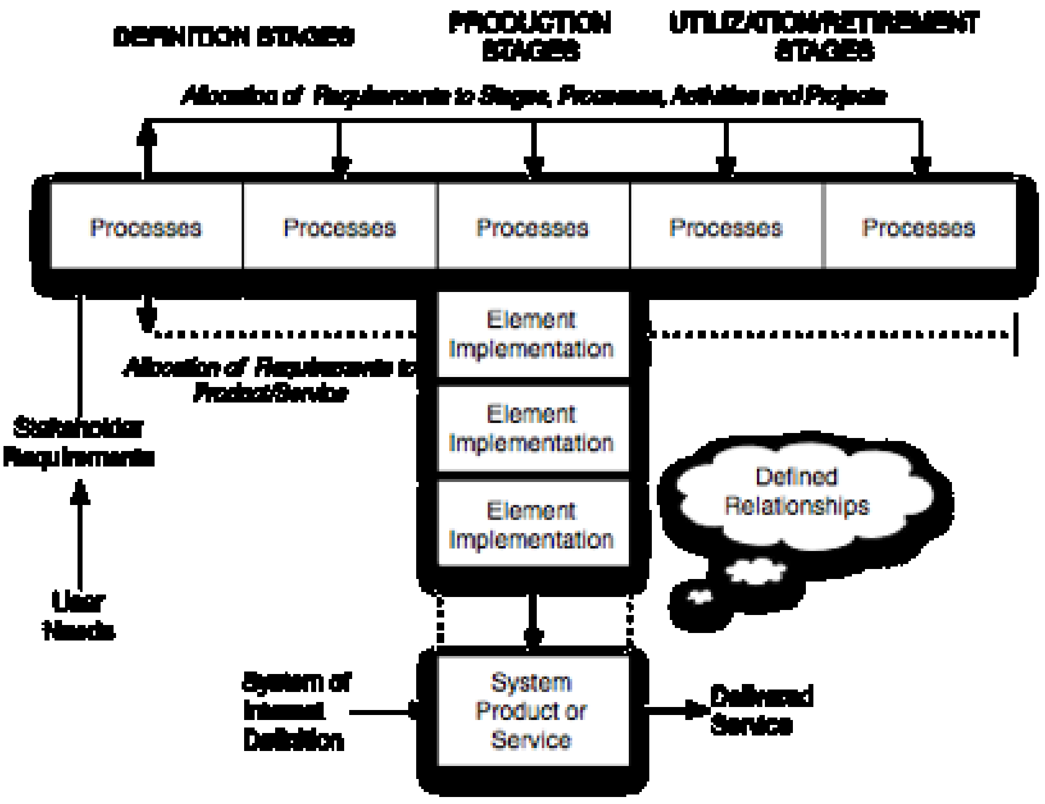


**Figure 4. Generic (T) Stage Structure of System Life Cycle Models (Lawson 2010).** Reprinted with permission of Harold Lawson. All other rights are reserved by the copyright owner.

Figure 5 shows the generic life cycle stages for a variety of stakeholders, from a standards organization (ISO/IEC) to commercial and government organizations. Although these stages differ in detail, they all have a similar sequential format that emphasizes the core activities as noted in Figure 2 (definition, production, and utilization/retirement).

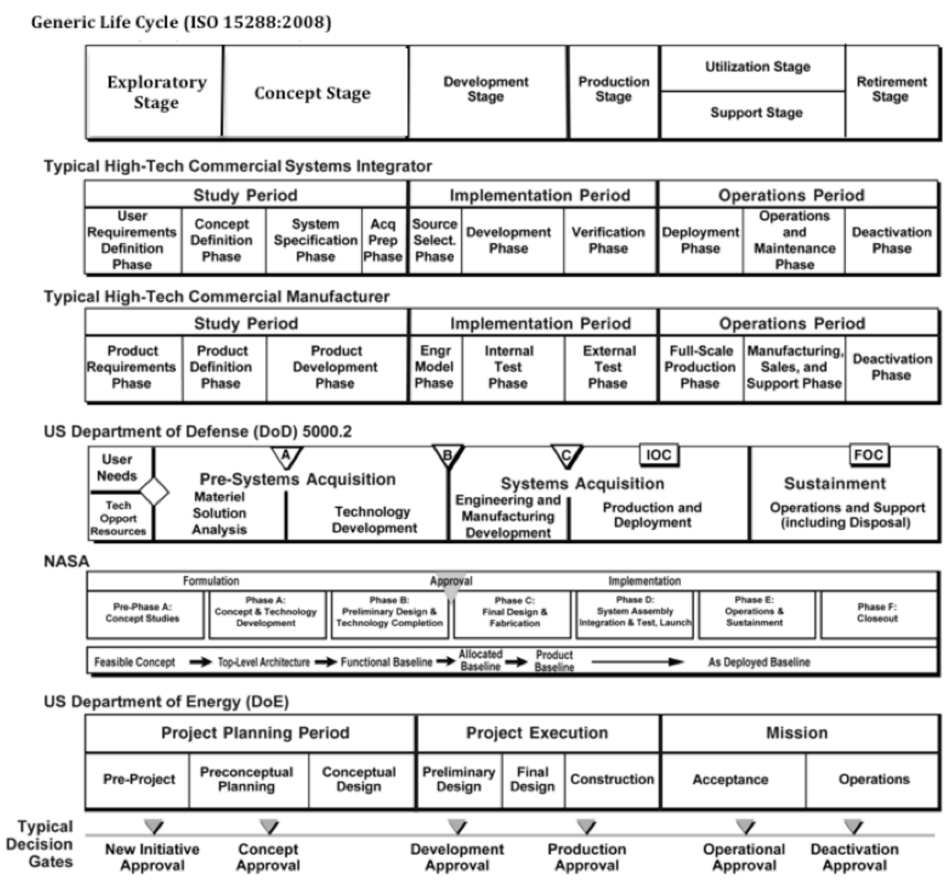


**Figure 5. Comparisons of Life Cycle Models (Forsberg, Mooz, and Cotterman 2005).** Reprinted with permission of John Wiley & Sons. All other rights are reserved by the copyright owner.

It is important to note that many of the activities throughout the life cycle are iterated. This is an example of recursion (glossary) as discussed in the Part 3 Introduction.

## Fundamentals of Life Cycle Stages and Program Management Phase

For this discussion, it is important to note that:

- The term **stage** refers to the different states of a system during its life cycle; some stages may overlap in time, such as the utilization stage and the support stage. The term "stage" is used in ISO/IEC/IEEE 15288.
- The term **phase** refers to the different steps of the program that support and manage the life of the system; the phases usually do not overlap. The term "phase" is used in many well-established models as an equivalent to the term "stage."

Program management employs phases, milestones, and decision gates which are used to assess the evolution of a system through its various stages. The stages contain the activities performed to achieve goals and serve to control and manage the sequence of stages and the transitions between each stage. For each project, it is essential to define and publish the terms and related definitions used on respective projects to minimize confusion.

A typical program is composed of the following phases:

- The **pre-study phase**, which identifies potential opportunities to address user needs with new solutions that make business sense.
- The **feasibility phase** consists of studying the feasibility of alternative concepts to reach a second decision gate before initiating the execution stage. During the feasibility phase, stakeholders' requirements and system requirements are identified, viable solutions are identified and studied, and virtual prototypes (glossary) can be

implemented. During this phase, the decision to move forward is based on:

  - whether a concept is feasible and is considered able to counter an identified threat or exploit an opportunity;
  - whether a concept is sufficiently mature to warrant continued development of a new product or line of products; and
  - whether to approve a proposal generated in response to a request for proposal.
- The **execution phase** includes activities related to four stages of the system life cycle: *development, production, utilization, and support*. Typically, there are two decision gates and two milestones associated with execution activities. The first milestone provides the opportunity for management to review the plans for execution before giving the go-ahead. The second milestone provides the opportunity to review progress before the decision is made to initiate production. The decision gates during execution can be used to determine whether to produce the developed SoI and whether to improve it or retire it.

These program management views apply not only to the SoI, but also to its elements and structure.

## Life Cycle Stages

Variations of the Vee model deal with the same general stages of a life cycle:

- New projects typically begin with an exploratory research phase which generally includes the activities of concept definition, specifically the topics of business or mission analysis and the understanding of stakeholder needs and requirements. These mature as the project goes from the exploratory stage to the concept stage to the development stage.
- The production phase includes the activities of system definition and system realization, as well as the development of the system requirements (glossary) and architecture (glossary) through verification and validation.
- The utilization phase includes the activities of system deployment and system operation.
- The support phase includes the activities of system maintenance, logistics, and product and service life management, which may include activities such as service life extension or capability updates, upgrades, and modernization.
- The retirement phase includes the activities of disposal and retirement, though in some models, activities such as service life extension or capability updates, upgrades, and modernization are grouped into the "retirement" phase.

Additional information on each of these stages can be found in the sections below (see links to additional Part 3 articles above for further detail). It is important to note that these life cycle stages, and the activities in each stage, are supported by a set of systems engineering management processes.

### Exploratory Research Stage

User requirements analysis and agreement is part of the exploratory research stage and is critical to the development of successful systems. Without proper understanding of the user needs, any system runs the risk of being built to solve the wrong problems. The first step in the exploratory research phase is to define the user (and stakeholder) requirements and constraints. A key part of this process is to establish the feasibility of meeting the user requirements, including technology readiness assessment. As with many SE activities this is often done iteratively, and stakeholder needs and requirements are revisited as new information becomes available.

A recent study by the National Research Council (National Research Council 2008) focused on reducing the development time for US Air Force projects. The report notes that, “simply stated, systems engineering is the translation of a user’s needs into a definition of a system and its architecture through an iterative process that results in an effective system design.” The iterative involvement with stakeholders is critical to the project success.

Except for the first and last decision gates of a project, the gates are performed simultaneously. See Figure 6 below.

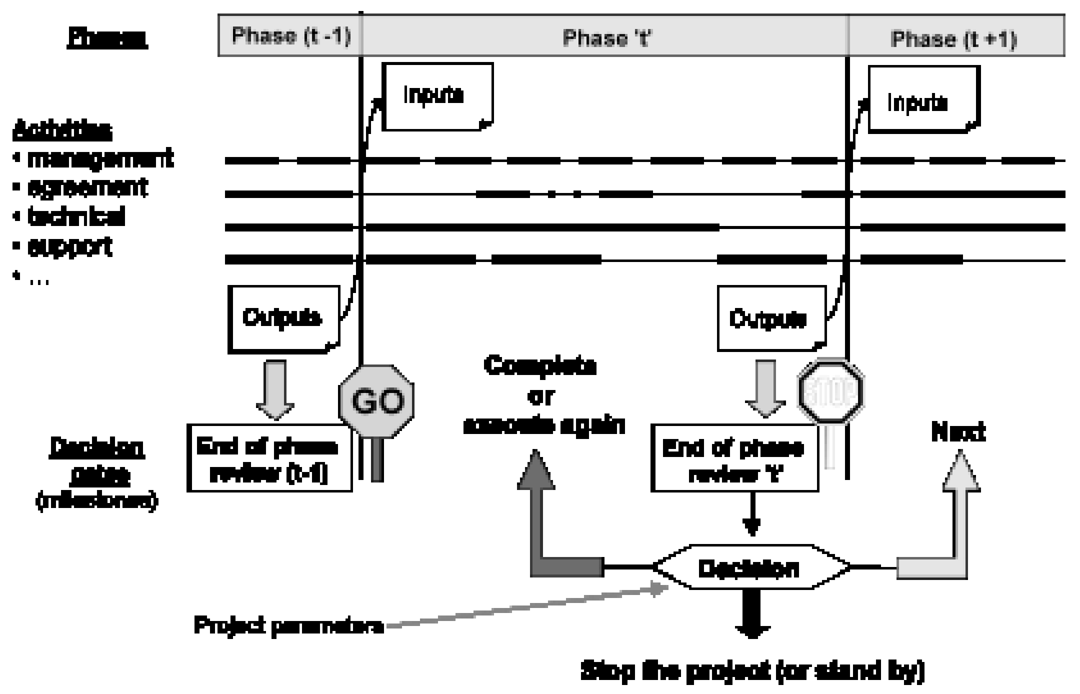


**Figure 6. Scheduling the Development Phases.** (SEBoK Original)

### Concept Stage

During the concept stage, alternate concepts are created to determine the best approach to meet stakeholder needs. By envisioning alternatives and creating models, including appropriate prototypes, stakeholder needs will be clarified and the driving issues highlighted. This may lead to an incremental or evolutionary approach to system development. Several different concepts may be explored in parallel.

### Development Stage

The selected concept(s) identified in the concept stage are elaborated in detail down to the lowest level to produce the solution that meets the stakeholder requirements. Throughout this stage, it is vital to continue with user involvement through in-process validation (the upward arrow on the Vee models). On hardware, this is done with frequent program reviews and a customer resident representative(s) (if appropriate). In agile development, the practice is to have the customer representative integrated into the development team.

### Production Stage

The production stage is where the SoI is built or manufactured. Product modifications may be required to resolve production problems, to reduce production costs, or to enhance product or SoI capabilities. Any of these modifications may influence system requirements and may require system re-qualification, re-verification, or re-validation. All such changes require SE assessment before changes are approved.

### Utilization Stage

A significant aspect of product life cycle management is the provisioning of supporting systems which are vital in sustaining operation of the product. While the supplied product or service may be seen as the narrow system-of-interest (NSOI) for an acquirer, the acquirer also must incorporate the supporting systems into a wider system-of-interest (WSOI). These supporting systems should be seen as system assets that, when needed, are activated in response to a situation that has emerged in respect to the operation of the NSOI. The collective name for the set of supporting systems is the integrated logistics support (ILS) system.

It is vital to have a holistic view when defining, producing, and operating system products and services. In Figure 7, the relationship between system design and development and the ILS requirements is portrayed.

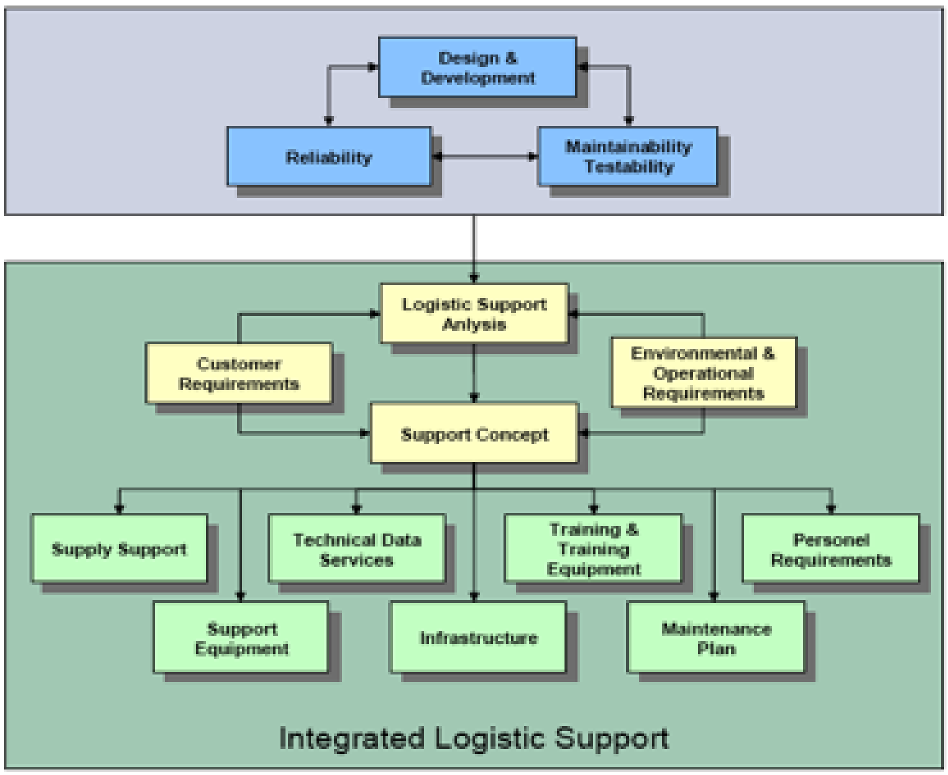


**Figure 7. Relating ILS to the System Life Cycle (Eichmueller and Foreman 2009)**. Reprinted with permission of of ASD/AIA S3000L Steering Committee. All other rights are reserved by the copyright owner.

The requirements for reliability, resulting in the need of maintainability and testability, are driving factors.

## Support Stage

In the support stage, the SoI is provided services that enable continued operation. Modifications may be proposed to resolve supportability problems, to reduce operational costs, or to extend the life of a system. These changes require SE assessment to avoid loss of system capabilities while under operation. The corresponding technical process is the maintenance process.

## Retirement Stage

In the retirement stage, the SoI and its related services are removed from operation. SE activities in this stage are primarily focused on ensuring that disposal requirements are satisfied. In fact, planning for disposal is part of the system definition during the concept stage. Experiences in the 20th century repeatedly demonstrated the consequences when system retirement and disposal was not considered from the outset. Early in the 21st century, many countries have changed their laws to hold the creator of a SoI accountable for proper end-of-life disposal of the system.

## Life Cycle Reviews

To control the progress of a project, different types of reviews are planned. The most commonly used are listed as follows, although the names are not universal:

- The **system requirements review** (SRR) is planned to verify and validate the set of system requirements before starting the detailed design activities.
- The preliminary design review (PDR) is planned to verify and validate the set of system requirements, the design artifacts, and justification elements at the end of the first engineering loop (also known as the "design-to" gate).
- The critical design review (CDR) is planned to verify and validate the set of system requirements, the design artifacts, and justification elements at the end of the last engineering loop (the “build-to” and “code-to” designs are released after this review).
- The integration, verification, and validation reviews are planned as the components are assembled into higher level subsystems and elements. A sequence of reviews is held to ensure that everything integrates properly and that there is objective evidence that all requirements have been met. There should also be an in-process validation that the system, as it is evolving, will meet the stakeholders’ requirements (see Figure 7).
- The final validation review is carried out at the end of the integration phase.
- Other management related reviews can be planned and conducted in order to control the correct progress of work, based on the type of system and the associated risks.

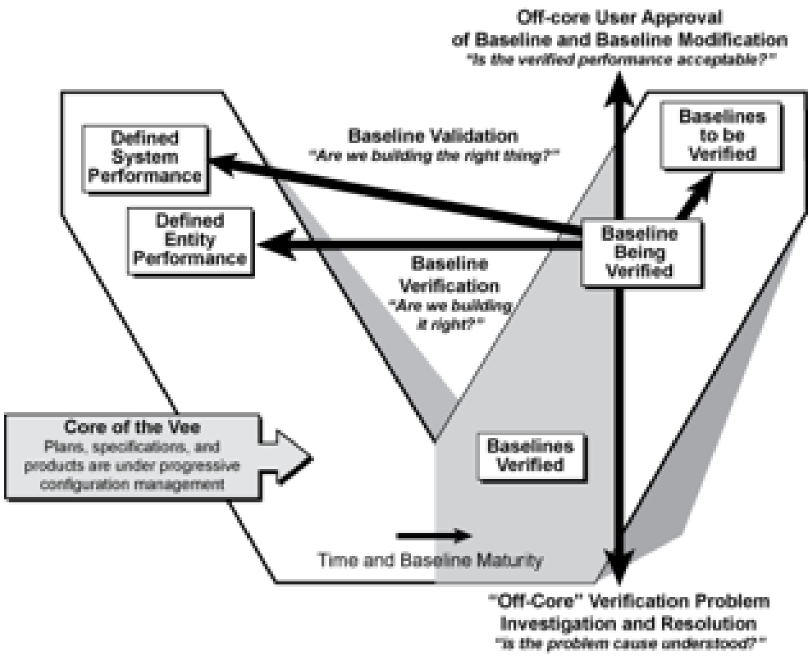

Off-core User Approval
of Baseline and Baseline Modification
"Is the verified performance acceptable?"
Defined System Performance
Baseline Validation
"Are we building the right thing?"
Baselines to be Verified
Defined Entity Performance
Baseline Verification
"Are we building it right?"
Baseline Being Verified
Core of the Vee
Plans, specifications, and products are under progressive configuration management
Baselines Verified
Time and Baseline Maturity
"Off-Core" Verification Problem
Investigation and Resolution
"Is the problem cause understood?"

**Figure 8. Right Side of the Vee Model (Forsberg, Mooz, and Cotterman 2005).** Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

## References

### Works Cited

Eichmueller, P. and B. Foreman. 2010. *S3000LTM*. Brussels, Belgium: Aerospace and Defence Industries Association of Europe (ASD)/Aerospace Industries Association (AIA).

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management,* 3rd ed. New York, NY, USA: J. Wiley & Sons.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications, Kings College, UK.

### Primary References

Beedle, M., et al. 2009. "The Agile Manifesto: Principles behind the Agile Manifesto". in *The Agile Manifesto* [database online]. Accessed December 04 2014 at www.agilemanifesto.org/principles.html

Boehm, B. and R. Turner. 2004. *Balancing Agility and Discipline.* New York, NY, USA: Addison-Wesley.

Fairley, R. 2009. *Managing and Leading Software Projects.* New York, NY, USA: J. Wiley & Sons.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management.* 3rd ed. New York, NY, USA: J. Wiley & Sons.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* Kings College, UK: College Publications.

Pew, R., and A. Mavor (eds.) 2007. *Human-System Integration in the System Development Process: A New Look.* Washington, DC, USA: The National Academies Press.

Royce, W.E. 1998. *Software Project Management: A Unified Framework*. New York, NY, USA: Addison Wesley.

### Additional References

Anderson, D. 2010. *Kanban*. Sequim, WA, USA: Blue Hole Press.

Baldwin, C. and K. Clark. 2000. *Design Rules: The Power of Modularity.* Cambridge, MA, USA: MIT Press.

Beck, K. 1999. *Extreme Programming Explained.* New York, NY, USA: Addison Wesley.

Beedle, M., et al. 2009. "The Agile Manifesto: Principles behind the Agile Manifesto". in The Agile Manifesto [database online]. Accessed 2010. Available at: www.agilemanifesto.org/principles.html

Biffl, S., A. Aurum, B. Boehm, H. Erdogmus, and P. Gruenbacher (eds.). 2005. *Value-Based Software Engineering*. New York, NY, USA: Springer.

Boehm, B. 1988. “A Spiral Model of Software Development.” IEEE *Computer* 21(5): 61-72.

Boehm, B. 2006. “Some Future Trends and Implications for Systems and Software Engineering Processes.” *Systems Engineering.* 9(1): 1-19.

Boehm, B., A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy. 1998. “Using the WinWin Spiral Model: A Case Study.” IEEE *Computer.* 31(7): 33-44.

Boehm, B., R. Turner, J. Lane, S. Koolmanojwong. 2014 (in press). *Embracing the Spiral Model: Creating Successful Systems with the Incremental Commitment Spiral Model*. Boston, MA, USA: Addison Wesley.

Castellano, D.R. 2004. “Top Five Quality Software Projects.” *CrossTalk.* 17(7) (July 2004): 4-19. Available at: http:/ /www.crosstalkonline.org/storage/issue-archives/2004/200407/200407-0-Issue.pdf

Checkland, P. 1981. *Systems Thinking, Systems Practice*. New York, NY, USA: Wiley.

Crosson, S. and B. Boehm. 2009. “Adjusting Software Life cycle Anchorpoints: Lessons Learned in a System of Systems Context.” Proceedings of the Systems and Software Technology Conference, 20-23 April 2009, Salt Lake City, UT, USA.

Dingsoyr, T., T. Dyba. and N. Moe (eds.). 2010. "Agile Software Development: Current Research and Future Directions.” Chapter in B. Boehm, J. Lane, S. Koolmanjwong, and R. Turner, *Architected Agile Solutions for Software-Reliant Systems.* New York, NY, USA: Springer.

Dorner, D. 1996. *The Logic of Failure*. New York, NY, USA: Basic Books.

Forsberg, K. 1995. "'If I Could Do That, Then I Could…' System Engineering in a Research and Development Environment.” Proceedings of the Fifth Annual International Council on Systems Engineering (INCOSE) International Symposium. 22-26 July 1995. St. Louis, MO, USA.

Forsberg, K. 2010. “Projects Don’t Begin With Requirements.” Proceedings of the IEEE Systems Conference, 5-8 April 2010, San Diego, CA, USA.

Gilb, T. 2005. *Competitive Engineering*. Maryland Heights, MO, USA: Elsevier Butterworth Heinemann.

Goldratt, E. 1984. *The Goal*. Great Barrington, MA, USA: North River Press.

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology.* New York, NY, USA: Wiley.

Holland, J. 1998. *Emergence*. New York, NY, USA: Perseus Books.

ISO/IEC. 2010. Systems and Software Engineering, Part 1: Guide for Life Cycle Management. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 24748-1:2010.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes.* Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC. 2003. *Systems Engineering — A Guide for The Application of ISO/IEC 15288 System Life Cycle Processes*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 19760:2003 (E).

Jarzombek, J. 2003. “Top Five Quality Software Projects.” *CrossTalk.* 16(7) (July 2003): 4-19. Available at: http:// www.crosstalkonline.org/storage/issue-archives/2003/200307/200307-0-Issue.pdf.

Kruchten, P. 1999. *The Rational Unified Process.* New York, NY, USA: Addison Wesley.

Landis, T. R. 2010. *Lockheed Blackbird Family (A-12, YF-12, D-21/M-21 & SR-71).* North Branch, MN, USA: Specialty Press.

Madachy, R. 2008. *Software Process Dynamics*. Hoboken, NJ, USA: Wiley.

Maranzano, J.F., S.A. Rozsypal, G.H. Zimmerman, G.W. Warnken, P.E. Wirth, D.W. Weiss. 2005. “Architecture Reviews: Practice and Experience.” IEEE *Software.* 22(2): 34-43.

National Research Council of the National Academies (USA). 2008. *Pre-Milestone A and Early-Phase Systems Engineering*. Washington, DC, USA: The National Academies Press.

Osterweil, L. 1987. “Software Processes are Software Too.” Proceedings of the SEFM 2011: 9th International Conference on Software Engineering. Monterey, CA, USA.

Poppendeick, M. and T. Poppendeick. 2003. *Lean Software Development: an Agile Toolkit.* New York, NY, USA: Addison Wesley.

Rechtin, E. 1991. *System Architecting: Creating and Building Complex Systems.* Upper Saddle River, NY, USA: Prentice-Hall.

Rechtin, E., and M. Maier. 1997. *The Art of System Architecting*. Boca Raton, FL, USA: CRC Press.

Schwaber, K. and M. Beedle. 2002. *Agile Software Development with Scrum*. Upper Saddle River, NY, USA: Prentice Hall.

Spruill, N. 2002. "Top Five Quality Software Projects." *CrossTalk.* 15(1) (January 2002): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2002/200201/200201-0-Issue.pdf.

Stauder, T. 2005. "Top Five Department of Defense Program Awards." *CrossTalk.* 18(9) (September 2005): 4-13. Available at http://www.crosstalkonline.org/storage/issue-archives/2005/200509/200509-0-Issue.pdf.

Warfield, J. 1976. *Societal Systems: Planning, Policy, and Complexity*. New York, NY, USA: Wiley.

Womack, J. and D. Jones. 1996. *Lean Thinking.* New York, NY, USA: Simon and Schuster.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# System Life Cycle Process Models: Iterative

***Contributing Author:*** *Kevin Forsberg*

There are a large number of life cycle process models. As discussed in the System Life Cycle Process Drivers and Choices article, these models fall into three major categories: (1) primarily pre-specified and sequential processes; (2) primarily evolutionary and concurrent processes (e.g., the rational unified process and various forms of the Vee and spiral models); and (3) primarily interpersonal and unconstrained processes (e.g., agile development, Scrum, extreme programming (XP), dynamic system development methods, and innovation-based processes).

This article discusses incremental and evolutionary development models (the second and third categories listed above) beyond variants of the Vee model. While there are a number of different models describing the project environment, the spiral model and the Vee Model have become the dominant approaches to visualizing the development process. Both the Vee and the spiral are useful models that emphasize different aspects of a system life cycle.

General implications of using iterative models for system design and development are discussed below. For a more specific understanding of how this life cycle model impacts systems engineering activities, please see the other knowledge areas (KAs) in Part 3. This article is focused on the use of iterative life cycle process models in systems engineering; however, because iterative process models are commonly used in software development, many of the examples below come from software projects. (See Systems Engineering and Software Engineering in Part 6 for more information on life cycle implications in software engineering.)

# Incremental and Evolutionary Development

## Overview of the Incremental Approach

Incremental and iterative development (IID) methods have been in use since the 1960s (and perhaps earlier). They allow a project to provide an initial capability followed by successive deliveries to reach the desired system-of-interest (SoI).

The IID approach, shown in Figure 1, is used when:

- rapid exploration and implementation of part of the system is desired;
- the requirements are unclear from the beginning;
- funding is constrained;
- the customer wishes to hold the SoI open to the possibility of inserting new technology at a later time; and/or
- experimentation is required to develop successive prototype (glossary) versions.

The attributes that distinguish IID from the single-pass, plan-driven approach are velocity and adaptability.

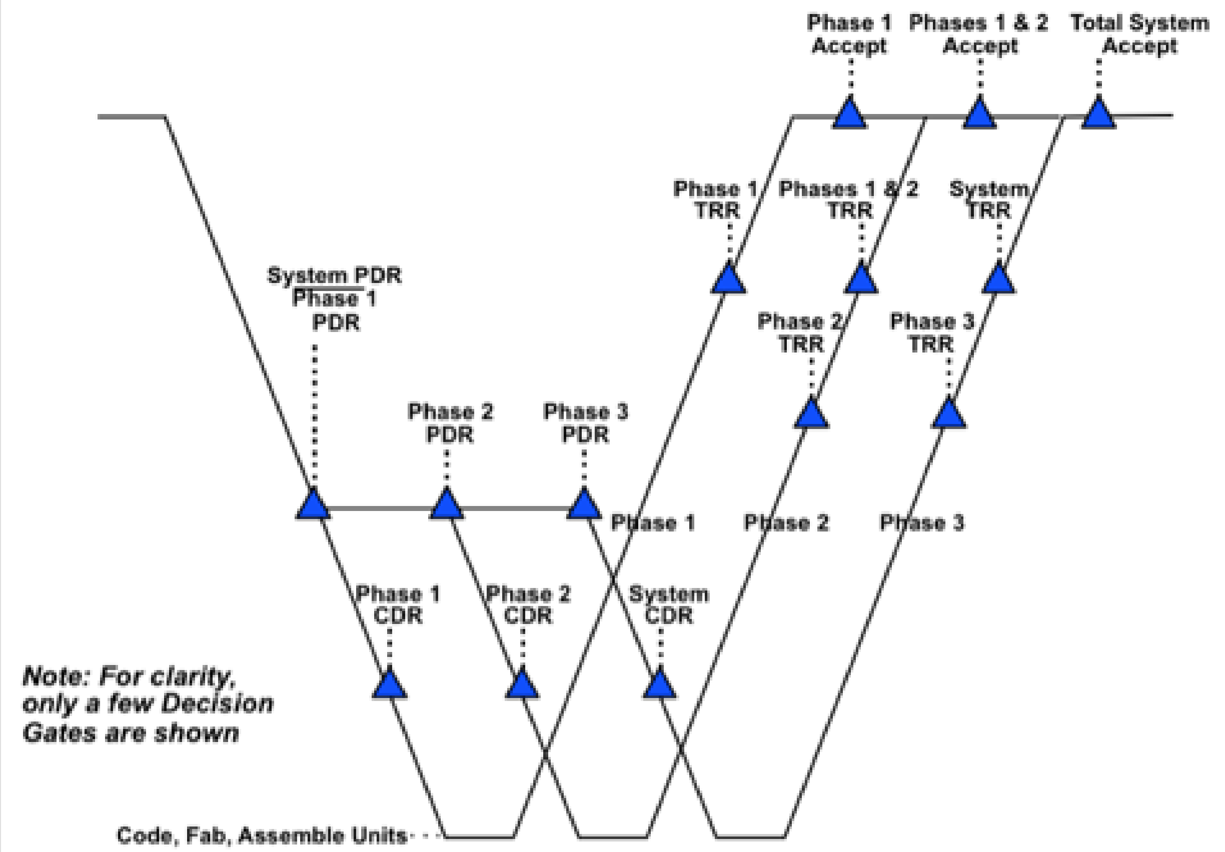


**Figure 1. Incremental Development with Multiple Deliveries (Forsberg, Mooz, and Cotterman 2005).** Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

Incremental development may also be “plan-driven” in nature if the requirements are known early on in the life cycle. The development of the functionality is performed incrementally to allow for insertion of the latest technology or for potential changes in needs or requirements. IID also imposes constraints. The example shown in Figure 2 uses the increments to develop high-risk subsystems (or components) early, but the system cannot function until all increments are complete.

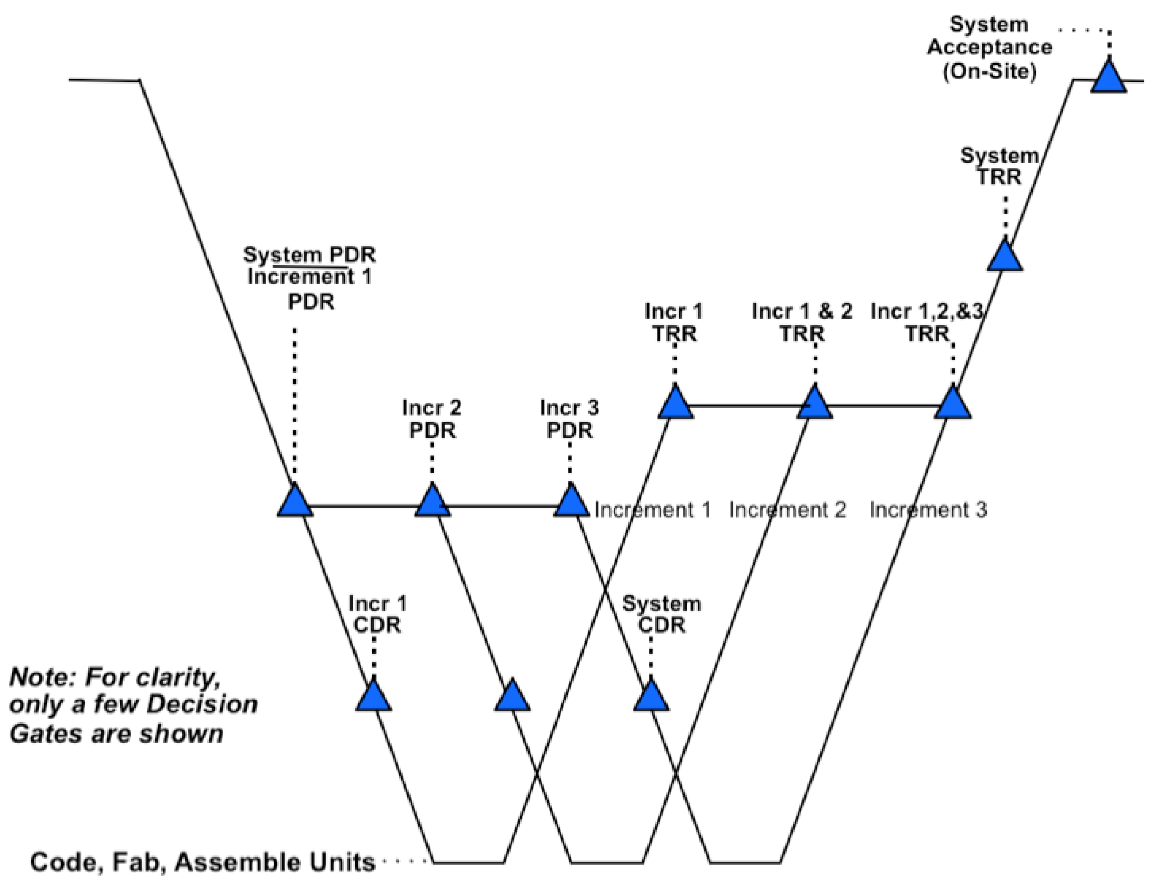


**Figure 2. Incremental Development with a Single Delivery (Forsberg, Mooz, Cotterman 2005).** Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

## Overview of the Evolutionary Approach

A specific IID methodology called evolutionary development is common in research and development (R&D) environments in both the government and commercial sector. Figure 3 illustrates this approach, which was used in the evolution of the high temperature tiles for the NASA Space Shuttle (Forsberg 1995). In the evolutionary approach, the end state of each phase of development is unknown, though the goal is for each phase to result in some sort of useful product.

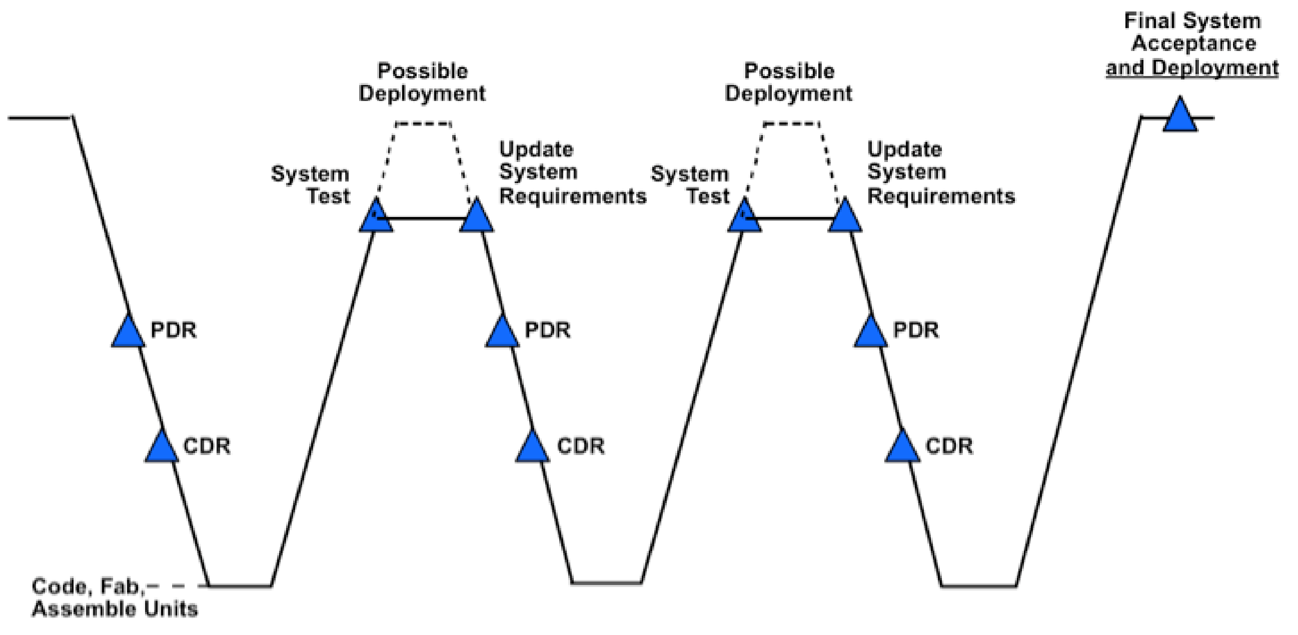

Final System Acceptance and Deployment
Possible Deployment
Possible Deployment
System Test
Update System Requirements
System Test
Update System Requirements
PDR
CDR
PDR
CDR
PDR
CDR
Code, Fab, Assemble Units

**Figure 3. Evolutionary Generic Model (Forsberg, Mooz, Cotterman 2005).** Reprinted with permission of John Wiley & Sons, Inc. All other rights are reserved by the copyright owner.

The real-world development environment is complex and difficult to map because many different project cycles are underway simultaneously. Figure 4 shows the applied research era for the development of the space shuttle Orbiter and illustrates multi-levels of simultaneous development, trade-studies, and ultimately, implementation.

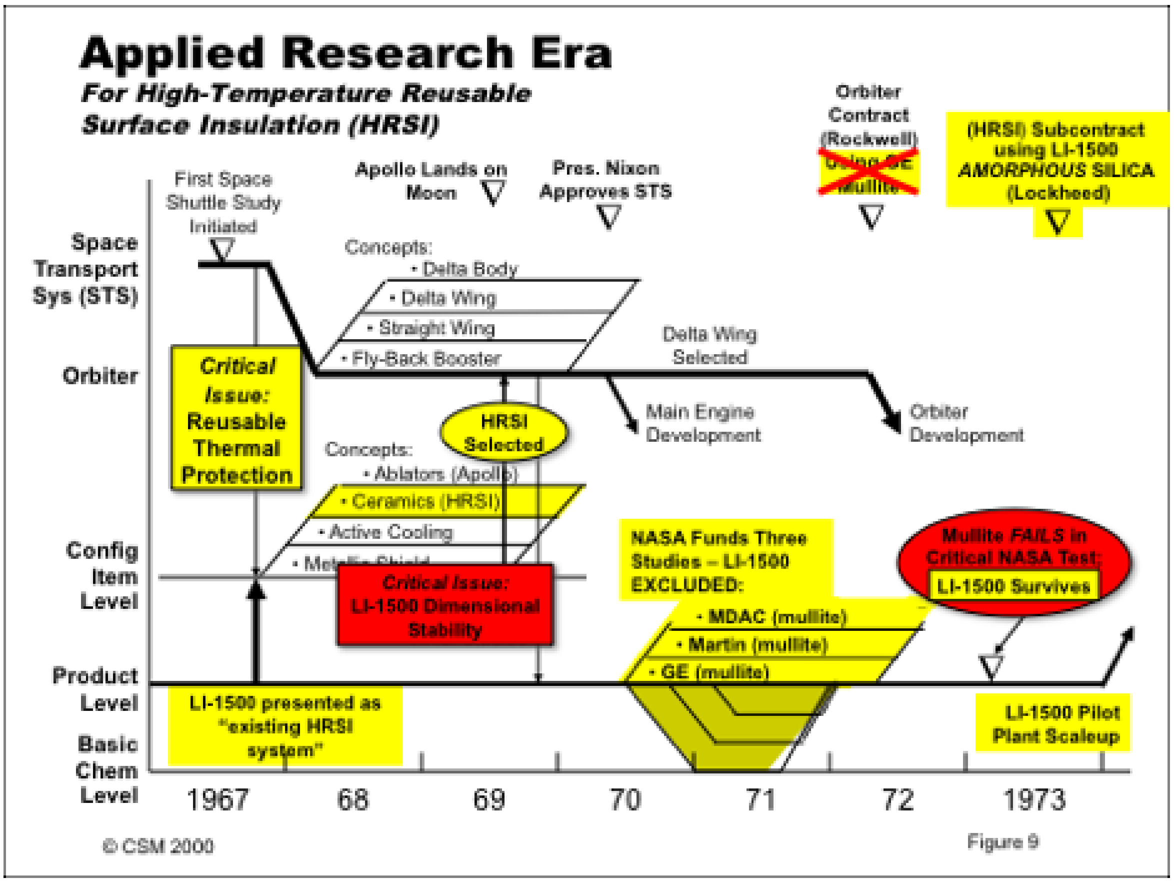

Applied Research Era
For High-Temperature Reusable Surface Insulation (HRSI)
First Space Shuttle Study Initiated
Apollo Lands on Moon
Pres. Nixon Approves STS
Orbiter Contract (Rockwell)
~~Using GE Mullite~~
(HRSI) Subcontract using LI-1500 AMORPHOUS SILICA (Lockheed)
Space Transport Sys (STS)
Orbiter
Config Item Level
Product Level
Basic Chem Level
Concepts:
• Delta Body
• Delta Wing
• Straight Wing
• Fly-Back Booster
Delta Wing Selected
Main Engine Development
Orbiter Development
Critical Issue: Reusable Thermal Protection
HRSI Selected
Concepts:
• Ablators (Apollo)
• Ceramics (HRSI)
• Active Cooling
• Metallic Shield
Critical Issue: LI-1500 Dimensional Stability
NASA Funds Three Studies – LI-1500 EXCLUDED:
• MDAC (mullite)
• Martin (mullite)
• GE (mullite)
Mullite FAILS in Critical NASA Test; LI-1500 Survives
LI-1500 presented as "existing HRSI system"
LI-1500 Pilot Plant Scaleup
1967
68
69
70
71
72
1973
© CSM 2000
Figure 9

**Figure 4. Evolution of Components and Orbiter Subsystems (including space shuttle tiles) During Creation of a Large "Single-Pass" Project (Forsberg 1995).** Reprinted with permission of Kevin Forsberg. All other rights are reserved by the copyright owner.

## Iterative Software Development Process Models

Software is a flexible and malleable medium which facilitates iterative analysis, design, construction, verification, and validation to a greater degree than is usually possible for the purely physical components of a system. Each repetition of an iterative development model adds material (code) to the growing software base; the expanded code base is tested, reworked as necessary, and demonstrated to satisfy the requirements for the baseline.

Process models for software development support iterative development on cycles of various lengths. Table 1 lists three iterative software development models which are presented in more detail below, as well as the aspects of software development that are emphasized by those models.

**Table 1. Primary Emphases of Three Iterative Software Development Models.**

| Iterative Model | Emphasis |
|---|---|
| **Incremental-build** | Iterative implementation-verification-validations-demonstration cycles |
| **Spiral** | Iterative risk-based analysis of alternative approaches and evaluation of outcomes |
| **Agile** | Iterative evolution of requirements and code |

Please note that the information below is focused specifically on the utilization of different life cycle models for software systems. In order to better understand the interactions between software engineering (SwE) and systems engineering (SE), please see the Systems Engineering and Software Engineering KA in Part 6.

### Overview of Iterative-Development Process Models

Developing and modifying software involves creative processes that are subject to many external and changeable forces. Long experience has shown that it is impossible to “get it right” the first time, and that iterative development processes are preferable to linear, sequential development process models, such as the well-known Waterfall model. In iterative development, each cycle of the iteration subsumes the software of the previous iteration and adds new capabilities to the evolving product to create an expanded version of the software. Iterative development processes provide the following advantages:

- Continuous integration, verification, and validation of the evolving product;
- Frequent demonstrations of progress;
- Early detection of defects;
- Early warning of process problems;
- Systematic incorporation of the inevitable rework that occurs in software development; and
- Early delivery of subset capabilities (if desired).

Iterative development takes many forms in SwE, including the following:

- An incremental-build process, which is used to produce periodic (typically weekly) builds of increasing product capabilities;
- Agile development, which is used to closely involve a prototypical customer in an iterative process that may repeat on a daily basis; and
- The spiral model, which is used to confront and mitigate risk factors encountered in developing the successive versions of a product.

## The Incremental-Build Model

The incremental-build model is a build-test-demonstrated model of iterative cycles in which frequent demonstrations of progress, verification, and validation of work-to-date are emphasized. The model is based on stable requirements and a software architectural specification. Each build adds new capabilities to the incrementally growing product. The process ends when the final version is verified, validated, demonstrated, and accepted by the customer.

Table 2 lists some partitioning criteria for incremental development into incremental build units of (typically) one calendar week each. The increments and the number of developers available to work on the project determine the number of features that can be included in each incremental build. This, in turn, determines the overall schedule.

**Table 2. Some partitioning criteria for incremental builds (Fairley 2009). Reprinted with permission of the IEEE Computer Society and John Wiley & Sons Inc. All other rights are reserved by the copyright owner.**

| Kind of System | Partitioning Criteria |
|---|---|
| Application package | Priority of features |
| Safety-critical systems | Safety features first; prioritized others follow |
| User-intensive systems | User interface first; prioritized others follow |
| System software | Kernel first; prioritized utilities follow |

Figure 5 illustrates the details of the build-verify-validate-demonstrate cycles in the incremental build process. Each build includes detailed design, coding, integration, review, and testing done by the developers. In cases where code is to be reused without modification, some or all of an incremental build may consist of review, integration, and testing of the base code augmented with the reused code. It is important to note that development of an increment may result in reworking previous components developed for integration to fix defects.

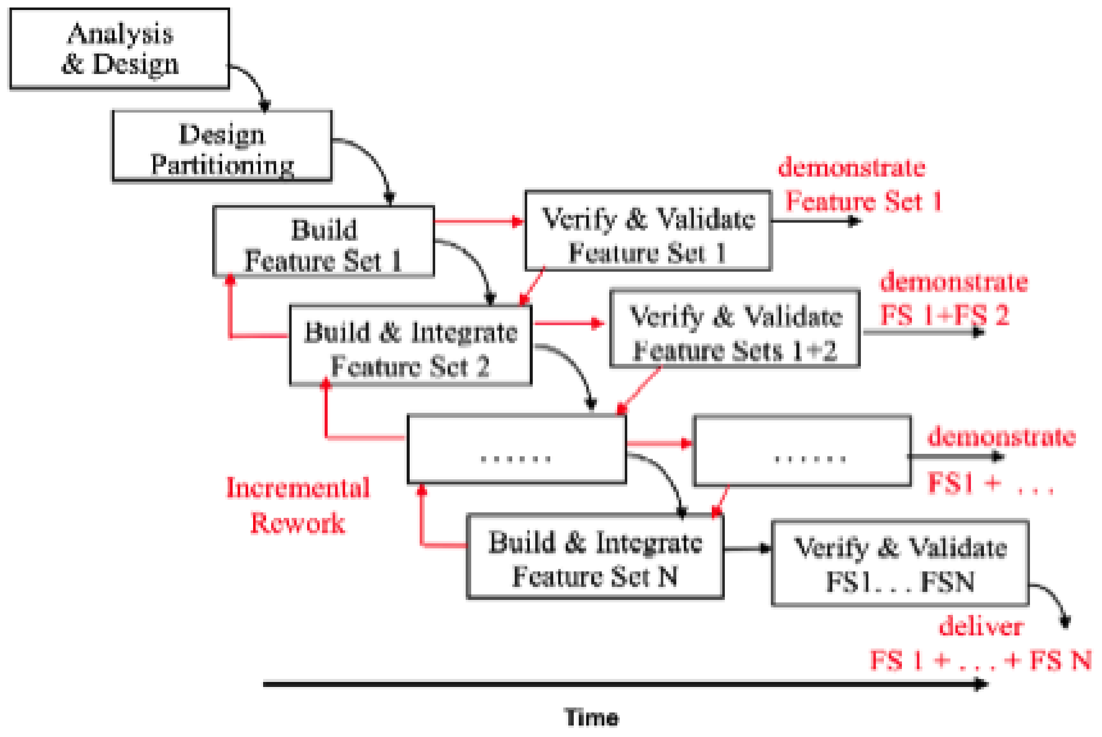

Analysis & Design
Design Partitioning
Build Feature Set 1
Verify & Validate Feature Set 1
demonstrate Feature Set 1
Build & Integrate Feature Set 2
Verify & Validate Feature Sets 1+2
demonstrate FS 1+FS 2
. . . . . .
. . . . . .
demonstrate FS1 + . . .
Incremental Rework
Build & Integrate Feature Set N
Verify & Validate FS1 . . . FSN
deliver FS 1 + . . . + FS N
Time

**Figure 5. Incremental Build-Verify-Validate-Demonstrate Cycles (Fairley 2009).** Reprinted with permission of the IEEE Computer Society and John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

Incremental verification, validation, and demonstration, as illustrated in Figure 5, overcome two of the major problems of a waterfall approach by:

- exposing problems early so they can be corrected as they occur; and
- incorporating minor in-scope changes to requirements that occur as a result of incremental demonstrations in subsequent builds.

Figure 5 also illustrates that it may be possible to overlap successive builds of the product. It may be possible, for example, to start a detailed design of the next version while the present version is being validated.

Three factors determine the degree of overlap that can be achieved:

1. Availability of personnel;
2. Adequate progress on the previous version; and
3. The risk of significant rework on the next overlapped build because of changes to the previous in-progress build.

The incremental build process generally works well with small teams, but can be scaled up for larger projects.

A significant advantage of an incremental build process is that features built first are verified, validated, and demonstrated most frequently because subsequent builds incorporate the features of the earlier iterations. In building the software to control a nuclear reactor, for example, the emergency shutdown software could be built first, as it would then be verified and validated in conjunction with the features of each successive build.

In summary, the incremental build model, like all iterative models, provides the advantages of continuous integration and validation of the evolving product, frequent demonstrations of progress, early warning of problems, early delivery of subset capabilities, and systematic incorporation of the inevitable rework that occurs in software development.

## The Role of Prototyping in Software Development

In SwE, a prototype is a mock-up of the desired functionality of some part of the system. This is in contrast to physical systems, where a prototype is usually the first fully functional version of a system (Fairley 2009, 74).

In the past, incorporating prototype software into production systems has created many problems. Prototyping is a useful technique that should be employed as appropriate; however, prototyping is *not* a process model for software development. When building a software prototype, the knowledge gained through the development of the prototype is beneficial to the program; however, the prototype code may not be used in the deliverable version of the system. In many cases, it is more efficient and more effective to build the production code from scratch using the knowledge gained by prototyping than to re-engineer the existing code.

## Life Cycle Sustainment of Software

Software, like all systems, requires sustainment efforts to enhance capabilities, adapt to new environments, and correct defects. The primary distinction for software is that sustainment efforts change the software; unlike physical entities, software components do not have to be replaced because of physical wear and tear. Changing the software requires re-verification and re-validation, which may involve extensive regression testing to determine that the change has the desired effect and has not altered other aspects of functionality or behavior.

## Retirement of Software

Useful software is rarely retired; however, software that is useful often experiences many upgrades during its lifetime. A later version may bear little resemblance to the initial release. In some cases, software that ran in a former operational environment is executed on hardware emulators that provide a virtual machine on newer hardware. In other cases, a major enhancement may replace and rename an older version of the software, but the enhanced version provides all of the capabilities of the previous software in a compatible manner. Sometimes, however, a newer version of software may fail to provide compatibility with the older version, which necessitates other changes to a system.

## Primarily Evolutionary and Concurrent Processes: The Incremental Commitment Spiral Model

### Overview of the Incremental Commitment Spiral Model

A view of the Incremental Commitment Spiral Model (ICSM) is shown in Figure 6.

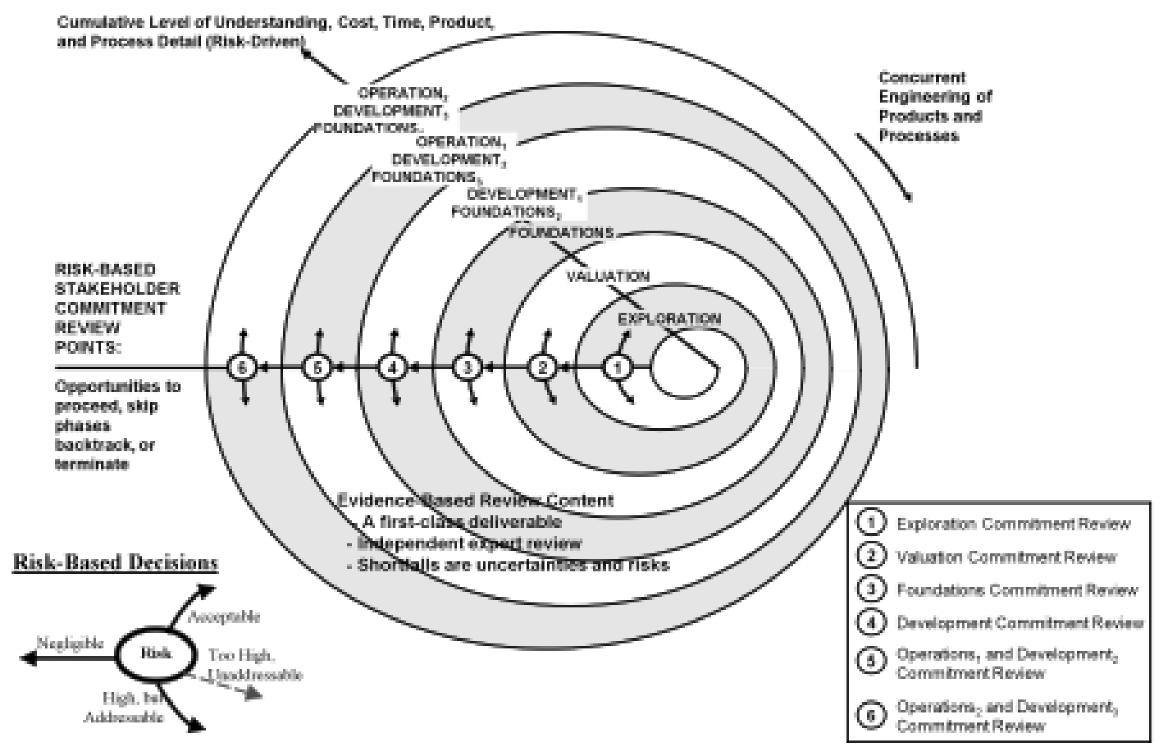

Cumulative Level of Understanding, Cost, Time, Product, and Process Detail (Risk-Driven)
OPERATION₁ DEVELOPMENT₂ FOUNDATIONS₃
OPERATION₂ DEVELOPMENT₃ FOUNDATIONS₄
DEVELOPMENT₁ FOUNDATIONS₂
FOUNDATIONS
VALUATION
EXPLORATION
Concurrent Engineering of Products and Processes
RISK-BASED STAKEHOLDER COMMITMENT REVIEW POINTS:
Opportunities to proceed, skip phases backtrack, or terminate
6 5 4 3 2 1
Evidence-Based Review Content
- A first-class deliverable
- Independent expert review
- Shortfalls are uncertainties and risks
Risk-Based Decisions
Risk
Acceptable
Negligible
Too High, Unaddressable
High, but Addressable
1 Exploration Commitment Review
2 Valuation Commitment Review
3 Foundations Commitment Review
4 Development Commitment Review
5 Operations₁ and Development₂ Commitment Review
6 Operations₂ and Development₃ Commitment Review

**Figure 6. The Incremental Commitment Spiral Model (ICSM) (Pew and Mavor 2007).** Reprinted with permission by the National Academy of Sciences, Courtesy of National Academies Press, Washington, D.C. All other rights are reserved by the copyright owner.

In the ICSM, each spiral addresses requirements and solutions concurrently, rather than sequentially, as well as products and processes, hardware, software, human factors aspects, and business case analyses of alternative product configurations or product line investments. The stakeholders consider the risks and risk mitigation plans and decide on a course of action. If the risks are acceptable and covered by risk mitigation plans, the project proceeds into the next spiral.

The development spirals after the first development commitment review follow the three-team incremental development approach for achieving both agility and assurance shown and discussed in Figure 2, "Evolutionary-Concurrent Rapid Change Handling and High Assurance" of System Life Cycle Process Drivers and Choices.

### Other Views of the Incremental Commitment Spiral Model

Figure 7 presents an updated view of the ICSM life cycle process recommended in the National Research Council *Human-System Integration in the System Development Process* study (Pew and Mavor 2007). It was called the Incremental Commitment Model (ICM) in the study. The ICSM builds on the strengths of current process models, such as early verification and validation concepts in the Vee model, concurrency concepts in the concurrent engineering model, lighter-weight concepts in the agile and lean models, risk-driven concepts in the spiral model, the phases and anchor points in the rational unified process (RUP) (Kruchten 1999; Boehm 1996), and recent extensions of the spiral model to address systems of systems (SoS) capability acquisition (Boehm and Lane 2007).

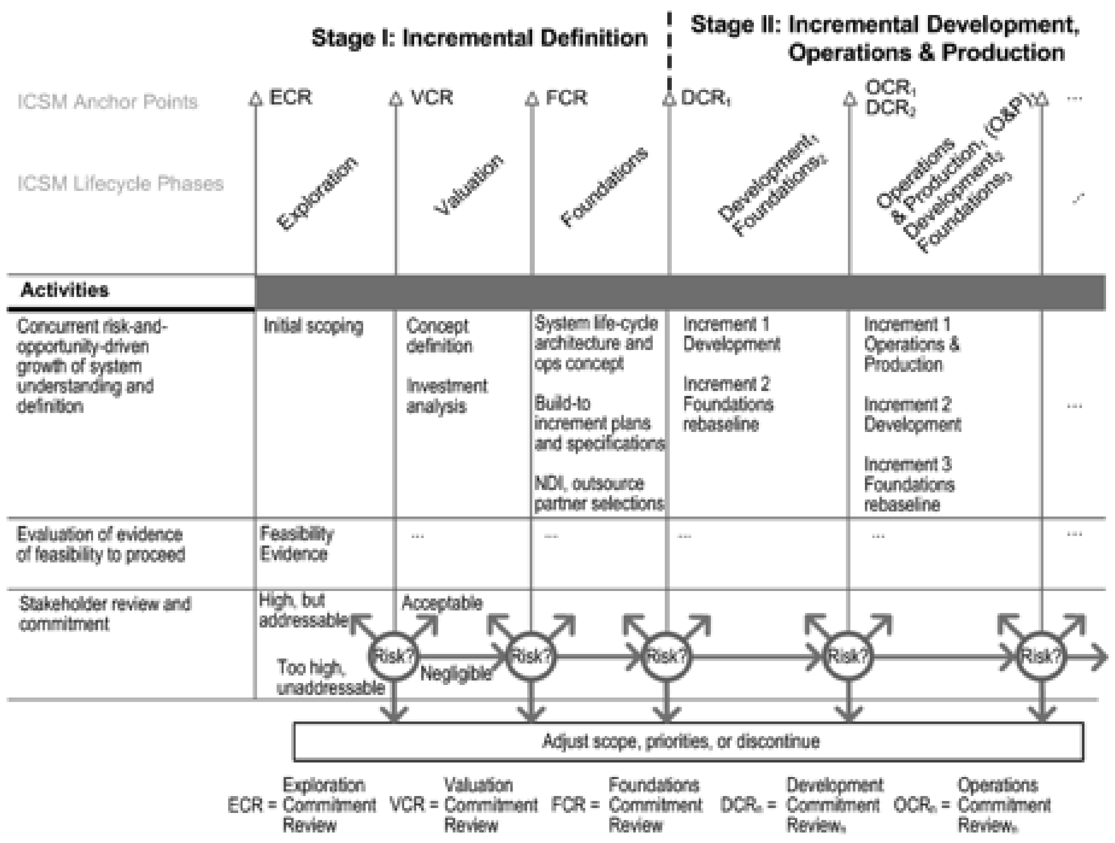

Stage I: Incremental Definition
Stage II: Incremental Development, Operations & Production
ICSM Anchor Points
ECR
VCR
FCR
DCR$_1$
OCR$_1$
DCR$_2$
...
ICSM Lifecycle Phases
Exploration
Valuation
Foundations
Development$_1$ Foundations$_2$
Operations$_1$ & Production$_1$ (O&P$_1$) Development$_2$ Foundations$_3$
Activities
Concurrent risk-and-opportunity-driven growth of system understanding and definition
Initial scoping
Concept definition
Investment analysis
System life-cycle architecture and ops concept
Build-to increment plans and specifications
NDI, outsource partner selections
Increment 1 Development
Increment 2 Foundations rebaseline
Increment 1 Operations & Production
Increment 2 Development
Increment 3 Foundations rebaseline
Evaluation of evidence of feasibility to proceed
Feasibility Evidence
Stakeholder review and commitment
High, but addressable
Acceptable
Risk?
Too high, unaddressable
Negligible
Adjust scope, priorities, or discontinue
ECR = Exploration Commitment Review
VCR = Valuation Commitment Review
FCR = Foundations Commitment Review
DCR$_n$ = Development Commitment Review$_n$
OCR$_n$ = Operations Commitment Review$_n$

**Figure 7. Phased View of the Generic Incremental Commitment Spiral Model Process (Pew and Mavor 2007).** Reprinted with permission by the National Academy of Sciences, Courtesy of National Academies Press, Washington, D.C. All other rights are reserved by the copyright owner.

The top row of activities in Figure 7 indicates that a number of system aspects are being concurrently engineered at an increasing level of understanding, definition, and development. The most significant of these aspects are shown in Figure 8, an extension of a similar *"hump diagram"* view of concurrently engineered software activities developed as part of the RUP (Kruchten 1999).

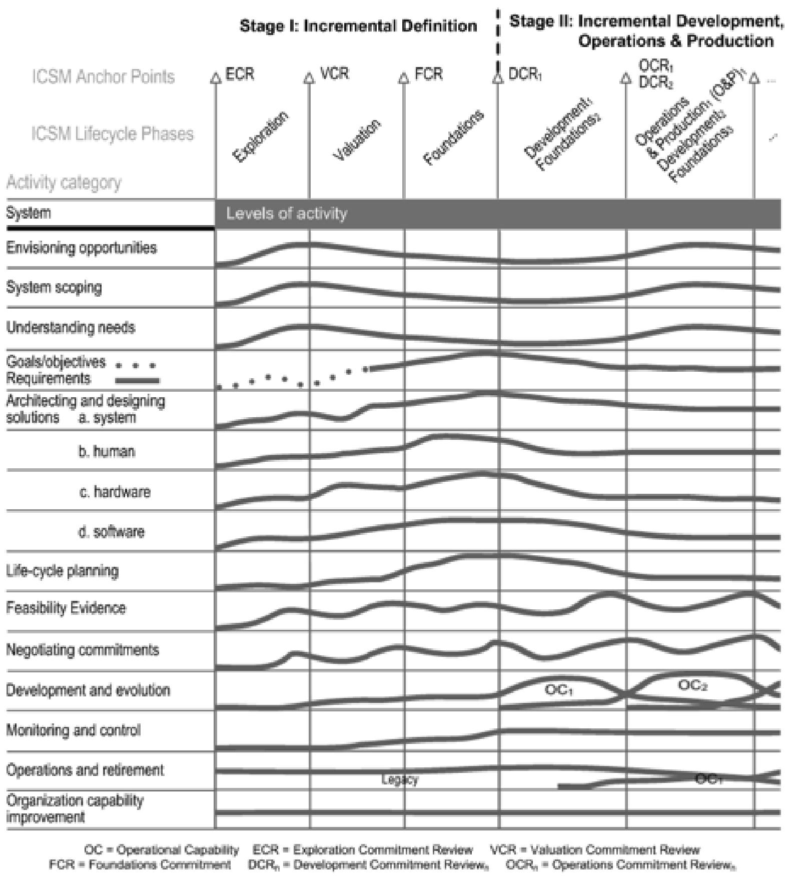
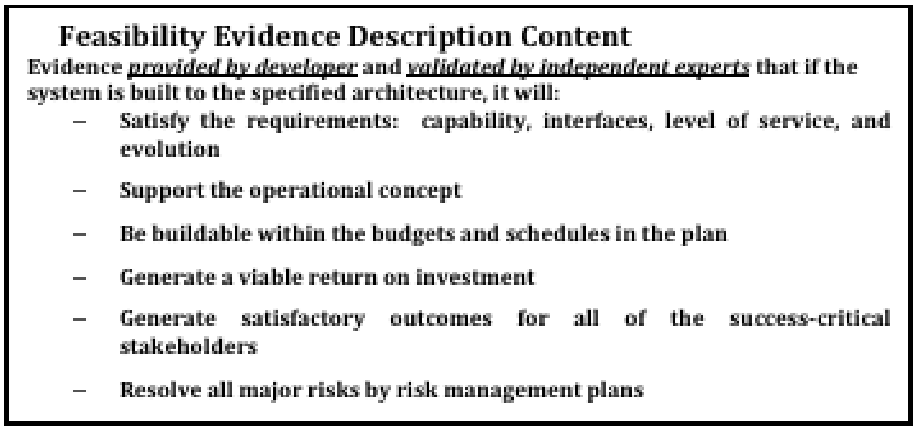


**Figure 8. ICSM Activity Categories and Level of Effort (Pew and Mavor 2007).**
Reprinted with permission by the National Academy of Sciences, Courtesy of National Academies Press, Washington, D.C. All other rights are reserved by the copyright owner.

As with the RUP version, the magnitude and shape of the levels of effort will be risk-driven and likely to vary from project to project. Figure 8 indicates that a great deal of concurrent activity occurs within and across the various ICSM phases, all of which need to be *"synchronized and stabilized,"* a best-practice phrase taken from *Microsoft Secrets* (Cusumano and Selby 1996) to keep the project under control.

The review processes and use of independent experts are based on the highly successful AT&T Architecture Review Board procedures described in “Architecture Reviews: Practice and Experience” (Maranzano et al. 2005). Figure 9 shows the content of the feasibility evidence description. Showing feasibility of the concurrently developed elements helps synchronize and stabilize the concurrent activities.



**Feasibility Evidence Description Content**

Evidence ***provided by developer*** and ***validated by independent experts*** that if the system is built to the specified architecture, it will:

– Satisfy the requirements: capability, interfaces, level of service, and evolution

– Support the operational concept

– Be buildable within the budgets and schedules in the plan

– Generate a viable return on investment

– Generate satisfactory outcomes for all of the success-critical stakeholders

– Resolve all major risks by risk management plans

**Figure 9. Feasibility Evidence Description Content (Pew and Mavor 2007).** Reprinted with permission by the National Academy of Sciences, Courtesy of National Academies Press, Washington, D.C. All other rights are reserved by the copyright owner.

The operations commitment review (OCR) is different in that it addresses the often-higher operational risks of fielding an inadequate system. In general, stakeholders will experience a two- to ten-fold increase in commitment level while going through the sequence of engineering certification review (ECR) to design certification review (DCR) milestones, but the increase in going from DCR to OCR can be much higher. These commitment levels are based on typical cost profiles across the various stages of the acquisition life cycle.

### Underlying ICSM Principles

ICSM has four underlying principles which must be followed:

1. Stakeholder value-based system definition and evolution;
2. Incremental commitment and accountability;
3. Concurrent system and software definition and development; and
4. Evidence and risk-based decision making.

### Model Experience to Date

The National Research Council Human-Systems Integration study (2008) found that the ICSM processes and principles correspond well with best commercial practices, as described in the Next Generation Medical Infusion Pump Case Study in Part 7. Further examples are found in *Human-System Integration in the System Development Process: A New Look* (Pew and Mavor 2007, chap. 5), *Software Project Management* (Royce 1998, Appendix D), and the annual series of "Top Five Quality Software Projects", published in CrossTalk (2002-2005).

## Agile and Lean Processes

According to the INCOSE *Systems Engineering Handbook* 3.2.2, *"Project execution methods can be described on a continuum from 'adaptive' to 'predictive.' Agile methods exist on the 'adaptive' side of this continuum, which is not the same as saying that agile methods are 'unplanned' or 'undisciplined,'"* (INCOSE 2011, 179). Agile development methods can be used to support iterative life cycle models, allowing flexibility over a linear process that better aligns with the planned life cycle for a system. They primarily emphasize the development and use of tacit interpersonal knowledge as compared to explicit documented knowledge, as evidenced in the four value propositions in the **"Agile Manifesto"**:

> *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value*

- ***Individuals and interactions*** *over processes and tools;*

- ***Working software** over comprehensive documentation;*
- ***Customer collaboration** over contract negotiation; and*
- ***Responding to change** over following a plan.*

  *That is, while there is value in the items on the right, we value the items on the left more.* (Agile Alliance 2001)

Lean processes are often associated with agile methods, although they are more scalable and applicable to high-assurance systems. Below, some specific agile methods are presented, and the evolution and content of lean methods is discussed. Please see "Primary References", "Additional References", and the Lean Engineering article for more detail on specific agile and lean processes.

## Scrum

Figure 10 shows an example of Scrum as an agile process flow. As with most other agile methods, Scrum uses the evolutionary sequential process shown in Table 1 (above) and described in Fixed-Requirements and Evolutionary Development Processes section in which systems capabilities are developed in short periods, usually around 30 days. The project then re-prioritizes its backlog of desired features and determines how many features the team (usually 10 people or less) can develop in the next 30 days.

Figure 10 also shows that once the features to be developed for the current Scrum have been expanded (usually in the form of informal stories) and allocated to the team members, the team establishes a daily rhythm of starting with a short meeting at which each team member presents a roughly one-minute summary describing progress since the last Scrum meeting, potential obstacles, and plans for the upcoming day.

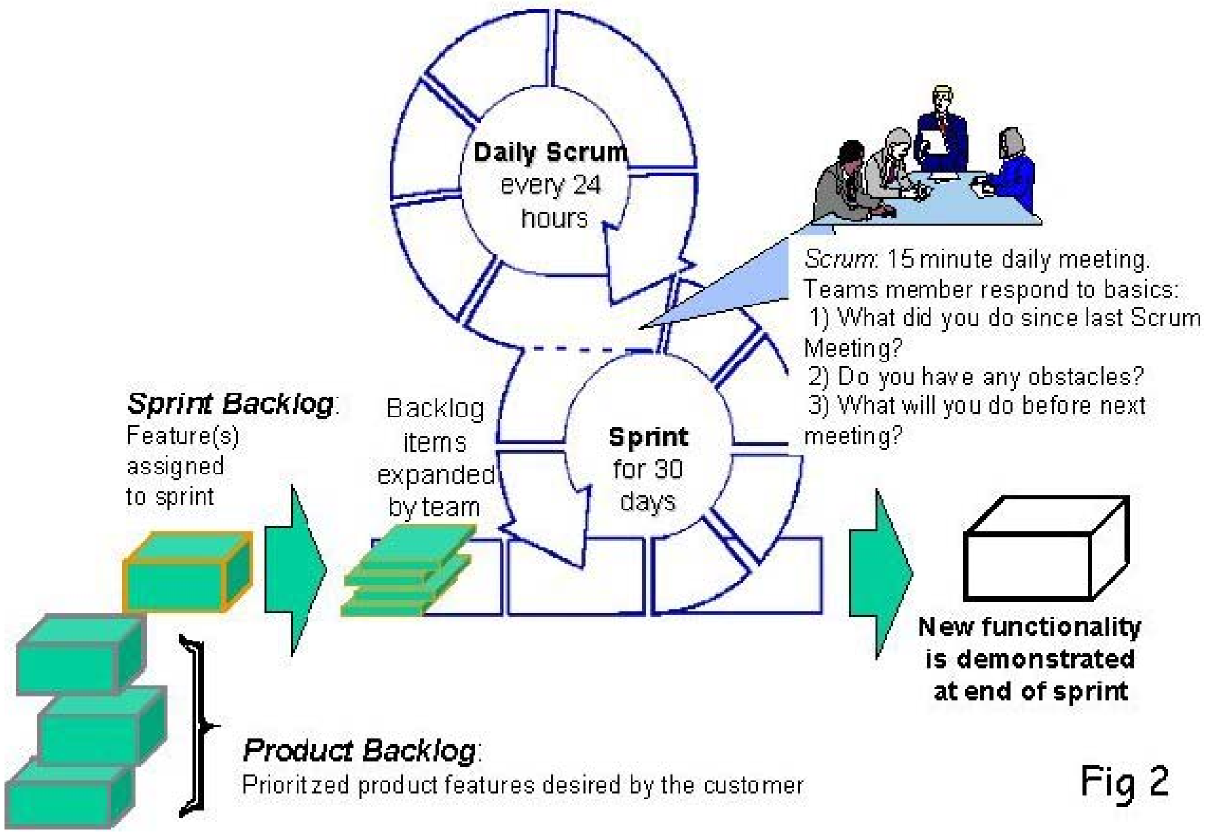

Daily Scrum every 24 hours
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog: Feature(s) assigned to sprint
Backlog items expanded by team
Sprint for 30 days
New functionality is demonstrated at end of sprint
Product Backlog: Prioritized product features desired by the customer
Fig 2

**Figure 10. Example Agile Process Flow: Scrum (Boehm and Turner 2004).** Reprinted with permission of Ken Schwaber. All other rights are reserved by the copyright owner.

### Architected Agile Methods

Over the last decade, several organizations have been able to scale up agile methods by using two layers of ten-person Scrum teams. This involves, among other things, having each Scrum team's daily meeting followed up by a daily meeting of the Scrum team leaders discussing up-front investments in evolving system architecture (Boehm et al. 2010). Figure 11 shows an example of the Architected Agile approach.

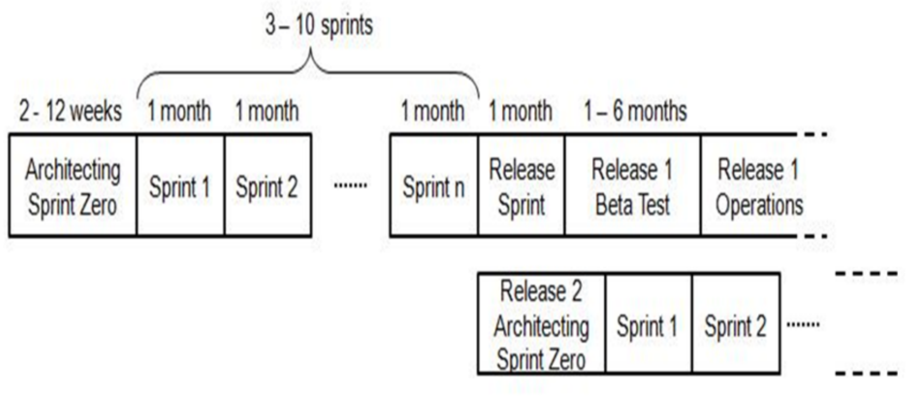


**Figure 11. Example of Architected Agile Process (Boehm 2009).** Reprinted with permission of Barry Boehm on behalf of USC-CSSE. All other rights are reserved by the copyright owner.

## Agile Practices and Principles

As seen with the Scrum and architected agile methods, "generally-shared" principles are not necessarily "uniformly followed". However, there are some general practices and principles shared by most agile methods:

- The project team understands, respects, works, and behaves within a defined SE process;
- The project is executed as fast as possible with minimum down time or staff diversion during the project and the critical path is managed;
- All key players are physically or electronically collocated, and "notebooks" are considered team property available to all;
- Baseline management and change control are achieved by formal, oral agreements based on "make a promise—keep a promise" discipline. Participants hold each other accountable;
- Opportunity exploration and risk reduction are accomplished by expert consultation and rapid model verification coupled with close customer collaboration;
- Software development is done in a rapid development environment while hardware is developed in a multi-disciplined model shop; and
- A culture of constructive confrontation pervades the project organization. The team takes ownership for success; it is never "someone else's responsibility."

Agile development principles (adapted for SE) are as follows (adapted from *Principles behind the Agile Manifesto* (Beedle et al. 2009)):

1. First, satisfy the customer through early and continuous delivery of valuable software (and other system elements).
2. Welcome changing requirements, even late in development; agile processes harness change for the customer's competitive advantage.

3. Deliver working software (and other system elements) frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business personnel and developers must work together daily throughout the project.
5. Build projects around motivated individuals; give them the environment, support their needs, and trust them to get the job done.
6. The most efficient and effective method of conveying information is face-to-face conversation.
7. Working software (and other system elements) is the primary measure of progress.
8. Agile processes promote sustainable development; the sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.

A team should reflect on how to become more effective at regular intervals and then tune and adjust its behavior accordingly. This self-reflection is a critical aspect for projects that implement agile processes.

## Lean Systems Engineering and Development

### Origins

As the manufacturing of consumer products such as automobiles became more diversified, traditional pre-planned mass-production approaches had increasing problems with quality and adaptability. Lean manufacturing systems such as the Toyota Production System (TPS) (Ohno 1988) were much better suited to accommodate diversity, to improve quality, and to support just-in-time manufacturing that could rapidly adapt to changing demand patterns without having to carry large, expensive inventories.

Much of this transformation was stimulated by the work of W. Edwards Deming, whose Total Quality Management (TQM) approach shifted responsibility for quality and productivity from planners and inspectors to the production workers who were closer to the real processes (Deming 1982). Deming's approach involved everyone in the manufacturing organization in seeking continuous process improvement, or "Kaizen".

Some of the TQM techniques, such as statistical process control and repeatability, were more suited to repetitive manufacturing processes than to knowledge work such as systems engineering (SE) and software engineering (SwE). Others, such as early error elimination, waste elimination, workflow stabilization, and Kaizen, were equally applicable to knowledge work. Led by Watts Humphrey, TQM became the focus for the Software Capability Maturity Model (Humphrey 1987; Paulk et al. 1994) and the CMM-Integrated or CMMI, which extended its scope to include systems engineering (Chrissis et al. 2003). One significant change was the redefinition of Maturity Level 2 from "Repeatable" to "Managed".

The Massachusetts Institute of Technology (MIT) conducted studies of the TPS, which produced a similar approach that was called the "Lean Production System" (Krafcik 1988; Womack et al. 1990). Subsequent development of "lean thinking" and related work at MIT led to the Air Force-sponsored Lean Aerospace Initiative (now called the Lean Advancement Initiative), which applied lean thinking to SE (Murman 2003, Womack-Jones 2003). Concurrently, lean ideas were used to strengthen the scalability and dependability aspects of agile methods for software (Poppendieck 2003; Larman-Vodde 2009). The Kanban flow-oriented approach has been successfully applied to software development (Anderson 2010).

**Principles**

Each of these efforts has developed a similar but different set of Lean principles. For systems engineering, the current best source is *Lean for Systems Engineering*, the product of several years' work by the INCOSE Lean SE working group (Oppenheim 2011). It is organized into six principles, each of which is elaborated into a set of lean enabler and sub-enabler patterns for satisfying the principle:

1. **Value.** Guide the project by determining the value propositions of the customers and other key stakeholders. Keep them involved and manage changes in their value propositions.
2. **Map the Value Stream (Plan the Program).** This includes thorough requirements specification, the concurrent exploration of trade spaces among the value propositions, COTS evaluation, and technology maturity assessment, resulting in a full project plan and set of requirements.
3. **Flow.** Focus on the project's critical path activities to avoid expensive work stoppages, including coordination with external suppliers.
4. **Pull.** Pull the next tasks to be done based on prioritized needs and dependencies. If a need for the task can't be found, reject it as waste.
5. **Perfection.** Apply continuous process improvement to approach perfection. Drive defects out early to get the system Right The First #**Time,** vs. fixing them during inspection and test. Find and fix root causes rather than symptoms.
6. **Respect for People.** Flow down responsibility, authority, and accountability to all personnel. Nurture a learning environment. Treat people as the organization's most valued assets. (Oppenheim 2011)

These lean SE principles are highly similar to the four underlying incremental commitment spiral model principles.

- **Principle 1: Stakeholder value-based system definition and evolution**, addresses the lean SE principles of value, value stream mapping, and respect for people (developers are success-critical stakeholders in the ICSM).
- **Principle 2: Incremental commitment and accountability**, partly addresses the pull principle, and also addresses respect for people (who are accountable for their commitments).
- **Principle 3: Concurrent system and software definition and development**, partly addresses both value stream mapping and flow.
- **Principle 4: Evidence and risk-based decision making**, uses evidence of achievability as its measure of success. Overall, the ICSM principles are somewhat light on continuous process improvement, and the lean SE principles are somewhat insensitive to requirements emergence in advocating a full pre-specified project plan and set of requirements.

See Lean Engineering for more information.

## References

### Works Cited

Agile Alliance. 2001. "Manifesto for Agile Software Development." http://agilemanifesto.org/.

Anderson, D. 2010. *Kanban*, Sequim, WA: Blue Hole Press.

Boehm, B. 1996. "Anchoring the Software Process." IEEE *Software* 13(4): 73-82.

Boehm, B. and J. Lane. 2007. "Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering." *CrossTalk.* 20(10) (October 2007): 4-9.

Boehm, B., J. Lane, S. Koolmanjwong, and R. Turner. 2010. "Architected Agile Solutions for Software-Reliant Systems," in Dingsoyr, T., T. Dyba., and N. Moe (eds.), *Agile Software Development: Current Research and Future Directions.* New York, NY, USA: Springer.

Boehm, B. and R. Turner. 2004. *Balancing Agility and Discipline.* New York, NY, USA: Addison-Wesley.

Castellano, D.R. 2004. “Top Five Quality Software Projects.” *CrossTalk.* 17(7) (July 2004): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2004/200407/200407-0-Issue.pdf.

Chrissis, M., M. Konrad, and S. Shrum. 2003. *CMMI: Guidelines for Process Integration and Product Improvement.* New York, NY, USA, Addison Wesley.

Deming, W.E. 1982. *Out of the Crisis.* Cambridge, MA, USA: MIT.

Fairley, R. 2009. *Managing and Leading Software Projects.* New York, NY, USA: John Wiley & Sons.

Forsberg, K. 1995. "If I Could Do That, Then I Could…’ System Engineering in a Research and Development Environment." Proceedings of the Fifth International Council on Systems Engineering (INCOSE) International Symposium. 22-26 July 1995. St Louis, MO, USA.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management,* 3rd ed. New York, NY, USA: John Wiley & Sons.

Humphrey, W., 1987. “Characterizing the Software Process: A Maturity Framework.” Pittsburgh, PA, USA: CMU Software Engineering Institute. CMU/SEI-87-TR-11.

Jarzombek, J. 2003. “Top Five Quality Software Projects.” *CrossTalk.* 16(7) (July 2003): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2003/200307/200307-0-Issue.pdf.

Krafcik, J. 1988. "Triumph of the lean production system". *Sloan Management Review.* 30(1): 41–52.

Kruchten, P. 1999. *The Rational Unified Process*. New York, NY, USA: Addison Wesley.

Larman , C. and B. Vodde. 2009. *Scaling Lean and Agile Development.* New York, NY, USA: Addison Wesley.

Maranzano, J.F., S.A. Rozsypal, G.H. Zimmerman, G.W. Warnken, P.E. Wirth, D.M. Weiss. 2005. “Architecture Reviews: Practice and Experience.” IEEE *Software.* 22(2): 34-43.

Murman, E. 2003. *Lean Systems Engineering I, II, Lecture Notes*, MIT Course 16.885J, Fall. Cambridge, MA, USA: MIT.

Oppenheim, B. 2011. *Lean for Systems Engineering.* Hoboken, NJ: Wiley.

Paulk, M., C. Weber, B. Curtis, and M. Chrissis. 1994. *The Capability Maturity Model: Guidelines for Improving the Software Process.* Reading, MA, USA: Addison Wesley.

Pew, R. and A. Mavor (eds.). 2007. *Human-System Integration in The System Development Process: A New Look.* Washington, DC, USA: The National Academies Press.

Poppendieck, M. and T. Poppendieck. 2003. *Lean Software Development: An Agile Toolkit for Software Development Managers.* New York, NY, USA: Addison Wesley.

Spruill, N. 2002. “Top Five Quality Software Projects.” *CrossTalk.* 15(1) (January 2002): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2002/200201/200201-0-Issue.pdf.

Stauder, T. “Top Five Department of Defense Program Awards.” *CrossTalk.* 18(9) (September 2005): 4-13. Available at http://www.crosstalkonline.org/storage/issue-archives/2005/200509/200509-0-Issue.pdf.

Womack, J., D. Jones, and D Roos. 1990. *The Machine That Changed the World: The Story of Lean Production.* New York, NY, USA: Rawson Associates.

Womack, J. and D. Jones. 2003. *Lean Thinking*. New York, NY, USA: The Free Press.

## Primary References

Beedle, M., et al. 2009. "The Agile Manifesto: Principles behind the Agile Manifesto". in *The Agile Manifesto* [database online]. Accessed 2010. Available at: www.agilemanifesto.org/principles.html.

Boehm, B. and R. Turner. 2004. *Balancing Agility and Discipline.* New York, NY, USA: Addison-Wesley.

Fairley, R. 2009. *Managing and Leading Software Projects.* New York, NY, USA: J. Wiley & Sons.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management,* 3rd ed. New York, NY, USA: J. Wiley & Sons.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* Kings College, UK: College Publications.

Pew, R., and A. Mavor (eds.). 2007. *Human-System Integration in the System Development Process: A New Look.* Washington, DC, USA: The National Academies Press.

Royce, W.E. 1998. *Software Project Management: A Unified Framework*. New York, NY, USA: Addison Wesley.

## Additional References

Anderson, D. 2010. *Kanban*. Sequim, WA, USA: Blue Hole Press.

Baldwin, C. and K. Clark. 2000. *Design Rules: The Power of Modularity.* Cambridge, MA, USA: MIT Press.

Beck, K. 1999. *Extreme Programming Explained.* New York, NY, USA: Addison Wesley.

Beedle, M., et al. 2009. "The Agile Manifesto: Principles behind the Agile Manifesto" in The Agile Manifesto [database online]. Accessed 2010. Available at: www.agilemanifesto.org/principles.html

Biffl, S., A. Aurum, B. Boehm, H. Erdogmus, and P. Gruenbacher (eds.). 2005. *Value-Based Software Engineering*. New York, NY, USA: Springer.

Boehm, B. 1988. “A Spiral Model of Software Development.” IEEE *Computer.* 21(5): 61-72.

Boehm, B. 2006. “Some Future Trends and Implications for Systems and Software Engineering Processes.” *Systems Engineering.* 9(1): 1-19.

Boehm, B., A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy. 1998. “Using the WinWin Spiral Model: A Case Study.” IEEE *Computer.* 31(7): 33-44.

Boehm, B., J. Lane, S. Koolmanojwong, and R. Turner. 2013 (in press). *Embracing the Spiral Model: Creating Successful Systems with the Incremental Commitment Spiral Model.* New York, NY, USA: Addison Wesley.

Castellano, D.R. 2004. “Top Five Quality Software Projects.” *CrossTalk.* 17(7) (July 2004): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2004/200407/200407-0-Issue.pdf.

Checkland, P. 1981. *Systems Thinking, Systems Practice*. New York, NY, USA: Wiley.

Crosson, S. and B. Boehm. 2009. “Adjusting Software Life Cycle Anchorpoints: Lessons Learned in a System of Systems Context.” Proceedings of the Systems and Software Technology Conference, 20-23 April 2009, Salt Lake City, UT, USA.

Dingsoyr, T., T. Dyba. and N. Moe (eds.). 2010. "Agile Software Development: Current Research and Future Directions.” Chapter in B. Boehm, J. Lane, S. Koolmanjwong, and R. Turner, *Architected Agile Solutions for Software-Reliant Systems.* New York, NY, USA: Springer.

Dorner, D. 1996. *The Logic of Failure*. New York, NY, USA: Basic Books.

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

Forsberg, K. 1995. "'If I Could Do That, Then I Could…' System Engineering in a Research and Development Environment." Proceedings of the Fifth Annual International Council on Systems Engineering (INCOSE) International Symposium. 22-26 July 1995. St. Louis, MO, USA.

Forsberg, K. 2010. "Projects Don't Begin With Requirements." Proceedings of the IEEE Systems Conference. 5-8 April 2010. San Diego, CA, USA.

Gilb, T. 2005. *Competitive Engineering*. Maryland Heights, MO, USA: Elsevier Butterworth Heinemann.

Goldratt, E. 1984. *The Goal*. Great Barrington, MA, USA: North River Press.

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology.* New York, NY, USA: Wiley.

Holland, J. 1998. *Emergence*. New York, NY, USA: Perseus Books.

ISO/IEC. 2010. *Systems and Software Engineering, Part 1: Guide for Life Cycle Management.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 24748-1:2010.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes.* Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2015.

ISO/IEC. 2003. *Systems Engineering — A Guide for The Application of ISO/IEC 15288 System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 19760:2003 (E).

Jarzombek, J. 2003. "Top Five Quality Software Projects." *CrossTalk.* 16(7) (July 2003): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2003/200307/200307-0-Issue.pdf.

Kruchten, P. 1999. *The Rational Unified Process.* New York, NY, USA: Addison Wesley.

Landis, T. R. 2010. *Lockheed Blackbird Family (A-12, YF-12, D-21/M-21 & SR-71).* North Branch, MN, USA: Specialty Press.

Madachy, R. 2008. *Software Process Dynamics*. New York, NY, USA: Wiley.

Maranzano, J., et al. 2005. "Architecture Reviews: Practice and Experience." IEEE *Software.* 22(2): 34-43.

National Research Council of the National Academies (USA). 2008. *Pre-Milestone A and Early-Phase Systems Engineering*. Washington, DC, USA: The National Academies Press.

Osterweil, L. 1987. "Software Processes are Software Too." Proceedings of the SEFM 2011: 9th International Conference on Software Engineering. Monterey, CA, USA.

Poppendeick, M. and T. Poppendeick. 2003. *Lean Software Development: an Agile Toolkit.* New York, NY, USA: Addison Wesley.

Rechtin, E. 1991. *System Architecting: Creating and Building Complex Systems.* Upper Saddle River, NY, USA: Prentice-Hall.

Rechtin, E., and M. Maier. 1997. *The Art of System Architecting*. Boca Raton, FL, USA: CRC Press.

Schwaber, K. and M. Beedle. 2002. *Agile Software Development with Scrum*. Upper Saddle River, NY, USA: Prentice Hall.

Spruill, N. 2002. "Top Five Quality Software Projects." *CrossTalk.* 15(1) (January 2002): 4-19. Available at: http://www.crosstalkonline.org/storage/issue-archives/2002/200201/200201-0-Issue.pdf.

Stauder, T. 2005. "Top Five Department of Defense Program Awards." *CrossTalk.* 18(9) (September 2005): 4-13. Available at http://www.crosstalkonline.org/storage/issue-archives/2005/200509/200509-0-Issue.pdf.

Warfield, J. 1976. *Societal Systems: Planning, Policy, and Complexity*. New York, NY, USA: Wiley.

Womack, J. and D. Jones. 1996. *Lean Thinking.* New York, NY, USA: Simon and Schuster.

**SEBoK v. 2.1, released 31 October 2019**

# Integration of Process and Product Models

***Lead Authors:*** *Kevin Forsberg, Bud Lawson*

When performing systems engineering activities, it is important to consider the mutual relationship between processes and the desired system. The type of system (see Types of Systems) being produced will affect the needed processes, as indicated in system life cycle process drivers and choices. This may cause the tailoring of defined processes as described in application of systems engineering standards.

## Process and Product Models

Figure 1 of life cycle models introduced the perspective of viewing stage work products provided by process execution as versions of a system-of-interest (SoI) at various life stages. The fundamental changes that take place during the life cycle of any man-made system include definition, production, and utilization. When building upon these, it is useful to consider the structure of a generic process and product life cycle stage model as portrayed in Figure 1 below.

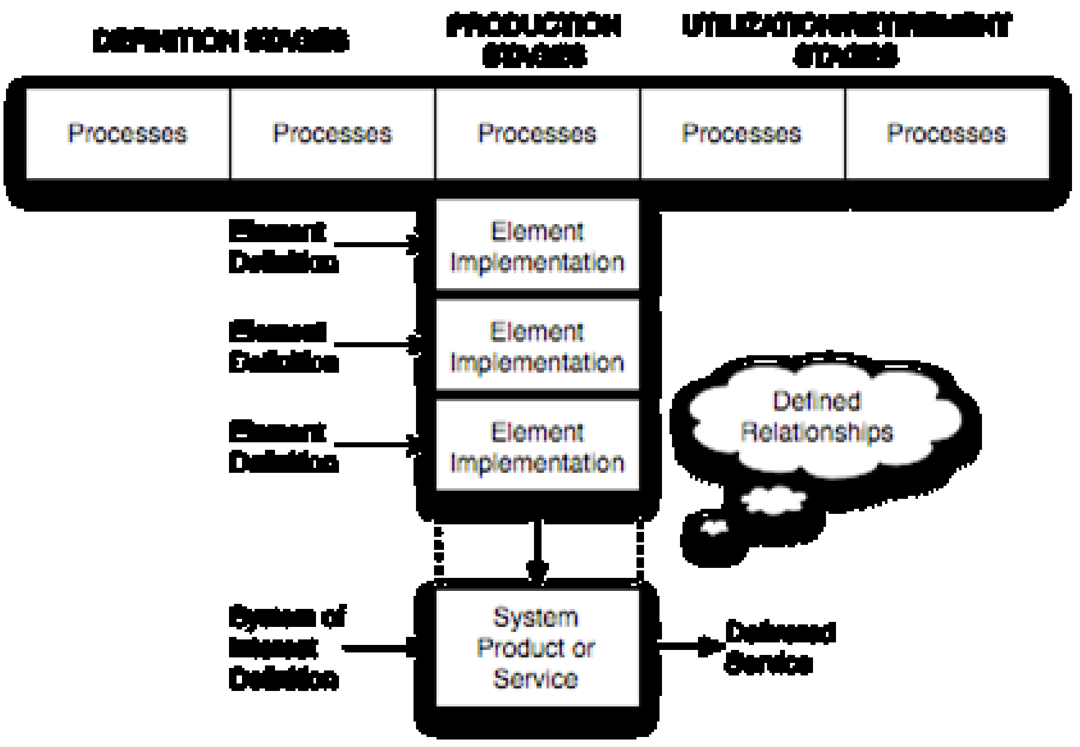


**Figure 1. Generic (T) Stage Structure of System Life Cycle (Lawson 2010).** Reprinted with permission of Harold "Bud" Lawson. All other rights are reserved by the copyright owner.

The (T) model indicates that a definition stage precedes a production stage where the implementation (acquisition, provisioning, or development) of two or more system elements has been accomplished. The system elements are integrated according to defined relationships into the SoI. Thus, both the process and product aspects are portrayed. The implementation and integration processes are followed in providing the primary stage results—namely, in assembled system product or service instances. However, as noted in life cycle models, the definition of the SoI when provided in a development stage can also be the result of first versions of the system. For example, a prototype, which may be viewed as a form of production or pre-production stage. Following the production stage is a utilization stage. Further relevant stages can include support and retirement. Note that this model also displays the important distinction between definition versus implementation and integration.

According to ISO/IEC/IEEE 15288 (2015), this structure is generic for any type of man-made SoI to undergo life cycle management. The production stage thus becomes the focal point of the (T) model at which system elements are implemented and integrated into system product or service instances based upon the definitions. For defined physical systems, this is the point at which product instances are manufactured and assembled (singularly or mass-produced). For non-physical systems, the implementation and integration processes are used in service preparation (establishment) prior to being instantiated to provide a service. For software systems, this is the point at which builds that combine software elements into versions, releases, or some other form of managed software product are produced.

Using recursive decomposition, the implementation of each system element can involve the invocation of the standard again at the next lowest level, thus treating the system element as a SoI in its own right. A new life cycle structure is then utilized for the lower level SoIs.

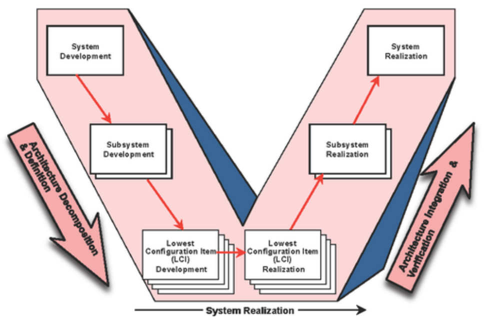
This is illustrated in the Dual Vee model (Figures 2a and 2b). The Dual Vee model is a three-dimensional system development model that integrates product and process in the creation of the system and component architectures. It emphasizes

- concurrent opportunity and risk management;
- user in-process validation;
- integration, verification, and validation planning; and
- verification problem resolution.

When decomposition terminates according to the practical need and risk-benefit analysis, system elements are then implemented (acquired, provisioned, or developed) according to the type of element involved.

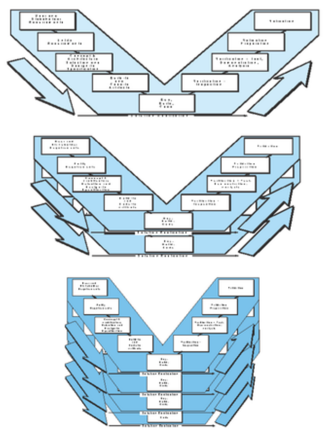
## Dual Vee Model

### Architecture Vee
for architecture management


System Development
Subsystem Development
Lowest Configuration Item (LCI) Development
Lowest Configuration Item (LCI) Realization
Subsystem Realization
System Realization
Architecture Decomposition & Definition
Architecture Integration & Verification
System Realization

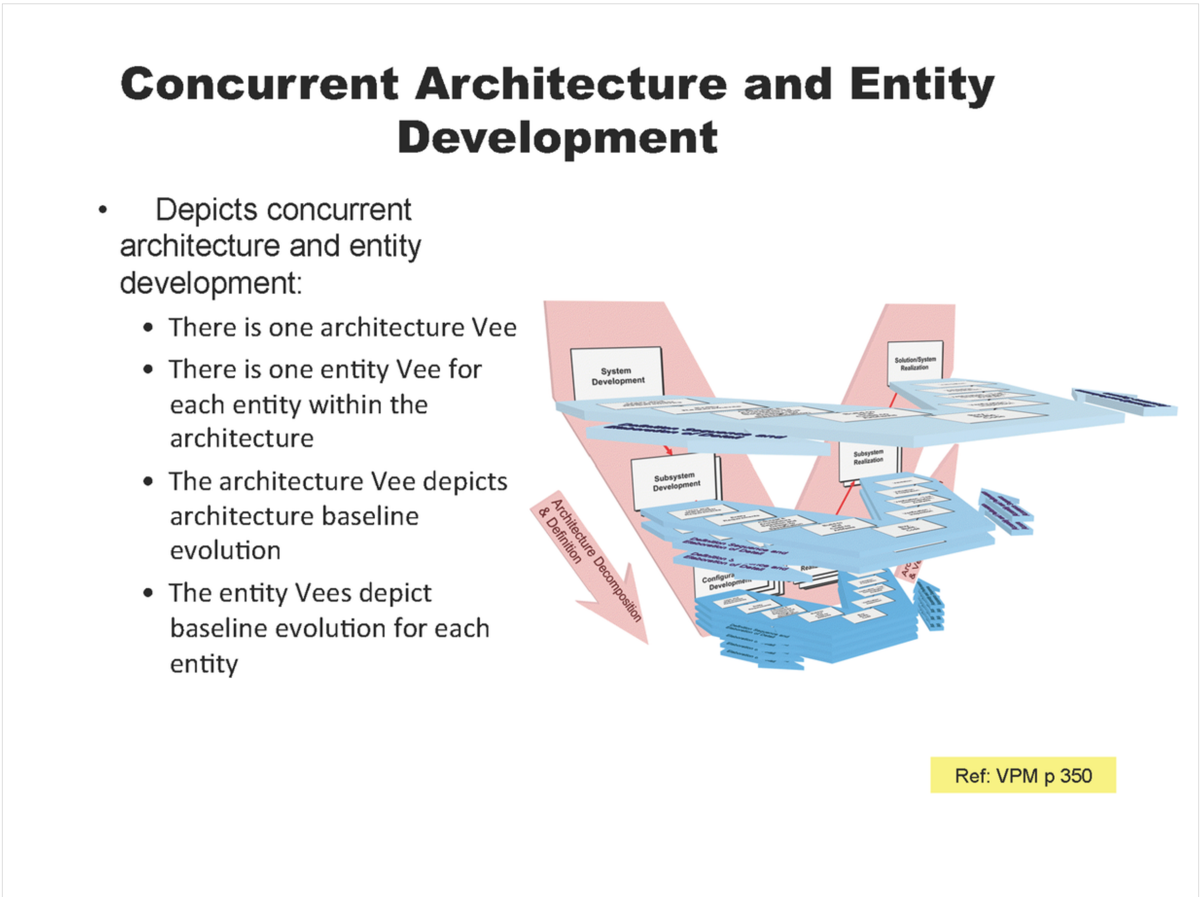
Depicts architecture baseline evolution. Vertical dimension is architecture decomposition. Horizontal dimension is system realization. Third dimension normal to the image is quantity of entities and their interfaces.

### Entity Vee
for entity management



An entity is any item of the architecture. The Vee depicts entity baseline elaboration. Vertical dimension is entity detail. Horizontal dimension is entity realization.

**Figure 2a. The Dual Vee Model (2a) (Forsberg, Mooz, Cotterman 2005).** Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.



## Concurrent Architecture and Entity Development

- Depicts concurrent architecture and entity development:
  - There is one architecture Vee
  - There is one entity Vee for each entity within the architecture
  - The architecture Vee depicts architecture baseline evolution
  - The entity Vees depict baseline evolution for each entity

Ref: VPM p 350

System Development
Solution/System Realization
Subsystem Development
Subsystem Realization
Architecture Decomposition & Definition

**Figure 2b. The Dual Vee Model (2b) (Forsberg, Mooz, Cotterman 2005).** Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

A practical aspect that can impact the process and product aspect is the decision to use off-the-shelf elements in commercial-off-the-shelf (COTS) form. In this case, further decomposition of the element is not necessary. The use of COTS elements (and their internally created neighbor or non-development item (NDI)) has become widespread, and they have proven their value. However, developers must make sure that the COTS product is appropriate for their environment.

A known flaw which occurs infrequently in normal use of the product in its intended environment may be benign and easily dealt with. In a new situation, it could have dramatic adverse consequences, such as those that occurred on the USS Yorktown Cruiser in 1998 (Wired News Contributors 1998). The customer mandated that Windows NT be used as the primary operating system for the ship. A *divide by zero* fault caused the operating system to fail, and the ship was dead in the water. It had to be towed back to port on three occasions.

Spiral models concurrently engineer not only process and product models, but also property and success models. Figure 3 shows how these models provide checks and balances, both at milestone reviews and as individual model choices are made. Methods and tools supporting this concurrent engineering are provided in “When Models Collide: Lessons from Software System Analysis” (Boehm and Port 1999), “Avoiding the Software Model-Clash Spiderweb” (Boehm, Port, and Al-Said 2000), and “Detecting Model Clashes During Software Systems Development” (Al-Said 2003).

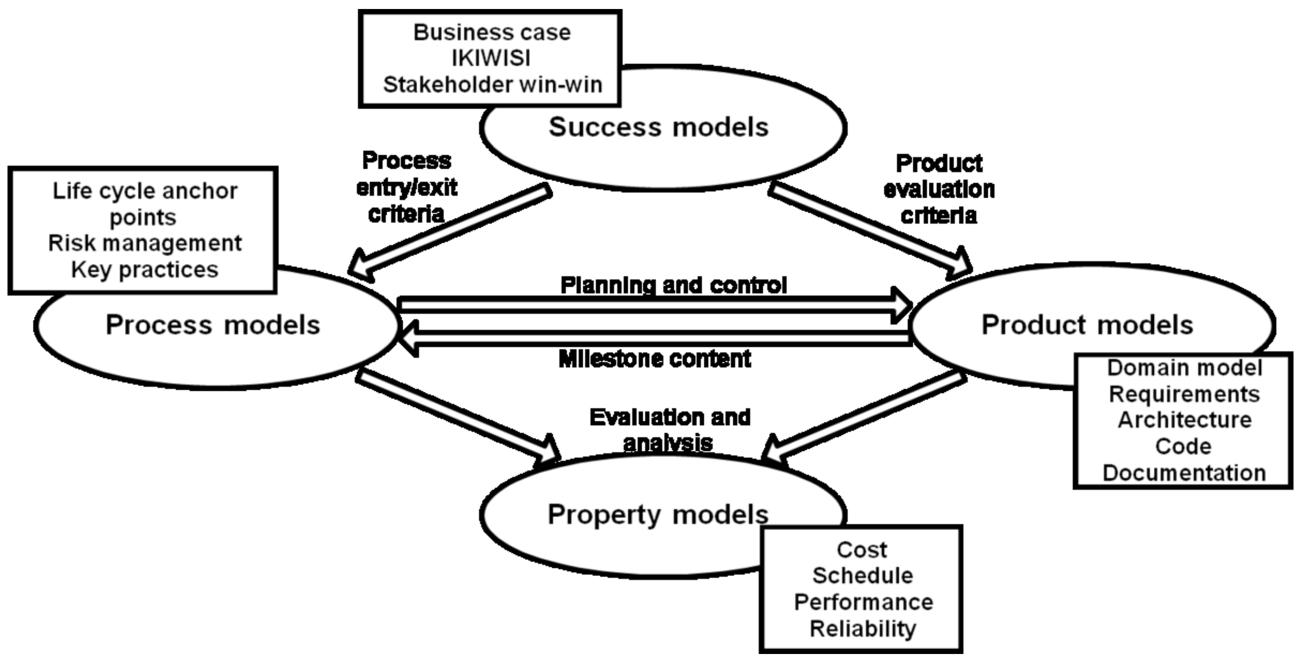


**Figure 3. Spiral Model Support for Process Models, Product Models, Success Models, Property Models (Boehm and Port 1999).** Reprinted with permission of © Copyright IEEE – All rights reserved. All other rights are reserved by the copyright owner.

For software systems, entry into the production stages is the point at which builds that combine software elements (code modules) into versions, releases, or some other form of managed software product are created. Thus, the major difference between systems in general and software systems is the slight variant of the generic model as presented in Figure 4.

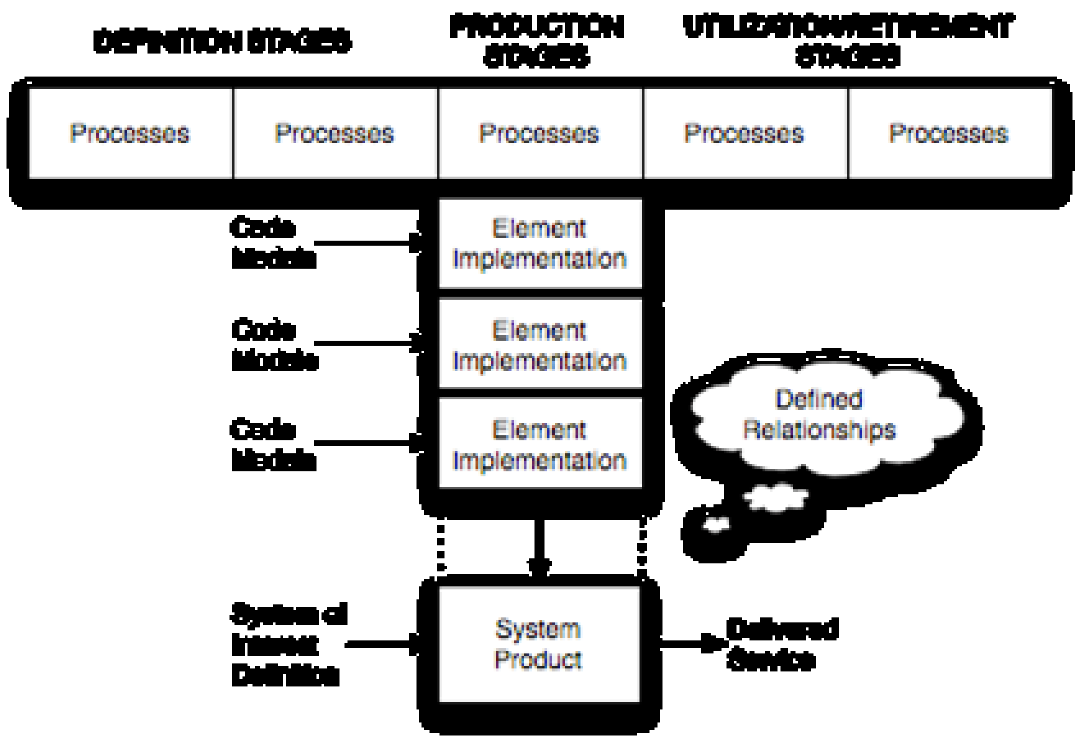


**Figure 4. T-Model for Software System (Lawson 2010).** Reprinted with permission of Harold "Bud" Lawson. All other rights are reserved by the copyright owner.

## Stage Execution Order

A sequential execution of life cycle stages is the most straightforward. As presented in System Life Cycle Process Models: Vee and System Life Cycle Process Models: Iterative, variants of the Vee model and the spiral model provide non-sequential models when practical considerations require a non-linear execution of life cycle stages. Building upon these two models, it is important to note that various types of complex systems require that the stages of the life cycle model be revisited as insight (knowledge) is gained, as well as when stakeholder requirements change. The iterations may involve necessary changes in the processes and in the product or service system. Thus, within the context of the (T) stage model, various orderings of stage execution - reflecting forms of non-sequential stage ordering - can be conveniently described, as portrayed in Figure 5.

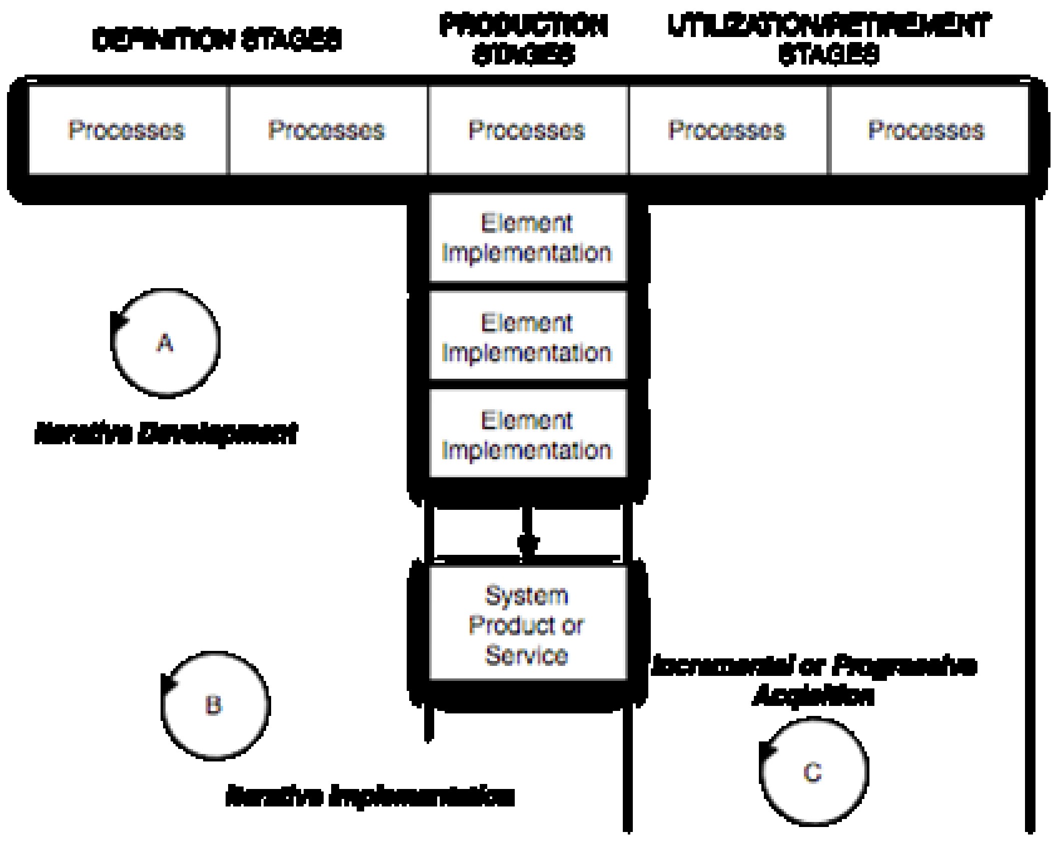


**Figure 5. Iteration Through Life Cycle Stages (Lawson 2010).** Reprinted with permission of Harold "Bud" Lawson. All other rights are reserved by the copyright owner.

Each pattern of stage execution involves iteration of the previous stages, perhaps with altered requirements for the processes or the system. The heavy lines in Figure 5 denote the demarcation of the revisited end points. Three are iterative forms, for which several variants can be extracted:

1. **Iterative development** is quite frequently deployed in order to assess stakeholder requirements, analyze the requirements, and develop a viable architectural design. Thus, it is typical that the concept stage may be revisited during the development stage. For systems where products are based upon physical structures (electronics, mechanics, chemicals, and so on), iteration after production has begun can involve significant costs and schedule delays. It is, therefore, important to get it *"right"* before going to production. The early stages are thus used to build confidence (verify and validate) that the solution works properly and will meet the needs of the stakeholders. Naturally, such an approach could be used for software and human activity systems as well; however, due to their soft nature, it can be useful to go further by experimenting and evaluating various configurations of the system.

2. **Iterative development and implementation** involves producing (defining, implementing, and integrating) various versions of the system, evaluating how well they meet stakeholder requirements, perhaps in the context of changing requirements, and then revisiting the concept and/or development stages. Such iterations are typical within software system development, where the cost of production is not as significant as for defined physical systems. A variant of this approach is the spiral model, where successive iterations fill in more detail (Boehm and May 1998). The use of this approach requires careful attention to issues related to baseline and configuration management. In this approach, significant verification (testing) should be performed on software systems in order to build confidence that the system delivered will meet stakeholder requirements.
3. **Incremental or progressive acquisition** involves releasing systems in the form of products and/or services to the consumers. This approach is appropriate when structural and capability (functions) changes are anticipated in a controlled manner after deployment. The use of this approach can be due to not knowing all of the requirements at the beginning, which leads to progressive acquisition/deployment, or due to a decision to handle the complexity of the system and its utilization in increments—namely, incremental acquisition. These approaches are vital for complex systems in which software is a significant system element. Each increment involves revisiting the definition and production stages. The utilization of these approaches must be based upon well-defined, agreed relationships between the supplying and acquiring enterprises. In fact, the iteration associated with each resulting product and/or service instance may well be viewed as a joint project, with actor roles being provided by both enterprises.

In all of the approaches it is wise to use modeling and simulation techniques and related tools to assist in understanding the effect of changes made in the complex systems being life cycle managed. These techniques are typically deployed in the earlier stages; however, they can be used in gaining insight into the potential problems and opportunities associated with the latter stages of utilization and maintenance (for example, in understanding the required logistics and help-desk aspects).

## Allocating and Meeting Requirements - Integration of Process and Product Models

Regardless of the order in which life cycle stages are executed, stakeholder requirements for the system, including changed requirements in each iteration, must be allocated into appropriate activities of the processes used in projects for various stages as well as to the properties of the elements of the product system or service system and their defined relationships. This distribution was illustrated in the fourth variant of Lawson’s T-model as presented in System Life Cycle Process Models: Iterative and System Life Cycle Process Models: Vee.

Ideally, the project management team should implement proven processes that will integrate the technical process models with the project management product models to manage any of the processes discussed earlier, including incremental and evolutionary development. The processes shown are the project management flow, starting with the beginning of the development phase (Forsberg, Mooz, and Cotterman 2005, 201).

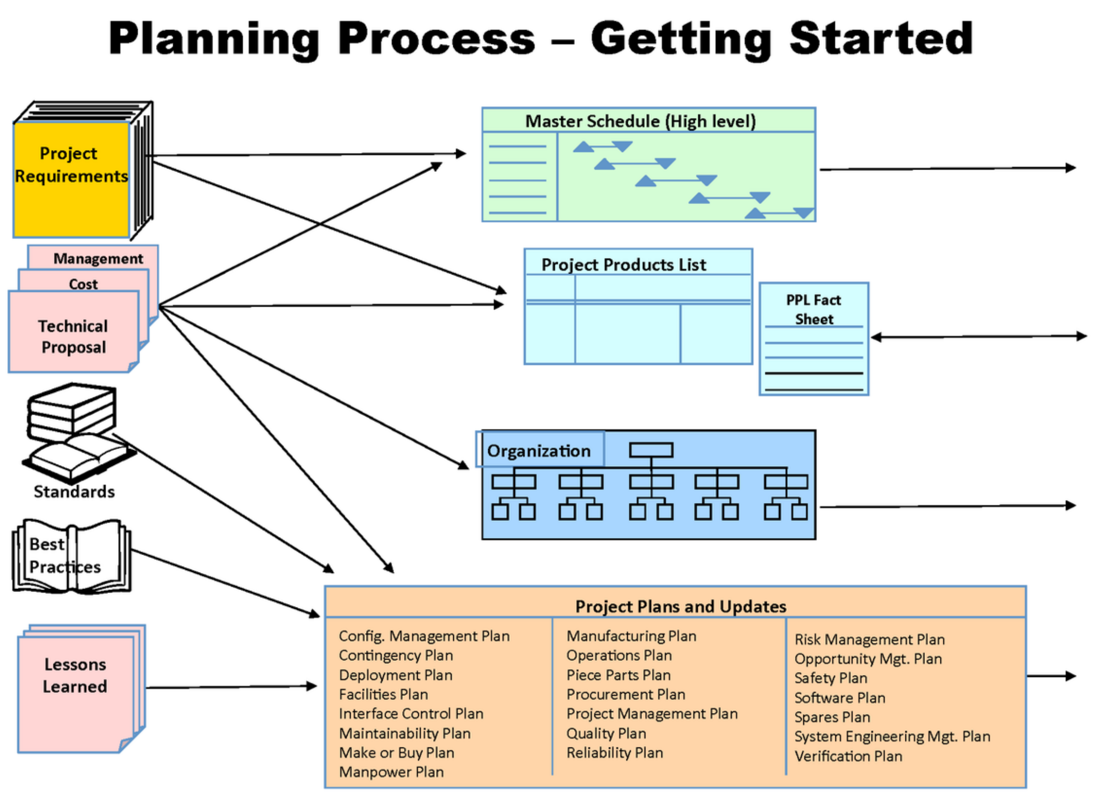


**Figure 6a. New Product Planning Process – Getting Started (Forsberg, Mooz, and Cotterman 2005).** Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

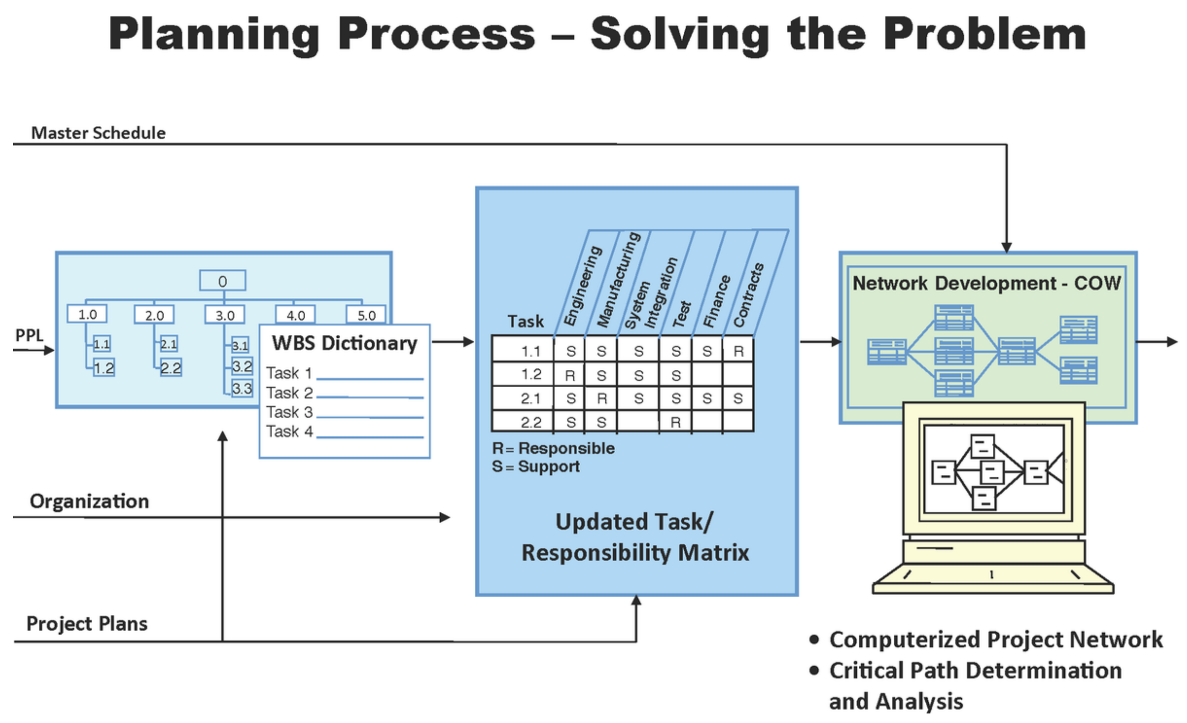


**Figure 6b. New Product Planning Process Solving the Problem (Forsberg, Mooz, and Cotterman 2005).** Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

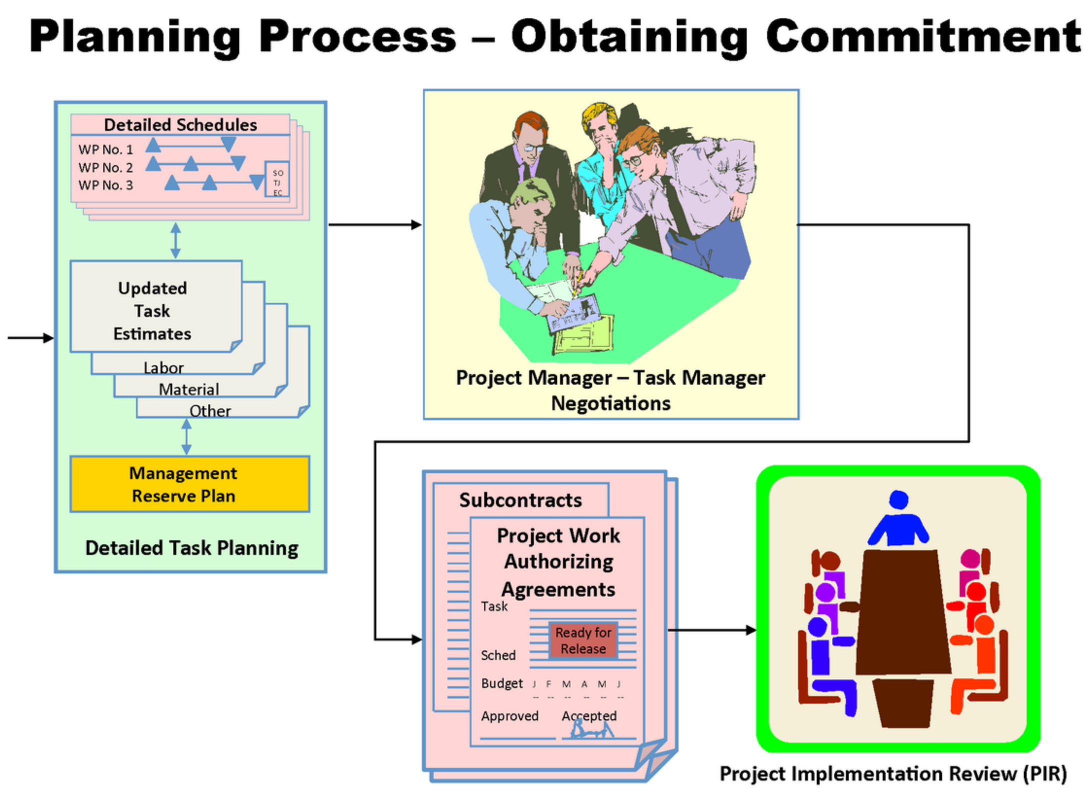


**Figure 6c. New Product Planning Process – Getting Commitment (Forsberg, Mooz, and Cotterman 2005).** Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

## References

### Works Cited

Boehm, B. and W. May. 1988. "A Spiral Model of Software Development and Enhancement." IEEE *Computer* 21(5): 61-72.

Boehm, B. and D. Port. 1999. "When Models Collide: Lessons From Software System Analysis." *IT Professional* 1(1): 49-56.

Boehm, B., J. Lane, S. Koolmanojwong, and R. Turner (forthcoming). *Embracing the Spiral Model: Creating Successful Systems with the Incremental Commitment Spiral Model.* New York, NY, USA: Addison Wesley.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management.* 3rd ed. New York, NY, USA: J. Wiley & Sons.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering-- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions.ISO/IEC/IEEE 15288:2015

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

Wired News Contributors. 2011. "Sunk by Windows NT," *Wired News*, last modified July 24, 1998. Accessed on September 11, 2011. Available at http://www.wired.com/science/discoveries/news/1998/07/13987.

## Primary References

Boehm, B. and W. May. 1988. “A Spiral Model of Software Development and Enhancement.” IEEE *Computer.* 21(5): 61-72.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management,* 3rd ed. New York, NY, USA: John Wiley & Sons.

Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, UK: College Publications.

## Additional References

Al-Said, M. 2003. "Detecting Model Clashes During Software Systems Development." PhD Diss. Department of Computer Science, University of Southern California, December 2003.

Boehm, B., J. Lane, S. Koolmanojwong, and R. Turner. (forthcoming). *Embracing the Spiral Model: Creating Successful Systems with the Incremental Commitment Spiral Model.* New York, NY, USA: Addison Wesley.

Boehm, B. and D. Port. 1999. "Escaping the Software Tar Pit: Model Clashes and How to Avoid Them." *ACM Software Engineering Notes.* (January, 1999): p. 36-48.

Boehm, B. and D. Port. 1999. "When Models Collide: Lessons From Software System Analysis." *IT Professional.* 1(1): 49-56.

Boehm, B., D. Port, and M. Al-Said. 2000. "Avoiding the Software Model-Clash Spiderweb." IEEE *Computer.* 33(11): 120-122.

Lawson, H. and M. Persson. 2010. “Portraying Aspects of System Life Cycle Models.” Proceedings of the European Systems Engineering Conference (EuSEC). 23-26 May 2010. Stockholm, Sweden.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Lean Engineering

Lean Systems Engineering (LSE) is the application of lean thinking (Womack 2003) to systems engineering (SE) and related aspects of enterprise and project management. LSE is an approach that is applicable throughout the system life cycle. The goal of LSE is to deliver the best life-cycle value for technically complex systems with minimal waste. Lean engineering is relevant to all of the traditional SE technical processes (see concept definition, system definition, system realization, system deployment and use, etc.). Lean engineering also interacts with and utilizes many of the specialty engineering disciplines discussed in Part 6.

## Lean Systems Engineering

SE is an established, sound practice, but not always delivered effectively. Most programs are burdened with some form of waste such as: poor coordination, unstable requirements, quality problems, delays, rework, or management frustration. Recent U.S. Government Accountability Office (GAO), National Aeronautics and Space Association (NASA), and Massachusetts Institute of Technology (MIT) studies of government programs document major budget and schedule overruns and a significant amount of waste in government programs - some reaching seventy percent of charged time. This waste represents a productivity reserve in programs and major opportunities to improve program efficiency.

LSE is the application of lean thinking to systems engineering and related aspects of enterprise and project management. SE is focused on the discipline that enables development of complex technical systems. Lean thinking is a holistic paradigm that focuses on delivering maximum value to the customer and minimizing wasteful practices. It has been successfully applied in manufacturing, aircraft depots, administration, supply chain management, healthcare, and product development, which includes engineering. LSE is the area of synergy between lean thinking and SE, which aims to deliver the best life-cycle value for technically complex systems with minimal waste. LSE does not mean less SE. It means more and better SE with higher responsibility, authority, and accountability (RAA), leading to better, waste-free workflow with increased mission assurance. Under the LSE philosophy, mission assurance is non-negotiable and any task which is legitimately required for success must be included; however, it should be well-planned and executed with minimal waste.

### Lean Principles

Oppenheim (2011) describes the six lean principles for product development (PD) as follows:

1. **Capture the *value* defined by the customer.** One cannot over-emphasize the importance of capturing task or program value (requirements, CONOPS, etc.) with precision, clarity, and completeness before resource expenditures ramp up to avoid unnecessary rework.
2. ***Map the value stream (plan the program)* and eliminate waste.** Map all end-to-end linked tasks, control/decision nodes, and the interconnecting information flows necessary to realize customer value. During the mapping process, eliminate all non-value-added activities, and enable the remaining activities to flow (without rework, backflow or stopping). The term *information flow* refers to the packets of information (knowledge) created by different tasks and flowing to other tasks for subsequent value adding, such as: design, analysis, test, review, decision, or integration. Each task adds value if it increases the level of useful information and reduces risk in the context of delivering customer value.
3. ***Flow* the work through planned and streamlined value-adding steps and processes, without stopping or idle time, unplanned rework, or backflow.** To optimize flow, one should plan for maximum concurrency of tasks, up to near capacity of an enterprise. Legitimate engineering iterations are frequently needed in PD, but they tend to be time consuming and expensive if they extend across disciplines. Lean PD encourages efficient methodology of *fail early - fail often* through rapid architecting and discovery techniques during early design

phases. Lean flow also makes every effort to use techniques that prevent lengthy iterations, such as design frontloading, trade space explorations, set designs, modular designs, legacy knowledge, and large margins. Where detailed cross-functional iterations are indeed necessary, lean flow optimizes iteration loops for overall value.

4. **Let customers *pull* value.** In PD, the pull principle has two important meanings: (1) the inclusion of any task in a program must be justified by a specific need from an internal or external customer and coordinated with them, and (2) the task should be completed when the customer needs the output. Excessively early completion leads to “shelf life obsolescence” including possible loss of human memory or changed requirements and late completion leads to schedule slips. This is the reason that every task owner or engineer needs to be in close communication with their internal customers to fully understand their needs and expectations and to coordinate their work.
5. **Pursue *perfection* of all processes.** Global competition requires continuous improvements of processes and products. Yet, no organization can afford to spend resources improving everything all the time. Systems engineers must make a distinction between processes and process outputs. Perfecting and refining the *work output* in a given task must be bounded by the overall value proposition (system or mission success, program budget and schedule) which define when an output is "good enough." In contrast, engineering and other *processes* must be continuously improved for competitive reasons.
6. ***Respect* for people.** A lean enterprise is an organization that recognizes that its people are the most important resource. In a lean enterprise, people are not afraid to identify problems and imperfections honestly and openly in real time, brainstorm about root causes and corrective actions without fear, or plan effective solutions together by consensus to prevent a problem from occurring again.

## Lean Enablers for Systems

In 2009, the International Council on Systems Engineering's (INCOSE's) Lean SE Working Group (LSE WG) released an online product entitled *Lean Enablers for Systems Engineering (LEfSE)*. It is a collection of practices and recommendations formulated as “dos” and “don’ts” of SE, based on lean thinking. The practices cover a large spectrum of SE and other relevant enterprise management practices, with a general focus on improving the program value and stakeholder satisfaction and reduce waste, delays, cost overruns, and frustrations. LEfSE are grouped under the six lean principles outlined above. The LEfSE are not intended to become a mandatory practice but should be used as a checklist of good practices. LEfSE do not replace the traditional SE; instead, they amend it with lean thinking.

LEfSE were developed by fourteen experienced INCOSE practitioners, some recognized leaders in lean and SE from industry, academia, and governments (such as the U.S., United Kingdom, and Israel), with cooperation from the 160-member international LSE WG. They collected best practices from the many companies, added collective tacit knowledge, wisdom, and experience of the LSE WG members, and inserted best practices from lean research and literature. The product has been evaluated by surveys and comparisons with the recent programmatic recommendations by GAO and NASA.

Oppenheim (2011) includes a comprehensive explanation of the enablers, as well as the history of LSE, the development process of LEfSE, industrial examples, and other material. Oppenheim, Murman, and Secor (2011) provide a scholarly article about LEfSE. A short summary was also published by Oppenheim in 2009.

## References

### Works Cited

Lean Systems Engineering Working Group. 2009. "Lean Enablers for Systems Engineering." Accessed 13 January 2016 at http:/ / www. lean-systems-engineering. org/ wp-content/ uploads/ 2012/ 07/ Lean-Enablers-for-SE-Version-1_03-.pdf.

Lean Systems Engineering Working Group. 2009. "Quick Reference Guide Lean Enablers for Systems Engineering." Accessed 13 January 2016 at http:/ / www. lean-systems-engineering. org/ wp-content/ uploads/ 2012/ 07/ LEfSE-Quick-Reference-Guide-8-pages.pdf.

Oppenheim, B.W. 2009. "Process Replication: Lean Enablers for Systems Engineering." *CrossTalk, The Journal of Defense Software Engineering*. July/August 2009.

Oppenheim, B.W. 2011. *Lean for Systems Engineering, with Lean Enablers for Systems Engineering*. Hoboken, NJ, USA: Wiley.

Oppenheim, B.W., E.M. Murman, and D. Secor. 2011. "Lean Enablers for Systems Engineering." *Journal of Systems Engineering*. 1(14).

Womack, J.P. 2003. *Lean Thinking*. Columbus, OH, USA: Free Press.

### Primary References

Lean Systems Engineering Working Group. 2009. "Lean Enablers for Systems Engineering." Accessed 13 January 2016 athttp:/ / www. lean-systems-engineering. org/ wp-content/ uploads/ 2012/ 07/ Lean-Enablers-for-SE-Version-1_03-.pdf.

Oppenheim, B., E. Murman, and D. Sekor. 2010. "Lean Enablers for Systems Engineering." Systems Engineering. 14(1). Accessed 13 January 2016. Available at http:/ / www. lean-systems-engineering. org/ wp-content/ uploads/ 2012/07/LEfSE-JSE-compressed.pdf.

### Additional References

Lean Enterprise Institute. 2009. "Principles of Lean." Accessed 1 March 2012 at http://www.lean.org/WhatsLean/ Principles.cfm.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Knowledge Area: Concept Definition

## Concept Definition

***Lead Author:*** *Garry Roedler*, ***Contributing Author:*** *Rick Adcock*

Concept Definition is the set of systems engineering (SE) activities in which the problem space and the needs and requirements of the business or enterprise and stakeholders are closely examined. The activities are grouped and described as generic processes which include Mission Analysis and Stakeholder Needs and Requirements. Concept Definition begins before any formal definition of the system-of-interest (SoI) is developed.

Mission Analysis focuses on the needs and requirements of business or enterprise — that is, on defining the problem or opportunity that exists (in what is often called the problem space or problem situation), as well as understanding the constraints on and boundaries of the selected system when it is fielded (in what is often called the solution space). The Stakeholder Needs and Requirements process explores and defines the operational aspects of a potential solution for the stakeholders from their point of view, independent of any specific solution. In these two Concept Definition activities, business or enterprise decision makers and other stakeholders describe *what* a solution should accomplish and *why* it is needed. Both *why* and *what* need to be answered before consideration is given to *how* the problem will be addressed (i.e., what type of solution will be implemented) and *how* the solution will be defined and developed.

If a new or modified system is needed, then System Definition activities are performed to assess the system. See Life Cycle Processes and Enterprise Need for further detail on the transformation of needs and requirements from the business or enterprise and stakeholder levels of abstraction addressed in Concept Definition to the system and system element level of abstraction addressed in System Definition.

The specific activities and sequence of Concept Definition activities and their involvement with the life cycle activities of any system, and in particular the close integration with System Definition activities, will be dependent upon the type of life cycle model being utilized. See Applying Life Cycle Processes for further discussion of the concurrent, iterative and recursive nature of these relationships.

### Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Business or Mission Analysis
- Stakeholder Needs and Requirements

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 as well as topics covered in Part 3.

## Concept Definition Activities

There are two primary activities discussed under concept definition: Mission Analysis and the definition of Stakeholder Needs and Requirements:

1. Mission Analysis begins an iteration of the life cycle of a potential SoI that could solve a problem or realize an opportunity for developing a new product, service, or enterprise. These activities assist business or enterprise decision makers to define the problem space, identify the stakeholders, develop preliminary operational concepts, and distinguish environmental conditions and constraints that bound the solution space. In other words, mission analysis takes the enterprise capability gap or opportunity and defines the problem/opportunity in a manner that provides a common understanding encapsulated in what are referred to as “business or mission needs.” Business or mission needs are then used to produce a clear, concise, and verifiable set of business requirements.
2. The Stakeholder Needs and Requirements activity works with stakeholders across the life cycle to elicit and capture a set of needs, expectations, goals, or objectives for a desired solution to the problem or opportunity, referred to as "stakeholder needs". The stakeholder needs are used to produce a clear, concise, and verifiable set of stakeholder requirements. Stakeholder needs and requirements identify and define the needs and requirements of the stakeholders in a manner that enables the characterization of the solution alternatives.

Mission Analysis takes the business and stakeholders' needs and requirements and carries the analysis down from problem space to solution space, including concept, mission, and boundary or context so that a solution concept (at the black-box level) can be selected from the alternatives. Figure 1 in the Mission Analysis topic depicts this interaction. The products and artifacts produced during Concept Definition are then used in System Definition.

The different aspects of how systems thinking is applicable to concept definition are discussed in SEBoK Part 2. In particular, the use of a combination of hard system and soft system approaches depending on the type of problem or class of solution is discussed in Identifying and Understanding Problems and Opportunities and the contrast between top-down and bottom-up approaches in Synthesizing Possible Solutions.

## Drivers of Solution on Problem Definition: Push Versus Pull

Problem definition and solution design depend on each other. Solutions should be developed to respond appropriately to well-defined problems. Problem definitions should be constrained to what is feasible in the solution space. System Analysis activities are used to provide the link between problems and solutions.

There are two paradigms that drive the ways in which concept definition is done: *push* and *pull*. The *pull* paradigm is based on providing a solution to an identified problem or gap, such as a missing mission capability for defense or infrastructure. The *push* paradigm is based on creating a solution to address a perceived opportunity, such as the emergence of an anticipated product or service that is attractive to some portion of the population (i.e. whether a current market exists or not). This can impact other life cycle processes, such as in verification and validation, or alpha/beta testing as done in some commercial domains.

As systems generally integrate existing and new system elements in a mixture of push and pull, it is often best to combine a bottom-up approach with a top-down approach to take into account legacy elements, as well as to identify the services and capabilities that must be provided in order to define applicable interface requirements and constraints. This is discussed in Applying Life Cycle Processes.

## References

### Works Cited

None.

### Primary References

ANSI/EIA. 1998. *Processes for Engineering a System*. Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA), ANSI/EIA 632-1998.

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Requirements Engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

### Additional References

Hitchins, D. 2007. *Systems Engineering: A 21st Century Systems Methodology.* Hoboken, NJ, USA: John Wiley & Sons.

*ISO/IEC. 2003. Systems Engineering – A Guide for The Application of ISO/IEC 15288 System Life Cycle Processes*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 19760:2003 (E). http://www.hitchins.net/EmergenceEtc.pdf.

ISO/IEC. 2007. *Systems Engineering – Application and Management of The Systems Engineering Process*. Geneva, Switzerland: International Organization for Standards (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 26702:2007.

Jackson, S., D. Hitchins, and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight.* (April 2010): 41-43.

NASA. 2007. *Systems Engineering Handbook*. Washington, D.C., USA: National Aeronautics and Space Administration (NASA). NASA/SP-2007-6105.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Business or Mission Analysis

***Lead Authors:*** *Alan Faisandier, Scott Jackson,* ***Contributing Authors:*** *Richard Turner, Garry Roedler, Rick Adcock*

The starting point of engineering any system-of-interest (SoI) is understanding the socio-economic and technological context in which potential problems or opportunities reside. Then, the enterprise strategic goals and stakeholder needs, expectations, and requirements represent the problem or the opportunity from the viewpoint of business or enterprise decision makers while also taking into account the views of users, acquirers, and customers.

Mission Analysis (MA) is part of the larger set of concept definition activities - the set of systems engineering activities in which the problem space and the needs of the business or enterprise and stakeholders are closely examined. This occurs before any formal definition of the (SoI) is developed but may need to be revisited through the life cycle. In fact, the activities of Concept Definition determine whether the enterprise strategic goals and business needs will be addressed by a new system, a change to an existing system, a service, an operational change or some other solution. The MA activity focuses on the identification of the primary purpose(s) of the solution (its "mission"), while Stakeholder Needs and Requirements activity explores what capabilities stakeholders desire in accomplishing the mission and may include some detail on the performance of certain aspects of the solution. MA is often performed iteratively with the Stakeholder Needs and Requirements activity to better understand the problem (or opportunity) space, as well as the solution space.

## Purpose and Definition

The purpose of MA is to understand a mission/market problem or opportunity, analyze the solution space, and initiate the life cycle of a potential solution that could address the problem or take advantage of an opportunity. MA is a type of strategic or operations analysis related to needs, capability gaps, or opportunities and solutions that can be applied to any organization that evolves its strategy for its business objectives.

MA, in some domains called market analysis or business analysis, is the identification, characterization, and assessment of an operational problem or opportunity within an enterprise. The definition of a mission or business function in a problem space frames the solution, both in terms of the direct application to the mission or business function, and in terms of the context for the resulting solution.

MA is used to define needed (or desired) operational actions, not hardware/software functions; that is, it is focused on defining the problem space, not the solution space. MA begins with the business vision and Concept of Operations (ConOps) (IEEE. 1998), and other organization strategic goals and objectives including the mission (or business function). The primary products of MA are Business or Mission Needs, which are supported by preliminary life-cycle concepts—including a preliminary acquisition concept, a preliminary operational concept (OpsCon), a preliminary deployment concept, a preliminary support concept, and a preliminary retirement concept. Business or Mission Needs are then elaborated and formalized into Business or Mission Requirements. The preliminary operational concept includes the operational scenarios for the mission and the context in which the solution will exist.

MA may include mathematical analysis, modeling, simulation, visualization, and other analytical tools to characterize the intended mission and determine how to best achieve the needs/objectives. MA evaluates alternative approaches to determine which best supports the stakeholder needs (among both materiel and non-materiel solution alternatives, also known as product solutions and service/operational solutions). Thus, MA defines the problem space and analyzes the solution space alternatives using quality attribute constraints driven by the enterprise objectives.

## Principles and Concepts

### Mission Analysis and Concept of Operations

MA and the terms ConOps and OpsCon are broadly used in U.S. and UK defense and aerospace organizations to analyze and define how a system is intended to operate, as well as how the major operations or operational scenarios are intended to be performed. They take into account the strategic, operational, and tactical aspects of the identified scenarios. ANSI/AIAA G-043A-2012 (ANSI 2012) identifies that the terms 'concept of operations' and 'operational concept' are often used interchangeably but notes that an important distinction exists because each has a separate purpose and is used to meet different ends. The ConOps is at an organizational level, prepared by enterprise management and refined by business management:

> *The ConOps, at the organization level, addresses the leadership's intended way of operating the organization. It may refer to the use of one or more systems (as black boxes) to forward the organization's goals and objectives. The ConOps document describes the organization's assumptions or intent in regard to an overall operation or series of operations within the business in regards to the system to be developed, existing systems, and possible future systems. This document is frequently embodied in long-range strategic plans and annual operational plans. The ConOps document serves as a basis for the organization to direct the overall characteristics of future business and systems.* (ISO/IEC 2011)

The ConOps informs the OpsCon, which is drafted by business management in the Mission Analysis activity and refined by stakeholders in the Stakeholder Needs and Requirements activity:

> *A system OpsCon document describes what the system will do (not how it will do it) and why (rationale). An OpsCon is a user-oriented document that describes system characteristics of the to-be-delivered system from the user's viewpoint. The OpsCon document is used to communicate overall quantitative and qualitative system characteristics to the acquirer, user, supplier and other organizational elements.* (ISO/IEC 2011)

It should be noted that the OpsCon has an operational focus and should be supported by the development of other concepts, including a deployment concept, a support concept, and a retirement concept.

In order to determine appropriate technical solutions for evolving enterprise capabilities, systems engineering (SE) leaders interact with enterprise leaders and operations analysts to understand:

- the enterprise ConOps and future mission, business, and operational (MBO) objectives;
- the characterization of the operational concept and objectives (i.e., constraints, mission or operational scenarios, tasks, resources, risks, assumptions, and related missions or operations); and
- how specific missions or operations are currently conducted and what gaps exist in those areas.

They then conceptually explore and select from alternative candidate solutions. This interaction ensures a full understanding of both the problem space and the solution space. The alternative candidate solutions can include a wide range of approaches to address the need, as well as variants for an approach to optimize specific characteristics (e.g., using a different distribution of satellite orbit parameters to maximize coverage or events while minimizing the number of satellites). Analysis, modeling and simulation, and trade studies are employed to select alternative approaches (NDIA 2010).

The notions of mission analysis, ConOps and OpsCon are also used in industrial sectors, such as aviation administrations and aeronautic transportation, health care systems, and space with adapted definitions and/or terms, such as operational concepts, usage concepts and/or technological concepts. For example, "mission analysis" is the term used to describe the mathematical analysis of satellite orbits performed to determine how best to achieve the objectives of a space mission (ESA 2008).

In commercial sectors, MA is often primarily performed as market analysis. Wikipedia defines market analysis as a process that:

> *. . . studies the attractiveness and the dynamics of a special market within a special industry. It is part of the industry analysis and this in turn of the global environmental analysis. Through all these analyses, the chances, strengths, weaknesses, and risks of a company can be identified. Finally, with the help of a Strengths, Weaknesses, Opportunities, and Threats (SWOT) analysis, adequate business strategies of a company will be defined. The market analysis is also known as a documented investigation of a market that is used to inform a firm's planning activities, particularly around decisions of inventory, purchase, work force expansion/contraction, facility expansion, purchases of capital equipment, promotional activities, and many other aspects of a company.* (Wikipedia Contributors, 2012)

Anywhere these notions are used, it is evident that they are based on fundamental concepts, such as the operational mode (or state of the system), scenario (of actions), the enterprise level ConOps and the system level operational concepts, functions, etc. For more explanations about the ConOps and operational concept, refer to *Systems and Software Engineering - Requirements Engineering* (ISO/IEC 2011); useful information can be found in Annex A, "System Operational Concept," and Annex B, "Concept of Operations" (ISO/IEC 2011).

## Mission Analysis as Part of Enterprise Strategy Development

Periodically, most enterprises re-evaluate their strategy with respect to their mission, vision, and positioning to accomplish their goals. Figure 1 shows the interactions of the enterprise strategy development and the concept definition, including the MA and Stakeholder Needs and Requirements activities that are involved in an iterative manner to fully develop the strategy and define future capabilities and solutions.

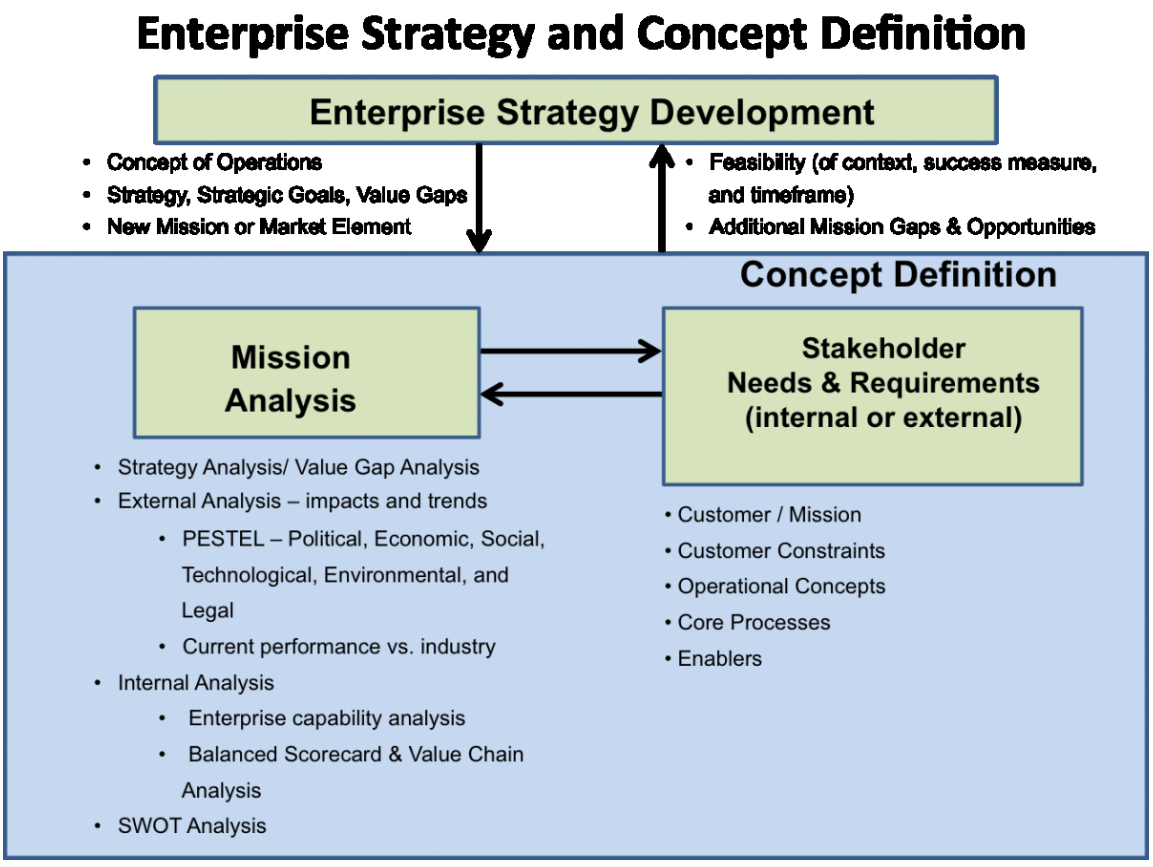

Enterprise Strategy and Concept Definition
Enterprise Strategy Development
• Concept of Operations
• Strategy, Strategic Goals, Value Gaps
• New Mission or Market Element
• Feasibility (of context, success measure, and timeframe)
• Additional Mission Gaps & Opportunities
Concept Definition
Mission Analysis
Stakeholder Needs & Requirements (internal or external)
• Strategy Analysis/ Value Gap Analysis
• External Analysis – impacts and trends
• PESTEL – Political, Economic, Social, Technological, Environmental, and Legal
• Current performance vs. industry
• Internal Analysis
• Enterprise capability analysis
• Balanced Scorecard & Value Chain Analysis
• SWOT Analysis
• Customer / Mission
• Customer Constraints
• Operational Concepts
• Core Processes
• Enablers

**Figure 1. Enterprise Strategy and Concept Development (Roedler 2012).** Used with permission of Garry Roedler. All other rights are reserved by the copyright owner.

As the enterprise evolves the strategy, it is essential to conduct the supporting MA or strategic analysis for each element of the enterprise to determine readiness to achieve future objectives. This analysis examines the current state to identify any problems or opportunities related to the objective achievement and aids the enterprise in fully understanding and defining the problem space. The analysis examines the external environment and interfaces in search of impacts and trends, as well as the internal enterprise to gauge its capabilities and value stream gaps. Additionally, a strengths, weaknesses, opportunities, and threats (SWOT) analysis may be performed. As the problem space is defined, the stakeholder needs are defined and transformed into stakeholder requirements that define the solutions needed. These requirements include those that address customer and mission needs, the future state of core processes and capabilities of the enterprise, and the enablers to support performance of those processes and capabilities. Finally, MA is engaged again to examine the solution space. Candidate solutions that span the potential solution space are identified, from simple operational changes to various system developments or modifications. Various techniques are applied to analyze the candidates, understand their feasibility and value, and select the best alternative.

## Process Approach

### Activities of the Process

It is necessary to perform the following major activities and tasks during the MA process:

1. Review and understand the enterprise mission, vision, and ConOps.
2. Identify and define any gaps and opportunities related to future evolution of the strategy:
    1. Examine the current state to identify any problems or opportunities related to the objective achievement, including any deficiencies of the existing system.
    2. Analyze the context of the actual political, economic, social, technological, environmental, and legal (PESTAL) factors, while studying sensitive factors such as cost and effectiveness, security and safety improvement, performance improvement or lack of existing systems, market opportunities, regulation changes, users' dissatisfaction, etc. External, internal, and SWOT analysis should be included as well. For the technological considerations, an appropriate architecture framework representation, such as the U.S. Department of Defense Architecture Framework (DoDAF) operations view (DoD 2010), the Zachman Framework (Rows 1 and 2) (Zachman 2008), and The Open Group Architecture Framework (TOGAF) Architecture Development Method (ADM) (The Open Group 2010) Phases A and B should be included within the concept definition when performing mission analysis and stakeholders needs and requirements.
    3. Define the mission, business, and/or operational problem or opportunity, as well as its context, and any key parameters, without focusing on a solution.
3. Examine and evaluate the solution space:
    1. Identify the main stakeholders (customers, users, administrations, regulations, etc.).
    2. Identify high level operational modes or states, or potential use cases.
    3. Identify candidate solutions that span the potential solution space, from simple operational changes to various system developments or modifications.
    4. Identify existing systems, products, and services that may address the need for operational or functional modifications.
    5. Deduce what potential expected services may be needed. The SoI is a potential and not yet existing product, service or enterprise. Additionally, the solution could be an operational change or a change to an existing product or service.
4. Perform appropriate modeling, simulation, and analytical techniques to understand the feasibility and value of the alternative candidate solutions. Model or simulate operational scenarios from these services and use cases, and enrich them through reviews with stakeholders and subject matter experts.

5. Define basic operational concept or market strategy, and/or business models.
    1. From previous modeled operational scenarios and operational modes, deduce and express the usage of operational concepts, or technical concepts.
    2. Collect and enrich needs, expectations, scenarios, and constraints.
    3. Validate the mission of any potential SoI in the context of any proposed market strategy or business model.
6. Evaluate the set of alternatives and select the best alternative.
    1. Perform a trade study of the alternatives to discriminate between the alternatives.
7. Provide feedback on feasibility, market factors, and alternatives for use in completion of the enterprise strategy and further actions.
8. Define preliminary deployment concept, preliminary support concept, and preliminary retirement concept.

## Mission Analysis Artifacts

This process may create several artifacts, such as:

- recommendations for revisions to the enterprise ConOps;
- preliminary operational concept document or inputs;
- mission analysis and definition reports (perhaps with recommendations for revisions of the mission);
- a set of business needs;
- preliminary life-cycle concepts (preliminary operational concept, preliminary deployment concept, preliminary support concept, and preliminary retirement concept;
- system analysis artifacts (e.g., use case diagrams, context diagrams, sequence/activity diagrams, functional flow block diagrams);
- trade study results (alternatives analysis);
- market study/analysis reports; and
- a set of business (or mission) requirements (often captured in a business requirement specification).

## Methods and Modeling Techniques

MA uses several techniques, such as:

- use case analysis;
- operational analysis;
- functional analysis;
- technical documentation review;
- trade studies;
- modeling;
- simulation;
- prototyping;
- workshops, interviews, and questionnaires;
- market competitive assessments;
- benchmarking; and
- organizational analysis techniques (e.g., strengths, weaknesses, opportunities, threats (SWOT analysis), and product portfolios).

## Practical Considerations

Major pitfalls encountered with mission analysis and marketing analysis are presented in Table 1.

**Table 1. Major Pitfalls for Mission Analysis. (SEBoK Original)**

| Pitfall | Description |
|---|---|
| Wrong level of system addressed | When delineating the boundaries of the SoI and defining the mission and purpose of the system at the very beginning of systems engineering, a classic mistake is to place the system-of-interest at the wrong level of abstraction. The level of abstraction can be too high or too low (sitting respectively in the upper-system or in a sub-system). This is the consequence of the principle stating that a system is always included in a larger system and of confusing the purpose and the mission of the SoI. |
| Operational modes or scenarios missing | In commercial products or systems, the lack or insufficient description of operational modes and scenarios (how the SoI will be used, in which situations, etc.) is often encountered. |

Proven practices with mission analysis and marketing analysis are presented in Table 2.

**Table 2. Mission Analysis Proven Practices. (SEBoK Original)**

| Practice | Description |
|---|---|
| Models of operational scenarios | Using modeling techniques as indicated in sections above for operational scenarios in any kind of SoI (including commercial systems). |
| Models of the context | Consider the context of use as a system and force oneself to use modeling techniques for main aspects of the context (functional, behavioral, physical, etc.). |

## References

### Works Cited

ANSI/AIAA G-043-2012e, Guide to the Preparation of Operational Concept Documents.

DoD. 2010. *DoD Architecture Framework*, version 2.02. Arlington, VA: U.S. Department of Defense. Accessed August 29, 2012. Available at: http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf.

ESA. 2008. *Mission Analysis: Towards a European Harmonization.* Paris, France: European Space Agency. Accessed August 29, 2012. Available at: http://www.esa.int/esapub/bulletin/bulletin134/bul134b_schoenmaekers.pdf.

IEEE. 1998. *Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, IEEE 1362:1998.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Life Cycle Processes - Requirements Engineering.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 29148:2011.

NDIA. 2010. "Mission Analysis Committee Charter". Website of the National Defense Industrial Association, Systems Engineering Division, Mission Analysis Committee. Accessed August 29, 2012. Available at: http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/Committees/Mission%20Analysis%20Committee/Mission%20Analysis%20Committee%20Charter.pdf.

The Open Group. 2011. *TOGAF*, version 9.1. Hogeweg, The Netherlands: Van Haren Publishing. Accessed August 29, 2012. Available at: https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?catalogno=g116.

Wikipedia contributors, "Market analysis," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Market_analysis&oldid=508583878 (accessed August 29, 2012).

Zachman, J. 2008. "John Zachman's Concise Definition of The Zachman Framework™." Zachman International Enterprise Architecture. Accessed August 29, 2012. Available at: http://www.zachman.com/about-the-zachman-framework.

## Primary References

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Requirements Engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0.

Lamsweerde, A. van. 2009. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. New York, NY, USA: Wiley.

## Additional References

Center for Quality Management. 1993. "Special Issue on Kano's Methods for Understanding Customer Defined Quality." *Center for Quality Management Journal.* 2(4) (Fall 1993).

Faisandier, A. 2012. *Systems Opportunities and Requirements*. Belberaud, France: Sinergy'Com.

Freeman, R. "Chapter 27: Achieving Strategic Aims: Moving Toward a Process Based Military Enterprise," in *Handbook of Military Industrial Engineering.* A.B. Badiru and M.U. Thomas (eds). Boca Raton, FL, USA: Taylor & Francis Group, CRC Press.

IEEE. 1998. *Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, IEEE 1362:1998.

Hull, M.E.C., K. Jackson, A.J.J. Dick. 2010. *Systems Engineering.* 3rd ed. London, UK: Springer.

Kaplan, R.S. and D.P. Norton. 2008. "Developing the Strategy: Vision, Value Gaps, and Analysis," Balanced Scorecard Report. Cambridge, MA, USA: Harvard Business School Publishing, Jan-Feb 2008.

Kano, N. 1984. "Attractive Quality and Must-Be Quality." *Quality JSQC.* 14(2) (October 1984).

Kohda, T., M. Wada, and K. Inoue. 1994. "A Simple Method for Phased Mission Analysis." *Reliability Engineering & System Safety.* 45(3): 299-309.

Marca, D. A. and C. L. McGowan. 1987. "SADT: Structured analysis and design techniques." *Software Engineering*. New York, NY: McGraw-Hill.

MITRE. 2011. "Concept Development." *Systems Engineering Guide.* Accessed 9 March 2012 at http://www.mitre.org/work/systems_engineering/guide/se_lifecycle_building_blocks/concept_development/ [1].

MITRE. 2011. "Requirements Engineering." *Systems Engineering Guide.* Accessed 9 March 2012 at http://www.mitre.org/work/systems_engineering/guide/se_lifecycle_building_blocks/requirements_engineering/ [2].

MITRE. 2011. "Stakeholder Assessment and Management." *Systems Engineering Guide.* Accessed 9 March 2012 at http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/transformation_planning_org_change/stakeholder_assessment_management.html/ [3].

Shupp, J.K. 2003. "The Mission Analysis Discipline: Bringing focus to the fuzziness about Attaining Good Architectures." Proceedings of INCOSE 13th International Symposium, July 2003.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

## References

[1] http://www.mitre.org/work/systems_engineering/guide/se_lifecycle_building_blocks/concept_development/
[2] http://www.mitre.org/work/systems_engineering/guide/se_lifecycle_building_blocks/requirements_engineering/
[3] http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/transformation_planning_org_change/stakeholder_assessment_management.html/

# Stakeholder Needs and Requirements

***Lead Authors:*** *Alan Faisandier, Garry Roedler,* ***Contributing Author:*** *Rick Adcock*

Stakeholder needs and requirements represent the views of those at the business or enterprise operations level—that is, of users, acquirers, customers, and other stakeholders as they relate to the problem (or opportunity), as a set of requirements for a solution that can provide the services needed by the stakeholders in a defined environment. Using enterprise-level life cycle concepts (see Business or Mission Analysis for details) as guidance, stakeholders are led through a structured process to elicit stakeholder needs (in the form of a refined set of system-level life-cycle concepts). Stakeholder needs are transformed into a defined set of Stakeholder Requirements, which may be documented in the form of a model, a document containing textual requirement statements or both.

Stakeholder requirements play major roles in systems engineering, as they:

- Form the basis of system requirements activities.
- Form the basis of system validation and stakeholder acceptance .
- Act as a reference for integration and verification activities.
- Serve as means of communication between the technical staff, management, finance department, and the stakeholder community.

This topic describes the definition of stakeholder needs and requirements which involves the activities necessary to elicit and prioritize the needs of the stakeholder(s), and transform those needs into a set of defined stakeholder requirements. Defining the problem or the issue to be solved, identifying the opportunity for developing a new solution, or improving a system-of-interest (SoI) must begin prior to starting the activities necessary to define stakeholder needs and requirements. This means that an initial context of use of the new or modified mission, operation, or capability has already been characterized (see Business or Mission Analysis). System requirements are considered in detail during system definition. None of the above can be considered complete until consistency between the two has been achieved, as demonstrated by traceability, for which a number of iterations may be needed.

## Purpose and Definition

The purpose of the Stakeholder Needs and Requirements definition activities are to elicit a set of clear and concise needs related to a new or changed mission for an enterprise (see mission analysis (MA) for information relevant to identifying and defining the mission or operation), and to transform these stakeholder needs into verifiable stakeholder requirements.

Stakeholders may well begin with desires, and expectations that may contain vague, ambiguous statements that are difficult to use for SE activities. Care must be taken to ensure that those desires and expectations are coalesced into a set of clear and concise need statements that are useful as a start point for system definition. These need statements will then need to be further clarified and translated into more *engineering-oriented* language in a set of stakeholder requirements to enable proper architecture definition and requirement activities. As an example, a need or an expectation such as, *to easily* manoeuvre *a car in order to park*, will be transformed in a set of stakeholder requirements to a statement such as, *increase the driviability of the car*, *decrease the effort for handling*, *assist the piloting*, *protect the coachwork against shocks or scratches*, etc.

To allow a clear description of the activities of stakeholder needs and requirements to be described, a generic view of the business teams and roles involved in a typical enterprise has been used below., tThis includes teams such as business management and business operations;, and roles including requirements engineer and business analyst. For an overview of these roles and how they enable both stakeholder and business requirements across the layers of a typical enterprise see Life Cycle Processes and Enterprise Need.

## Principles and Concepts

### Identifying Stakeholders

Stakeholders of a SoI may vary throughout the life cycle. Thus, in order to get a complete set of needs and subsequent requirements, it is important to consider all stages of the life cycle model when identifying the stakeholders or classes of stakeholders.

Every system has its own stages of life, which typically include stages such as concept, development, production, operations, sustainment, and retirement (for more information, please see Life Cycle Models). For each stage, a list of all stakeholders having an interest in the future system must be identified. The goal is to get every stakeholder's point of view for every stage of the system life in order to consolidate a complete set of stakeholder needs that can be prioritized and transformed into the set of stakeholder requirements as exhaustively as possible. Examples of stakeholders are provided in Table 1.

**Table 1. Stakeholder Identification Based on Life Cycle Stages. (SEBoK Original)**

| Life Cycle Stage | Example of Related Stakeholders |
|---|---|
| Engineering | Acquirer, panel of potential users, marketing division, research and development department, standardization body, suppliers, verification and validation team, production system, regulator/certification authorities, etc. |
| Development | Acquirer, suppliers (technical domains for components realization), design engineers, integration team, etc. |
| Transfer for Production or for Use | Quality control, production system, operators, etc. |
| Logistics and Maintenance | Supply chain, support services, trainers, etc. |
| Operation | Normal users, unexpected users, etc. |
| Disposal | Operators, certifying body, etc. |

### Identifying Stakeholder Needs

Once business management is satisfied that their needs and requirements are reasonably complete, they pass them on to the business operations team. Here, the Stakeholder Needs and Requirements (SNR) Definition Process uses the ConOps, or Strategic Business Plan (SBP), and the life-cycle concepts as guidance. The requirements engineer (RE) or business analyst (BA) leads stakeholders from the business operations layer through a structured process to elicit stakeholder needs—in the form of a refined OpsCon (or similar document) and other life-cycle concepts. The RE or BA may use a fully or partially structured process to elicit specific needs, as described in models such as user stories, use cases, scenarios, system concepts, and operational concepts.

## Identifying Stakeholder Requirements

Stakeholder needs are transformed into a formal set of stakeholder requirements, which are captured as models or documented as textual requirements in and output typically called a Stakeholder Requirement Specification (StRS), Stakeholder Requirement Document (StRD) or similar. That transformation should be guided by a well-defined, repeatable, rigorous, and documented process of requirements analysis. This requirements analysis may involve the use of functional flow diagrams, timeline analysis, N2 Diagrams, design reference missions, modeling and simulations, movies, pictures, states and modes analysis, fault tree analysis, failure modes and effects analysis, and trade studies.

## Collecting Stakeholder Needs and Requirements

There are many ways to collect stakeholder needs and requirements. It is recommended that several techniques or methods be considered during elicitation activities to better accommodate the diverse set of sources, including:

- Structured brainstorming workshops
- Interviews and questionnaires
- Technical, operational, and/or strategy documentation review
- Simulations and visualizations
- Prototyping
- Modeling
- Feedback from verification and validation processes,
- Review of the outcomes from the system analysis process (ISO/IEC 2015)
- Quality function deployment (QFD) - can be used during the needs analysis and is a technique for deploying the "voice of the customer". It provides a fast way to translate customer needs into requirements. (Hauser and Clausing 1988)
- Use case diagrams (OMG 2010)
- Activity diagrams (OMG 2010)
- Functional flow block diagrams (Oliver, Kelliher, and Keegan 1997)

## From the Capture of Stakeholder Needs to the Definition of Stakeholder Requirements

Several steps are necessary to understand the maturity of stakeholder needs and to understand how to improve upon that maturity. Figure 1 presents the *cycle of needs* as it can be deduced from Professor Shoji Shiba's and Professor Noriaki Kano's works and courses, and is adapted here for systems engineering (SE) purposes.

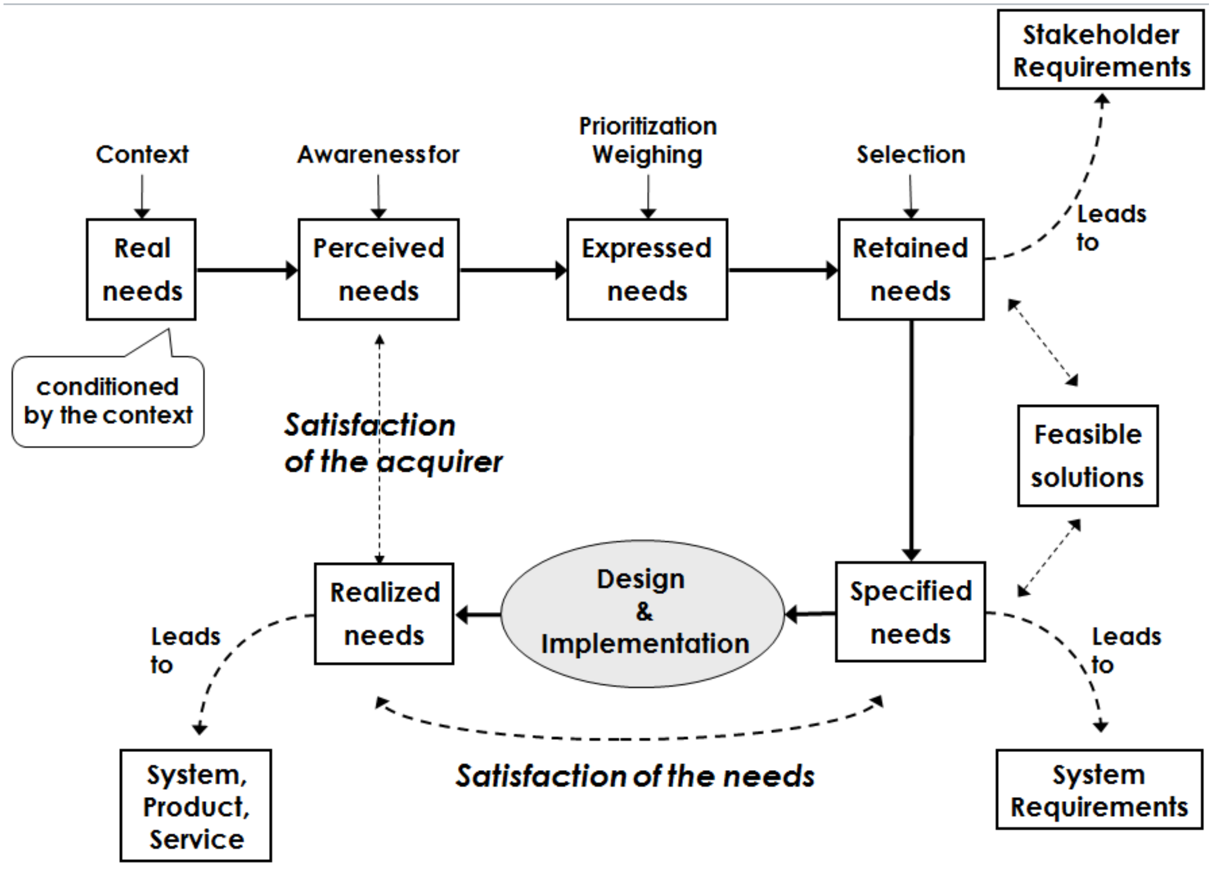


**Figure 1. Cycle of Needs (Faisandier 2012).** Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

Figure 1 shows the steps and the position of the stakeholder requirements and system requirements in the engineering cycle. Below are explanations of each stage of requirements (Faisandier 2012); to illustrate this, consider this example of a system related to infectious disease identification:

- **Real needs** are those that lie behind any perceived needs (see below); they are conditioned by the context in which people live. As an example, a generic need could be the ability to *identify infectious diseases easily.*Often, real needs appear to be simple tasks.
- **Perceived needs** are based on a person's awareness that something is wrong, that something is lacking, that improvements could be made, or that there are business, investment, or market opportunities that are not being capitalized upon. Perceived needs are often presented as a list of organized expectations resulting from an analysis of the usage conditions for the considered action (see Business or Mission Analysis). Following from the infectious disease example above, the real need might be perceived as a need to *carry out medical tests in particular circumstances (laboratories, points of care, hospitals, and/or human dispensaries).* Since the real need is seldom clearly expressed, richness of the knowledge of the perceived needs is used as a basis for potential solutions. This step has to be as complete as possible to cover all the contexts of use.
- **Expressed needs** originate from perceived needs in the form of generic actions or constraints, and are typically prioritized. In the example, if safety is the primary concern, the expressed need to *protect the operator against contamination* may take priority over other expressed needs such as *assist in the execution of tests.* When determining the expressed needs, the analysis of the expected mission or services in terms of operational

scenarios takes place.

- **Retained needs** are selected from the expressed needs. The selection process uses the prioritization of expressed needs to achieve a solution or to make attaining solutions feasible. The retained needs allow the consideration of potential solutions for a SoI. These retained *stakeholder intentions do not serve as stakeholder requirements, since they often lack definition, analysis, and possibly consistency and feasibility. Using the concept of operations to aid the understanding of the stakeholder intentions at the organizational level and the system operational concept from the system perspective, requirements engineering leads stakeholders from those initial intentions to structured and more formal stakeholder requirement statements,* ISO/IEC/IEEE 29148 *Systems and software engineering - Requirements engineering* (ISO 2011). Characteristics of good requirements can be found in (ISO 2011). Exploration of potential solutions must start from this step. The various solutions suggested at this step are not yet products, but describe means of satisfying the stakeholder requirements. Each potential solution imposes constraints on the potential future SoI.
- **Specified needs**, are the translation of the stakeholder needs to represent the views of the supplier, keeping in mind the potential, preferred, and feasible solutions. Specified needs are translated into system requirements. Consistent practice has shown this process requires iterative and recursive steps in parallel with other life cycle processes through the system design hierarchy (ISO 2011).
- **Realized needs** are the product, service, or enterprise realized, taking into account every specified need (and hence, the retained needs).

Each class of needs listed above aligns with an area of the SE process. For example, the development of *specified needs* requirements is discussed in the System Requirements topic. For more information on how requirements are used in the systems engineering process, please see the System Definition knowledge area (KA).

### Classification of Stakeholder Requirements

Several classifications of stakeholder requirements are possible, e.g. ISO/IEC 29148, section 9.4.2.3 (ISO 2011) provides a useful set of elements for classification. Examples of classification of stakeholder requirements include: service or functional, operational, interface, environmental, human factors, logistical, maintenance, design, production, verification requirements, validation, deployment, training, certification, retirement, regulatory, environmental, reliability, availability, maintainability, design, usability, quality, safety, and security requirements. Stakeholders will also be faced with a number of constraints, including: enterprise, business, project, design, realization, and process constraints.

## Process Approach

### Activities of the Process

Major activities and tasks performed during this process include the following:

- Identify the stakeholders or classes of stakeholders across the life cycle.
- Elicit, capture, or consolidate the stakeholder needs, expectations, and objectives as well as any constraints coming from the mission and business analysis processes.
- Refine the OpsCon and other life-cycle concepts (acquisition concept, deployment concept, support concept, and retirement concept).
- Prioritize the stakeholder needs.
- Transform the prioritized and retained stakeholder needs into stakeholder requirements.
- Verify the quality of each stakeholder requirement and of the set of stakeholder requirements using the characteristics of good requirements identified in the System Requirements article.
- Validate the content and the relevance of each stakeholder requirement with corresponding stakeholder representatives providing rationale (glossary) for the existence of the requirement.

- Identify potential risks (or threats and hazards) that could be generated by the stakeholder requirements (for further information, see Risk Management).
- Synthesize, record, and manage the stakeholder requirements and potential associated risks.

### Artifacts, Methods and Modeling Techniques

This process may create several artifacts, such as:

- Recommendations to refine the Business Requirement Specification (if necessary)
- Refined life-cycle concepts (OpsCon, acquisition concept, deployment concept, support concept, and retirement concept)
- Stakeholder requirements (in the form of a model or a document containing textual requirements, such as the Stakeholder Requirement Specification)
- Stakeholder interview reports
- Stakeholder requirements database
- Stakeholder requirements justification documents (for traceability purposes)
- Input for draft verification and validation plans

The content, format, layout and ownership of these artifacts will vary depending on who is creating them and in which domains they will be used. Between these artifacts and the outputs of the process, activities should cover the information identified in the first part of this article.

## Practical Considerations

Major pitfalls encountered with stakeholder requirements are presented in Table 3.

**Table 3. Major Pitfalls for Stakeholder Requirements. (SEBoK Original)**

| Pitfall | Description |
|---|---|
| **Operator Role Not Considered** | Sometimes engineers do not take into account the humans acting as operators inside a system or those who use the system and are outside of the system. As a consequence, elements are forgotten (e.g. roles of operators). |
| **Exchanges with External Objects Forgotten** | The exhaustiveness of requirements can be an issue; in particular, the interfaces with external objects of the context of the system can be forgotten (exchanges of matter, energy, information). |
| **Physical Connections with External Objects Forgotten** | Within the interface issue, physical connections of the system-of-interest with external objects can be forgotten (technological constraints). |
| **Forgotten Stakeholders** | Stakeholders can be forgotten, as everyone thinks of direct users, customers, and suppliers; however, one may fail to consider those who do not want the system to exist and malevolent persons. |

Proven practices with stakeholder requirements are presented in Table 4.

**Table 4. Stakeholder Requirements Proven Practices. (SEBoK Original)**

| Practice | Description |
|---|---|
| **Involve Stakeholders** | Involve the stakeholders early in the stakeholder requirements development process. |
| **Presence of Rationale** | Capture the rationale for each stakeholder requirement. |
| **Analyze Sources before Starting** | Complete stakeholder requirements as much as possible before starting the definition of the system requirements. |
| **Modeling Techniques** | Use modeling techniques as indicated in sections above. |
| **Requirements Management Tool** | Consider using a requirements management tool. This tool should have the capability to trace linkages between the stakeholder requirements and the system requirements and to record the source of each stakeholder requirement. |

## References

### Works Cited

Faisandier, A. 2012. *Systems Architecture and Design.* Belberaud, France: Sinergy'Com.

Hauser, J. and D. Clausing. 1988. "The House of Quality." *Harvard Business Review.* (May - June 1988).

OMG. 2010. *OMG Systems Modeling Language specification*, version 1.2. Needham, MA: Object Management Group. July 2010.

Oliver, D., T. Kelliher, and J. Keegan. 1997. *Engineering complex systems with models and objects*. New York, NY, USA: McGraw-Hill.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Requirements engineering.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

### Primary References

ISO/IEC/IEEE. 2011. *Systems and software engineering - Requirements engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Architecture Description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

### Additional References

Buede, D.M. 2009. *The engineering design of systems: Models and methods*. 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

MITRE. 2011. "Requirements Engineering." *Systems Engineering Guide.* Accessed 9 March 2012 at http://www.mitre.org/work/systems_engineering/guide/se_lifecycle_building_blocks/requirements_engineering/.

MITRE. 2011. "Stakeholder Assessment and Management." *Systems Engineering Guide.* Accessed 9 March 2012 at http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/transformation_planning_org_change/stakeholder_assessment_management.html.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Knowledge Area: System Definition

## System Definition

***Lead Authors:*** *Alan Faisandier, Garry Roedler*, ***Contributing Author:*** *Rick Adcock*

System definition activities are conducted to create and describe in detail a system-of-interest (SoI) to satisfy an identified need. The activities are grouped and described as generic processes. which consist of system requirements definition, system architecture definition, system design definition and system analysis. The architecture definition of the system may include the development of related logical architecture models and physical architecture models. During and/or at the end of any iteration, gap analysis is performed to ensure that all system requirements have been mapped to the architecture and design.

System definition activities build on the artifacts and decisions from concept definition, primarily the articulation of the mission of the (SoI), the needs and requirements of stakeholders, and preliminary operational concepts. See Life Cycle Processes and Enterprise Need for further detail on the transformation of needs and requirements from the business or enterprise and stakeholder levels of abstraction addressed in concept definition to the system and system element level of abstraction addressed in system definition.

The products of system definition activities (system requirements, architecture and design) are inputs to system realization.

The specific activities and sequence of system definition activities and their involvement with the life cycle activities of any system, and in particular the close integration with concept definition and system realization activities, will be dependent upon the type of life cycle model being utilized. See Applying Life Cycle Processes for further discussion of the concurrent, iterative and recursive nature of these relationships.

### Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- System Requirements
- System Architecture
- Logical Architecture Model Development
- Physical Architecture Model Development
- System Design
- System Analysis

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

## System Views and System Elements

An engineered system solution to a defined concept includes a set of engineering elements, characteristics, and properties. These elements are grouped in two ways:

- Needs and requirements views
- Architecture and design views

Architecture views include the identification of the boundary and interfaces of a system-of-interest (SoI), which may then be further refined as a collection of system elements and their relationships.

### Needs and Requirements Views

Requirements provide an overall view of the purpose and mission which the system as a whole is intended to satisfy, as well as a technology-independent view of that the system solutions(s) should do. They are conventionally organized into two types:

- Business or mission requirements and Stakeholder requirements are defined and discussed in the Concept Definition KA.
- System requirements, which describe the functions which the system as a whole should fulfill in order to satisfy the stakeholder requirements and are expressed in an appropriate set of views, and non-functional requirements expressing the levels of safety, security, reliability, etc., which are called for. These collectively form the basis for verification later in the life cycle.

System requirements and stakeholder requirements are closely related. Neither can be considered complete until consistency between the two has been achieved, as demonstrated by traceability, for which a number of iterations may be needed.

The process activities that are used to identify, engineer and manage system requirements are described further in the System Requirements article in the KA.

### Architecture and Design Views

A given engineered system is one solution that could address/answer a problem or an opportunity (represented through requirements views); the solution may be more or less complex. A complex solution cannot be comprehended with a single view or model, because of the characteristics or properties of the problem/solution (see system complexity). The characteristics are structured as types or entities; types are related to each other. An instantiation of the set of types can be understood as THE architecture of the system. The majority of interpretations of system architecture are based on the fairly intangible notion of structure. Therefore, the system architecture and design is formally represented with sets of types or entities such as functions, interfaces, resource flow items, information elements, physical elements, nodes, links, etc. These entities may possess attributes/characteristics such as dimensions, environmental resilience, availability, reliability, learnability, execution efficiency, etc. The entities are interrelated by the means of relationships and are generally grouped into sets to represent views/models of the system architecture and design.

Viewpoints and views are sometimes specified in architecture frameworks. Views are usually generated from models. Many systems engineering practices use logical and physical views for modeling the system architecture and design.

- The **logical view of the architecture** supports the logical operation of the system all along its life cycle, and may include functional, behavioral, and temporal views/models. Operational scenarios refine the mission into a collection of functions and dynamic structures that describe how the mission is performed (behavior).
- The **physical view of the architecture** is a set of system elements performing the functions of the system. Those system elements can be either material or immaterial (e.g., equipment made of hardware, software and/or human

roles).

The boundary of the system architecture depends on what engineers include within the scope of the SoI and outside of it. This decision marks the transition from the characterization of the problem context to the beginnings of solution definition.

Facing the potential number of system elements that constitute the physical architecture, sets of system elements can be grouped to form systems. The decomposition of the SoI (highest level) may include the decomposition of several layers of systems (intermediate levels of systems) until technological system elements (lowest level) are defined. Any layer of the decomposition may include systems and non-decomposable technological system elements. The relationship between each layer is recursive; as a system element is also an engineered system it can be characterized in its turn using the previous views in its own context.

The logical and physical representations of the system architecture are mapped onto each other. The interactions between system elements are defined by interfaces whose complexity strongly depends on the way the system architecture and design is defined. The relationships between the outputs of concept definition and the system solution, as well as the range of other views of a system that are available to describe a more complete set of characteristics between the system elements are discussed further in the Logical Architecture Model Development and Physical Architecture Model Development sections of system definition.

## System Synthesis and Decomposition

System definition is managed through methodical synthesis of the SoI into systems and system elements. Solution synthesis may be top down or bottom up, as discussed in Synthesizing Possible Solutions. However it is done, as the system architecture definition advances, a decomposition of systems and system elements emerges, this forms a system breakdown structure (SBS). For project management purposes, every system of the SBS may be included in a *building block*, a notion introduced in (ANSI/EIA 1998), also called *system blocks*.

Stakeholder requirements and system requirements exist at all layers of the SBS. In ISO/IEC/IEEE 29148 *Systems and software engineering - Requirements Engineering* (ISO 2011), these layers are known as levels of abstraction. Along with systematically introducing layers of systems, the architecture and design process manages the transformation of the system requirements through levels of abstraction. Figure 1 illustrates this approach.

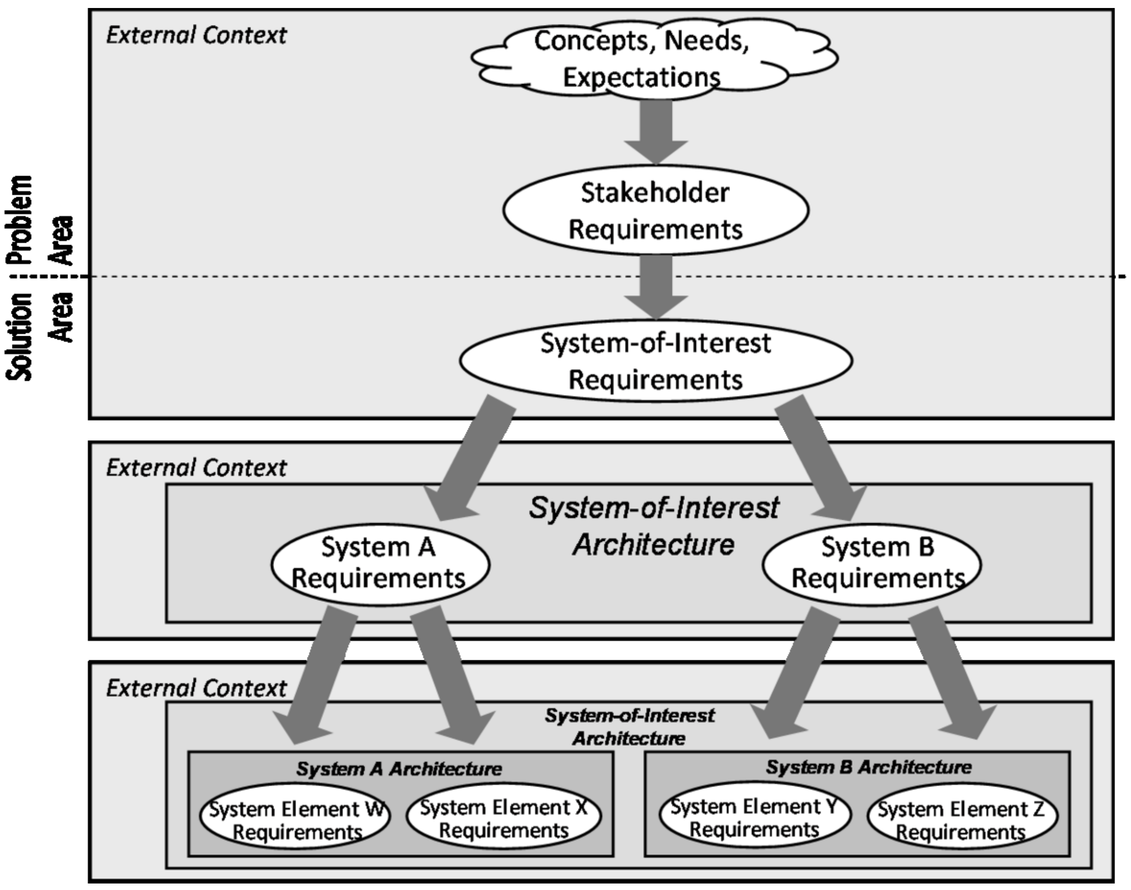

External Context
Concepts, Needs, Expectations
Stakeholder Requirements
Problem Area
Solution Area
System-of-Interest Requirements
External Context
System-of-Interest Architecture
System A Requirements
System B Requirements
External Context
System-of-Interest Architecture
System A Architecture
System B Architecture
System Element W Requirements
System Element X Requirements
System Element Y Requirements
System Element Z Requirements

**Figure 1. Top-down Development of Architecture and Design, and Requirements (Faisandier 2012).** Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

As shown in Figure 1

- The white ovals represent requirements at decreasing levels of abstraction, and the arrows represent the transformation of those requirements through the levels using the architecture and design process. Stakeholder expressions of needs, expectations, and constraints are transformed into stakeholder requirements.
- The next transformation crosses the boundary between the problem and solution areas by converting stakeholder requirements into system requirements, reflecting the bounded solution space.
- At the SoI level, the system architecture is developed which serves to identify systems and system elements and establishes how they operate together to address the SoI requirements.

This approach is applied recursively for each level of abstraction/decomposition recognizing that the same generic processes are applied at multiple levels of abstraction. At any level of this decomposition one or more solution options may be presented as system architectures. The process by which the solution which best fits the system requirements, associated stakeholder needs and wider life cycle concerns is selected and justified is discussed in the System Analysis process.

Figure 2 below portrays the engineering that occurs in each system block. As necessary, system elements are defined through sets of system element requirements, which become inputs to other system blocks (*level n+1*). The approach is then recursively applied using the system definition processes.

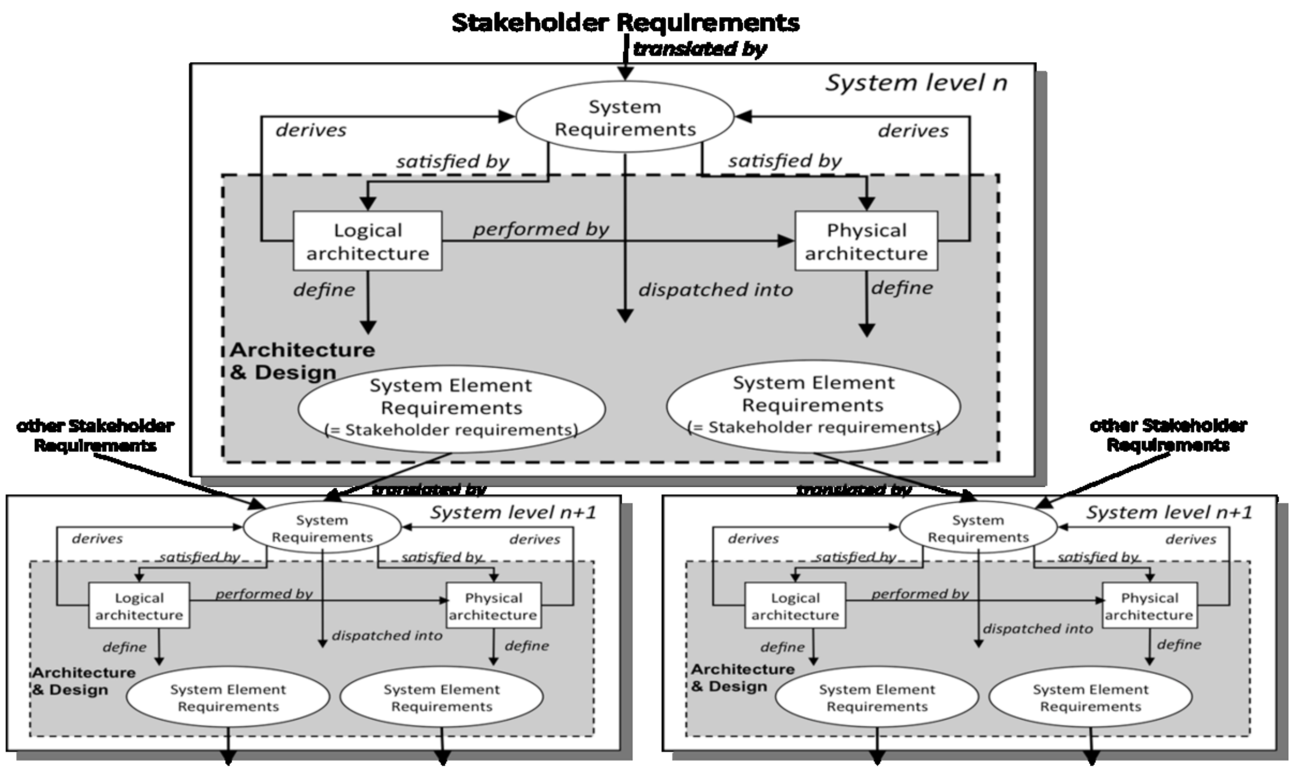


**Figure 2. Recursive Instantiation of Definition Processes (Faisandier 2012).** Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

At the *n+1* level, the systems or system elements may also collect other stakeholder requirements that are directly pertinent to this level of architecture and design. Processes within each system are generic, but unique in local purpose, scope and context.

See Applying Life Cycle Processes for a discussion of the iterative and recursive application of system requirements and architecture processes, and Life Cycle Processes and Enterprise Need for further detail on the transformation of needs and requirements to system and system element levels of abstraction.

The different aspects of how systems thinking is applicable to system definition are discussed in SEBoK Part 2. In particular, see discussion of the recursive nature of systems and engineered system contexts in Engineered System Context; the contrast between top-down and bottom up approaches in Synthesizing Possible Solutions and the role of solution architecture options and selection in Analysis and Selection between Alternative Solutions.

## References

### Works Cited

ANSI/EIA. 1998. *Processes for Engineering a System*. Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA), ANSI/EIA-632-1998.

Faisandier, A. 2012. *Systems Architecture and Design.* Belberaud, France: Sinergy'Com.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Requirements Engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

## Primary References

ANSI/EIA. 1998. *Processes for Engineering a System.* Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA), ANSI/EIA 632-1998.

Blanchard, B.S., and W.J. Fabrycky. 2005. *Systems Engineering and Analysis.* 4th ed. Prentice-Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

ISO/IEC. 2007. *Systems Engineering – Application and Management of The Systems Engineering Process*. Geneva, Switzerland: International Organization for Standards (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 26702:2007.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Requirements Engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Architecture Description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Martin, J.N. 1997. *Systems Engineering Guidebook: A process for developing systems and products,* 1st ed. Boca Raton, FL, USA: CRC Press.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

## Additional References

Baldwin, C.Y. and K.B. Clark. 2000. *Design Rules*. Cambridge, Mass: MIT Press.

Buede, D.M. 2009. *The Engineering Design of Systems: Models and Methods*. 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

Hatley, D.J., and I.A. Pirbhai. 1987. *Strategies for Real-Time System Specification*. New York, NY: Dorset House Pub.

MOD. 2010. *MOD Architecture Framework,* Version 1.2.004. UK Ministry of Defence. Available at: http://www.mod.uk/DefenceInternet/AboutDefence/WhatWeDo/InformationManagement/MODAF/.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# System Requirements

***Lead Authors:*** *Alan Faisandier, Garry Roedler,* **Contributing Author:** *Richard Turner, Rick Adcock, Ariela Sofer*

System requirements are all of the requirements at the *system level* that describe the functions which the system as a whole should fulfill to satisfy the stakeholder needs and requirements, and is expressed in an appropriate combination of textual statements, views, and non-functional requirements; the latter expressing the levels of safety, security, reliability, etc., that will be necessary.

System requirements play major roles in systems engineering, as they:

- Form the basis of system architecture and design activities.
- Form the basis of system integration and verification activities.
- Act as reference for validation and stakeholder acceptance.
- Provide a means of communication between the various technical staff that interact throughout the project.

Elicitation of stakeholder requirements starts in Concept Definition, and will be initially developed though interview and mission analysis. System requirements are considered in detail during System Definition. Neither can be considered complete until consistency between the two has been achieved, as demonstrated by traceability, for which a number of iterations may be needed.

## Definition and Purpose of Requirements

A requirement is a statement that identifies a product or processes operational, functional, or design characteristic or constraint, which is unambiguous, testable, or measurable and necessary for product or process acceptability (ISO 2007).

To avoid confusion in the multitude of terms pertaining to *requirements*, consider the following classifications:

- **Process Role or State**: The role the requirement plays in the definition process; for instance, its position in the system block (e.g. translated, derived, satisfied) or its state of agreement (e.g. proposed, approved, cancelled).
- **Level of Abstraction**: The level within the definition process that the requirement stands; for instance, stakeholder requirement, system requirement, system element requirement.
- **Type of Requirement**: The nature of the requirement itself; for instance, functional, performance, constraint, etc.

Any single requirement may simultaneously be in a particular state, at a particular level abstraction, and of a particular type. For additional explanations about differences between the types of requirements, refer to (Martin 1997, Chapter 2).

## Principles Governing System Requirements

### Relationship to Stakeholder Requirements and Logical Architecture

A set of stakeholder requirements are clarified and translated from statements of need into *engineering-oriented* language in order to enable proper architecture definition, design, and verification activities that are needed as the basis for system requirements analysis.

The system requirements are based around identification and synthesis of the functions required of any solution system associated with performance and other quality measures and provide the basis for the assessment of candidate solutions and verification of the completed system. The system requirements are expressed in technical language that is useful for architecture and design: unambiguous, consistent, coherent, exhaustive, and verifiable. Of course, close coordination with the stakeholders is necessary to ensure the translation is accurate and traceability is maintained. This results in a set of system functions and requirements specifying measurable characteristics which can form the basis for system realization.

The logical architecture defines system boundary and functions, from which more detailed system requirements can be derived. The starting point for this process may be to identify functional requirements from the stakeholder requirements and to use this to start the architectural definition, or to begin with a high level functional architecture view and use this as the basis for structuring system requirements. The exact approach taken will often depend on whether the system is an evolution of an already understood product or service, or a new and unprecedented solution (see Synthesizing Possible Solutions). However, when the process is initiated it is important that the stakeholder requirements, system requirements, and logical architecture are all complete, consistent with each other, and assessed together at the appropriate points in the systems life cycle model.

#### Traceability and the Assignment of System Requirements during Architecture and Design

Requirements traceability provides the ability to track information from the origin of the stakeholder requirements, to the top level of requirements and other system definition elements at all levels of the system hierarchy (see Applying Life Cycle Processes). Traceability is also used to provide an understanding as to the extent of a change as an input when impact analyses is performed in cases of proposed engineering improvements or requests for change.

During architecture definition and design, the assignment of requirements from one level to lower levels in the system hierarchy can be accomplished using several methods, as appropriate - see Table 1.

**Table 1. Assessment Types for a System Requirement. (SEBoK Original)**

| Assignment Type for a System Requirement | Description |
|---|---|
| **Direct Assignment** | The system requirement from the higher level is directly assigned to a system or a system element for a lower level (e.g. the color used to paint visible parts of the product). |
| **Indirect Assignment (Simply Decomposed)** | The system requirement is distributed across several systems or system elements and the sum of a more complex calculation for distribution is equal to the requirement of higher level (e.g. a mass requirement, power distribution, reliability allocation, etc.) with sufficient margin or tolerance. A documented and configuration-managed "assignment budget" for each assignment must be maintained. |
| **Indirect Assignment (Modeled and Decomposed)** | The system requirement is distributed to several systems or system elements using an analysis or mathematical modeling technique. The resulting design parameters are assigned to the appropriate systems or system elements (with appropriate margin). For example, in the case of a radar detection requirement that is being analyzed, these lower-level parameters for output power, beam size, frequencies, etc. will be assigned to the appropriate hardware and software elements. Again, the analysis (or model) must be documented and configuration-managed. |
| **Derived Requirement (from Design)** | Such system requirements are developed during the design activities as a result of the decision of the design team, not the stakeholder community. These requirements may include the use of commercial-off-the-shelf (COTS) items, existing systems or system elements in inventory, common components, and similar design decisions in order to produce a "best value" solution for the customer. As such, these derived requirements may not directly trace to a stakeholder requirement, but they do not conflict with a stakeholder requirement or a constraint. |

## Classification of System Requirements

Several classifications of system requirements are possible, depending on the requirements definition methods and/or the architecture and design methods being applied. (ISO 2011) provides a classification which is summarized in Table 2 (see references for additional classifications).

**Table 2. Example of System Requirements Classification. (SEBoK Original)**

| Types of System Requirement | Description |
|---|---|
| **Functional Requirements** | Describe qualitatively the system functions or tasks to be performed in operation. |
| **Performance Requirements** | Define quantitatively the extent, or how well, and under what conditions a function or task is to be performed (e.g. rates, velocities). These are quantitative requirements of system performance and are verifiable individually. Note that there may be more than one performance requirement associated with a single function, functional requirement, or task. |
| **Usability Requirements** | Define the quality of system use (e.g. measurable effectiveness, efficiency, and satisfaction criteria). |
| **Interface Requirements** | Define how the system is required to interact or to exchange material, energy, or information with external systems (external interface), or how system elements within the system, including human elements, interact with each other (internal interface). Interface requirements include physical connections (physical interfaces) with external systems or internal system elements supporting interactions or exchanges. |
| **Operational Requirements** | Define the operational conditions or properties that are required for the system to operate or exist. This type of requirement includes: human factors, ergonomics, availability, maintainability, reliability, and security. |
| **Modes and/or States Requirements** | Define the various operational modes of the system in use and events conducting to transitions of modes. |
| **Adaptability Requirements** | Define potential extension, growth, or scalability during the life of the system. |
| **Physical Constraints** | Define constraints on weight, volume, and dimension applicable to the system elements that compose the system. |
| **Design Constraints** | Define the limits on the options that are available to a designer of a solution by imposing immovable boundaries and limits (e.g., the system shall incorporate a legacy or provided system element, or certain data shall be maintained in an online repository). |
| **Environmental Conditions** | Define the environmental conditions to be encountered by the system in its different operational modes. This should address the natural environment (e.g. wind, rain, temperature, fauna, salt, dust, radiation, etc.), induced and/or self-induced environmental effects (e.g. motion, shock, noise, electromagnetism, thermal, etc.), and threats to societal environment (e.g. legal, political, economic, social, business, etc.). |
| **Logistical Requirements** | Define the logistical conditions needed by the continuous utilization of the system. These requirements include sustainment (provision of facilities, level support, support personnel, spare parts, training, technical documentation, etc.), packaging, handling, shipping, transportation. |
| **Policies and Regulations** | Define relevant and applicable organizational policies or regulatory requirements that could affect the operation or performance of the system (e.g. labor policies, reports to regulatory agony, health or safety criteria, etc.). |
| **Cost and Schedule Constraints** | Define, for example, the cost of a single exemplar of the system, the expected delivery date of the first exemplar, etc. |

### Requirements Management

Requirements management is performed to ensure alignment of the system and system element requirements with other representations, analysis, and artifacts of the system. It includes providing an understanding of the requirements, obtaining commitment, managing changes, maintaining bi-directional traceability among the requirements and with the rest of the system definition, and alignment with project resources and schedule.

There are many tools available to provide a supporting infrastructure for requirements management; the best choice is the one that matches the processes of the project or enterprise. Requirements management is also closely tied to configuration management for baseline management and control. When the requirements have been defined, documented, and approved, they need to be put under baseline management and control. The baseline allows the project to analyze and understand the impact (technical, cost, and schedule) of ongoing proposed changes.

## Process Approach

### Purpose and Principle of the Approach

The purpose of the system requirements analysis process is to transform the stakeholder, user-oriented view of desired services and properties into a technical view of the product that meets the operational needs of the user. This process builds a representation of the system that will meet stakeholder requirements and that, as far as constraints permit, does not imply any specific implementation. It results in measurable system requirements that specify, from the supplier’s perspective, what performance and non-performance characteristics it must possess in order to satisfy stakeholders' requirements (ISO 2015).

### Activities of the Process

Major activities and tasks during this process include:

1. Analyzing the stakeholder requirements to check completeness of expected services and operational scenarios, conditions, operational modes, and constraints.
2. Defining the system requirements and their rationale.
3. Classifying the system requirements using suggested classifications (see examples above).
4. Incorporating the derived requirements (coming from architecture and design) into the system requirements baseline.
5. Establishing the upward traceability with the stakeholder needs and requirements.
6. Establishing bi-directional traceability between requirements at adjacent levels of the system hierarchy.
7. Verifying the quality and completeness of each system requirement and the consistency of the set of system requirements.
8. Validating the content and relevance of each system requirement against the set of stakeholder requirements.
9. Identifying potential risks (or threats and hazards) that could be generated by the system requirements.
10. Synthesizing, recording, and managing the system requirements and potential associated risks.
11. Upon approval of the requirements, establishing control baselines along with the other system definition elements in conjunction with established configuration management practices.

### Checking Correctness of System Requirements

System requirements should be checked to gauge whether they are well expressed and appropriate. There are a number of characteristics that can be used to check system requirements, such as standard peer review techniques and comparison of each requirement against the set of requirements characteristics, which are listed in Table 2 and Table 3 of the "Presentation and Quality of Requirements" section (below). Requirements can be further validated using the requirements elicitation and rationale capture described in the section "Methods and Modeling Techniques" (below).

## Methods and Modeling Techniques

### Requirements Elicitation and Prototyping

Requirements elicitation requires user involvement and can be effective in gaining stakeholder involvement and buy-in. Quality Function Deployment (QFD) and prototyping are two common techniques that can be applied and are defined in this section. In addition, interviews, focus groups, and Delphi techniques are often applied to elicit requirements.

QFD is a powerful technique to elicit requirements and compare design characteristics against user needs (Hauser and Clausing 1988). The inputs to the QFD application are user needs and operational concepts, so it is essential that the users participate. Users from across the life cycle should be included to ensure that all aspects of user needs are accounted for and prioritized.

Early prototyping can help the users and developers interactively identify functional and operational requirements as well as user interface constraints. This enables realistic user interaction, discovery, and feedback, as well as some sensitivity analysis. This improves the users' understanding of the requirements and increases the probability of satisfying their actual needs.

### Capturing Requirements Rationale

One powerful and cost-effective technique to translate stakeholder requirements to system requirements is to capture the rationale for each requirement. Requirements rationale is merely a statement as to why the requirement exists, any assumptions made, the results of related design studies, or any other related supporting information. This supports further requirements analysis and decomposition. The rationale can be captured directly in a requirements database (Hull, Jackson, and Dick 2010).

Some of the benefits of this approach include:

- **Reducing the total number of requirements** - The process aids in identifying duplicates. Reducing requirements count will reduce project cost and risk.
- **Early exposure of bad assumptions**
- **Removes design implementation** - Many poorly written stakeholder requirements are design requirements in disguise, in that the customer is intentionally or unintentionally specifying a candidate implementation.
- **Improves communication with the stakeholder community** - By capturing the requirements rationale for all stakeholder requirements, the line of communication between the users and the designers is greatly improved. (Adapted from Chapter 8 of (Hooks and Farry 2000)).

### Modeling Techniques

Modeling techniques that can be used when requirements must be detailed or refined, or in cases in which they address topics not considered during the stakeholder requirements definition and mission analysis, include:

- State-charts models (ISO 2011, Section 8.4)
- Scenarios modeling (ISO 2011, Section 6.2.3.1)
- Simulations, prototyping (ISO 2011, Section 6.3.3.2)
- Quality Function Deployment (INCOSE 2011, p. 83)
- Systems Modeling Language (SysML) sequence diagrams, activity diagrams, use cases, state machine diagrams, requirements diagrams (OMG 2010)
- Functional Flow Block Diagram for operational scenarios (Oliver, Kelliher, and Keegan 1997)

### Presentation and Quality of Requirements

Generally, requirements are provided in a textual form. Guidelines exist for writing good requirements; they include recommendations about the syntax of requirements statements, wording (exclusions, representation of concepts, etc.), and characteristics (specific, measurable, achievable, feasible, testable, etc.). Refer to (INCOSE 2011, Section 4.2.2.2) and (ISO 2011).

There are several characteristics of both requirements and sets of requirements that are used to aid their development and to verify the implementation of requirements into the solution. Table 3 provides a list and descriptions of the characteristics for individual requirements and Table 4 provides a list and descriptions of characteristics for a set of requirements, as adapted from (ISO 2011, Sections 5.2.5 and 5.2.6).

**Table 3. Characteristics of Individual Requirements. (SEBoK Original)**

| Characteristic | Description |
|---|---|
| **Necessary** | The requirement defines an essential capability, characteristic, constraint, and/or quality factor. If it is not included in the set of requirements, a deficiency in capability or characteristic will exist, which cannot be fulfilled by implementing other requirements |
| **Appropriate** | The specific intent and amount of detail of the requirement is appropriate to the level of the entity to which it refers (level of abstraction). This includes avoiding unnecessary constraints on the architecture or design to help ensure implementation independence to the extent possible |
| **Unambiguous** | The requirement is stated in such a way so that it can be interpreted in only one way. |
| **Complete** | The requirement sufficiently describes the necessary capability, characteristic, constraint, or quality factor to meet the entity need without needing other information to understand the requirement |
| **Singular** | The requirement should state a single capability, characteristic, constraint, or quality factor. |
| **Feasible** | The requirement can be realized within entity constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk |
| **Verifiable** | The requirement is structured and worded such that its realization can be proven (verified) to the customer's satisfaction at the level the requirements exists. |
| **Correct** | The requirement must be an accurate representation of the entity need from which it was transformed. |
| **Conforming** | The individual requirements should conform to an approved standard template and style for writing requirements, when applicable. |

Note: Traceability is considered by some sources as a characteristic (ISO 2011). However, a recent viewpoint is that Traceability is actually an attribute of a requirement; that is, something that is appended to the requirement, not an intrinsic characteristic of a requirement (INCOSE 2011). The traceability characteristic or attribute is defined as: The requirement is upwards traceable to specific documented stakeholder statement(s) of need, higher tier requirement, or another source (e.g., a trade or design study). The requirement is also downwards traceable to the specific requirements in the lower tier requirements specifications or other system definition artifacts. That is, all parent-child relationships for the requirement are identified in tracing such that the requirement traces to its source and implementation.

### Table 4. Characteristics of a Set of Requirements. (SEBoK Original)

| Characteristic | Description |
|---|---|
| **Complete** | The requirement set stands alone such that it sufficiently describes the necessary capabilities, characteristics, constraints, and/or quality factors to meet the entity needs without needing other information. In addition, the set does not contain any to be defined (TBD), to be specified (TBS), or to be resolved (TBR) clauses. |
| **Consistent** | The set of requirements contains individual requirements that are unique, do not conflict with or overlap with other requirements in the set, and the units and measurement systems they use are homogeneous. The language used within the set of requirements is consistent, i.e., the same word is used throughout the set to mean the same thing. |
| **Feasible** | The requirement set can be realized within entity constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk. (Note: Feasible includes the concept of "affordable".) |
| **Comprehensible** | The set of requirements must be written such that it is clear as to what is expected by the entity and its relation to the system of which it is a part. |
| **Able to be validated** | It must be able to be proven the requirement set will lead to the achievement of the entity needs within the constraints (such as cost, schedule, technical, legal and regulatory compliance). |

### Requirements in Tables

Requirements may be provided in a table, especially when specifying a set of parameters for the system or a system element. It is good practice to make standard table templates available. For tables, the following conventions apply:

- Invoke each requirements table in the requirements set that clearly points to the table.
- Identify each table with a unique title and table number.
- Include the word “requirements” in the table title.
- Identify the purpose of the table in the text immediately preceding it and include an explanation of how to read and use the table, including context and units.
- For independent-dependent variable situations, organize the table in a way that best accommodates the use of the information.
- Each cell should contain, at most, a single requirement.

### Requirements in Flow Charts

Flow charts often contain requirements in a graphical form. These requirements may include logic that must be incorporated into the system, operational requirements, process or procedural requirements, or other situations that are best defined graphically by a sequence of interrelated steps. For flow charts, the following conventions apply:

- Invoke flow charts in the requirements set that clearly points to the flow chart.
- Identify each flow chart with a unique title and figure number.
- Include the word “requirements” in the title of the flow chart.
- Clearly indicate and explain unique symbols that represent requirements in the flow chart.

### Requirements in Drawings

Drawings also provide a graphical means to define requirements. The type of requirement defined in a drawing depends on the type of drawing. The following conventions apply:

- Drawings are used when they can aid in the description of the following:
  - Spatial Requirements
  - Interface Requirements
  - Layout Requirements
- Invoke drawings in the requirements set that clearly point to the drawing.

### Artifacts

This process may create several artifacts, such as:

- System Requirements Document
- System Requirements Justification Document (for traceability purpose)
- System Requirements Database, including traceability, analysis, rationale, decisions, and attributes, where appropriate.
- System External Interface Requirements Document (this document describes the interfaces of the system with external elements of its context of use; the interface requirements can be integrated or not to the system requirements document.

The content, format, layout and ownership of these artifacts will vary depending on who is creating them as well as in which domain they will be utilized. Between them and the outputs of the process, activities should cover the information identified in the first part of this article.

## Practical Considerations about System Requirements

There are several **pitfalls** that will inhibit the generation and management of an optimal set of system requirements, as discussed in Table 5.

**Table 5. Major Pitfalls with Definition of System Requirements. (SEBoK Original)**

| Pitfall | Description |
|---|---|
| **Insufficient Analysis of Stakeholder Requirements** | If the receivers of the stakeholder requirements do not perform a sufficient critical analysis of them, the consequence could be difficulties translating them into system requirements and the obligation to come back to the stakeholders, losing time. |
| **Insufficient Analysis of Operational Modes and Scenarios** | The operational modes and operational scenarios are not sufficiently analyzed or defined by the person in charge of writing the system requirements. Those elements allow the structuring of the system and its use early in the engineering process and help the designer to remember functions and interfaces. |
| **Incomplete Set of System Requirements** | If the system requirements are not sufficiently precise and complete, there is a great risk that the design will not have the expected level of quality and that the verification and validation of the system will be delayed. |
| **Lack of Verification Method** | Delaying the capture of verification methods and events for each system requirement; identification of the verification approach for each requirement often provides additional insight as to the correctness and necessity of the requirement itself. |
| **Missing traceability** | Incorrect or missing traceability of each requirement, both to an upper-level "parent" requirement as well as allocation to an inappropriate system or system element. |

The **proven practices** in Table 6 have repeatedly been shown to reduce project risk and cost, foster customer satisfaction, and produce successful system development.

**Table 6. Proven Practices for System Requirements. (SEBoK Original)**

| Practice | Description |
|---|---|
| **Involve Stakeholders** | Involve the stakeholders as early as possible in the system requirements development process. |
| **Presence of Rationale** | Capture the rationale for each system requirement. |
| **Always Complete before Starting** | Check that stakeholder requirements are complete as much as possible before starting the definition of the system requirements. |
| **Peer Reviews** | Organize peer reviews of system requirements with applicable subject matter experts. |
| **Modeling Techniques** | Use modeling techniques as indicated in sections above. |
| **Requirements Management Tool** | Consider using a requirements management tool, especially for more complex projects. This tool should have the capability to trace linkages between system requirements to display relationships. A requirements management tool is intended to facilitate and support the systematic managing of system requirements throughout the project life cycle. |
| **Measures for Requirement Engineering** | Use typical measures for requirement engineering; for further information, refer to the *Systems Engineering Leading Indicators Guide* (Roedler et al. 2010). Both process and product measures should be used for requirements engineering. To get the desired insight to facilitate risk-managed requirements engineering, it may be necessary to use more than one measure based on the information needs (risks, objectives, issues) for the requirements. Useful measures include: <br>• Requirements Volatility<br>• Requirements Trends<br>• Requirements Verification Progress (plan vs. actual)<br>• Requirements Validation Progress (plan vs. actual)<br>• TBD and TBR Closure Per Plan<br>• Peer Review Defects |

## References

### Works Cited

Hauser, J. and D. Clausing. 1988. "The House of Quality." *Harvard Business Review.* (May - June 1988).

Hooks, I.F. and K.A. Farry. 2000. *Customer-centered products: Creating successful products through smart requirements management*. New York, NY, USA: American Management Association.

Hull, M.E.C., K. Jackson, A.J.J. Dick. 2010. *Systems Engineering,* 3rd ed. London, UK: Springer.

INCOSE. 2011. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.1.

ISO/IEC. 2007. *Systems and Software Engineering -- Recommended Practice for Architectural Description of Software-Intensive Systems*. Geneva, Switzerland: International Organization for Standards (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 42010:2007.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Requirements Engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Martin, J.N. 1997. *Systems Engineering Guidebook: A Process for Developing Systems and Products,* 1st ed. Boca Raton, FL, USA: CRC Press.

Oliver, D., T. Kelliher, and J. Keegan. 1997. *Engineering complex systems with models and objects*. New York, NY, USA: McGraw-Hill.

OMG. 2010. *OMG Systems Modeling Language specification*, version 1.2. Needham, MA: Object Management Group. July 2010.

## Primary References

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Requirements Engineering*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC/IEEE 15288:2015.

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

Lamsweerde, A. van. 2009. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. New York, NY, USA: Wiley.

## Additional References

Faisandier, A. 2012. *Systems Opportunities and Requirements*. Belberaud, France: Sinergy'Com.

Hooks, I.F. and K.A. Farry. 2000. *Customer-Centered Products: Creating Successful Products through Smart Requirements Management.* New York, NY, USA: American Management Association.

Hull, M.E.C., K. Jackson, A.J.J. Dick. 2010. *Systems Engineering,* 3rd ed. London, UK: Springer.

Roedler, G., D. Rhodes, C. Jones, and H. Schimmoller. 2010. *Systems Engineering Leading Indicators Guide,* version 2.0. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2005-001-03.

SEI. 2007. "Requirements Management Process Area" and "Requirements Development Process Area." in *Capability Maturity Model Integrated (CMMI) for Development*, version 1.2. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# System Architecture

***Lead Authors:*** *Alan Faisandier, Garry Roedler* **Contributing Author:** *Rick Adcock*

The purpose of system architecture activities is to define a comprehensive solution based on principles, concepts, and properties logically related and consistent with each other. The solution architecture has features, properties, and characteristics satisfying, as far as possible, the problem or opportunity expressed by a set of system requirements (traceable to mission/business and stakeholder requirements) and life cycle concepts (e.g., operational, support) and are implementable through technologies (e.g., mechanics, electronics, hydraulics, software, services, procedures, human activity).

System Architecture is abstract, conceptualization-oriented, global, and focused to achieve the mission and life cycle concepts of the system. It also focuses on high-level structure in systems and system elements. It addresses the architectural principles, concepts, properties, and characteristics of the system-of-interest. It may also be applied to more than one system, in some cases forming the common structure, pattern, and set of requirements for classes or families of similar or related systems.

## General Concepts and Principles

### Notion of Structure

The SEBoK considers systems engineering to cover all aspects of the creation of a system, including system architecture.

The majority of interpretations of system architecture are based on the fairly intangible notion of *structure* (i.e. relationships between elements). Some authors limit the types of structure considered to be architectural; for example, restricting themselves to *functional* and *physical* structure. Recent practice has extended consideration to include *behavioral*, *temporal* and other dimensions of structure.

ISO/IEC/IEEE 42010 *Systems and software engineering - Architecture description* (ISO 2011) provides a useful description of the architecture considering the stakeholder concerns, architecture viewpoints, architecture views, architecture models, architecture descriptions, and architecting throughout the life cycle.

A discussion of the features of systems architectures can be found in (Maier and Rechtin 2009).

An attempt to develop and apply a systematic approach to characterizing architecture belief systems in systems engineering has been described by the INCOSE UK Architecture Working Group (Wilkinson et al.2010, Wilkinson 2010).

### Architecture Description of the System

An architecture framework contains standardized viewpoints, view templates, meta-models, model templates, etc. that facilitate the development of the views of a system architecture (see architecture framework for examples). ISO/IEC/IEEE 42010 (ISO 2011) specifies the normative features of architecture frameworks, viewpoints, and views as they pertain to architecture description. A viewpoint addresses a particular stakeholder concern (or set of closely related concerns). The viewpoint specifies the kinds of model to be used in developing the system architecture to address that concern (or set of concerns), the ways in which the models should be generated, and how the models are related and used to compose a view.

Logical and physical models (or views) are often used for representing fundamental aspects of the system architecture. Other complementary viewpoints and views are necessarily used to represent how the system architecture addresses stakeholder concerns, for example, cost models, process models, rule models, ontological models, belief models, project models, capability models, data models, etc.

## Classification of Principles and Heuristics

Engineers and architects use a mixture of mathematical principles and heuristics (heuristics are lessons learned through experience, but not mathematically proven). When an issue is identified and defined through system requirements, principles and heuristics may or may not be able to address it. Principles and heuristics that are used in system views/models can be classified according to the domains in which those system views/models are used, as follows:

1. **Static domain** relates to physical structure or organization of the SoI broken down into systems and system elements. It deals with partitioning systems, system elements, and physical interfaces.
2. **Dynamic domain** relates to logical architecture models; in particular, to the representation of the behavior of the system. It includes a description of functions (i.e. transformations of input flows into output flows) and interactions between functions of the system and between those of the external objects or systems. It takes into account reactions to events that launch or stop the execution of functions of the system. It also deals with the effectiveness (i.e. performances, operational conditions) of the system.
3. **Temporal domain** relates to temporal invariance levels of the execution of functions of the system. This means that every function is executed according to cyclic or synchronous characteristics. It includes decisional levels that are asynchronous characteristics of the behavior of some functions.
4. **Environmental domain** relates to enablers (production, logistics support, etc.), but also to the survivability of the system in reaction to natural hazards or threats and to the integrity of the system in reaction to internal potential hazards. This includes, for example, climatic, mechanical, electromagnetic, and biological aspects.

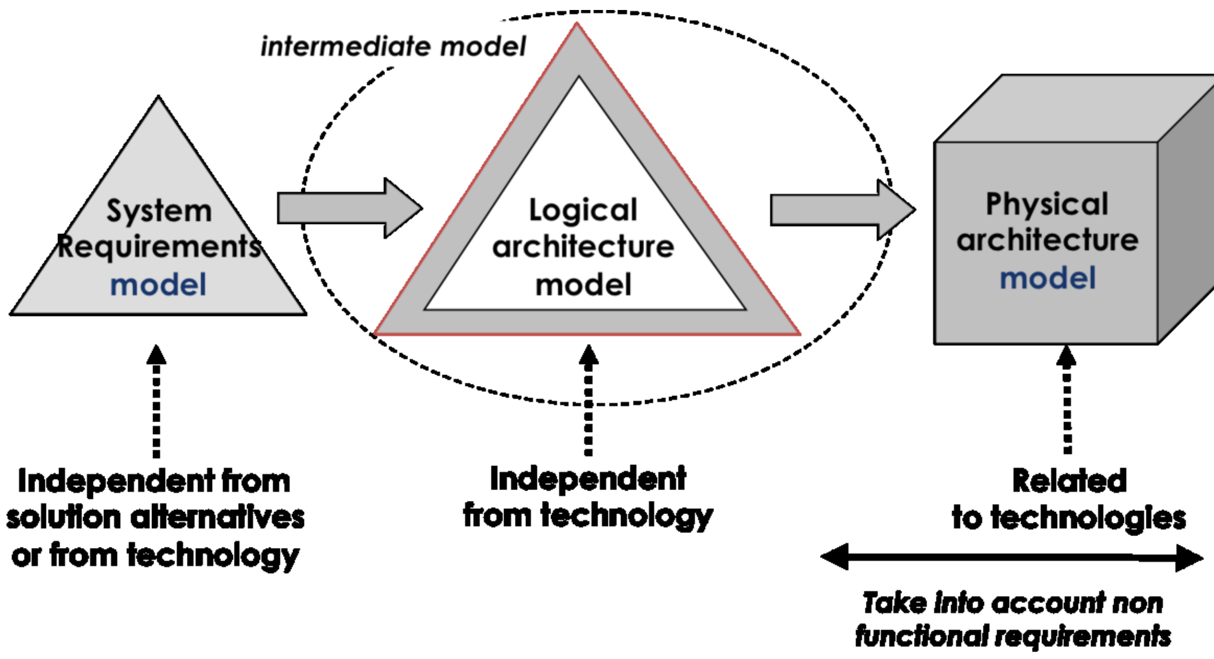
More detailed classification of heuristics can be found in (Maier and Rechtin 2009).

## Transition from System Requirements to Logical and Physical Architecture Models

The aim of the approach is to progress from system requirements (representing the problem from a supplier/designer point of view, as independent of technology as possible) through an intermediate model of logical architecture, to allocate the elements of the logical architecture model to system elements of candidate physical architecture models.

(System requirements and logical architecture models share many characteristics, as they are both organized on functional lines, independently of the implementation. Some authors (Stevens et al 1998) go so far as to conflate the two, which simplifies the handling of multiple simultaneous views. Whether this approach is adopted depends on the specific practices of the development organization and where contractual boundaries are drawn.)

Design decisions and technological solutions are selected according to performance criteria and non-functional requirements, such as operational conditions and life cycle constraints (e.g., environmental conditions, maintenance constraints, realization constraints, etc.), as illustrated in Figure 1. Creating intermediate models, such as logical architecture models, facilitates the validation of functional, behavioral, and temporal properties of the system against the system requirements that have no major technological influence impacts during the life of the system, the physical interfaces, or the technological layer without completely questioning the logical functioning of the system.


intermediate model
System Requirements model
Logical architecture model
Physical architecture model
Independent from solution alternatives or from technology
Independent from technology
Related to technologies
Take into account non functional requirements

## Iterations between Logical and Physical Architecture Model Development

As discussed in system requirements, the exact approach taken in the synthesis of solutions will often depend on whether the system is an evolution of an already understood product or service, or a new and unprecedented solution (see Synthesizing Possible Solutions).

Whatever the approach, architecture activities require spending several iterations between logical architecture models development and physical architecture models development, until both logical and physical architecture models are consistent and provide the necessary level of detail. One of the first architecture activities is the creation of a logical architecture model based on nominal scenarios (of functions). The physical architecture model is used to determine main system elements that could perform system functions and to organize them.

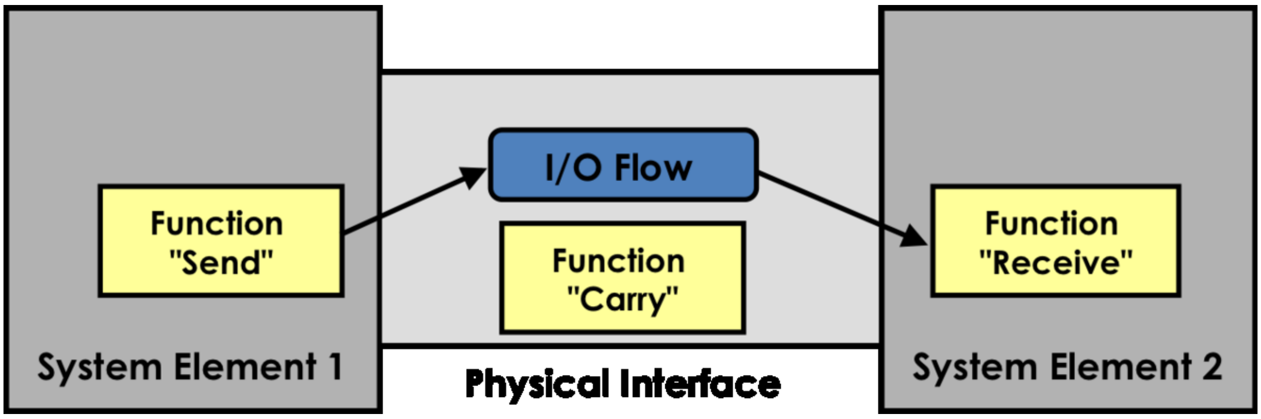
Subsequent logical architecture model iterations can take into account allocations of functions to system elements and derived functions coming from physical solution choices. It also supplements the initial logical architecture model by introducing other scenarios, failure analyses, and operational requirements not previously considered. Derived functions are allocated to system elements; in turn, this affects the physical architecture models.

Additional iterations are focused on producing complete and consistent logical and physical views of the solution.

During system design, technological choices can potentially lead to new functions, new input/output and control flows, and new physical interfaces. These new elements can lead to creation of new system requirements, called *derived requirements*.

## Notion of Interface

The notion of interface is one of the most important to consider when defining the architecture of a system. The fundamental aspect of an interface is functional and is defined as inputs and outputs of functions. As functions are performed by physical elements (system elements), inputs/outputs of functions are also carried by physical elements; these are called physical interfaces. Consequentially, both functional and physical aspects are considered in the notion of interface. A detailed analysis of an interface shows the function *“send”* located in one system element, the function *“receive”* located in the other one, and the function *“carry"* as being performed by the physical interface that supports the input/output flow (see Figure 2).



In the context of complex exchanges between system elements, particularly in software-intensive systems, a protocol is seen as a physical interface that carries exchanges of data. However, the input/output flows can include many other exchanges than data, such as energy.

## Emergent Properties

The overarching architecture of a system may have design properties or operational effects that emerge from the arrangement and interaction between system elements, but which may not be properties of any individual element or intended for the system as a whole.

The elements of an engineered system interact among themselves and can create desirable or undesirable phenomena, such as inhibition, interference, resonance, or the reinforcement of any property. The definition of the system includes an analysis of interactions between system elements in order to prevent undesirable properties and reinforce desirable ones.

A property which emerges from a system can have various origins, from a single system element to the interactions among several elements (Thome, B. 1993). The term emergent properties is used by some authors to identify any property which emerges from a system, while other may refer to this as synergy and reserve emergent property for explaining unexpected properties or properties not considered fully during system development, but have emerged during operation. The system concept of emergence is discussed in SEBoK Part 2 (see Emergence).

| Broad Categories of Properties | Description and Examples |
|---|---|
| **Local Property** | The property is located in a single system element – e.g. the capacity of a container is the capacity of the system. |
| **Accumulative System Property** | The property is located in several system elements and is obtained through the simple summation of elemental properties – e.g. the weight of the system results from the sum of the weights of its system elements. |
| **Emergent Property Modified by Architecture and/or Interactions.** | The property exists in several system elements and is modified by their interactions – e.g. the reliability/safety of a system results from the reliability/safety of each system element and the way they are organized. Architectural steps are often critical to meeting system requirements. |
| **Emergent Property Created by Interactions** | The property does not exist in system elements and results only from their interactions – e.g. electromechanical interfaces, electromagnetism, static electricity, etc. |
| **Controlled Emergent Property** | Property controlled or inhibited before going outside the system – e.g.: unbalance removed by the addition of a load; vibration deadened by a damper. |

Physical architecture design will include the identification of likely synergies and emergent properties and the inclusion of derived functions, components, arrangements, and/or environmental constraints in the logical or physical architectures models to avoid, mitigate or restrain them within acceptable limits. Corresponding *derived requirements* should be added to the system requirements baseline when they impact the system-of-interest(SoI). This may be achieved through the knowledge and experience of the systems engineer or through the application of system patterns. However, it is generally not possible to predict, avoid, or control all emergent properties during the architecture development. Fully dealing with the consequences of emergence can only be done via iteration between system definition, system realization and system deployment and use (Hitchins, 2008)

The notion of emergence is applied during architecture and design to highlight necessary derived functions; additionally, internal emergence is often linked to the notion of complexity. This is the case with complex adaptive systems (CAS), in which the individual elements act independently, but behave jointly according to common constraints and goals (Flood and Carson 1993). Examples of CAS include: the global macroeconomic network within a country or group of countries, stock market, complex web of cross border holding companies, manufacturing businesses, geopolitical organizations, etc. (Holland, J. 1999 and 2006).

### Reuse of System Elements

Systems engineers frequently utilize existing system elements. This reuse constraint has to be identified as a system requirement and carefully taken into account during architecture and design. One can distinguish three general cases involving system element reuse, as shown in Table 2.

| Re-use Case | Actions and Comments |
|---|---|
| **Case 1:**The requirements of the system element are up-to-date and it will be re-used with no modification required. | • The system architecture to be defined will have to adapt to the boundaries, interfaces, functions, effectiveness, and behavior of the re-used system element.<br>• If the system element is not adapted, it is probable that costs, complexity, and risks will increase. |
| **Case 2:**The requirements of the system element are up-to-date and it will be re-used with possible modifications. | • The system architecture to be defined is flexible enough to accommodate the boundaries, interfaces, functions, effectiveness, and behavior of the re-used system element.<br>• The design of the reused system element, including its test reports and other documentation, will be evaluated and potentially redesigned. |
| **Case 3:**The requirements are not up-to-date or do not exist. | • It is necessary to reverse engineer the system element to identify its boundaries, interfaces, functions, performances, and behavior.<br>• This is a difficult activity, since the extant documentation for the re-used system element is likely unavailable or insufficient.<br>• Reverse engineering is expensive in terms of both time and money, and brings with it increased risk. |

There is a common idea that reuse is *free*; however, if not approached correctly, reuse may introduce risks that can be significant for the project (costs, deadlines, complexity).

## Process Approach

### Purpose

The purpose of the System Architecture process is to generate system architecture alternatives, to select one or more alternative(s) that frame stakeholder concerns and meet system requirements, and to express this in a set of consistent views. (ISO 2015).

It should be noted that the architecture activities below overlap with both system definition and concept definition activities. In particular that key aspects of the operational and business context, and hence certain stakeholder needs, strongly influence the approach taken to architecture development and description. Also, the architecture activities will drive the selection of, and fit within, whatever approach to solution synthesis has been selected.

## Activities of the process

Major activities and tasks performed during this process include the following:

**1. Initialize the definition of the system architecture**

- Build an understanding of the environment/context of use for which a system is needed in order to establish insight into the stakeholder concerns. To do this, analyze relevant market, industry, stakeholder, enterprise, business, operations, mission, legal and other information that help to understand the perspectives that could guide the definition of the system architecture views and models.
- Capture stakeholder concerns (i.e., expectations or constraints) that span system life cycle stages. The concerns are often related to critical characteristics to the system that relate to the stages; they should be translated into or incorporated to system requirements.
- Tag system requirements that deal with operational conditions (e.g., safety, security, dependability, human factors, interfaces, environmental conditions) and life cycle constraints (e.g., maintenance, disposal, deployment) that would influence the definition of the architecture elements.
- Establish an architecture roadmap and strategy that should include methods, modeling techniques, tools, need for any enabling systems, products, or services, process requirements (e.g., measurement approach and methods), evaluation process (e.g., reviews and criteria).
- Plan enabling products or services acquisition (need, requirements, procurement).

**2. Define necessary architecture viewpoints**

- Based on the identified stakeholder concerns, identify relevant architecture viewpoints and architecture frameworks that may support the development of models and views.

**3. Develop candidate architectures models and views**

- Using relevant modeling techniques and tools, and in conjunction with the Stakeholder Needs and Requirements process and the System Requirements process, determine the system-of-interest context including boundary with elements of the external environment. This task includes the identification of relationships, interfaces or connections, exchanges and interactions of the system-of-interest with external elements. This task enables to define or understand the expected operational scenarios and/or system behaviors within its context of use.
- Define architectural entities (e.g., functions, input/output flows, system elements, physical interfaces, architectural characteristics, information/data elements, containers, nodes, links, communication resources, etc.), which address the different types of system requirements (e.g., functional requirements, interface requirements, environmental requirements, operational conditions – dependability, human factors, etc., constraints – physical dimensions, production, maintenance, disposal).
- Relate architectural entities to concepts, properties, characteristics, behaviors, functions, and/or constraints that are relevant to decisions of the system-of-interest architecture. This gives rise to architectural characteristics (e.g., generality, modularity, operability, efficiency, simplicity).
- Select, adapt, or develop models of the candidate architectures of the system, such as logical and physical models (see **Logical Architecture Model Development** and **Physical Architecture Model Development**). It is sometimes neither necessary nor sufficient to use logical and physical models. The models to be used are those that best address key stakeholder concerns.
- From the models of the candidate architectures, compose views that are relevant to the stakeholder concerns and critical or important requirements.
- Define derived system requirements induced by necessary instances of architectural entities (e.g., functions, interfaces) and by structural dispositions (e.g., constraints, operational conditions). Use the system requirements definition process to define and formalize them.

- Check models and views consistency and resolve any identified issues. ISO/IEC/IEEE 42010, 2011 maybe used for this.
- Verify and validate the models by execution or simulation, if modeling techniques and tools permit. Where possible, use design tools to check feasibility and validity; and/or implement partial mock-ups, or use executable architecture prototypes or simulators.

**4. Relate system architecture to system design**

- Define the system elements that reflect the architectural characteristics (when the architecture is intended to be design-agnostic, these system elements may be notional until the design evolves). To do this, partition, align, and allocate architectural characteristics and system requirements to system elements. Establish guiding principles for the system design and evolution. Sometimes, a “reference architecture” is created using these notional system elements as a means to convey architectural intent and to check for design feasibility.
- Define interfaces for those that are necessary for the level of detail and understanding of the architecture. This includes the internal interfaces between the system elements and the external interfaces with other systems.
- Determine the design properties applicable to system elements in order to satisfy the architectural characteristics.
- For each system element that composes the system, develop requirements corresponding to allocation, alignment, and partitioning of design properties and system requirements to system elements. To do this, use the stakeholder needs and requirements definition process and the system requirements definition process.

**5. Assess architecture candidates and select one**

- Assess the candidate architectures using the architecture evaluation criteria. This is done through application of the System Analysis, Measurement, and Risk Management processes.
- Select the preferred architecture(s). This is done through application of the Decision Management process.

**6. Manage the selected architecture**

- Establish and maintain the rationale for all selections among alternatives and decision for the architecture, architecture framework(s), viewpoints, kinds of models, and models of the architecture.
- Manage the maintenance and evolution of the architecture description, including the models, and views. This includes concordance, completeness, and changes due to environment or context changes, technological, implementation, and operational experiences. Allocation and traceability matrices are used to analyze impacts onto the architecture. The present process is performed at any time evolutions of the system occur.
- Establish a means for the governance of the architecture. Governance includes the roles, responsibilities, authorities, and other control functions.
- Coordinate reviews of the architecture to achieve stakeholder agreement. The stakeholder requirements and system requirements can serve as references.

## Artifacts, Methods and Modeling Techniques

This process may create several artifacts, such as system architecture description documents and system justification documents (traceability matrices and architectural choices).

The content, format, layout, and ownership of these artifacts may vary depending on the person creating them and the domains in which they are being used. The outputs of the process activities should cover the information identified in the first part of this article.

## Practical Considerations

### Pitfalls

Some of the key pitfalls encountered in planning and performing system architecture are provided in Table 3.

| Pitfall | Description |
|---|---|
| **Problem Relevance** | If the architecture is developed without input from the stakeholders' concerns, or cannot be understood and related back to their issues it might lose the investments of the stakeholder community. |
| **Reuse of System Elements** | In some projects, for industrial purposes, existing products or services are imposed very early as architecture/design constraints in the stakeholder requirements or in the system requirements, without paying sufficient attention to the new context of use of the system in which they are also included. It is better to work in the right direction from the beginning. Define the system first, taking note of other requirements, and then see if any suitable non-developmental items (NDI) are available. Do not impose a system element from the beginning, which would reduce the trade-space. The right reuse process consists of defining reusable system elements in every context of use. |

### Proven Practices

Some proven practices gathered from the references are provided in Table 4.

| Practice | Description |
|---|---|
| **Emerging properties** | Control the emergent properties of the interactions between the systems or the system elements; obtain the required synergistic properties and control or avoid the undesirable behaviors (vibration, noise, instability, resonance, etc.). |

## References

### Works Cited

Faisandier, A. 2012. *Systems Architecture and Design.* Belberaud, France: Sinergy'Com.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes.* Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Maier, M., and E. Rechtin. 2009. *The Art of Systems Architecting.* 3rd ed. Boca Raton, FL, USA: CRC Press.

Wilkinson, M., A. James, M. Emes, P. King, P. Bryant. 2010. “Belief Systems in Systems Architecting: Method and Preliminary Applications." Presented at the IEEE SMC Society’s 5th International Conference on System of Systems Engineering (SoSE). 22nd-24th June 2010. Loughborough University, UK.

Flood, R.L., and E.R. Carson. 1993. *Dealing with complexity: An Introduction to the Theory and Application of Systems Science*, 2nd ed. New York, NY, USA: Plenum Press

Holland, J.H. 1999. *Emergence: from chaos to order*. Reading, Mass: Perseus Books.

Hitchins, D. 2008. "Emergence, Hierarchy, Complexity, Architecture: How do they all fit together? A guide for seekers after enlightenment." Self-published white paper. Accessed 4 September 2012. Available at: http://www.hitchins.net/EmergenceEtc.pdf.

Holland, J.H. 2006. "Studying Complex Adaptive Systems." *Journal of Systems Science and Complexity.* 19(1): 1-8. http://hdl.handle.net/2027.42/41486

Thome, B. 1993. *Systems Engineering, Principles & Practice of Computer-Based Systems Engineering*. New York, NY, USA: Wiley.

## Primary References

ANSI/IEEE. 2000. *Recommended Practice for Architectural Description for Software-Intensive Systems*. New York, NY, USA: American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE), ANSI/IEEE 1471-2000.

INCOSE. 2015. 'Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities', version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Faisandier, A. 2012. *Systems Architecture and Design.*Belberaud, France: Sinergy'Com.

Blanchard, B.S., and W.J. Fabrycky. 2005. *Systems Engineering and Analysis.*4th ed. Prentice-Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

ISO/IEC. 2007. *Systems Engineering – Application and Management of The Systems Engineering Process*. Geneva, Switzerland: International Organization for Standards (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 26702:2007.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Architecture Description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Martin, J.N. 1997. *Systems Engineering Guidebook: A process for developing systems and products,*1st ed. Boca Raton, FL, USA: CRC Press.

NASA. 2007. *Systems Engineering Handbook.*Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

## Additional References

Checkland, P. B. 1999. *Systems Thinking, Systems Practice*. Chichester, UK: John Wiley & Sons Ltd.

OMG. 2010. *OMG Systems Modeling Language specification*, version 1.2, July 2010. http://www.omg.org/technology/documents/spec_catalog.htm.

Sillitto H, "Architecting Systems - Concepts, Principles and Practice", College Publications 2014

Wilkinson, M.K. 2010. “Z8: Systems Architecture”, in Z-guide series. INCOSE UK, available from INCOSE UK at: http://www.incoseonline.org.uk/Program_Files/Publications/zGuides.aspx?CatID=Publications.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Logical Architecture Model Development

***Lead Authors:*** *Alan Faisandier, Garry Roedler*, **Contributing Author:** *Rick Adcock*

Logical Architecture Model Development may be used as a task of the activity "Develop candidate architectures models and views", or a sub-process of the System Architecture Definition process (see **System Architecture**). Its purpose is to elaborate models and views of the functionality and behavior of the future engineered system as it should operate, while in service. The logical architecture model of a engineered system of interest (SoI) is composed of a set of related technical concepts and principles that support the logical operation of the system. It may include a functional architecture view, a behavioral architecture view, and a temporal architecture view. Other additional views are suggested in architecture frameworks, depending on the domain.

Note: The term *Logical Architecture* is a contraction of the expression *Logical View of the System Architecture*.

## Concepts and Principles

### Functional Architecture Model

A functional architecture model is a set of functions and their sub-functions that defines the transformations performed by the system to complete its mission.

**Function and Input-Output Flow** - In the context of System Architecture, functions and input-output flows are architecture entities. A function is an action that transforms inputs and generates outputs, involving data, materials, and/or energies. These inputs and outputs are the flow items exchanged between functions. The general mathematical notation of a function is $\boldsymbol{y} = f(\boldsymbol{x}, t)$, in which **y** and **x** are vectors that may be represented graphically and t = time.

In order to define the complete set of functions of the system, one must identify all the functions necessitated by the system and its derived requirements, as well as the corresponding inputs and outputs of those functions. Generally speaking, there are two kinds of functions:

1. Functions that are directly deduced from functional and interface requirements. These functions express the expected services of a system necessary to meet its system requirements.
2. Functions that are derived and issued from the alternative solutions of physical architecture model and are dependent upon the result of the design; additionally, they rely upon on technology choice to implement the logical architecture model elements.

**Functional Hierarchy/Decomposition of Functions** - At the highest level of a hierarchy (Figure 1), it is possible to represent a system as a unique, central function (defined as the system's mission) that in many ways is similar to a "black box" ("F0" in plan A-0 in Figure 1). In order to understand, in detail, what the system does, this "head-of-hierarchy" (F0) is broken down into sub-functions (F1, F2, F3, F4) grouped to form a sub-level of the hierarchy (plan A0), and so on. Functions of the last level of a functional hierarchy can be called leaf-functions (F21, F22, F23, F24 in plan A2). Hierarchies (or breakdowns) decompose a complex or global function into a set of functions for which physical solutions are known, feasible, or possible to imagine.

This view of functional hierarchy represents a static view of functions which would be populated at different levels over a number of iteration, depending upon the synthesis approach used. In general, it is not created by a single top-down decomposition. A static functional hierarchy on its own does not represent how effectively the flows of inputs and outputs are exchanged, and may need to be viewed alongside the other models below.

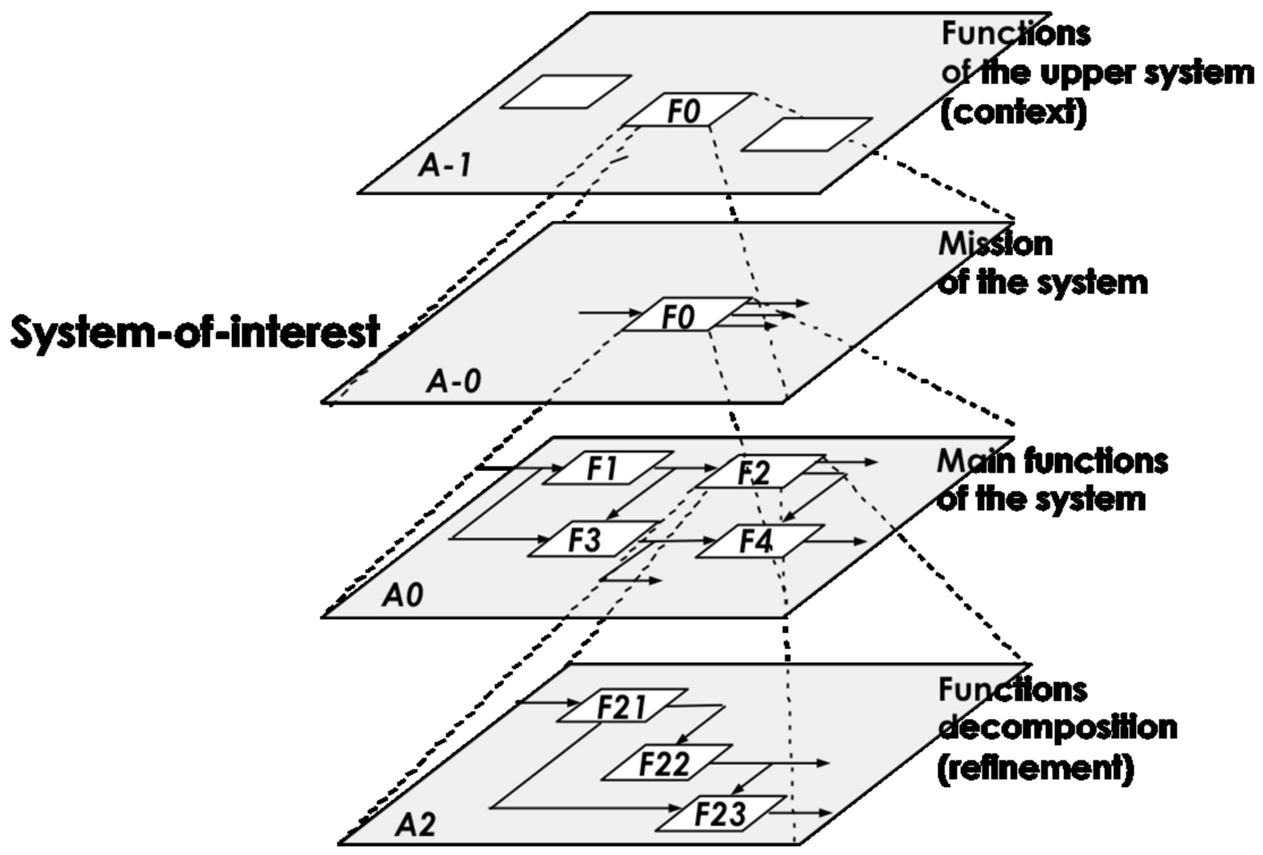


**Figure 1. Decomposition of Functions (Faisandier 2012).** Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

## Behavioral Architecture Model

A behavioral architecture model is an arrangement of functions and their sub-functions as well as interfaces (inputs and outputs) that defines the execution sequencing, conditions for control or data-flow, and performance level necessary to satisfy the system requirements ISO/IEC/IEEE 26702 (ISO 2007). A behavioral architecture model can be described as a set of inter-related scenarios of functions and/or operational modes.

**Control (Trigger)** - A control flow is an element that activates a function as a condition of its execution. The state of this element, or the condition it represents, activates or deactivates the function (or elements thereof). A control flow can be a signal or an event, such as a switch being moved to the *on* position, an alarm, a trigger, a temperature variation, or the push of a key on a keyboard.

**Scenario (of Functions)** - A scenario of functions is a chain of functions that are performed as a sequence and synchronized by a set of control flows to work to achieve a global transformation of inputs into outputs, as seen in the figures below. A scenario of functions expresses the dynamic of an upper level function. A behavioral architecture is developed by considering both scenarios for each level of the functional hierarchy and for each level of the system hierarchy. When representing scenarios of functions and behavioral architecture models, it is appropriate to use diagrams as modeling techniques, such as functional flow block diagrams (FFBD) (Oliver, Kelliher, and Keegan 1997) or activity diagrams, developed with SysML (OMG 2010). Figures 2 and 3 provide examples of these diagrams.

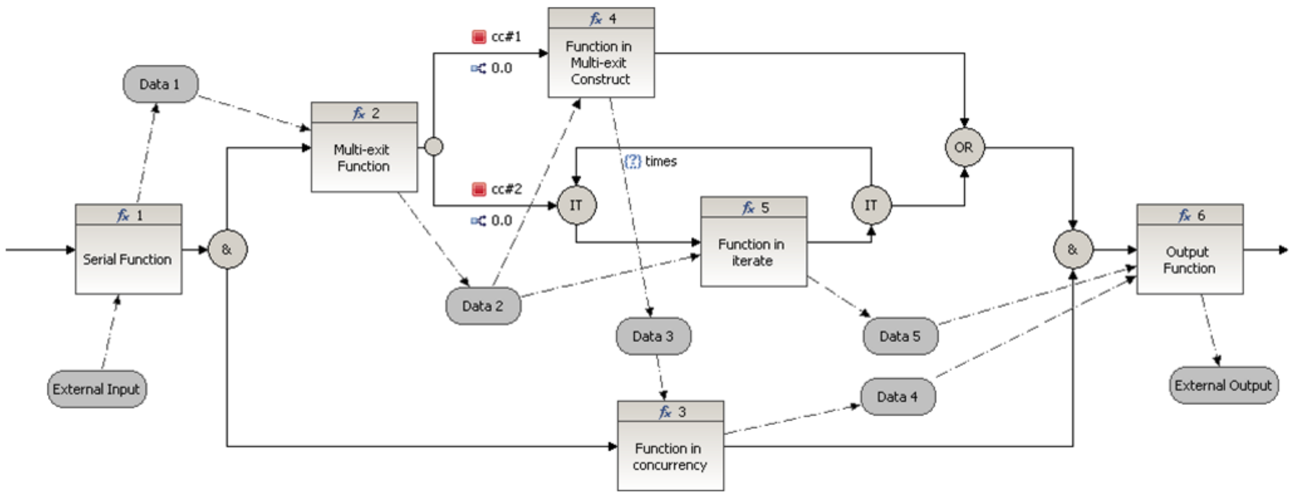


**Figure 2. Illustration of a Scenario (eFFBD).** (SEBoK Original)

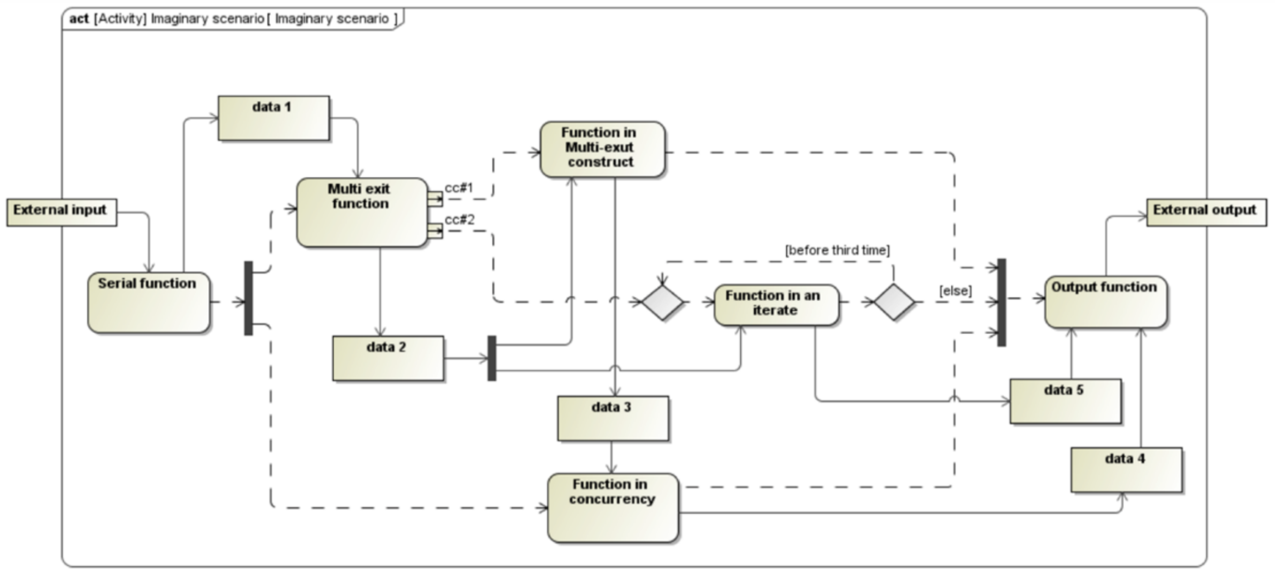


**Figure 3. Illustration of a Scenario (Activity Diagram).** (SEBoK Original)

**Operational Mode** - A scenario of functions can be viewed by abstracting the transformation of inputs into outputs of each function and focusing on the active or non-active state of the function and its controls. This view is called a *scenario of modes*, which is a chain of modes performed as a sequence of transitions between the various modes of the system. The transition from one mode to another is triggered by the arrival of a control flow (event/trigger). An action (function) can be generated within a transition between two modes following the arrival of an event or a trigger, as demonstrated in Figure 4 below.

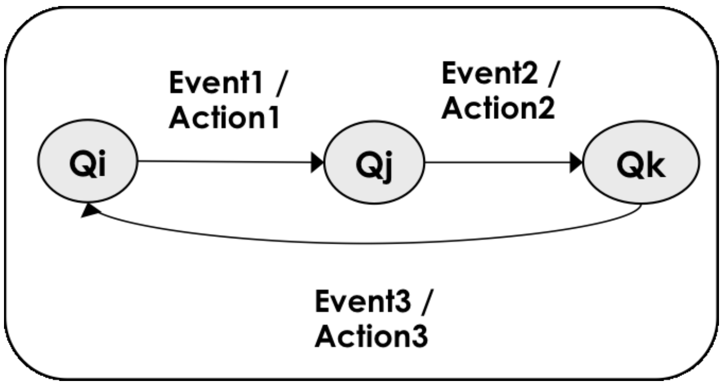


**Figure 4. Scenario of Operational Modes (Faisandier 2012).** Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

**Behavioral Patterns** - When defining scenarios or behavioral architecture models, architects may opt to recognize and use known models to represent the expected transformations and behaviors. Patterns are generic basic models that may be more or less sophisticated depending on the complexity of the treatment. (Gamma, Helm, Johnson, and Vlissides 1995) A pattern can be represented with different notations. Behavioral patterns are classified into several categories, which can be seen in the following examples (see also SEBoK Part 2: Patterns of Systems Thinking):

- Basic patterns or constructs linking functions - such as sequence, iteration, selection, concurrence, multiple exits, loops with an exit, and replication.
- Complex patterns - such as monitoring a treatment, exchanging a message, man machine interfaces, modes monitoring, real-time monitoring of processes, queue management, and continuous monitoring with supervision.
- Failure detection, identification, and recovery (FDIR) patterns - such as passive redundancies, active redundancies, semi-active redundancies, and treatments with reduced performance.

## Temporal Architecture Model

A temporal architecture model is a classification of the functions of a system that is derived according to the frequency level of execution. Temporal architecture models include the definition of synchronous and asynchronous aspects of functions. The decision monitoring that occurs inside a system follows the same temporal classification because the decisions are related to the monitoring of functions.

**Temporal and Decisional Hierarchy Concept** - Not every function of a system is performed at the same frequency. The frequencies change depending on the time and the manner in which the functions are started and executed. One must therefore consider several classes of performance. There are synchronous functions that are executed cyclically and asynchronous functions that are executed following the occurrence of an event or trigger.

To be more specific, *real-time* systems and *command-control* systems combine cyclical operations (synchronous) and factual aspects (asynchronous). Cyclical operations consist of sharing the execution of functions according to frequencies, which depend on either the constraints of capture or dispatching the input/output and control flows. Two types of asynchronous events can be distinguished:

1. Disturbances on High Frequencies (bottom of figure 5) - Decisions that are made at either the level they occur or one level above. The goal is to deter disturbances from affecting the low frequencies so that the system continues to achieve its mission objectives. This is the way to introduce exception operations, with the typical example relating to operations concerns, breakdowns, or failures.
2. Changes on Low Frequencies (top of figure 5) - Decisions pertaining to changes that are made at the upper levels. The ultimate goal is to transmit them toward bottom levels to implement the modifications. A typical example relates to operator actions, maintenance operations, etc.

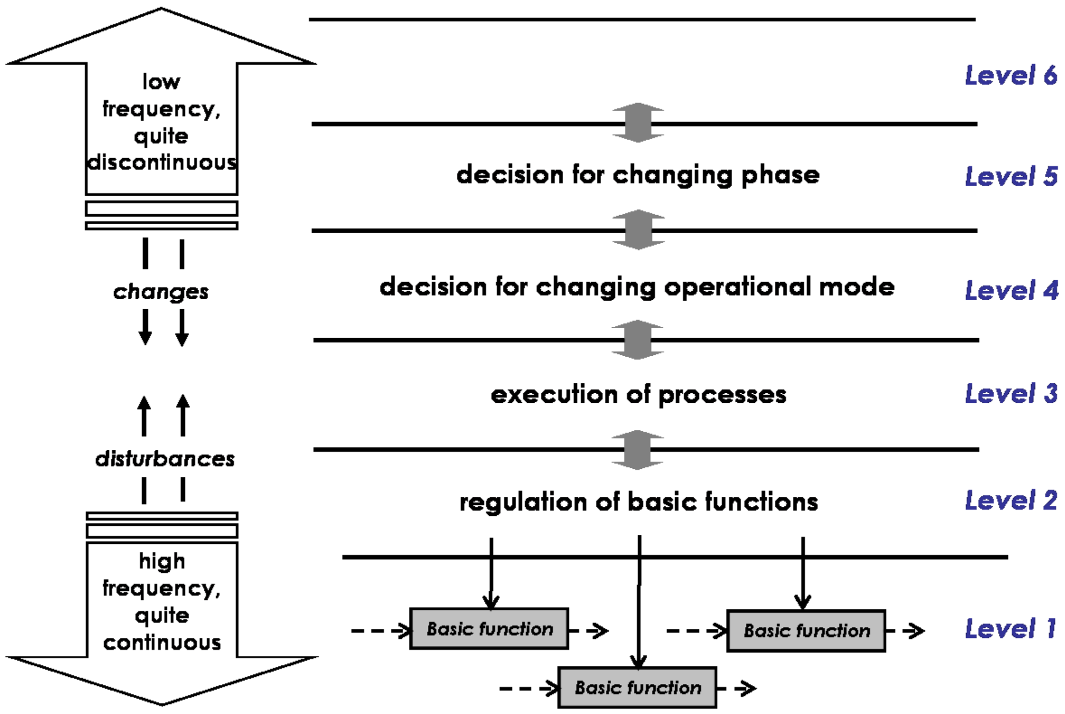

low frequency, quite discontinuous
changes
disturbances
high frequency, quite continuous
Level 6
decision for changing phase
Level 5
decision for changing operational mode
Level 4
execution of processes
Level 3
regulation of basic functions
Level 2
Basic function
Basic function
Basic function
Level 1

**Figure 5. Temporal and Decision Hierarchy Levels (Faisandier 2012).** Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

## Process Approach

### Purpose

The purpose of the Logical Architecture Model Development is to define, select, and synthesize a system's logical architecture model to provide a framework against which to verify that a future system will satisfy its system requirements in all operational scenarios, within which trade-offs between system requirements can be explored in developing such systems.

Generic inputs to the process include system requirements, generic architecture patterns that architects identify and use to answer requirements, outcomes from system analysis processes, and feedback from system verification and validation processes. Depending on the Life Cycle Model that is chosen, there will be iterations through which these inputs and outputs, and the relationships between them evolve and change throughout the process (see also Applying Life Cycle Processes).

Generic outputs from the process are either a single logical architecture model or a set of candidate logical architecture models together with the selected independent logical architecture model and a rationale for its selection. They include, at minimum, views and models. These involve functional, behavioral and temporal views; a traceability matrix between logical architecture model elements and system requirements.

## Activities of the Process

Major activities and tasks performed during this process include the following:

- Identify and analyze functional and behavioral elements:
  - Identify functions, input-output flows, operational modes, transition of modes, and operational scenarios from system requirements by analyzing the functional, interface, and operational requirements.
  - Define necessary inputs and controls (energy, material, and data flows) to each function and outputs that result in the deduction of the necessary functions to use, transform, move, and generate the input-output flows.
- Assign system requirements to functional and behavioral elements:
  - Formally characterize functions expressions and their attributes through the assignment of performance, effectiveness, and constraints requirements. In particular, study the temporal aspects from requirements to assign duration, response time, and frequency to functions.
  - Formally characterize the input, output, and control flows expressions and their attributes through assignment of interface, effectiveness, operational, temporal and constraints requirements.
  - Establish traceability between system requirements and these functional and behavioral elements.
- Define candidate logical architecture models For each candidate:
  - Analyze operational modes as stated in the system requirements (if any) and/or use previously defined elements to model sequences of operational modes and the transition of modes. Eventually decompose the modes into sub-modes and then establish for each operational mode one or several scenarios of functions recognizing and/or using relevant generic behavioral patterns.
  - Integrate these scenarios of functions in order to get a behavioral architecture model of the system (a complete picture of the dynamic behavior).
  - Decompose previously defined logical elements as necessary to look towards implementation.
  - Assign and incorporate temporal constraints to previously defined logical elements, such as the period of time, duration, frequency, response-time, timeout, stop conditions, etc.
  - Define several levels of execution frequency for functions that correspond to levels of decision, in order to monitor system operations, prioritize processing on this time basis, and share out functions among those execution frequency levels to get a temporal architecture model.
  - Perform functional failure modes and effects analysis and update the logical architecture elements as necessary.
  - Execute the models with simulators (when possible) and tune these models to obtain the expected characteristics.
- Synthesize the selected independent logical architecture model:
  - Select the logical architecture by assessing the candidate logical architecture models against assessment criteria (related to system requirements) and compare them, using the system analysis process to perform assessments and decision management process for the selection (see the System Analysis and Decision Management topics). This selected logical architecture model is called *independent logical architecture model* because, as much as possible, it is independent of implementation decisions.
  - Identify and define derived logical architecture model elements created for the necessity of design and corresponding with the derived system requirements. Assign these requirements to the appropriate system (current studied system or external systems).
  - Verify and validate the selected logical architecture models (using as executable models as possible), make corrections as necessary, and establish traceability between system requirements and logical architecture model elements.
- Feedback logical architecture model development and system requirements. This activity is performed after the physical architecture model development process:

- Model the *allocated logical architecture* to systems and system elements, if such a representation is possible, and add any functional, behavioral, and temporal elements as needed to synchronize functions and treatments.
- Define or consolidate derived logical and physical elements induced by the selected logical and physical architecture models. Define the corresponding derived requirements and allocate them to appropriate logical and physical architectures elements. Incorporate these derived requirements into the requirements baselines of impacted systems.

### Artifacts, Methods and Modeling Techniques

Logical architecture descriptions use modeling techniques that are grouped under the following types of models. Several methods have been developed to support these types of models (some are executable models):

- Functional Models – These include models such as the structured analysis design technique (SADT/IDEF0), system analysis & real time (SA-RT), enhanced Functional Flow Block Diagrams (eFFBD), and the function analysis system technique (FAST).
- Semantic Models- These include models such as entities-relationships diagrams, class diagrams, and data flow diagrams.
- Dynamic Models – These include such models as state-transition diagrams, state-charts, eFFBDs, state machine diagrams (SysML), activity diagrams (SysML) (OMG. 2010), and petri nets.

Depending on the type of domain (e.g. defense, enterprise), architecture frameworks provide descriptions that can help to represent additional aspects/views of architectures - see the section 'Enterprise Architecture Frameworks & Methodologies' in Enterprise Systems Engineering Key Concepts. See also practical means for using general templates related to ISO/IEC/IEEE 42010 (ISO 2011).

## Practical Considerations

As stated above, the purpose of the logical architecture model is to provide a description of what a system must be able to do to satisfy the stated need. This should help to ensure that the needs and/or concerns of all stakeholders are addressed by any solution, and that innovative solutions, as well as those based on current solution technologies, can be considered. In practice it is human nature for problem stakeholders to push their own agendas and for solution architects or designers to offer their familiar solutions. If a logical architecture model is not properly enforced with the chosen life cycle, it is easy for both problem and solution stakeholders to ignore it and revert to their own biases (see Part 5 Enabling Systems Engineering). This is exacerbated if the logical architecture model becomes an end in its own right or disconnected from the main lifecycle activities. This can occur either through the use of abstract language or notations, levels of detail, time taken, or an overly complex final architecture that does not match the purpose for which it was created. If the language, scope, and timeliness of the architecture are not matched to the problem stakeholder or solution providers, it is easier for them to overlook it. Key pitfalls and good practices which can help to avoid problems related to logical architecture model are described in the next two sections.

### Pitfalls

Some of the key pitfalls encountered in developing logical architecture are provided in Table 1.

## Table 1. Pitfalls with Logical Architecture Development. (SEBoK Original)

| Pitfall | Description |
|---|---|
| **Problem Relevance** | The logical architecture model should relate back to the operational scenarios produced by mission analysis. |
| **Inputs for Architecture Model** | The major input for architecture definition activity involves the set of system requirements and the instances in which they do not address the right level of architecture. The consequence is that the architect allows the requirements to fall to the side and invents a solution with what he or she understands through the input. |
| **Decomposition Too Deep** | A common mistake made by many beginners in architecture consists of decomposing the functions too deeply or having too many functions and input/output flows in scenarios or in the functional architecture model of the current system block. |
| **Not Considering Inputs and Outputs Together with Functions** | A common mistake is to consider only the actions supported by functions and decomposing them, while forgetting the inputs and the outputs or considering them too late. Inputs and outputs are integral parts of a function. |
| **Considering Static Decomposition of Functions Only** | Static function decomposition is the smallest functional architecture model task and answers the basic question, "How is this done?" The purpose of the static decomposition is to facilitate the management or navigation through the list of functions. The static decomposition should be established only when scenarios have been created and the logical architecture is close to complete. |
| **Mixing Governance, Management, and Operation** | Governance (strategic monitoring), management (tactical monitoring), and basic operations are often mixed in complex systems. Logical architecture model should deal with behavioral architecture model as well as with temporal architecture model. |

## Proven Practices

Some proven practices gathered from the references are provided in Table 2.

## Table 2. Proven Practices with Logical Architecture Development. (SEBoK Original)

| Practice | Description |
|---|---|
| **Constitute Scenarios of Functions** | Before constituting a decomposition tree of functions, one must model the behavior of the system, establish scenarios of functions, and decompose functions as scenarios of sub-functions. |
| **Analysis and Synthesis Cycles** | When facing a system that contains a large number of functions, one should attempt to synthesize functions into higher abstraction levels of functions with the assistance of criteria. Do not perform analysis only; instead, conduct small cycles of analysis (decomposition) and synthesis. The technique of using scenarios includes this design practice. |
| **Alternate Functional and Behavioral Views** | A function (action verb; e.g. "to move") and its state of execution/operational mode (e.g. "moving") are two similar and complimentary views. Utilize this to consider a behavioral view of the system that allows for the transition from one operational mode to another. |
| **The Order to Create a Scenario Of Functions** | When creating a scenario of functions, it is more efficient to first establish the (control) flow of functions, then to add input and output flows, and finally to add triggers or signals for synchronization. |

## References

### Works Cited

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, MA, USA: Addison-Wesley.

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

ISO/IEC. 2007.*Systems Engineering – Application and Management of the Systems Engineering Process.*Geneva, Switzerland: International Organization for Standards (ISO)/International Electronical Commission (IEC), ISO/IEC 26702:2007.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Oliver, D., T. Kelliher, and J. Keegan. 1997. *Engineering complex systems with models and objects*. New York, NY, USA: McGraw-Hill.

OMG. 2010. *OMG Systems Modeling Language specification*, version 1.2, July 2010. http://www.omg.org/technology/documents/spec_catalog.htm.

### Primary References

ANSI/IEEE. 2000. *Recommended practice for architectural description for software-intensive systems*. New York, NY: American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE), ANSI/IEEE 1471-2000.

INCOSE. 2015. *Systems Engineering Handbook* - A Guide for System Life Cycle Processes and Activities", version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

ISO/IEC. 2007. *Systems Engineering – Application and Management of the Systems Engineering Process.* Geneva, Switzerland: International Organization for Standards (ISO)/International Electronical Commission (IEC), ISO/IEC 26702:2007.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Architecture Description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

Maier, M. and E. Rechtin. 2009. *The Art of Systems Architecting,* 3rd ed. Boca Raton, FL, USA: CRC Press.

### Additional References

Alexander, C., S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel. 1977. *A Pattern Language: Towns, Buildings, Construction*. New York, NY, USA: Oxford University Press.

Buede, D.M. 2009. *The engineering design of systems: Models and methods*. 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

Oliver, D., T. Kelliher, and J. Keegan. 1997. *Engineering Complex Systems with Models and Objects*. New York, NY, USA: McGraw-Hill.

The Open Group. 2011. *TOGAF*, version 9.1. Hogeweg, The Netherlands: Van Haren Publishing. Accessed August 29, 2012. Available at: https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?catalogno=g116.

Zachman, J. 2008. "John Zachman's Concise Definition of The Zachman Framework™." Zachman International Enterprise Architecture. Accessed August 29, 2012. Available at: http:/ / www. zachman. com/ about-the-zachman-framework.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Physical Architecture Model Development

***Lead Authors:*** *Alan Faisandier, Rick Adcock*

Physical Architecture Model Development may be used as a task of the activity "Develop candidate architectures models and views", or a sub-process of the System Architecture Definition process (see **System Architecture** article).Its purpose is to elaborate models and views of a physical, concrete solution that accommodates the logical architecture model and satisfies and trades-off system requirements. Once a logical architecture model is defined (see Logical Architecture Model Development), concrete physical elements have to be identified that can support functional, behavioral, and temporal features as well as the expected properties of the system deduced from non-functional system requirements (e.g. constraint of replacement of obsolescence, and/or continued product support).

A physical architecture model is an arrangement of physical elements, (system elements and physical interfaces) that provides the solution for a product, service, or enterprise. It is intended to satisfy logical architecture elements and system requirements ISO/IEC/IEEE 26702 (ISO 2007). It is implementable through technological system elements. System requirements are allocated to both the logical and physical architectures. The resulting system architecture is assessed with system analysis and when completed becomes the basis for system realization.

In some cases, particularly when multiple systems are to be defined to a common physical architecture model, one of the drivers for the physical architecture model may be interface standards; these physical interfaces may well be one of the most important concerns for these systems. It is quite possible that such interface standards are mandated at a high level in the system requirements. On the other hand, it is equally possible for standards to be derived during physical architecture model development and these can be critical enablers for desirable engineering outcomes, such as: families of systems, technology insertion, interoperability and “open systems”. For example, today’s video, hi-fi, and computer systems have all benefited from adoption of interface standards. Other examples exist in most fields of engineering from nuts and bolts, plumbing, electrical installations, rail gauges, TCP/IP, IT systems and software to modular defense and space systems.

Note: The term *Physical Architecture* is a contraction of the expression *Physical View of the System Architecture*.

## Concepts and Principles

### System Element, Physical Interface, and Physical Architecture Model

A system element is a discrete part of a system that can be implemented to fulfill design properties. A system element can be hardware, software, data, humans, processes (e.g., processes that provide a service to users), procedures (e.g., operator instructions), facilities, materials, and naturally occurring entities (e.g., water, organisms, and minerals), or any combination of these ISO/IEC/IEEE 15288 (ISO 2015). A physical interface binds two system elements together; this is similar to a link or a connector. Table 1 provides some examples of system elements and physical interfaces.

**Table 1. Types of System Elements and Physical Interfaces. (SEBoK Original)**

| Element | Product System | Service System | Enterprise System |
|---|---|---|---|
| **System Element** | • Hardware Parts (mechanics, electronics, electrical, plastic, chemical, etc.)<br>• Operator Roles<br>• Software Pieces | • Processes, Data Bases, Procedures, etc.<br>• Operator Roles<br>• Software Applications | • Corporate, Direction, Division, Department, Project, Technical Team, Leader, etc.<br>• IT Components |
| **Physical Interface** | * Hardware Parts, Protocols, Procedures, etc. | * Protocols, Documents, etc. | * Protocols, Procedures, Documents, etc. |

A complex system composed of thousands of physical and/or intangible parts may be structured in several layers of systems and system elements. The number of elements in a level of the structure of one system is limited to only a few, in order to facilitate managing the system definition; a common guideline is *five plus or minus two* elements (see illustration in Figure 1).

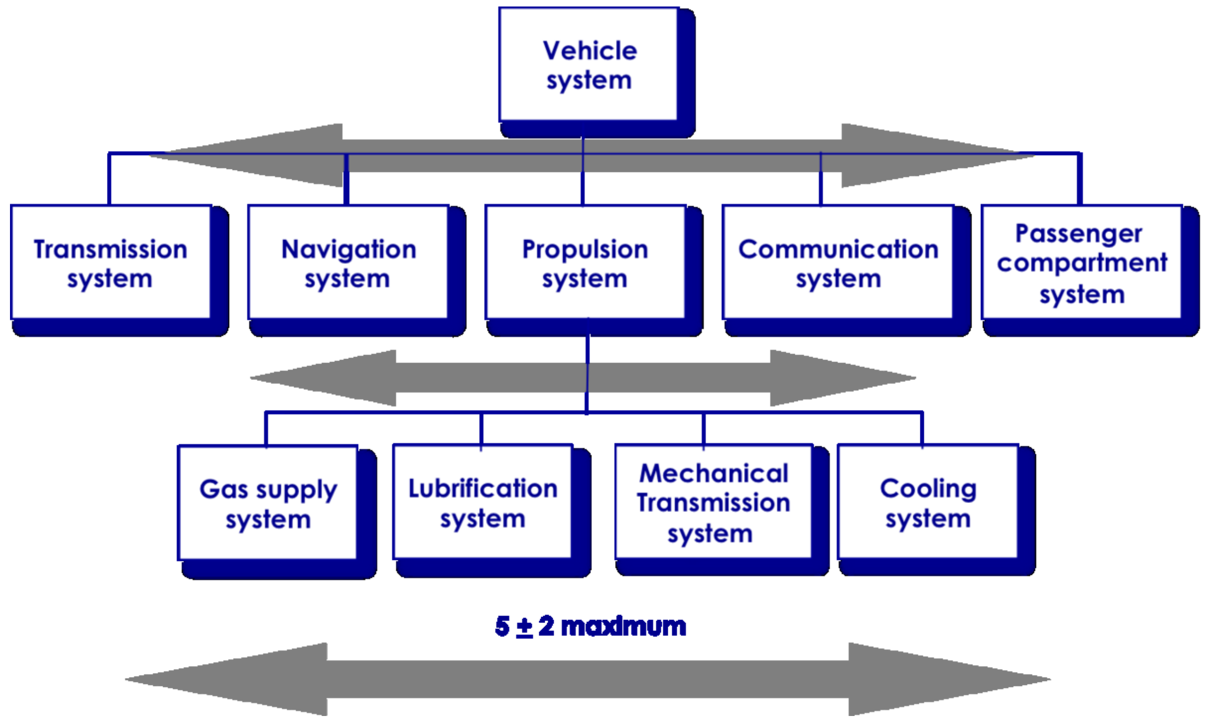


**Figure 1. Layers of Systems and System Elements (Faisandier 2012).** Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

A physical architecture model is built from systems, system elements, and all necessary physical interfaces between these elements, as well as from external elements (neighboring or enabling systems and/or system elements in the considered layer and concerned elements in the context of the global system-of-interest) - see illustration in Figure 2.

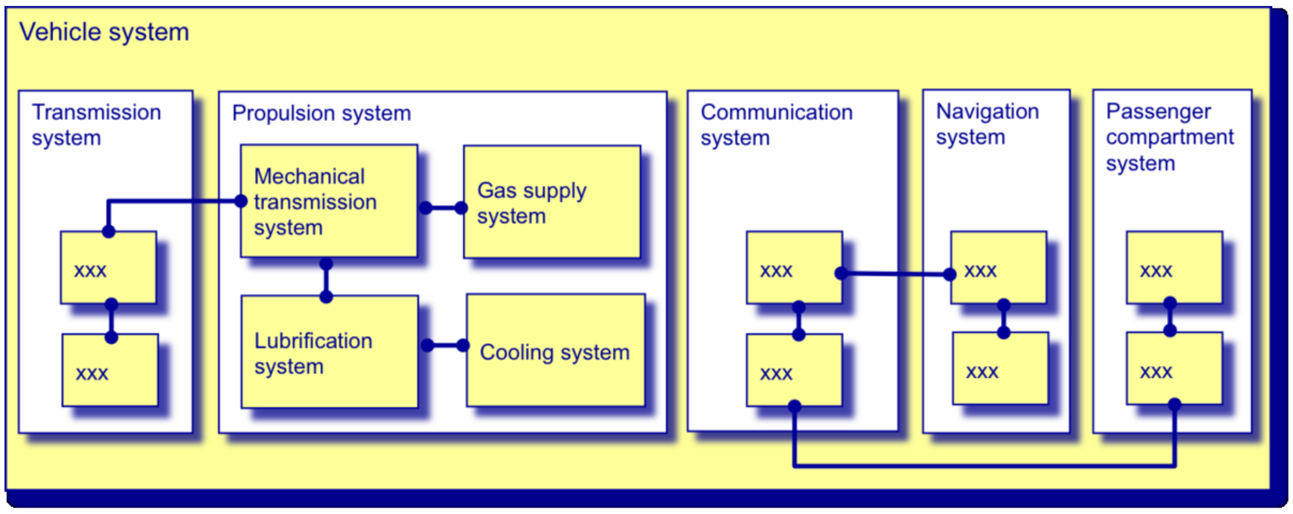


**Figure 2. Physical Architecture Model Representation (Faisandier 2012).** Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

## Design Property

A design property is a property that is obtained during system architecture and created through the assignment of non-functional requirements, estimates, analyses, calculations, simulations of a specific aspect, or through the definition of an existing element associated with a system element, a physical interface, and/or a physical architecture. If the defined element complies with a requirement, the design property will relate to (or may equal) the requirement. Otherwise, one has to identify any discrepancy that could modify the requirement or design property, and detect any deviations.

Stakeholders have concerns that correspond to the expected behavior of a system within operational, environmental, and/or physical constraints as well as to more general life cycle constraints. Stakeholder requirements and system requirements express these concerns as expected capabilities from the system (e.g., usability, interoperability, security, expandability, environment suitability, etc.). Architects and/or designers identify these capabilities from requirements and deduce corresponding quantitative or qualitative design properties to properly equip their physical architecture model (e.g., reliability, availability, maintainability, modularity, robustness, operability, climatic environment resistance, dimensions limits, etc.). For further discussion on how some of these properties may be included in architecture and design, please see the article Systems Engineering and Specialty Engineering in the Related Disciplines knowledge area (KA).

## Allocation of Logical Elements to Physical Elements and Partitioning

Developing a candidate physical architecture model for a system consists of first identifying the system elements that can perform functions of the logical architecture model as well as identifying the interfaces capable of carrying out the input-output flows and control flows. When identifying potential elements, a systems engineer needs to allocate design properties within the logical architecture; these properties are deduced from the system requirements. Partitioning and allocation are activities to decompose, gather, or separate functions in order to facilitate the identification of feasible system elements that support these functions. Either they exist and can be reused or re-purposed, or they can be developed and technically implemented.

Partitioning and allocation use criteria to find potential affinities between functions. Systems engineers use system requirements and/or design properties as criteria to assess and select candidate system elements and partitions of functions, such as similar transformations within the same technology, similar levels of efficiency, exchange of the same type of input-output flows (information, energy, and materials), centralized or distributed controls, execution with close frequency level, dependability conditions, environment resistance level, and other enterprise constraints.

A concurrent engineering approach is necessary when several different sets of technologies, knowledge, and skills are necessary to establish a candidate physical architecture model. This is particularly true during the partition and allocation of functions to various system elements, in which the systems engineer must account for compatibility issues and emergent properties. (see SEBoK Part 2: Synthesizing Possible Solutions for a discussion of possible approaches)

## Developing Candidate Physical Architecture Models

The goal of physical architecture model development activities is to provide the best possible physical architecture model made of suitable systems, technological system elements, and physical interfaces (i.e., the architecture that answers, at best, all system requirements, depending on agreed limits or margins of each requirement). The best way to do this is to produce several candidate physical architecture models, assess and compare them, and then select the most suitable one.

A candidate physical architecture model is elaborated according to affinity criteria in order to build a set of system elements (i.e., separate, gather, connect, and disconnect the network of system elements and their physical interfaces). These criteria are the same as those used for partitioning and allocating functions to system elements. The physical architecture model development may be focused in different ways, for example, it may address:

- Reduction in the number of physical interfaces
- System elements that can be tested separately
- Compatible technology
- Measures of the proximity of elements in space
- Ease of handling (weight, volume, and transportation facilities)
- Optimization of resources shared between elements
- Modularity (i.e. elements have low interdependence)
- Resilience (i.e. elements which are highly reliable, maintainable or replaceable)

## Evaluating and Selecting the Preferred Candidate

Viable physical architecture models enable all required functions or capabilities specified in the logical architecture model to be realized. Architecture and design activity includes evaluation to obtain a balance among design properties, costs, risks, etc. Generally, the physical architecture model of a system is determined more strongly by non-functional requirements (e.g., performance, safety, security, environmental conditions, constraints, etc.) than by functions. There may be many (physical) ways to establish functions but fewer ways of satisfying non-functional requirements. The preferred physical architecture model represents the selection of system elements, their physical relationships, and interfaces. Typically this physical architecture will still leave further systems engineering to be undertaken to achieve a fully optimized system after any remaining trade-offs are made and algorithms and parameters of the system are finalized.

Certain analyses (efficiency, dependability, cost, risks, etc.) are required to get sufficient data that characterize the global behavior and structure of the candidate architectures in regard to system requirements; this is often broadly referred to as system analysis. Other analyses and assessments require knowledge and skills from the different involved technologies and specialities (mechanics, electronics, software, thermodynamics, electro-magnetic compatibility, safety, security etc.). They are performed through corresponding specialist analysis of the system.

### Legacy Systems and Systems of Systems

Few systems come into existence or operate without interacting with others in a system context. These interactions may be with other operational systems, or maintenance and support systems, which in turn may be legacy (already in use) or future legacy (under development and likely to operate with the system of interest in the future).

The best chosen approach will be dependent on the strength of interactions between the system-of-interest (SoI) and its wider context. While these interactions are small, they may be accounted for by defining a set of static external interface requirements (for example technical standards) with which the system must comply, by including these as constraints in the system requirements and ensuring compliance through design assurance.

Where the interactions are more intense, for example where continuous information is to be exchanged with other systems, these will have to be recognized as part of a system of systems context and will instead be considered as part of an enterprise systems engineering approach.

Another important consideration may be the sharing of technology or system elements between the SoI and other systems, often as part of a family of systems (many examples occur in automotive and aerospace industries) or the re-use of system elements from existing legacy. Here a degree of top-down or middle-out design work will be necessary to ensure the system of interest embodies the required system elements, while conforming as far as possible to the stakeholder and system requirements, with any compromises being understood and managed.

If a System-of-Interest is intended to be used in one or more service systems or system of systems configurations this will affect its physical architecture model. One of the features of these SoS is the late binding of component systems in use. Such component systems must be architected with open or configurable interfaces, must have clearly defined functions packaged in such a way as to be relevant to the SoS using them, and must include some method by which they can be identified and included in the SoS when needed.

Both service systems and SoS will be defined by a high level physical architecture model, which will be utilized to define the relevant SoS relationships, interfaces, and constraints that should be included in Concept Definition. The results will be embedded in the stakeholder and system requirements and handled through interface agreements and across-project communication during development, realization, and use.

See SEBoK Part 4 Applications of Systems Engineering for more information on special considerations for architecting SoS.

## Process Approach

### Purpose

The purpose of the Physical Architecture Model Development is to define, select, and synthesize a system physical architecture model which can support the logical architecture model. A physical architecture model will have specific properties to address stakeholder concerns or environmental issues and to satisfy system requirements.

Because of the evolution of the context of use or technological possibilities, the physical architecture which is composed of system elements is supposed to evolve along the life cycle of the system in order for it to continue to perform its mission within the limits of its required effectiveness. Depending on whether or not evolution impacts logical architecture model elements, allocations to system elements may change. A physical architecture model is equipped with specific design properties (glossary) to continuously challenge the evolution.

**Generic inputs** include the selected logical architecture model, system requirements, generic patterns and properties that architects identify and utilize to answer requirements, outcomes from system analysis, and feedback from system verification and system validation.

**Generic outputs** are the selected physical architecture model, allocation matrix of functional elements to physical elements, traceability matrix with system requirements, stakeholder requirements of each system and system element composing the physical architecture model, and rejected solutions.

## Activities of the Process

Major activities and tasks to be performed during this process include the following:

- Partition and allocate functional elements to system elements:
  - Search for system elements or technologies able to perform functions and physical interfaces to carry input-output and control flows. Ensure system elements exist or can be engineered. Assess each potential system element using criteria deduced from design properties (themselves deduced from non-functional system requirements).
  - Partition functional elements (functions, scenarios, input-outputs, triggers, etc.) using the given criteria and allocate partitioned sets to system elements (using the same criteria).
  - When it is impossible to identify a system element that corresponds to a partitioned functional set, decompose the function until the identification of implementable system elements is possible.
  - Check the compatibility of technologies and the compatibility of interfaces between selected system elements.
- Constitute candidate physical architecture models.
  - Because partitioned sets of functions can be numerous, there are generally too many system elements. For defining controllable architectures, system elements have to be grouped into higher-level system elements known as system element groups, often called sub-systems in industry.
  - Constitute several different system element groups corresponding to different combinations of elementary system elements. One set of system element groups plus one or several non-decomposable system elements form a candidate physical architecture model of the considered system.
  - Represent (using patterns) the physical architecture model of each system element group connecting its system elements with physical Interfaces that carry input-output flows and triggers. Add physical interfaces as needed; in particular, add interfaces with external elements to the system element group.
  - Represent the synthesized physical architecture of the considered system built from system element groups, non-decomposable system, and physical interfaces inherited from the physical architecture model of system element groups.
  - Enhance the physical architecture model with design properties such as modularity, evolution capability, adaptability to different environments, robustness, scalability, resistance to environmental conditions, etc.
  - If possible, use executable architecture prototypes (e.g., hardware-software (HW-SW)-in-the-loop prototypes) for identifying potential deficiencies and correct the architecture as needed.
- Assess physical architecture model candidates and select the most suitable one:
  - Use the system analysis process to perform assessments (see the System Analysis topic).
  - Use the Decision Management process to support the trades and selection of the preferred alternative (see the Decision Management topic).
- Synthesize the selected physical architecture model:
  - Formalize physical elements and properties. Verify that system requirements are satisfied and that the solution is realistic.
  - Identify the derived physical and functional elements created for the necessity of architecture and design and the corresponding system requirements.
  - Establish traceability between system requirements and physical elements as well as allocate matrices between functional and physical elements.

### Artifacts, Methods and Modeling Techniques

Physical architecture descriptions use modeling techniques to create and represent physical architectures. Some common physical models include structural blocks, mass, layout and other models. Modeling techniques may be:

- Physical block diagrams (PBD)
- SysML block definition diagrams (BDD)
- Internal block diagrams (IBD) (OMG 2010)
- Executable architecture prototyping
- Etc.

Depending on the type of domain for which it is to be used (defense, enterprise, etc.), architecture frameworks may provide descriptions that can help to trade-off candidate architectures. Please see section 'Enterprise Architecture Frameworks & Methodologies' in Enterprise Systems Engineering Key Concepts.

## Practical Considerations

Key pitfalls and good practices related to physical architecture development are described in the next two sections.

### Pitfalls

Some of the key pitfalls encountered in performing physical architecture model development are provided in Table 3.

**Table 3. Pitfalls with Physical Architecture Development. (SEBoK Original)**

| Pitfall | Description |
|---|---|
| Too Many Levels in a Single System Block | The current system block includes too many levels of decomposition. The right practice is that the physical architecture model of a system block is composed of one single level of systems and/or system elements. |
| No Logical Architecture Model | The developers perform a direct passage from system requirements to physical architecture model without establishing a logical architecture model; this is a common wrong practice that mainly takes place when dealing with repeating systems and products because the functions are already known. The issue is that a function is always associated with input-output flows defined in a specific domain set. If the domain set changes, the performance of the function can become invalid. |
| Direct Allocation on Technologies | At a high level of abstraction of multidisciplinary systems, directly allocating the functions onto technologies of the lowest level of abstraction, such as hardware or software, does not reflect a system comprehension. The right practice is to consider criteria to decompose the architecture into the appropriate number of levels, alternating logical and physical before reaching the technology level ( the last level of the system). |

### Proven Practices

Some proven practices gathered from the references are provided in Table 4.

**Table 4. Proven Practices with Physical Architecture Development. (SEBoK Original)**

| Practice | Description |
|---|---|
| Modularity | Restrict the number of interactions between the system elements and consider the modularity principle (maximum of consistency inside the system element, minimum of physical interfaces with outside) as the right way for architecting systems. |
| Focus on Interfaces | Focusing on interfaces rather than on system elements is another key element of a successful architecture and design for abstract levels of systems. |

## References

### Works Cited

ISO/IEC. 2007. Systems Engineering – Application and Management of The Systems Engineering Process. Geneva, Switzerland: International Organization for Standards (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 26702:2007.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

OMG. 2010. *OMG Systems Modeling Language specification*, version 1.2, July 2010. http://www.omg.org/technology/documents/spec_catalog.htm.

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

### Primary References

ANSI/IEEE. 2000. *Recommended practice for architectural description for software-intensive systems*. New York, NY: American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE), ANSI/IEEE 1471-2000.

INCOSE. 2015. *Systems Engineering Handbook* - A Guide for System Life Cycle Processes and Activities", version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Architecture Description*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

### Additional References

Maier, M., and E. Rechtin. 2009. *The Art of Systems Architecting,* 3rd ed. Boca Raton, FL, USA: CRC Press.

Holland, J.H. 2006. "Studying Complex Adaptive Systems." *Journal of Systems Science and Complexity.*19(1): 1-8. http://hdl.handle.net/2027.42/41486

Thome, B. 1993. *Systems Engineering, Principles & Practice of Computer-Based Systems Engineering.* New York, NY, USA: Wiley.

The Open Group. 2011. *TOGAF*, version 9.1. Hogeweg, The Netherlands: Van Haren Publishing. Accessed August 29, 2012. Available at: https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?catalogno=g116.

Zachman, J. 2008. "John Zachman's Concise Definition of The Zachman Framework™." Zachman International Enterprise Architecture. Accessed August 29, 2012. Available at: http:/ / www. zachman. com/ about-the-zachman-framework.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# System Design

***Lead Authors:*** *Alan Faisandier, Rick Adcock*

The purpose of the System Design is to supplement the system architecture providing information and data useful and necessary for implementation of the system elements. Design definition is the process of developing, expressing, documenting, and communicating the realization of the architecture of the system through a complete set of design characteristics described in a form suitable for implementation.

## Concepts and principles

### Design Notion

In industrial practices, the term *design* is often used to mean both architecture and design. In the recent past, professionals used the term *design* when they dealt with simpler technological products - ones that do not include several different and interconnected technological components such as hardware, software, operators, services, etc. In the development of new multi-technology products and services, professionals have recognized the usefulness of the notion of *system* in dealing with complexity (interconnections level, multi-techno, emergence, etc.).

It was due to complexity that structuring the elements that comprise a system became necessary. This structure explains the functional, behavioral, temporal, physical, and other aspects of a system as described in System Architecture. Practitioners found the term *structure* inadequate to describe all of these aspects of a system. The terms *architecture* and *architectural design* have been used for approximately 30 years, especially in software intensive systems and other domains, such as the space industry. The set of different types and interrelated structures can be understood as the architecture of the system.

The trend today is to consider system architecture and system design as different and separate sets of activities, but concurrent and strongly intertwined.

System design includes activities to conceive a set of system elements that answers a specific, intended purpose, using principles and concepts; it includes assessments and decisions to select system elements that compose the system, fit the architecture of the system, and comply with traded-off system requirements. It is the complete set of detailed models, properties, and/or characteristics described into a form suitable for implementation.

### Design characteristics and design enablers

Every technological domain or discipline owns its peculiar laws, rules, theories, and enablers concerning transformational, structural, behavioral, and temporal properties of its composing parts of materials, energy, or information. These specific parts and/or their compositions are described with typical design characteristics and enablers. These allow achieving the implementation of every system element through various transformations and exchanges required by design characteristics (e.g., operability level, reliability rate, speed, safeguard level) that have been assigned during the system architecture definition process.

The design definition provides the description of the design characteristics and design enablers necessary for implementation. Design characteristics include dimensions, shapes, materials, and data processing structures. Design enablers include formal expressions or equations, drawings, diagrams, tables of metrics with their values and margins, patterns, algorithms, and heuristics.

- Examples of generic design characteristics in mechanics of solids: shape, geometrical pattern, dimension, volume, surface, curves, resistance to forces, distribution of forces, weight, velocity of motion, temporal persistence
- Examples of generic design characteristics in software: distribution of processing, data structures, data persistence, procedural abstraction, data abstraction, control abstraction, encapsulation, creational patterns (e.g., builder, factory, prototype, singleton), and structural patterns (e.g., adapter, bridge, composite, decorator, proxy)

## Relation with System Architecture

System design is intended to be the link between the system architecture (at whatever point this milestone is defined in the specific application of the systems engineering process) and the implementation of technological system elements that compose the physical architecture model of the system.

Design definition is driven by specified requirements, the system architecture, and more detailed analysis of performance and feasibility. It addresses the implementation technologies and their assimilation. Design provides the “how” or “implement-to” level of the definition.

Design concerns every system element composed of implementation technologies, such as for example mechanics, electronics, software, chemistry, human operations and services for which specific engineering processes are needed. System design provides feedback to the parent system architecture to consolidate or confirm the allocation and partitioning of architectural characteristics and design properties to system elements.

## Design Descriptor

A design descriptor is the set of generic design characteristics and of their possible values. If similar, but not exact system elements exist, it is possible to analyze these in order to identify their basic characteristics. Variations of the possible values of each characteristic determine potential candidate system elements.

## Holistic Design

Holistic design is an approach that considers the system being designed as an interconnected whole, which is also part of something larger. Holistic concepts can be applied to the system as a whole along with the system in its context (e.g., the enterprise or mission in which the system participates), as well as the design of mechanical devices, the layout of spaces, and so forth. This approach often incorporates concerns about the environment, considering how the design will impact the environment and attempting to reduce environmental impact. Holistic design is about more than merely trying to meet the system requirements.

## Process Approach

### Purpose

The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture. ISO/IEC/IEEE 15288 (ISO 2015)

Generic inputs include architecture description of the parent system, system element requirements.

Generic outputs are the description of the design characteristics and design enablers necessary for implementation.

### Activities of the Process

Major activities and tasks to be performed during this process include the following:

**1. Initialize design definition**

- Plan for technology management for the whole system. Identify the technologies (mechanics, electricity, electronics, software, biology, operators, etc.) that would compose and implement the system elements and their physical interfaces.
- Determine which technologies and system elements have a risk to become obsolete, or evolve during the operation stage of the system. Plan for their potential replacement.
- Identify types of design characteristics or properties for each technology of each system element.
- Periodically assess design characteristics and adjust as the system evolves.
- Document the design definition strategy, including the need for and requirements of any enabling systems, products, or services to perform the design.

**2. Establish design characteristics and design enablers related to each system element**

- Perform or consolidate or detail system requirements allocation to system elements for all requirements and system elements not fully addressed in the System Architecture process (normally, every system requirement would have been transformed into architectural entities and architectural characteristics within the System Architecture process, which are then allocated to system elements through direct assignment or some partitioning).
- Define the design characteristics relating to the architectural characteristics and check that they are implementable. Use design enablers, such as models (physical and analytical), design heuristics, etc. If the design characteristics are not feasible, then assess other design alternatives or implementation option, or perform trades of other system elements definition.
- Define the interfaces that were not defined by the System Architecture process or that need to be refined as the design details evolve. This includes both internal interfaces between the system elements and the external interfaces with other systems.
- Record the design characteristics of each system element within the applicable artifacts (they depend on the design methods and techniques used).
- Provide rationale about selection of major implementation options and enablers.

#### 3. Assess alternatives for obtaining system elements

- Identify existing implemented system elements (COTS/NDI, reused, or other non-developed system elements). Alternatives for new system elements to be developed may be studied.
- Assess design options for the system element, using selection criteria that are derived from the design characteristics.
- Select the most appropriate alternatives.
- If the decision is made to develop the system element, rest of the design definition process and the implementation process are used. If the decision is to buy or reuse a system element, the acquisition process may be used to obtain the system element.

#### 4. Manage the design

- Capture and maintain the rationale for all selections among alternatives and decisions for the design, architecture characteristics, design enablers, and sources of system elements.
- Assess and control the evolution of the design characteristics, including the alignment with the architecture.
- Establish and maintain traceability between design characteristics and architectural characteristics, and with requirements as necessary.
- Provide baseline information for configuration management.
- Maintain the design baseline and the design definition strategy.

## Practical Considerations

Key pitfalls and proven practices related to system design are described in the next two sections.

### Pitfalls

Some of the key pitfalls encountered in performing system design are provided in Table 1.

| Pitfall | Description |
|---|---|
| **Consider only separately the design of each system element** | This would conduct to use heterogeneous implementation of a given technology or between technologies within the system-of-interest. The design strategy for the complete system is defined to search synergies and/or commonalities that could help operation and maintenance of system elements. |

### Proven Practices

Some proven practices gathered from the references are provided in Table 2.

| Practice | Description |
|---|---|
| **Architecture and design mutual support** | Discipline engineers perform the design definition of each system element; they provide strong support (knowledge and competencies) to systems engineers, or architects, in the evaluation and selection of candidate system architectures and system elements. Inversely, systems engineers, or architects, must provide feedback to discipline engineers to improve knowledge and know-how. |

## References

### Works Cited

INCOSE. 2015. *INCOSE Systems Engineering Handbook,*Version 4. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

### Primary References

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

### Additional References

Baldwin, C.Y. and K.B. Clark. 2000. *Design Rules*. Cambridge, Mass: MIT Press.

Buede, D.M. 2009. *The Engineering Design of Systems: Models and Methods*. 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

DoD. 2010. *DOD Architecture Framework.*Version 2.02. Arlington, VA, USA: US Department of Defense. Available at: http://cio-nii.defense.gov/sites/dodaf20/

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# System Analysis

***Lead Authors:*** *Alan Faisandier, Ray Madachy* **Contributing Author:** *Rick Adcock*

System analysis allows developers to objectively carry out quantitative assessments of systems in order to select and/or update the most efficient system architecture and to generate derived engineering data. During engineering, assessments should be performed every time technical choices or decisions are made to determine compliance with system requirements.

System analysis provides a rigorous approach to technical decision-making. It is used to perform trade-off studies, and includes modeling and simulation, cost analysis, technical risks analysis, and effectiveness analysis.

## Principles Governing System Analysis

One of the major tasks of a systems engineer is to evaluate the engineering data and artifacts created during the systems engineering (SE) process. The evaluations are at the center of system analysis, providing means and techniques

- to define assessment criteria based on system requirements;
- to assess design properties of each candidate solution in comparison to these criteria;
- to score globally the candidate solutions and to justify the scores; and
- to decide on the appropriate solution(s).

The Analysis and Selection between Alternative Solutions article in the Systems Approach Applied to Engineered Systems knowledge area (KA) of Part 2 describes activities related to selecting between possible system solutions to an identified problem or opportunity. The following general principles of systems analysis are defined:

- Systems analysis is based on assessment criteria based upon a problem or opportunity system description.
  - These criteria will be based around an ideal system description, which assumes a hard system problem context can be defined.
  - Criteria must consider required system behavior and properties of the complete solution, in all possible wider system contexts and environments.
  - These must consider non-functional issues such as system safety, security, etc. (Please see Systems Engineering and Specialty Engineering for additional discussion on incorporating non-functional elements.)
  - This "ideal" system description may be supported by soft system descriptions, from which additional “soft” criteria may be defined. For example, a stakeholder preference for or against certain kinds of solutions, relevant social, political or cultural conventions to be considered, etc.
- The assessment criteria should include, at a minimum, the constraints on cost and time scales acceptable to stakeholders.
- Trade studies provide a mechanism for conducting analysis of alternative solutions.
  - A trade study should consider a set of assessment criteria, with appropriate awareness of the limitations and dependencies between individual criteria.
  - Trade studies need to deal with both objective and subjective criteria. Care must be taken to assess the sensitivity of the overall assessment to particular criteria.

## Trade-off studies

In the context of the definition of a system, a trade-off study consists of comparing the characteristics of each system element and of each candidate system architecture to determine the solution that best globally balances the assessment criteria. The various characteristics analyzed are gathered in cost analysis, technical risks analysis, and effectiveness analysis (NASA 2007).

Guidance on the conduct of trade studies for all types of system context are characterized in the above principles and described in more details in the Analysis and Selection between Alternative Solutions topic. Of particular interest to SE analysis are technical effectiveness, cost, and technical risk analysis.

## Effectiveness Analysis

The effectiveness of an engineered system solution includes several essential characteristics that are generally gathered in the following list of analyses, including (but not limited to): performance, usability, dependability, manufacturing, maintenance or support, environment, etc. These analyses highlight candidate solutions under various aspects.

It is essential to establish a classification that limits the number of analyses to the really significant aspects, such as key performance parameters. The main difficulties of effectiveness analysis are to sort and select the right set of effectiveness aspects; for example, if the product is made for a single use, maintainability will not be a relevant criterion.

## Cost Analysis

A cost analysis considers the full life cycle costs. A cost baseline can be adapted according to the project and the system. The global life cycle cost (LCC), or total ownership cost (TOC), may include examplar labor and non-labor cost items such as those indicated in Table 1.

**Table 1. Types of Costs. (SEBoK Original)**

| Type of Cost | Description and Examples |
|---|---|
| **Development** | Engineering, development tools (equipment and software), project management, test-benches, mock-ups and prototypes, training, etc. |
| **Product manufacturing or service realization** | Raw materials and supplying, spare parts and stock assets, necessary resources to operation (water, electricity power, etc.), risks and nuances, evacuation, treatment and storage of waste or rejections produced, expenses of structure (taxes, management, purchase, documentation, quality, cleaning, regulation, controls, etc.), packing and storage, documentation required. |
| **Sales and after-sales** | Expenses of structure (subsidiaries, stores, workshops, distribution, information acquisition, etc.), complaints and guarantees, etc. |
| **Customer utilization** | Taxes, installation (customer), resources necessary to the operation of the product (water, fuel, lubricants, etc.), financial risks and nuisances, etc. |
| **Supply chain** | Transportation and delivery |
| **Maintenance** | Field services, preventive maintenance, regulation controls, spare parts and stocks, cost of guarantee, etc. |
| **Disposal** | Collection, dismantling, transportation, treatment, waste recycling, etc. |

Methods for determining cost are described in the Planning topic.

### Technical Risks Analysis

Every risk analysis concerning every domain is based on three factors:

1. Analysis of potential threats or undesired events and their probability of occurrence.
2. Analysis of the consequences of these threats or undesired events and their classification on a scale of gravity.
3. Mitigation to reduce the probabilities of threats and/or the levels of harmful effect to acceptable values.

The technical risks appear when the system cannot satisfy the system requirements any longer. The causes reside in the requirements and/or in the solution itself. They are expressed in the form of insufficient effectiveness and can have multiple causes: incorrect assessment of the technological capabilities; over-estimation of the technical maturity of a system element; failure of parts; breakdowns; breakage, obsolescence of equipment, parts, or software, weakness from the supplier (non-compliant parts, delay for supply, etc.), human factors (insufficient training, wrong tunings, error handling, unsuited procedures, malice), etc.

Technical risks are not to be confused with project risks, even if the method to manage them is the same. Although technical risks may lead to project risks, technical risks address the system itself, not the process for its development. (See Risk Management for more details.)

## Process Approach

### Purpose and Principles of the Approach

The system analysis process is used to: (1) provide a rigorous basis for technical decision making, resolution of requirement conflicts, and assessment of alternative physical solutions (system elements and physical architectures); (2) determine progress in satisfying system requirements and derived requirements; (3) support risk management; and (4) ensure that decisions are made only after evaluating the cost, schedule, performance, and risk effects on the engineering or re-engineering of a system (ANSI/EIA 1998). This process is also called the decision analysis process by NASA (2007, 1-360) and is used to help evaluate technical issues, alternatives, and their uncertainties to support decision-making. (See Decision Management for more details.)

System analysis supports other system definition processes:

- Stakeholder requirements definition and system requirements definition processes use system analysis to solve issues relating to conflicts among the set of requirements; in particular, those related to costs, technical risks, and effectiveness (performances, operational conditions, and constraints). System requirements subject to high risks, or those which would require different architectures, are discussed.
- The Logical Architecture Model Development and Physical Architecture Model Development processes use it to assess characteristics or design properties of candidate logical and physical architectures, providing arguments for selecting the most efficient one in terms of costs, technical risks, and effectiveness (e.g., performances, dependability, human factors, etc.).

Like any system definition process, the system analysis process is iterative. Each operation is carried out several times; each step improves the precision of analysis.

### Activities of the Process

Major activities and tasks performed within this process include

- Planning the trade-off studies:
  - Determine the number of candidate solutions to analyze, the methods and procedures to be used, the expected results (examples of objects to be selected: behavioral architecture/scenario, physical architecture, system element, etc.), and the justification items.
  - Schedule the analyses according to the availability of models, engineering data (system requirements, design properties), skilled personnel, and procedures.

- Define the selection criteria model:
  - Select the assessment criteria from non-functional requirements (performances, operational conditions, constraints, etc.), and/or from design properties.
  - Sort and order the assessment criteria.
  - Establish a scale of comparison for each assessment criterion, and weigh every assessment criterion according to its level of relative importance with the others.
- Identify candidate solutions, related models, and data.
- Assess candidate solutions using previously defined methods or procedures:
  - Carry out costs analysis, technical risks analysis, and effectiveness analysis placing every candidate solution on every assessment criterion comparison scale.
  - Score every candidate solution as an assessment score.
- Provide results to the calling process: assessment criteria, comparison scales, solutions' scores, assessment selection, and possibly recommendations and related arguments.

## Artifacts and Ontology Elements

This process may create several artifacts, such as

- A selection criteria model (list, scales, weighing)
- Costs, risks, and effectiveness analysis reports
- Justification reports

This process handles the ontology elements of Table 2 within system analysis.

### Table 2. Main Ontology Elements as Handled within System Analysis. (SEBoK Original)

| | |
|---|---|
| **Assessment Criterion** | In the context of system analysis, an assessment criterion is a characteristic used to assess or compare system elements, physical interfaces, physical architectures, functional architectures/scenarios, or any engineering elements that can be compared. |
| | Identifier; name; description; relative weight; scalar weight |
| **Assessment Selection** | In the context of system analysis, an assessment selection is a technical management element based on an assessment score that justifies the selection of a system element, a physical interface, a physical architecture, or a functional architecture/scenario. |
| **Assessment Score** | In the context of system analysis, an assessment score is obtained assessing a system element, a physical interface, a physical architecture, a functional architecture/scenario using a set of assessment criteria. |
| | Identifier; name; description; value |
| **Cost** | In the context of systems engineering, a cost is an amount expressed in a given currency related to the value of a system element, a physical interface, and a physical architecture. |
| | Identifier; name; description; amount; type (development, production, utilization, maintenance, disposal); confidence interval; period of reference; estimation technique |
| **Risk** | An event having a probability of occurrence and consequences related to the system mission or on other characteristics. (Used for technical risk in engineering.). A risk is the combination of vulnerability a danger or threat. |
| | Identifier; name description; status |

## Checking Correctness of System Analysis

The main items to be checked within system analysis in order to get validated arguments are

- Relevance of the models and data in the context of use of the system,
- Relevance of assessment criteria related to the context of use of the system,
- Reproducibility of simulation results and of calculations,
- Precision level of comparisons' scales,
- Confidence of estimates, and
- Sensitivity of solutions' scores related to assessment criteria weights.

See Ring, Eisner, and Maier (2010) for additional perspective.

## Methods and Modeling Techniques

- **General usage of models**: Various types of models can be used in the context of system analysis:
  - **Physical models** are scale models allowing simulation of physical phenomena. They are specific to each discipline; associated tools include mock-ups, vibration tables, test benches, prototypes, decompression chamber, wind tunnels, etc.
  - **Representation models** are mainly used to simulate the behavior of a system. For example, enhanced functional flow block diagrams (eFFBDs), statecharts, state machine diagrams (based in systems modeling language (SysML)), etc.
  - **Analytical models** are mainly used to establish values of estimates. We can consider the deterministic models and probabilistic models (also known as stochastic models) to be analytical in nature. Analytical models use equations or diagrams to approach the real operation of the system. They can be very simple (addition) to incredibly complicated (probabilistic distribution with several variables).
- **Use right models** depending on the project progress
  - At the beginning of the project, first studies use simple tools, allowing rough approximations which have the advantage of not requiring too much time and effort. These approximations are often sufficient to eliminate unrealistic or outgoing candidate solutions.
  - Progressively with the progress of the project it is necessary to improve precision of data to compare the candidate solutions still competing. The work is more complicated if the level of innovation is high.
  - A systems engineer alone cannot model a complex system; he has to be supported by skilled people from different disciplines involved.
- **Specialist expertise**: When the values of assessment criteria cannot be given in an objective or precise way, or because the subjective aspect is dominating, we can ask specialists for expertise. The estimates proceed in four steps:

1. Select interviewees to collect the opinion of qualified people for the considered field.
2. Draft a questionnaire; a precise questionnaire allows an easy analysis, but a questionnaire that is too closed risks the neglection of significant points.
3. Interview a limited number of specialists with the questionnaire, including an in-depth discussion to get precise opinions.
4. Analyze the data with several different people and compare their impressions until an agreement on a classification of assessment criteria and/or candidate solutions is reached.

Often used analytical models in the context of system analysis are summarized in Table 3.

### Table 3. Often Used Analytical Models in the Context of System Analysis. (SEBoK Original)

| Type of Model | Description |
|---|---|
| **Deterministic models** | • Models containing **statistics** are included in this category. The principle consists in establishing a model based on a significant amount of data and number of results from former projects; they can apply only to system elements/components whose technology already exists.<br>• Models by **analogy** also use former projects. The system element being studied is compared to an already existing system element with known characteristics (cost, reliability, etc.). Then these characteristics are adjusted based on the specialists' expertise.<br>• **Learning curves** allow foreseeing the evolution of a characteristic or a technology. One example of evolution: "Each time the number of produced units is multiplied by two, the cost of this unit is reduced with a certain percentage, generally constant." |
| **Probabilistic models** (also called stochastic models) | The theory of probability allows classifying the possible candidate solutions compared to consequences from a set of events as criteria. These models are applicable if the number of criteria is limited and the combination of the possible events is simple. Take care that the sum of probabilities of all events is equal to one for each node. |
| **Multi-criteria decisions models** | When the number of criteria is greater than ten, it is recommended that a multi-criteria decision model be established. This model is obtained through the following actions:<br>• Organize the criteria as a hierarchy (or a decomposition tree).<br>• Associate each criterion of each branch of the tree with a relative weight compare to each other of the same level.<br>• Calculate a scalar weight for each leaf criterion of each branch multiplying all the weights of the branch.<br>• Score every candidate solution on the leaf criteria; sum the scores to get a global score for each candidate solution; compare the scores.<br>• Using a computerized tool allows to perform sensitivity analysis to get a robust choice. |

## Practical Considerations

Key pitfalls and good practices related to system analysis are described in the next two sections.

### Pitfalls

Some of the key pitfalls encountered in planning and performing system analysis are provided in Table 4.

### Table 4. Pitfalls with System Analysis. (SEBoK Original)

| Pitfall | Description |
|---|---|
| Analytical modeling is not a decision tool | Analytical modeling gives analytical results from analytical data. It has to be considered as a help and not as a decision tool. |
| Models and system levels of decomposition | A model can be well adapted to a level *n* of a system and to be incompatible with the model of the higher level which uses the data coming from the lower level. It is essential that the systems engineer ensures the coherence of the various models used. |
| Optimization is not a sum of optimized elements | The general optimization of the system-of-interest is not the sum of its optimized systems and/or system elements. |

### Proven Practices

Some proven practices gathered from the references are provided in Table 5.

**Table 5. Proven Practices with System Analysis. (SEBoK Original)**

| Practice | Description |
|---|---|
| Stay in the operational field | Models can never simulate all the behavior/reactions of a system: they operate only in one limited field with a restricted number of variables. When a model is used, it is always necessary to make sure that the parameters and data inputs are part of the operation field. If not, there is a high risk of irregular outputs. |
| Evolve models | Models shall evolve during the project: by modification of parameter settings, by entering new data when modified (modification of assessment criteria, functions to perform, requirements, etc.), by the use of new tools when those used reach their limits. |
| Use several types of models | It is recommended to concurrently use several types of models in order to compare the results and/or to take into account another aspect of the system. |
| Keep context elements consistent | Results of a simulation shall always be given in their modeling context: tool used, selected assumptions, parameters and data introduced, and variance of the outputs. |

## References

### Works Cited

ANSI/EIA. 1998. *Processes for Engineering a System.* Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA), ANSI/EIA-632-1998.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

Ring, J, H. Eisner, and M. Maier. 2010. "Key Issues of Systems Engineering, Part 3: Proving Your Design." INCOSE *Insight* 13(2).

### Primary References

ANSI/EIA. 1998. *Processes for Engineering a System.* Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA), ANSI/EIA 632-1998.

Blanchard, B.S., and W.J. Fabrycky. 2010. *Systems Engineering and Analysis,* 5th ed. Prentice-Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C., USA: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

### Additional References

Ring, J, H. Eisner, and M. Maier. 2010. "Key Issues of Systems Engineering, Part 3: Proving Your Design." INCOSE *Insight.* 13(2).

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Knowledge Area: System Realization

# System Realization

***Lead Authors:*** *John Snoderly, Alan Faisandier,* ***Contributing Author:*** *Rick Adcock*

System realization activities are conducted to create and test versions of a system as specified by system definition. The activities are grouped and described as generic processes that are performed iteratively and/or concurrently depending on the selected life cycle model. These activities include those required to build a system (system implementation), integrate disparate system elements (system integration), and ensure that the system meets both the needs of stakeholders (system validation) and aligns with the system requirements and architecture (system verification).

These activities are not sequential, but are performed concurrently, iteratively and recursively depending on the selected life cycle model. Figure 1 (see "Overview", below), also shows how these processes fit within the context of system definition and System Deployment and Use KAs. See also Applying Life Cycle Processes for further discussion of the relationships between process and life cycle model.

## Topics

Each part of the SEBoK is divided into KAs, which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- System Implementation
- System Integration
- System Verification
- System Validation

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

## Overview

Essentially, the outputs of system definition are used during system implementation to create system elements and during system integration to provide plans and criteria for combining these elements. The requirements are used to verify and validate system elements, systems, and the overall system-of-interest (SoI). These activities provide feedback into the system design, particularly when problems or challenges are identified.

Finally, when the system is considered, verified, and validated, it will then become an input to system deployment and use. It is important to understand that there is overlap in these activities; they do not have to occur in sequence as demonstrated in Figure 1. Every life cycle model includes realization activities, principally, verification and validation activities. The way these activities are performed is dependent upon the life cycle model in use. (For additional information on life cycles, see the Life Cycle Models KA.)

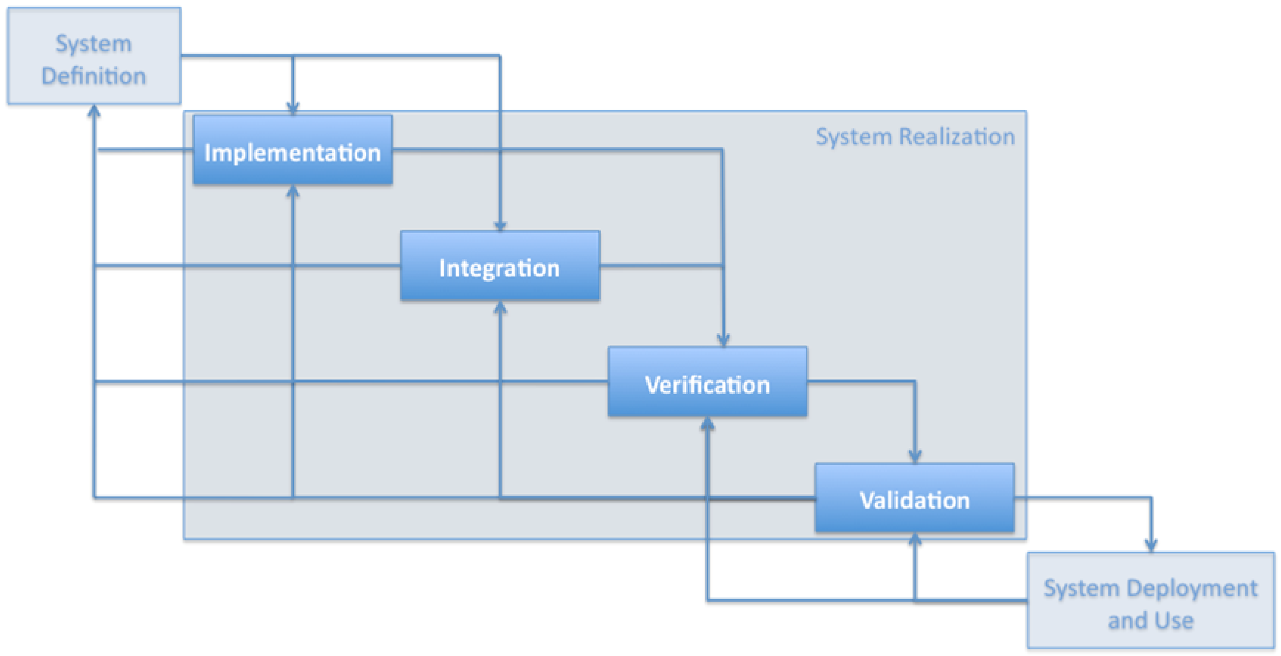


**Figure 1. System Realization.** (SEBoK Original)

The realization processes are performed to ensure that the system will be ready for transition and has the appropriate structure and behavior to enable the desired operation and functionality throughout the system's life span. Both DAU and NASA include transition in realization, in addition to implementation, integration, verification, and validation (Prosnik 2010; NASA December 2007, 1-360).

## Fundamentals

### Macro View of Realization Processes

Figure 2 illustrates a macro view of generic outputs from realization activities when using a Vee life cycle model. The left side of the Vee represents various design activities 'going down' the system.

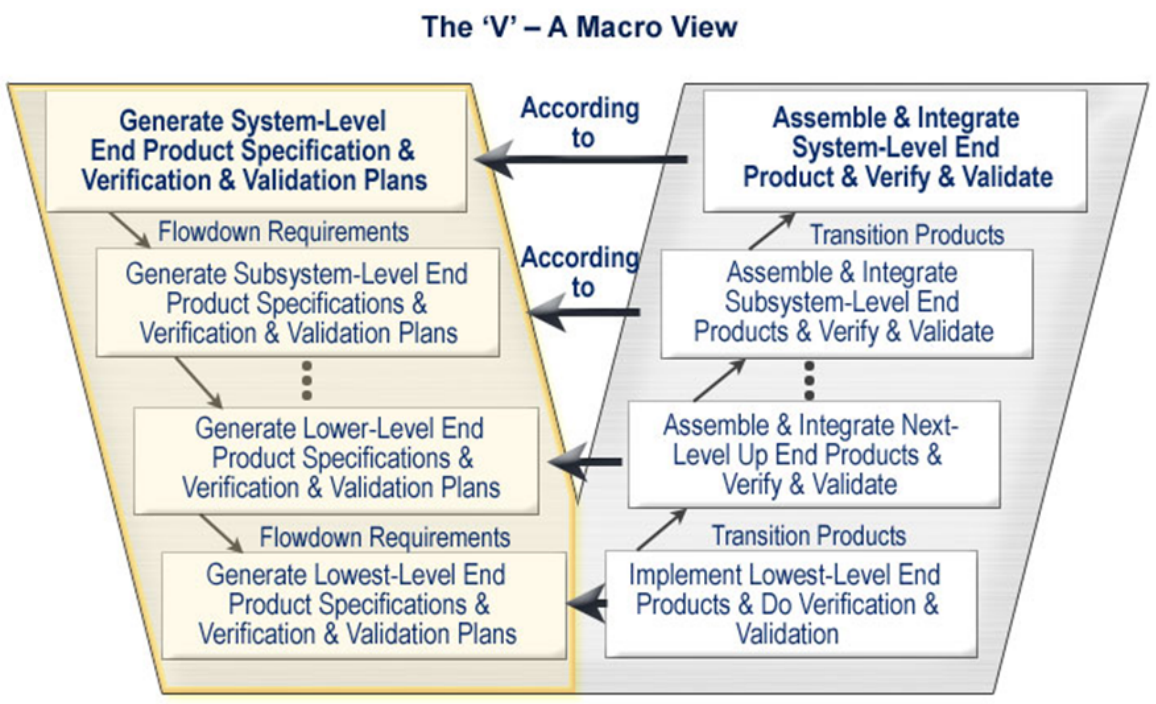

The 'V' – A Macro View
Generate System-Level End Product Specification & Verification & Validation Plans
Flowdown Requirements
Generate Subsystem-Level End Product Specifications & Verification & Validation Plans
Generate Lower-Level End Product Specifications & Verification & Validation Plans
Flowdown Requirements
Generate Lowest-Level End Product Specifications & Verification & Validation Plans
According to
According to
Assemble & Integrate System-Level End Product & Verify & Validate
Transition Products
Assemble & Integrate Subsystem-Level End Products & Verify & Validate
Assemble & Integrate Next-Level Up End Products & Verify & Validate
Transition Products
Implement Lowest-Level End Products & Do Verification & Validation

**Figure 2. The Vee Activity Diagram (Prosnik 2010).** Released by the Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

The left side of the Vee model demonstrates the development of system elements specifications and design descriptions. In this stage, verification and validation plans are developed, which are later used to determine whether realized system elements (products, services, or enterprises) are compliant with specifications and stakeholder requirements. Also, during this stage initial specifications become flow-down requirements for lower-level system models. In terms of time frame, these activities take place early in the system's life cycle. These activities are discussed further in the System Definition KA. However, it is important to understand that some of the system realization activities are initiated at the same time as system definition activities; this is the case with integration, verification and validation planning in particular.

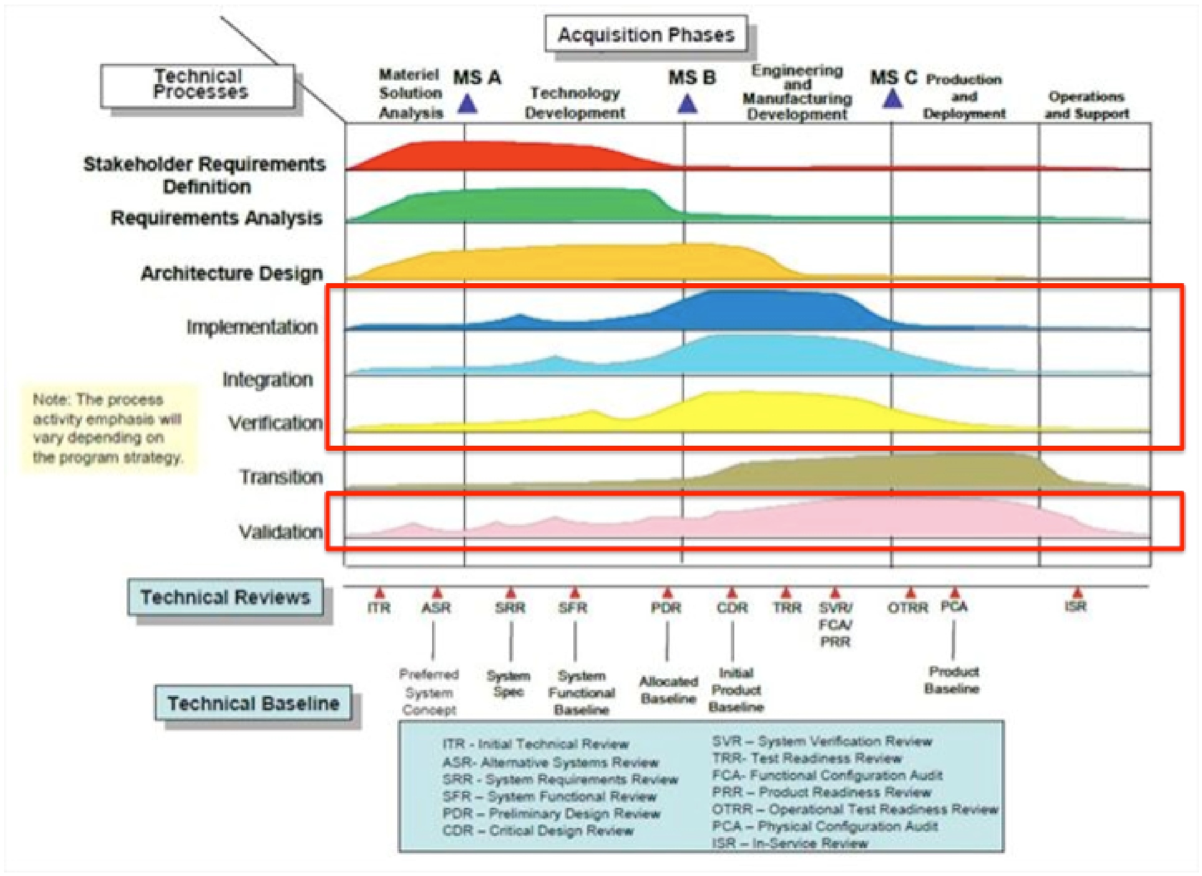
The right side of the Vee model, as illustrated in Figure 2, shows the system elements (products, services, or enterprises) are assembled according to the system model described on the left side of the Vee (integration). Verification and validation activities determine how well the realized system fulfills the stakeholder requirements, the system requirements, and design properties. These activities should follow the plans developed on the left side of the Vee. Integration can be done continuously, incrementally and/or iteratively, supported by verification and validation (V&V) efforts. For example, integration typically starts at the bottom of the Vee and continues upwards to the top of the Vee.

The U.S. Defense Acquisition University (DAU) provides an overview of what occurs during system realization:

> *Once the products of all system models have been fully defined, Bottom-Up End Product Realization can be initiated. This begins by applying the Implementation Process to buy, build, code or reuse end products. These implemented end products are verified against their design descriptions and specifications, validated against Stakeholder Requirements and then transitioned to the next higher*

> *system model for integration. End products from the Integration Process are successively integrated upward, verified and validated, transitioned to the next acquisition phase or transitioned ultimately as the End Product to the user.* (Prosnik 2010)

While the systems engineering (SE) technical processes are life cycle processes, the processes are concurrent, and the emphasis of the respective processes depends on the phase and maturity of the design. Figure 3 portrays (from left to right) a notional emphasis of the respective processes throughout the systems acquisition life cycle from the perspective of the U.S. Department of Defense (DoD). It is important to note that from this perspective, these processes do not follow a linear progression; instead, they are concurrent, with the amount of activity in a given area changing over the system's life cycle. The red boxes indicate the topics that will be discussed as part of realization.



Acquisition Phases
Technical Processes
Materiel Solution Analysis
MS A
Technology Development
MS B
Engineering and Manufacturing Development
MS C
Production and Deployment
Operations and Support
Stakeholder Requirements Definition
Requirements Analysis
Architecture Design
Implementation
Integration
Verification
Transition
Validation
Note: The process activity emphasis will vary depending on the program strategy.
Technical Reviews
ITR
ASR
SRR
SFR
PDR
CDR
TRR
SVR/ FCA/ PRR
OTRR
PCA
ISR
Technical Baseline
Preferred System Concept
System Spec
System Functional Baseline
Allocated Baseline
Initial Product Baseline
Product Baseline
ITR - Initial Technical Review
ASR- Alternative Systems Review
SRR - System Requirements Review
SFR – System Functional Review
PDR – Preliminary Design Review
CDR – Critical Design Review
SVR – System Verification Review
TRR- Test Readiness Review
FCA- Functional Configuration Audit
PRR – Product Readiness Review
OTRR – Operational Test Readiness Review
PCA – Physical Configuration Audit
ISR – In-Service Review

**Figure 3. Notional Emphasis of Systems Engineering Technical Processes and Program Life-Cycle Phases (DAU 2010).** Released by the Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

## References

### Works Cited

DAU. 2010. *Defense Acquisition Guidebook (DAG).* Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

Prosnik, G. 2010. Materials from "Systems 101: Fundamentals of Systems Engineering Planning, Research, Development, and Engineering". DAU distance learning program. eds. J. Snoderly, B. Zimmerman. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

## Primary References

INCOSE. 2011. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.1.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Martin, J.N. 1997. *Systems Engineering Guidebook: A process for developing systems and products,* 1st ed. Boca Raton, FL, USA: CRC Press.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

## Additional References

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

DAU. *Your Acquisition Policy and Discretionary Best Practices Guide.* In Defense Acquisition University (DAU)/U.S. Department of Defense (DoD) [database online]. Ft Belvoir, VA, USA. Available at: https://dag.dau.mil/Pages/Default.aspx (accessed 2010).

ECSS. 2009. *Systems Engineering General Requirements*. Noordwijk, Netherlands: Requirements and Standards Division, European Cooperation for Space Standardization (ECSS), 6 March 2009. ECSS-E-ST-10C.

IEEE. 2012. "Standard for System and Software Verification and Validation". Institute of Electrical and Electronics Engineers. IEEE-1012.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# System Implementation

***Lead Authors:*** *John Snoderly, Alan Faisandier*

System Implementation uses the structure created during architectural design and the results of system analysis to construct system elements that meet the stakeholder requirements and system requirements developed in the early life cycle phases. These system elements are then integrated to form intermediate aggregates and finally the complete system-of-interest (SoI). See System Integration.

## Definition and Purpose

Implementation is the process that actually yields the lowest-level system elements in the system hierarchy (system breakdown structure). System elements are made, bought, or reused. Production involves the hardware fabrication processes of forming, removing, joining, and finishing, the software realization processes of coding and testing, or the operational procedures development processes for operators' roles. If implementation involves a production process, a manufacturing system which uses the established technical and management processes may be required.

The purpose of the implementation process is to design and create (or fabricate) a system element conforming to that element’s design properties and/or requirements. The element is constructed employing appropriate technologies and industry practices. This process bridges the system definition processes and the integration process. Figure 1 portrays how the outputs of system definition relate to system implementation, which produces the implemented (system) elements required to produce aggregates and the SoI.

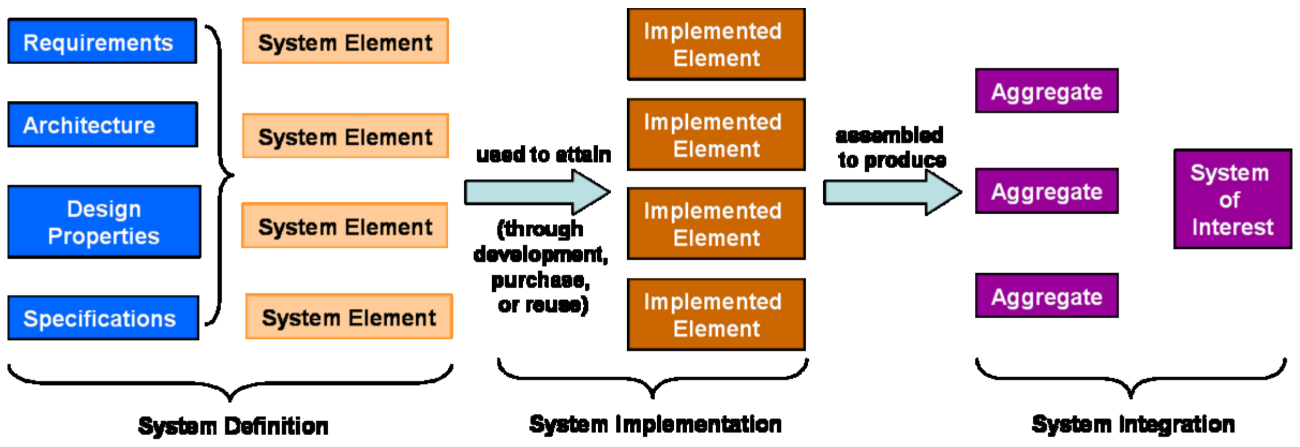


**Figure 1. Simplification of How the Outputs of System Definition Relate to System Implementation, which Produces the System Elements Required to Produce Systems and Subsystems.** (SEBoK Original)

## Process Approach

### Purpose and Principle of the Approach

During the implementation process, engineers apply the design properties and/or requirements allocated to a system element to design and produce a detailed description. They then fabricate, code, or build each individual element using specified materials, processes, physical or logical arrangements, standards, technologies, and/or information flows outlined in detailed descriptions (drawings or other design documentation). A system element will be verified against the detailed description of properties and validated against its requirements.

If subsequent verification and validation (V&V) actions or configuration audits reveal discrepancies, recursive interactions occur, which includes predecessor activities or processes, as required, to mitigate those discrepancies and to modify, repair, or correct the system element in question. Figure 2 provides the context for the implementation process from the perspective of the U.S. Defense Acquisition University (DAU).

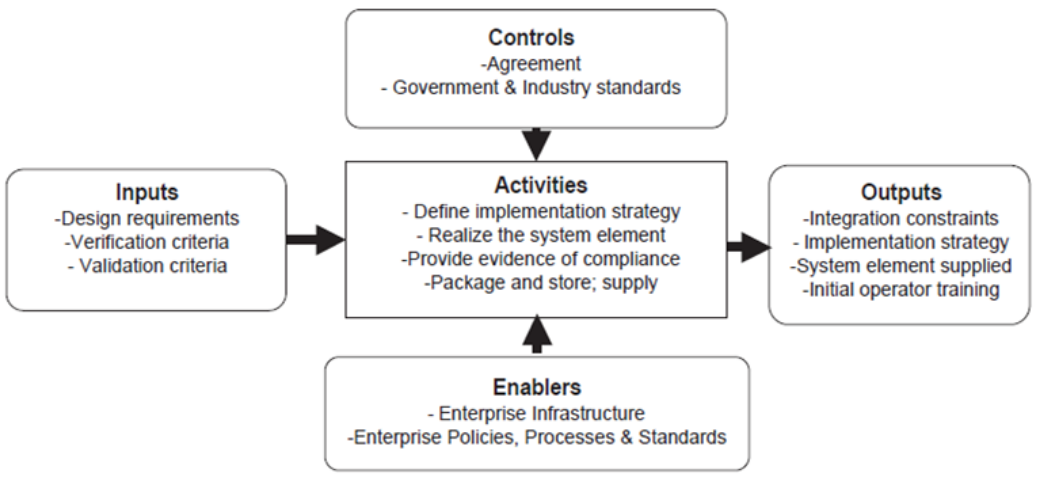

Controls
-Agreement
- Government & Industry standards
Inputs
-Design requirements
-Verification criteria
- Validation criteria
Activities
- Define implementation strategy
- Realize the system element
-Provide evidence of compliance
-Package and store; supply
Outputs
-Integration constraints
- Implementation strategy
-System element supplied
-Initial operator training
Enablers
- Enterprise Infrastructure
-Enterprise Policies, Processes & Standards

**Figure 2. Context Diagram for the Implementation Process (DAU 2010).** Released by the Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

Such figures provide a useful overview of the systems engineering (SE) community’s perspectives on what is required for implementation and what the general results of implementation may be. These are further supported by the discussion of implementation inputs, outputs, and activities found in the National Aeronautics and Space Association's (NASA's) *Systems Engineering Handbook* (NASA 2007). It is important to understand that these views are process -oriented. While this is a useful model, examining implementation only in terms of process can be limiting.

Depending on the technologies and systems chosen when a decision is made to produce a system element, the implementation process outcomes may generate constraints to be applied on the architecture of the higher-level system; those constraints are normally identified as derived system requirements and added to the set of system requirements applicable to this higher-level system. The architectural design has tomust take those constraints into account.

If the decision is made to purchase or reuse an existing system element, it has tomust be identified as a constraint or system requirement applicable to the architecture of the higher-level system. Conversely, the implementation process may involve some adaptation or adjustments to the system requirement in order to be integrated into a higher-level system or aggregate.

Implementation also involves packaging, handling, and storage, depending on the concerned technologies and where or when the system requirement needs to be integrated into a higher-level aggregate. Developing the supporting documentation for a system requirement, such as the manuals for operation, maintenance, and/or installation, is also a part of the implementation process; these artifacts are utilized in the system deployment and use phase. The system element requirements and the associated verification and validation criteria are inputs to this process; these inputs come from the architectural design process detailed outputs.

Execution of the implementation process is governed by both industrial and government standards and the terms of all applicable agreements. This may include conditions for packaging and storage, as well as preparation for use activities, such as operator training. In addition, packaging, handling, storage, and transportation (PHS&T) considerations will constrain the implementation activities. For more information, refer to the discussion of PHS&T in the System Deployment and Use article. The developing or integrating organization will likely have enterprise-level safety practices and guidelines that must also be considered.

## Activities of the Process

The following major activities and tasks are performed during this process:

- **Define the implementation strategy** - Implementation process activities begin with detailed design and include developing an implementation strategy that defines fabrication and coding procedures, tools and equipment to be used, implementation tolerances, and the means and criteria for auditing configuration of resulting elements to the detailed design documentation. In the case of repeated system element implementations (such as for mass manufacturing or replacement elements), the implementation strategy is defined and refined to achieve consistent and repeatable element production; it is retained in the project decision database for future use. The implementation strategy contains the arrangements for packing, storing, and supplying the implemented element.
- **Realize the system element** - Realize or adapt and produce the concerned system element using the implementation strategy items as defined above. Realization or adaptation is conducted with regard to standards that govern applicable safety, security, privacy, and environmental guidelines or legislation and the practices of the relevant implementation technology. This requires the fabrication of hardware elements, development of software elements, definition of training capabilities, drafting of training documentation, and the training of initial operators and maintainers.
- **Provide evidence of compliance** - Record evidence that the system element meets its requirements and the associated verification and validation criteria as well as the legislation policy. This requires the conduction of peer reviews and unit testing, as well as inspection of operation and maintenance manuals. Acquire measured properties that characterize the implemented element (weight, capacities, effectiveness, level of performance, reliability, availability, etc.).
- **Package, store, and supply the implemented element** - This should be defined in the implementation strategy.

## Artifacts and Ontology Elements

This process may create several artifacts such as:

- an implemented system
- implementation tools
- implementation procedures
- an implementation plan or strategy
- verification reports
- issue, anomaly, or trouble reports
- change requests (about design)

This process handles the ontology elements shown in Table 1 below.

### Table 1. Main Ontology Elements as Handled within System Element Implementation. (SEBoK Original)

| Element | Definition |
|---|---|
| | **Attributes (examples)** |
| **Implemented Element** | An implemented element is a system element that has been implemented. In the case of hardware it is marked with a part/serial number. |
| | Identifier, name, description, type (hardware, software application, software piece, mechanical part, electric art, electronic component, operator role, procedure, protocol, manual, etc.) |

| | |
|---|---|
| **Measured Property** | A measured property is a characteristic of the implemented element established after its implementation. The measured properties characterize the implemented system element when it is completely realized, verified, and validated. If the implemented element complies with a design property, the measured property should equal the design property. Otherwise one has tomust identify the difference or non-conformance which treatment could conclude to modify the design property and possibly the related requirements, or to modify (correct, repair) the implemented element, or to identify a deviation. |
| | Identifier, name, description, type (effectiveness, availability, reliability, maintainability, weight, capacity, etc.), value, unit, etc. |

The main relationships between ontology elements are presented in Figure 3.

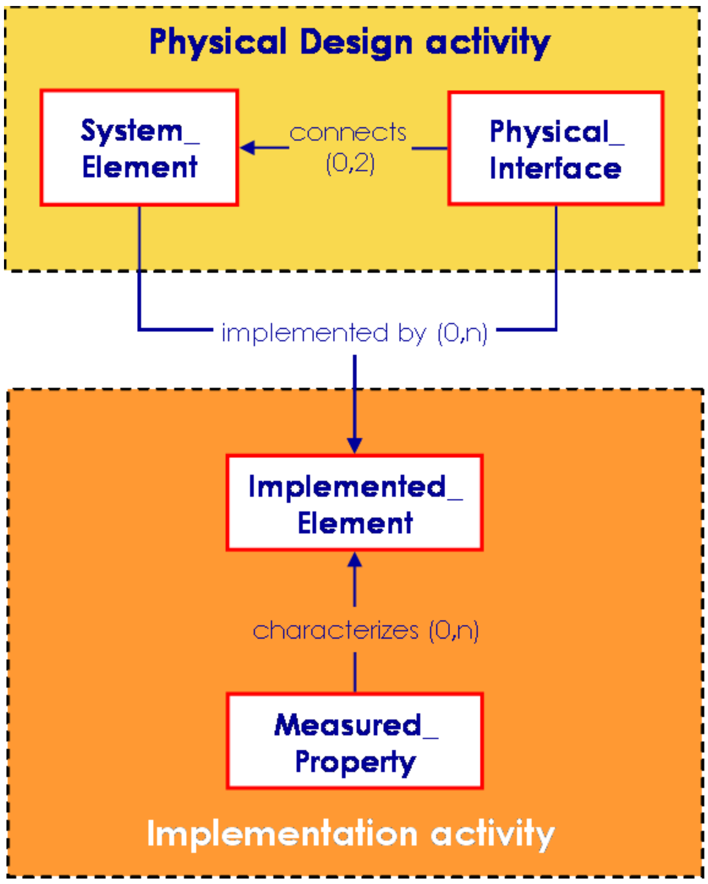


**Figure 3. Implementation Elements Relationships with Other Engineering Elements.** (SEBoK Original)

## Methods, Techniques, and Tools

There are many software tools available in the implementation and integration phases. The most basic method would be the use of N-squared diagrams as discussed in Jeff Grady's book *System Integration* (Grady 1994).

## Checking and Correctness of Implementation

Proper implementation checking and correctness should include testing to determine if the implemented element (i.e., piece of software, hardware, or other product) works in its intended use. Testing could include mockups and breadboards, as well as modeling and simulation of a prototype or completed pieces of a system. Once this is completed successfully, the next process would be system integration.

## References

### Works Cited

DAU. February 19, 2010. *Defense acquisition guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense.

Grady, J.O. 1994. *System integration.* Boca Raton, FL, USA: CRC Press, Inc.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

### Primary References

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

Grady, J.O. 1994. *System Integration*. Boca Raton, FL, USA: CRC Press, Inc.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

### Additional References

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# System Integration

***Lead Authors:*** *John Snoderly, Alan Faisandier, Scott Jackson*

System integration consists of taking delivery of the implemented system elements which compose the system-of-interest (SoI), assembling these implemented elements together, and performing the verification and validation actions (V&V actions) in the course of the assembly. The ultimate goal of system integration is to ensure that the individual system elements function properly as a whole and satisfy the design properties or characteristics of the system. System integration is one part of the realization effort and relates only to developmental items. Integration should not to be confused with the assembly of end products on a production line. To perform the production, the assembly line uses a different order from that used by integration.

## Definition and Purpose

System integration consists of a process that “*iteratively combines implemented system elements to form complete or partial system configurations in order to build a product or service. It is used recursively for successive levels of the system hierarchy*.” (ISO/IEC 15288 2015, 68). The process is extended to any kind of product system, service system, and enterprise system. The purpose of system integration is to prepare the SoI for final validation and transition either for use or for production. Integration consists of progressively assembling aggregates of implemented elements that compose the SoI as architected during design, and to check correctness of static and dynamic aspects of interfaces between the implemented elements.

The U.S. Defense Acquisition University (DAU) provides the following context for integration: *The integration process will be used . . . for the incorporation of the final system into its operational environment to ensure that the system is integrated properly into all defined external interfaces. The interface management process is particularly important for the success of the integration process, and iteration between the two processes will occur* (DAU 2010).

The purpose of system integration can be summarized as below:

- Completely assemble the implemented elements to make sure that the they are compatible with each other.
- Demonstrate that the aggregates of implemented elements perform the expected functions and meet measures of performance/effectiveness.
- Detect defects/faults related to design and assembly activities by submitting the aggregates to focused V&V actions.

Note: In the systems engineering literature, sometimes the term *integration* is used in a larger context than in the present topic. In this larger sense, it concerns the technical effort to simultaneously design and develop the system and the processes for developing the system through concurrent consideration of all life cycle stages, needs, and competences. This approach requires the "integration" of numerous skills, activities, or processes.

## Principles

### Boundary of Integration Activity

Integration can be understood as the whole bottom-up branch of the Vee Model, including the tasks of assembly and the appropriate verification tasks. See Figure 1 below:

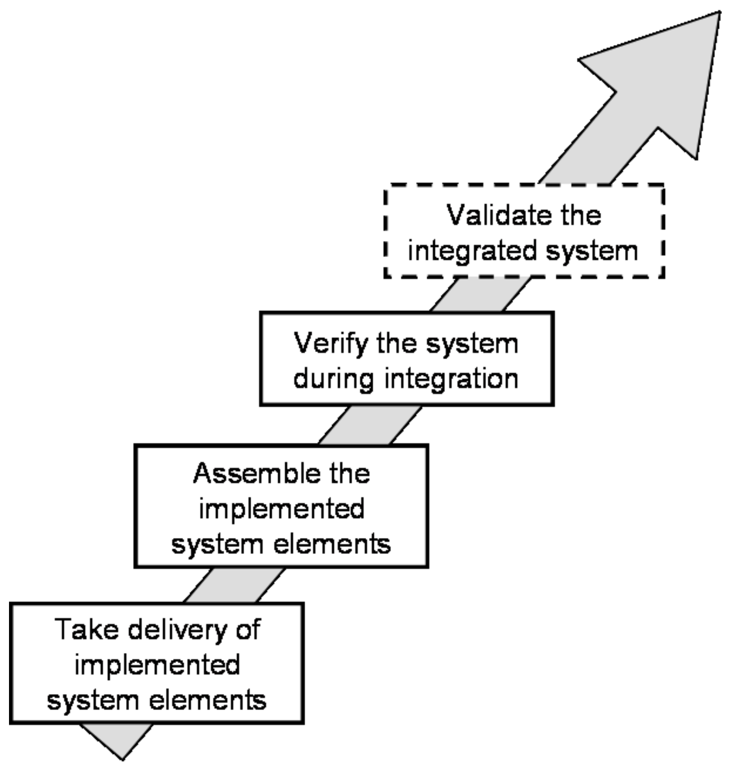

Validate the integrated system
Verify the system during integration
Assemble the implemented system elements
Take delivery of implemented system elements

**Figure 1. Limits of Integration Activities.** (SEBoK Original)

The assembly activity joins together, and physically links, the implemented elements. Each implemented element is individually verified and validated prior to entering integration. Integration then adds the verification activity to the assembly activity, excluding the final validation.

The final validation performs operational tests that authorize the transition for use or the transition for production. Remember that system integration only endeavors to obtain pre-production prototypes of the concerned product, service, or enterprise. If the product, service, or enterprise is delivered as a unique exemplar, the final validation activity serves as acceptance for delivery and transfer for use. If the prototype has to be produced in several exemplars, the final validation serves as acceptance to launch their production. The definition of the optimized operations of assembly which will be carried out on a production line relates to the manufacturing process and not to the integration process.

Integration activity can sometimes reveal issues or anomalies that require modifications of the design of the system. Modifying the design is not part of the integration process but concerns only the design process. Integration only deals with the assembly of the implemented elements and verification of the system against its properties as designed. During assembly, it is possible to carry out tasks of finishing touches which require simultaneous use of several implemented elements (e.g., paint the whole after assembly, calibrate a biochemical component, etc.). These tasks must be planned in the context of integration and are not carried out on separate implemented elements and do not include modifications related to design.

## Aggregation of Implemented Elements

The integration is used to systematically assemble a higher-level system from lower-level ones (implemented system elements) that have been implemented. Integration often begins with analysis and simulations (e.g., various types of prototypes) and progresses through increasingly more realistic systems and system elements until the final product, service, or enterprise is achieved.

System integration is based on the notion of an aggregate - a subset of the system made up of several implemented elements (implemented system elements and physical interfaces) on which a set of V&V actions is applied. Each aggregate is characterized by a configuration which specifies the implemented elements to be physically assembled and their configuration status.

To perform V&V actions, a V&V configuration that includes the aggregate plus V&V tools is constituted. The V&V tools are enabling products and can be simulators (simulated implemented elements), stubs or caps, activators (launchers, drivers), harness, measuring devices, etc.

## Integration by Level of System

According to the Vee Model, system definition (top-down branch) is done by successive levels of decomposition; each level corresponds to the physical architecture of systems and system elements. The integration (bottom-up branch) takes the opposite approach of composition (i.e., a level by level approach). On a given level, integration is done on the basis of the physical architecture defined during system definition.

## Integration Strategy

The integration of implemented elements is generally performed according to a predefined strategy. The definition of the integration strategy is based on the architecture of the system and relies on the way the architecture of the system has been designed. The strategy is described in an integration plan that defines the minimum configuration of expected aggregates, the order of assembly of these aggregates in order to support efficient subsequent verification and validation actions (e.g., inspections and/or testing), techniques to check or evaluate interfaces, and necessary capabilities in the integration environment to support combinations of aggregates. The integration strategy is thus elaborated starting from the selected verification and validation strategy. See the System Verification and System Validation topics.

To define an integration strategy, there are several possible integration approaches/techniques that may be used individually or in combination. The selection of integration techniques depends on several factors; in particular, the type of system element, delivery time, order of delivery, risks, constraints, etc. Each integration technique has strengths and weaknesses which should be considered in the context of the SoI. Some integration techniques are summarized in Table 1 below.

**Table 1. Integration Techniques. (SEBoK Original)**

| Integration Technique | Description |
|---|---|
| **Global Integration** | Also known as *big-bang integration*; all the delivered implemented elements are assembled in only one step.<br>• This technique is simple and does not require simulating the implemented elements not being available at that time.<br>• Difficult to detect and localize faults; interface faults are detected late.<br>• Should be reserved for simple systems, with few interactions and few implemented elements without technological risks. |

| | |
|---|---|
| **Integration "with the Stream"** | The delivered implemented elements are assembled as they become available. <br>• Allows starting the integration quickly. <br>• Complex to implement because of the necessity to simulate the implemented elements not yet available. Impossible to control the end-to-end "functional chains"; consequently, global tests are postponed very late in the schedule. <br>• Should be reserved for well-known and controlled systems without technological risks. |
| **Incremental Integration** | In a predefined order, either one or a very few implemented elements are added to an already integrated increment of implemented elements. <br>• Fast localization of faults: a new fault is usually localized in lately integrated implemented elements or dependent of a faulty interface. <br>• Require simulators for absent implemented elements. Require many test cases, as each implemented element addition requires the verification of the new configuration and regression testing. <br>• Applicable to any type of architecture. |
| **Subsets Integration** | Implemented elements are assembled by subsets, and then subsets are assembled together (a subset is an aggregate); could also be called "functional chains integration". <br>• Time saving due to parallel integration of subsets; delivery of partial products is possible. Requires less means and fewer test cases than integration by increments. <br>• Subsets shall be defined during the design. <br>• Applicable to architectures composed of sub-systems. |
| **Top-Down Integration** | Implemented elements or aggregates are integrated in their activation or utilization order. <br>• Availability of a skeleton and early detection of architectural faults, definition of test cases close to reality, and the re-use of test data sets possible. <br>• Many stubs/caps need to be created; difficult to define test cases of the leaf-implemented elements (lowest level). <br>• Mainly used in software domain. Start from the implemented element of higher level; implemented elements of lower level are added until leaf-implemented elements. |
| **Bottom-Up Integration** | Implemented elements or aggregates are integrated in the opposite order of their activation or utilization. <br>• Easy definition of test cases; early detection of faults (usually localized in the leaf-implemented elements); reduce the number of simulators to be used. An aggregate can be a sub-system. <br>• Test cases shall be redefined for each step, drivers are difficult to define and realize, implemented elements of lower levels are "over-tested", and does not allow architectural faults to be quickly detected. <br>• Mainly used in software domain, but can be used in any kind of system. |
| **Criterion Driven Integration** | The most critical implemented elements compared to the selected criterion are first integrated (dependability, complexity, technological innovation, etc.). Criteria are generally related to risks. <br>• Allows early and intensive testing of critical implemented elements; early verification of design choices. <br>• Test cases and test data sets are difficult to define. |

Usually, a mixed integration technique is selected as a trade-off between the different techniques listed above, allowing optimization of work and adaptation of the process to the system under development. The optimization takes into account the realization time of the implemented elements, their delivery scheduled order, their level of complexity, the technical risks, the availability of assembly tools, cost, deadlines, specific personnel capability, etc.

## Process Approach

### Activities of the Process

Major activities and tasks performed during this process include:

- **Establishing the integration plan** (this activity is carried out concurrently to the design activity of the system) that defines:
  - The optimized integration strategy – order of aggregates assembly using appropriate integration techniques.
  - The V&V actions to be processed for the purpose of integration.
  - The configurations of the aggregates to be assembled and verified.
  - The integration means and verification means (dedicated enabling products) that may include assembly procedures, assembly tools (harness, specific tools), V&V tools (simulators, stubs/caps, launchers, test benches, devices for measuring, etc.), and V&V procedures.
- **Obtain the integration means** and verification means as defined in the integration plan. The acquisition of the means can be accomplished through various ways such as procurement, development, reuse, and sub-contracting; usually the acquisition of the complete set of means is a mix of these methods.
- **Take delivery** of each implemented element:
  - Unpack and reassemble the implemented element with its accessories.
  - Check the delivered configuration, conformance of implemented elements and compatibility of interfaces, and ensure the presence of mandatory documentation.
- **Assemble the implemented elements** into aggregates:
  - Gather the implemented elements to be assembled, the integration means (assembly tools, assembly procedures), and the verification means (V&V tools and procedures).
  - Connect the implemented elements to each other using assembly tools to constitute aggregates in the order prescribed by the integration plan and in assembly procedures.
  - Add or connect the V&V tools to the aggregates as predefined.
  - Carry out eventual operations of welding, gluing, drilling, tapping, adjusting, tuning, painting, parametering, etc.
- **Verify each aggregate**:
  - Check the aggregate is correctly assembled according to established procedures.
  - Perform the verification process that uses verification and validation procedures and check that the aggregate shows the right design properties/specified requirements.
  - Record integration results/reports and potential issue reports, change requests, etc.

### Artifacts and Ontology Elements

This process may create several artifacts such as:

- an integrated system
- assembly tools
- assembly procedures
- integration plans
- integration reports
- issue/anomaly/trouble reports
- change requests (about design)

This process utilizes the ontology elements discussed in Table 2.

## Table 2. Main Ontology Elements as Handled within System Integration. (SEBoK Original)

| Element | Definition<br>Attributes |
|---|---|
| **Aggregate** | An aggregate is a subset of the system made up of several system elements or systems on which a set of verification actions is applied.<br>Identifier, name, description |
| **Assembly Procedure** | An assembly procedure groups a set of elementary assembly actions to build an aggregate of implemented system elements.<br>Identifier, name, description, duration, unit of time |
| **Assembly Tool** | An assembly tool is a physical tool used to connect, assemble, or link several implemented system elements to build aggregates (specific tool, harness, etc.).<br>Identifier, name, description |
| **Risk** | An event having a probability of occurrence and a gravity degree on its consequence onto the system mission or on other characteristics (used for technical risk in engineering). A risk is the combination of vulnerability and of a danger or a threat.<br>Identifier, name, description, status |
| **Rationale** | An argument that provides the justification for the selection of an engineering element.<br>Identifier, name, description (rationale, reasons for defining an aggregate, assembly procedure, assembly tool) |

Note: verification and validation ontology elements are described in the System Verification and System Validation topics.

The main relationships between ontology elements are presented in Figure 2.

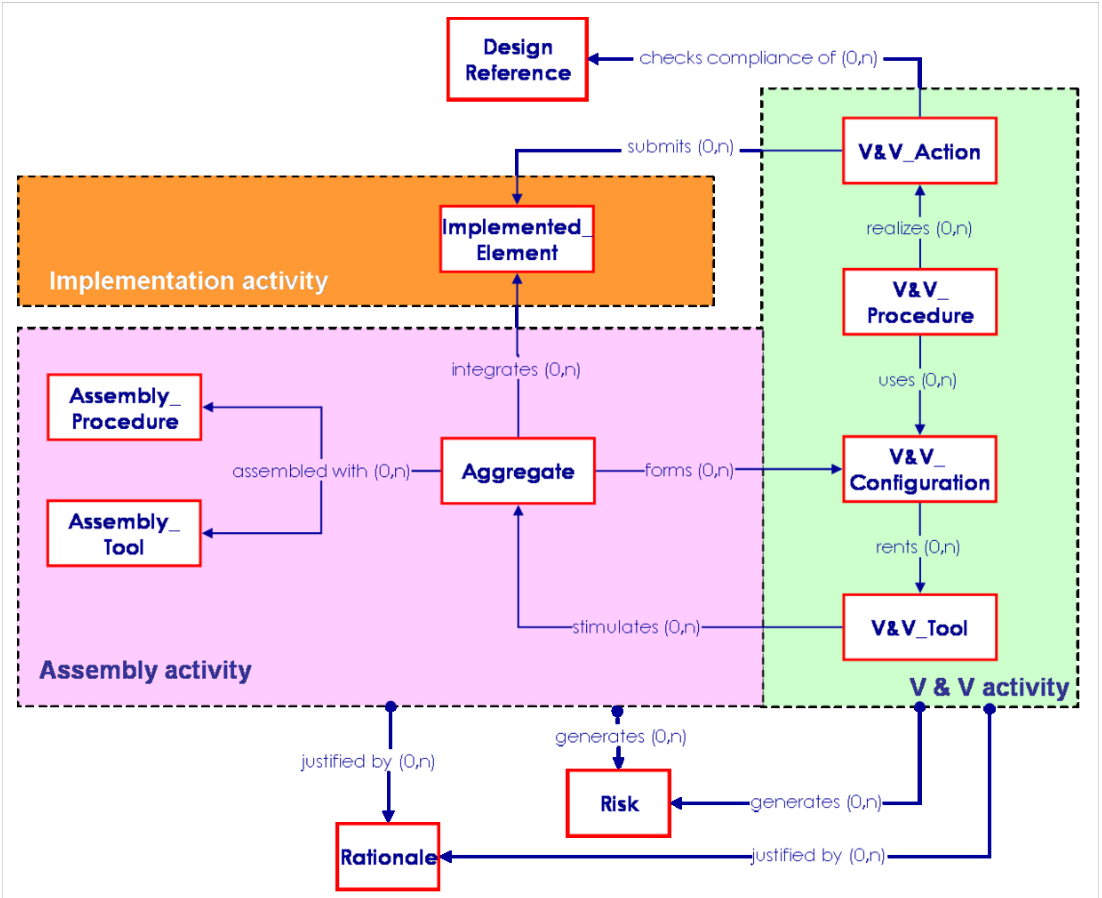


**Figure 2. Integration Elements Relationships with Other Engineering Elements.** (SEBoK Original)

## Checking and Correctness of Integration

The main items to be checked during the integration process include the following:

- The integration plan respects its template.
- The expected assembly order (integration strategy) is realistic.
- No system element and physical interface set out in the system design document is forgotten.
- Every interface and interaction between implemented elements is verified.
- Assembly procedures and assembly tools are available and validated prior to beginning the assembly.
- V&V procedures and tools are available and validated prior to beginning the verification.
- Integration reports are recorded.

## Methods and Techniques

Several different approaches are summarized above in the section Integration Strategy [1] (above) that may be used for integration, yet other approaches exist. In particular, important integration strategies for intensive software systems include: vertical integration, horizontal integration, and star integration.

### Coupling Matrix and N-squared Diagram

One of the most basic methods to define the aggregates and the order of integration would be the use of N-Squared diagrams (Grady 1994, 190).

In the integration context, the coupling matrices are useful for optimizing the aggregate definition and verification of interfaces:

- The integration strategy is defined and optimized by reorganizing the coupling matrix in order to group the implemented elements in aggregates, thus minimizing the number of interfaces to be verified between aggregates (see Figure 3).

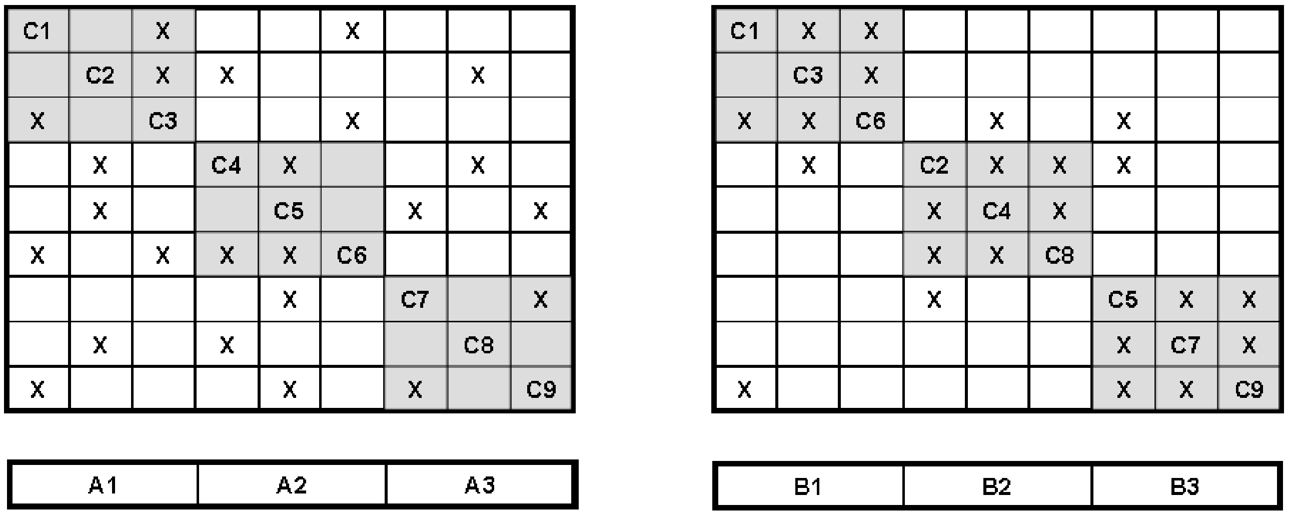


**Figure 3. Initial Arrangement of Aggregates on the Left; Final Arrangement After Reorganization on the Right.** (SEBoK Original)

- When verifying the interactions between aggregates, the matrix is an aid tool for fault detection. If by adding an implemented element to an aggregate an error is detected, the fault can be related to the implemented element, to the aggregate, or to the interfaces. If the fault is related to the aggregate, it can relate to any implemented element or any interface between the implemented elements internal to the aggregate.

## Application to Product Systems, Service Systems, and Enterprise Systems

As the nature of implemented system elements and physical interfaces is different for these types of systems, the aggregates, the assembly tools, and the V&V tools are different. Some integration techniques are more appropriate to specific types of systems. Table 3 below provides some examples.

### Table 3. Different Integration Elements for Product, Service, and Enterprise Systems. (SEBoK Original)

| Element | Product System | Service System | Enterprise System |
|---|---|---|---|
| **System Element** | Hardware Parts (mechanics, electronics, electrical, plastic, chemical, etc.)<br>Operator Roles<br>Software Pieces | Processes, data bases, procedures, etc.<br>Operator Roles<br>Software Applications | Corporate, direction, division, department, project, technical team, leader, etc.<br>IT components |
| **Physical Interface** | Hardware parts, protocols, procedures, etc. | Protocols, documents, etc. | Protocols, procedures, documents, etc. |
| **Assembly Tools** | Harness, mechanical tools, specific tools<br>Software Linker | Documentation, learning course, etc. | Documentation, learning, moving of office |
| **Verification Tools** | Test bench, simulator, launchers, stub/cap | Activity/scenario models, simulator, human roles rehearsal, computer, etc.<br>Skilled Experts | Activity/scenario models, simulator, human roles rehearsal |
| **Validation Tools** | Operational environment | Operational environment | Operational environment |

| | | | |
|---|---|---|---|
| **Recommended Integration Techniques** | Top down integration technique<br>Bottom Up Integration technique | Subsets integration technique (functional chains) | Global integration technique<br>Incremental integration |

## Practical Considerations

Key pitfalls and good practices related to system integration are described in the next two sections.

### Pitfalls

Some of the key pitfalls encountered in planning and performing SE Measurement are provided in Table 4.

**Table 4. Major Pitfalls with System Integration. (SEBoK Original)**

| Pitfall | Description |
|---|---|
| **Delay of expected element** | The experience shows that the implemented elements always do not arrive in the expected order and the tests never proceed or result as foreseen; therefore, the integration strategy should allow a great flexibility. |
| **Big-bang not appropriate** | The "big-bang" integration technique is not appropriate for a fast detection of faults. It is thus preferable to verify the interfaces progressively all along the integration. |
| **Integration plan too late** | The preparation of the integration activities is planned too late in the project schedule, typically when first implemented elements are delivered. |

### Good Practices

Some good practices gathered from the references are provided in Table 5.

**Table 5. Proven Practices with System Integration. (SEBoK Original)**

| Practice | Description |
|---|---|
| **Start earlier development of means** | The development of assembly tools and verification and validation tools can be take as long as the system development itself. It should be started as early as possible as soon as the preliminary design is nearly frozen. |
| **Integration means seen as enabling systems** | The development of integration means (assembly tools, verification, and validation tools) can be seen as enabling systems, using system definition and system realization processes as described in this SEBoK, and managed as projects. These projects can be led by the project of the corresponding system-of-interest, but assigned to specific system blocks, or can be subcontracted as separate projects. |
| **Use coupling matrix** | A good practice consists in gradually integrating aggregates in order to detect faults more easily. The use of the coupling matrix applies for all strategies and especially for the bottom up integration strategy. |
| **Flexible integration plan and schedule** | The integration process of complex systems cannot be easily foreseeable and its progress control difficult to observe. This is why it is recommended to plan integration with specific margins, using flexible techniques, and integrating sets by similar technologies. |
| **Integration and design teams** | The integration responsible should be part of the design team. |

## References

### Works Cited

DAU. February 19, 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

### Primary References

INCOSE. 2010. *Systems Engineering Handbook: A Guide for Systems Life Cycle Processes and Activities*. Version 3.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

### Additional References

Buede, D.M. 2009. *The Engineering Design of Systems: Models and Methods.* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense. February 19, 2010.

Gold-Bernstein, B. and W.A. Ruh. 2004. *Enterprise integration: The essential guide to integration solutions*. Boston, MA, USA: Addison Wesley Professional.

Grady, J.O. 1994. *System integration*. Boca Raton, FL, USA: CRC Press, Inc.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight* 12(4):59-63.

Jackson, S. 2010. *Architecting Resilient Systems: Accident Avoidance and Survival and Recovery from Disruptions*. Hoboken, NJ, USA: John Wiley & Sons.

Reason, J. 1997. *Managing the Risks of Organizational Accidents.* Aldershot, UK: Ashgate Publishing Limited.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

## References

[1] http://sebokwiki.org/1.0.1/index.php?title=System_Integration#Integration_Strategy

# System Verification

***Lead Authors:*** *John Snoderly, Alan Faisandier*

System Verification is a set of actions used to check the *correctness* of any element, such as a system element, a system, a document, a service, a task, a requirement, etc. These types of actions are planned and carried out throughout the life cycle of the system. Verification is a generic term that needs to be instantiated within the context it occurs. As a process, verification is a transverse activity to every life cycle stage of the system. In particular, during the development cycle of the system, the verification process is performed in parallel with the system definition and system realization processes and applies to any activity and any product resulting from the activity. The activities of every life cycle process and those of the verification process can work together. For example, the integration process frequently uses the verification process. It is important to remember that verification, while separate from validation, is intended to be performed in conjunction with validation.

## Definition and Purpose

Verification is the confirmation, through the provision of objective evidence, that specified requirements have been fulfilled. With a note added in ISO/IEC/IEEE 15288, the scope of verification includes a set of activities that compares a system or system element against the requirements, architecture and design characteristics, and other properties to be verified (ISO/IEC/IEEE 2015). This may include, but is not limited to, specified requirements, design description, and the system itself.

The purpose of verification, as a generic action, is to identify the faults/defects introduced at the time of any transformation of inputs into outputs. Verification is used to provide information and evidence that the transformation was made according to the selected and appropriate methods, techniques, standards, or rules.

Verification is based on tangible evidence; i.e., it is based on information whose veracity can be demonstrated by factual results obtained from techniques such as inspection, measurement, testing, analysis, calculation, etc. Thus, the process of verifying a system (product, service, enterprise, or system of systems (SoS)) consists of comparing the realized characteristics or properties of the product, service, or enterprise against its expected design properties.

## Principles and Concepts

### Concept of Verification Action

#### Why Verify?

In the context of human realization, any human thought is susceptible to error. This is also the case with any engineering activity. Studies in human reliability have shown that people trained to perform a specific operation make around 1-3 errors per hour in best case scenarios. In any activity, or resulting outcome of an activity, the search for potential errors should not be neglected, regardless of whether or not one thinks they will happen or that they should not happen; the consequences of errors can cause extremely significant failures or threats.

A **verification action** is defined, and then performed, as shown in Figure 1.

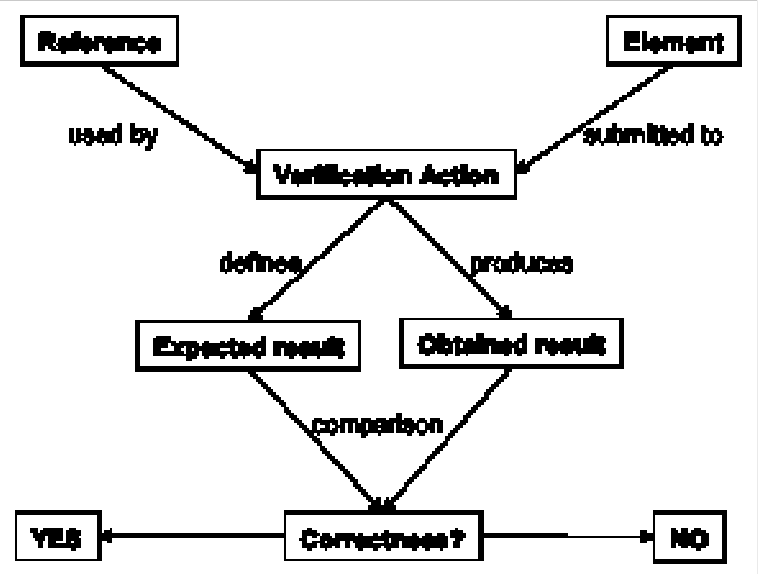


**Figure 1. Definition and Usage of a Verification Action.** (SEBoK Original)

The definition of a verification action applied to an engineering element includes the following:

- Identification of the element on which the verification action will be performed
- Identification of the reference to define the expected result of the verification action (see examples of reference in Table 1)

The performance of a verification action includes the following:

- Obtaining a result by performing the verification action onto the submitted element
- Comparing the obtained result with the expected result
- Deducing the degree of correctness of the element

### What to Verify?

Any engineering element can be verified using a specific reference for comparison: stakeholder requirement, system requirement, function, system element, document, etc. Examples are provided in Table 1.

**Table 1. Examples of Verified Items. (SEBoK Original)**

| Items | Explanation for Verification |
|---|---|
| **Document** | To verify a document is to check the application of drafting rules. |
| **Stakeholder Requirement and System Requirement** | To verify a stakeholder requirement or a system requirement is to check the application of syntactic and grammatical rules, characteristics defined in the stakeholder requirements definition process, and the system requirements definition process such as necessity, implementation free, unambiguous, consistent, complete, singular, feasible, traceable, and verifiable. |
| **Design** | To verify the design of a system is to check its logical and physical architecture elements against the characteristics of the outcomes of the design processes. |
| **System** | To verify a system (product, service, or enterprise) is to check its realized characteristics or properties against its expected design characteristics. |
| **Aggregate** | To verify an aggregate for integration is to check every interface and interaction between implemented elements. |
| **Verification Procedure** | To verify a verification procedure is to check the application of a predefined template and drafting rules. |

### Verification versus Validation

The term *verification* is often associated with the term *validation* and understood as a single concept of *V&V*. Validation is used to ensure that *one is working the right problem*, whereas verification is used to ensure that *one has solved the problem right* (Martin 1997). From an actual and etymological meaning, the term verification comes from the Latin *verus*, which means truth, and *facere*, which means to make/perform. Thus, verification means to prove that something is *true* or correct (a property, a characteristic, etc.). The term validation comes from the Latin *valere*, which means to become strong, and has the same etymological root as the word *value*. Thus, validation means to prove that something has the right features to produce the expected effects. (Adapted from "Verification and Validation in plain English" (Lake INCOSE 1999).)

The main differences between the verification process and the validation process concern the references used to check the correctness of an element, and the acceptability of the effective correctness.

- Within verification, comparison between the expected result and the obtained result is generally binary, whereas within validation, the result of the comparison may require a judgment of value regarding whether or not to accept the obtained result compared to a threshold or limit.
- Verification relates more to one element, whereas validation relates more to a set of elements and considers this set as a whole.
- Validation presupposes that verification actions have already been performed.
- The techniques used to define and perform the verification actions and those for validation actions are very similar.

### Integration, Verification, and Validation of the System

There is sometimes a misconception that verification occurs after integration and before validation. In most cases, it is more appropriate to begin verification activities during development or implementation and to continue them into deployment and use.

Once the system elements have been realized, they are integrated to form the complete system. Integration consists of assembling and performing verification actions as stated in the integration process. A final validation activity generally occurs when the system is integrated, but a certain number of validation actions are also performed parallel to the system integration in order to reduce the number of verification actions and validation actions while controlling the risks that could be generated if some checks are excluded. Integration, verification, and validation are intimately processed together due to the necessity of optimizing the strategy of verification and validation, as well as the strategy of integration.

## Process Approach

### Purpose and Principle of the Approach

The purpose of the verification process is to confirm that the system fulfills the specified design requirements. This process provides the information required to effect the remedial actions that correct non-conformances in the realized system or the processes that act on it - see ISO/IEC/IEEE 15288 (ISO/IEC/IEEE 2015).

Each system element and the complete system itself should be compared against its own design references (specified requirements). As stated by Dennis Buede, *verification is the matching of [configuration items], components, sub-systems, and the system to corresponding requirements to ensure that each has been built right* (Buede 2009). This means that the verification process is instantiated as many times as necessary during the global development of the system. Because of the generic nature of a process, the verification process can be applied to any engineering element that has conducted to the definition and realization of the system elements and the system itself.

Facing the huge number of potential verification actions that may be generated by the normal approach, it is necessary to optimize the verification strategy. This strategy is based on the balance between what must be verified and constraints, such as time, cost, and feasibility of testing, which naturally limit the number of verification actions and the risks one accepts when excluding some verification actions.

Several approaches exist that may be used for defining the verification process. The International Council on Systems Engineering (INCOSE) dictates that two main steps are necessary for verification: planning and performing verification actions (INCOSE 2012). NASA has a slightly more detailed approach that includes five main steps: prepare verification, perform verification, analyze outcomes, produce a report, and capture work products (NASA December 2007, 1-360, p. 102). Any approach may be used, provided that it is appropriate to the scope of the system, the constraints of the project, includes the activities of the process listed below in some way, and is appropriately coordinated with other activities.

**Generic inputs** are baseline references of the submitted element. If the element is a system, inputs are the logical and physical architecture elements as described in a system design document, the design description of internal interfaces to the system and interfaces requirements external to the system, and by extension, the system requirements. **Generic outputs** define the verification plan that includes verification strategy, selected verification actions, verification procedures, verification tools, the verified element or system, verification reports, issue/trouble reports, and change requests on design.

## Activities of the Process

To establish the verification strategy drafted in a verification plan (this activity is carried out concurrently to system definition activities), the following steps are necessary:

- Identify verification scope by listing as many characteristics or properties as possible that should be checked. The number of verification actions can be extremely high.
- Identify constraints according to their origin (technical feasibility, management constraints as cost, time, availability of verification means or qualified personnel, and contractual constraints that are critical to the mission) that limit potential verification actions.
- Define appropriate verification techniques to be applied, such as inspection, analysis, simulation, peer-review, testing, etc., based on the best step of the project to perform every verification action according to the given constraints.
- Consider a tradeoff of what should be verified (scope) taking into account all constraints or limits and deduce what can be verified; the selection of verification actions would be made according to the type of system, objectives of the project, acceptable risks, and constraints.
- Optimize the verification strategy by defining the most appropriate verification technique for every verification action while defining necessary verification means (tools, test-benches, personnel, location, and facilities) according to the selected verification technique.
- Schedule the execution of verification actions in the project steps or milestones and define the configuration of elements submitted to verification actions (this mainly involves testing on physical elements).

Performing verification actions includes the following tasks:

- Detail each verification action; in particular, note the expected results, the verification techniques to be applied, and the corresponding means required (equipment, resources, and qualified personnel).
- Acquire verification means used during system definition steps (qualified personnel, modeling tools, mocks-up, simulators, and facilities), and then those used during the integration step (qualified personnel, verification tools, measuring equipment, facilities, verification procedures, etc.).
- Carry out verification procedures at the right time, in the expected environment, with the expected means, tools, and techniques.

- Capture and record the results obtained when performing verification actions using verification procedures and means.

The obtained results must be analyzed and compared to the expected results so that the status may be recorded as either *compliant* or *non-compliant*. Systems engineering (SE) practitioners will likely need to generate verification reports, as well as potential issue/trouble reports, and change requests on design as necessary.

Controlling the process includes the following tasks:

- Update the verification plan according to the progress of the project; in particular, planned verification actions can be redefined because of unexpected events.
- Coordinate verification activities with the project manager: review the schedule and the acquisition of means, personnel, and resources. Coordinate with designers for issues/trouble/non-conformance reports and with the configuration manager for versions of the physical elements, design baselines, etc.

## Artifacts and Ontology Elements

This process may create several artifacts such as:

- verification plans (contain the verification strategy)
- verification matrices (contain the verification action, submitted element, applied technique, step of execution, system block concerned, expected result, obtained result, etc.)
- verification procedures (describe verification actions to be performed, verification tools needed, the verification configuration, resources and personnel needed, the schedule, etc.)
- verification reports
- verification tools
- verified elements
- issue / non-conformance / trouble reports
- change requests to the design

This process utilizes the ontology elements displayed in Table 2 below.

**Table 2. Main Ontology Elements as Handled within Verification. (SEBoK Original)**

| Element | Definition |
|---|---|
| | **Attributes (examples)** |
| **Verification Action** | A verification action describes what must be verified (the element as reference) on which element, the expected result, the verification technique to apply, on which level of decomposition. |
| | Identifier, name, description |
| **Verification Procedure** | A verification procedure groups a set of verification actions performed together (as a scenario of tests) in a gin verification configuration. |
| | Identifier, name, description, duration, unit of time |
| **Verification Tool** | A verification tool is a device or physical tool used to perform verification procedures (test bench, simulator, cap/stub, launcher, etc.). |
| | Identifier, name, description |
| **Verification Configuration** | A verification configuration groups all physical elements (aggregates and verification tools) necessary to perform a verification procedure. |
| | Identifier, name, description |

| | |
|---|---|
| **Risk** | An event having a probability of occurrence and a gravity degree on its consequence onto the system mission or on other characteristics (used for technical risk in engineering). A risk is the combination of vulnerability and of a danger or a threat. |
| **Rationale** | An argument that provides the justification for the selection of an engineering element.<br><br>Identifier, name, description (rationale, reasons for defining a verification action, a verification procedure, for using a verification tool, etc.) |

## Methods and Techniques

There are several verification techniques to check that an element or a system conforms to its design references or its specified requirements. These techniques are almost the same as those used for validation, though the application of the techniques may differ slightly. In particular, the purposes are different; verification is used to detect faults/defects, whereas validation is used to provide evidence for the satisfaction of (system and/or stakeholder) requirements. Table 3 below provides descriptions of some techniques for verification.

**Table 3. Verification Techniques. (SEBoK Original)**

| Verification Technique | Description |
|---|---|
| **Inspection** | Technique based on visual or dimensional examination of an element; the verification relies on the human senses or uses simple methods of measurement and handling. Inspection is generally non-destructive, and typically includes the use of sight, hearing, smell, touch, and taste, simple physical manipulation, mechanical and electrical gauging, and measurement. No stimuli (tests) are necessary. The technique is used to check properties or characteristics best determined by observation (e.g. paint color, weight, documentation, listing of code, etc.). |
| **Analysis** | Technique based on analytical evidence obtained without any intervention on the submitted element using mathematical or probabilistic calculation, logical reasoning (including the theory of predicates), modeling and/or simulation under defined conditions to show theoretical compliance. Mainly used where testing to realistic conditions cannot be achieved or is not cost-effective. |
| **Analogy or Similarity** | Technique based on evidence of similar elements to the submitted element or on experience feedback. It is absolutely necessary to show by prediction that the context is invariant that the outcomes are transposable (models, investigations, experience feedback, etc.). Similarity can only be used if the submitted element is similar in design, manufacture, and use; equivalent or more stringent verification actions were used for the similar element, and the intended operational environment is identical to or less rigorous than the similar element. |
| **Demonstration** | Technique used to demonstrate correct operation of the submitted element against operational and observable characteristics without using physical measurements (no or minimal instrumentation or test equipment). Demonstration is sometimes called 'field testing'. It generally consists of a set of tests selected by the supplier to show that the element response to stimuli is suitable or to show that operators can perform their assigned tasks when using the element. Observations are made and compared with predetermined/expected responses. Demonstration may be appropriate when requirements or specification are given in statistical terms (e.g. mean time to repair, average power consumption, etc.). |
| **Test** | Technique performed onto the submitted element by which functional, measurable characteristics, operability, supportability, or performance capability is quantitatively verified when subjected to controlled conditions that are real or simulated. Testing often uses special test equipment or instrumentation to obtain accurate quantitative data to be analyzed. |
| **Sampling** | Technique based on verification of characteristics using samples. The number, tolerance, and other characteristics must be specified to be in agreement with the experience feedback. |

## Practical Considerations

Key pitfalls and good practices related to this topic are described in the next two sections.

### Pitfalls

Some of the key pitfalls encountered in planning and performing System Verification are provided in Table 4.

**Table 4. Major Pitfalls with System Verification (SEBoK Original)**

| Pitfall | Description |
|---|---|
| Confusion between verification and validation | Confusion between verification and validation causes developers to take the wrong reference/baseline to define verification and validation actions and/or to address the wrong level of granularity (detail level for verification, global level for validation). |
| No verification strategy | One overlooks verification actions because it is impossible to check every characteristic or property of all system elements and of the system in any combination of operational conditions and scenarios. A strategy (justified selection of verification actions against risks) must be established. |
| Save or spend time | Skip verification activity to save time. |
| Use only testing | Use only testing as a verification technique. Testing requires checking products and services only when they are implemented. Consider other techniques earlier during design; analysis and inspections are cost effective and allow discovering early potential errors, faults, or failures. |
| Stop verifications when funding is diminished | Stopping the performance of verification actions when budget and/or time are consumed. Prefer using criteria such as coverage rates to end verification activity. |

### Proven Practices

Some proven practices gathered from the references are provided in Table 5.

**Table 5. Proven Practices with System Verification. (SEBoK Original)**

| Practice | Description |
|---|---|
| Start verifications early in the development | The earlier characteristics of an element are verified in the project, the easier the corrections are to do and the consequences on schedule and cost will be fewer. |
| Define criteria ending verifications | Carrying out verification actions without limits generates a risk of drift for costs and deadlines. Modifying and verifying in a non-stop cycle until arriving at a perfect system is the best way to never supply the system. Thus, it is necessary to set limits of cost, time, and a maximum number of modification loops back for each verification action type, ending criteria (percentages of success, error count detected, coverage rate obtained, etc.). |
| Involve design responsible with verification | Include the verification responsible in the designer team or include some designer onto the verification team. |

## References

### Works Cited

Buede, D.M. 2009. *The Engineering Design of Systems: Models and Methods.* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

INCOSE. 2012. *INCOSE Systems Engineering Handbook,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Lake, J. 1999. "V & V in Plain English." International Council on Systems Engineering (INCOSE) 9th Annual International Symposium, Brighton, UK, 6-10 June 1999.

NASA. 2007. *Systems Engineering Handbook*. Washington, DC, USA: National Aeronautics and Space Administration (NASA), December 2007. NASA/SP-2007-6105.

### Primary References

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/ Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), December 2007. NASA/SP-2007-6105.

### Additional References

Buede, D.M. 2009. *The Engineering Design of Systems: Models and Methods,* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

ECSS. 2009. *Systems Engineering General Requirements.* Noordwijk, Netherlands: Requirements and Standards Division, European Cooperation for Space Standardization (ECSS), 6 March 2009. ECSS-E-ST-10C.

MITRE. 2011. "Verification and Validation." in *Systems Engineering Guide.* Accessed 11 March 2012 at [[1]].

SAE International. 1996. *Certification Considerations for Highly-Integrated or Complex Aircraft Systems.* Warrendale, PA, USA: SAE International, ARP475.

SEI. 2007. "Measurement and Analysis Process Area" in *Capability Maturity Model Integrated (CMMI) for Development, version 1.2.* Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

## References

[1] http://mitre.org/work/systems_engineering/guide/se_lifecycle_building_blocks/test_evaluation/verification_validation.html

# System Validation

***Lead Author:*** *Alan Faisandier*, ***Contributing Author:*** *Rick Adcock*

System Validation is a set of actions used to check the compliance of any element (a system element, a system, a document, a service, a task, a system requirement, etc.) with its purpose and functions. These actions are planned and carried out throughout the life cycle of the system. Validation is a generic term that needs to be instantiated within the context it occurs. When understood as a process, validation is a transverse activity to every life cycle stage of the system. Particularly during the development cycle of the system, the validation process is performed in parallel with the system definition and system realization processes and applies to any activity and product resulting from this activity. The validation process is not limited to a phase at the end of system development, but generally occurs at the end of a set of life cycle tasks or activities, and always at the end of each milestone of a development project. It may be performed on an iterative basis on every produced engineering element during development and may begin with the validation of the expressed stakeholder requirements. When the validation process is applied to the system when completely integrated, it is often called *final validation*. It is important to remember that while system validation is separate from verification, the activities are complementary and intended to be performed in conjunction.

## Definition and Purpose

Validation is the confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled. With a note added in ISO 9000:2005: *validation is the set of activities that ensure and provide confidence that a system is able to accomplish its intended use, goals, and objectives (i.e., meet stakeholder requirements) in the intended operational environment* (ISO 2005).

The purpose of validation, as a generic action, is to establish the compliance of any activity output as compared to inputs of the activity. It is used to provide information and evidence that the transformation of inputs produced the expected and *right* result. Validation is based on tangible evidence; i.e., it is based on information whose veracity can be demonstrated by factual results obtained from techniques or methods such as inspection, measurement, test, analysis, calculation, etc. Thus, to validate a system (product, service, or enterprise) consists of demonstrating that it satisfies its system requirements and eventually the stakeholder's requirements depending on contractual practices. From a global standpoint, the purpose of validating a system is to acquire confidence in the system's ability to achieve its intended mission, or use, under specific operational conditions.

## Principles

### Concept of Validation Action

#### Why Validate?

The primary goal of systems engineering (SE) is to develop a solution that meets the needs and requirements of stakeholders. Validation is the process by which engineers ensure that the system will meet these needs and requirements.

A **validation action** is defined and then performed (see Figure 1, below).

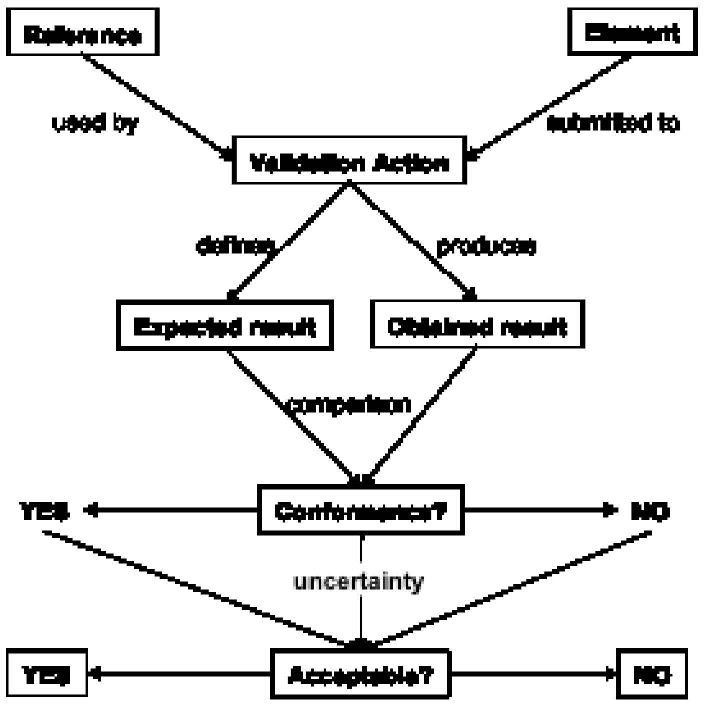


**Figure 1. Definition and Usage of a Validation Action.** (SEBoK Original)

A validation action applied to an engineering element includes the following:

- Identification of the element on which the validation action will be performed.
- Identification of the reference that defines the expected result of the validation action.

Performing the validation action includes the following:

- Obtaining a result by performing the validation action onto the submitted element.
- Comparing the obtained result with the expected result.
- Deducing the degree of compliance of the element.
- Deciding on the acceptability of this compliance, because sometimes the result of the comparison may require a value judgment to decide whether to accept the obtained result as compared to the relevance of the context of use.

Note: If there is uncertainty about compliance, the cause could come from ambiguity in the requirements.

### What to Validate?

Any engineering element can be validated using a specific reference for comparison, such as stakeholder requirements, system requirements, functions, system elements, documents, etc. Examples are provided in Table 1 below:

### Table 1. Examples of Validated Items (SEBoK Original)

| Items | Explanation for Validation |
|---|---|
| **Document** | To validate a document is to make sure its content is compliant with the inputs of the task that produced the document. |
| **Stakeholder Requirement and System Requirement** | To validate a stakeholder requirement is to make sure its content is justified and relevant to stakeholders' expectations, complete and expressed in the language of the customer or end user. To validate a system requirement is to make sure its content translates correctly and/or accurately a stakeholder requirement to the language of the supplier. |
| **Design** | To validate the design of a system (logical and physical architectures) is to demonstrate that it satisfies its system requirements. |
| **System** | To validate a system (product, service, or enterprise) is to demonstrate that the product, service, or enterprise satisfies its system requirements and/or its stakeholder requirements. |
| **Activity** | To validate an activity or a task is to make sure its outputs are compliant with its inputs. |
| **Process** | To validate a process is to make sure its outcomes are compliant with its purpose. |

## Validation versus Verification

The Verification versus Validation section of the System Verification article gives fundamental differences between the two concepts and associated processes. The Table 2 provides information to help understand these differences.

### Table 2. Verification and Validation Differences (may vary with context). (SEBoK Original)

| Point of View | Verification | Validation |
|---|---|---|
| Purpose of the Activity | Detect, identify faults/defects (supplier oriented) | Acquire confidence (end user oriented) |
| Idea behind the Term | Based on truth (objective/unbiased) | Based on value judgement (more subjective) |
| Level of Concern | Detail and local | Global in the context of use |
| Vision | Glass box (how it runs inside) | Black box (application of inputs provides the expected effect) |
| Basic Method | Fine-tooth comb | Traceability matrix |
| System (Product, Service, Enterprise) | "Done Right" (respects the state of the art); focus on (physical) characteristics | "Does Right" (produces the expected effect); focus on services, functions |
| Baseline Reference for Comparison (Product, Service, Enterprise) | System design | System requirements (and stakeholder requirements) |
| Order of Performance | First | Second |
| Organization of Activity | Verification actions are defined and/or performed by development/designer team | Validation actions are defined and/or performed by experts and external members to development/designer team |

## Validation, Final Validation, and Operational Validation

System validation concerns the global system seen as a whole and is based on the totality of requirements (system requirements, stakeholders' requirements, etc.), but it is obtained gradually throughout the development stage in three non-exclusive ways:

- accumulating the results of verification actions and validation actions provided by the application of corresponding processes to every engineering element;
- performing final validation actions to the complete, integrated system in an industrial environment (as close as possible to the operational environment); and
- performing operational validation actions on the complete system in its operational environment (context of use).

## Verification and Validation Level per Level

It is impossible to carry out only a single global validation on a complete, integrated complex system. The sources of faults/defects could be important, and it would be impossible to determine the causes of non-conformance manifested during this global check. Generally, the system-of-interest (SoI) has been decomposed during design in a set of layers of systems. Thus, every system and system element is verified, validated, and possibly corrected before being integrated into the parent system of the higher level, as shown in Figure 2.

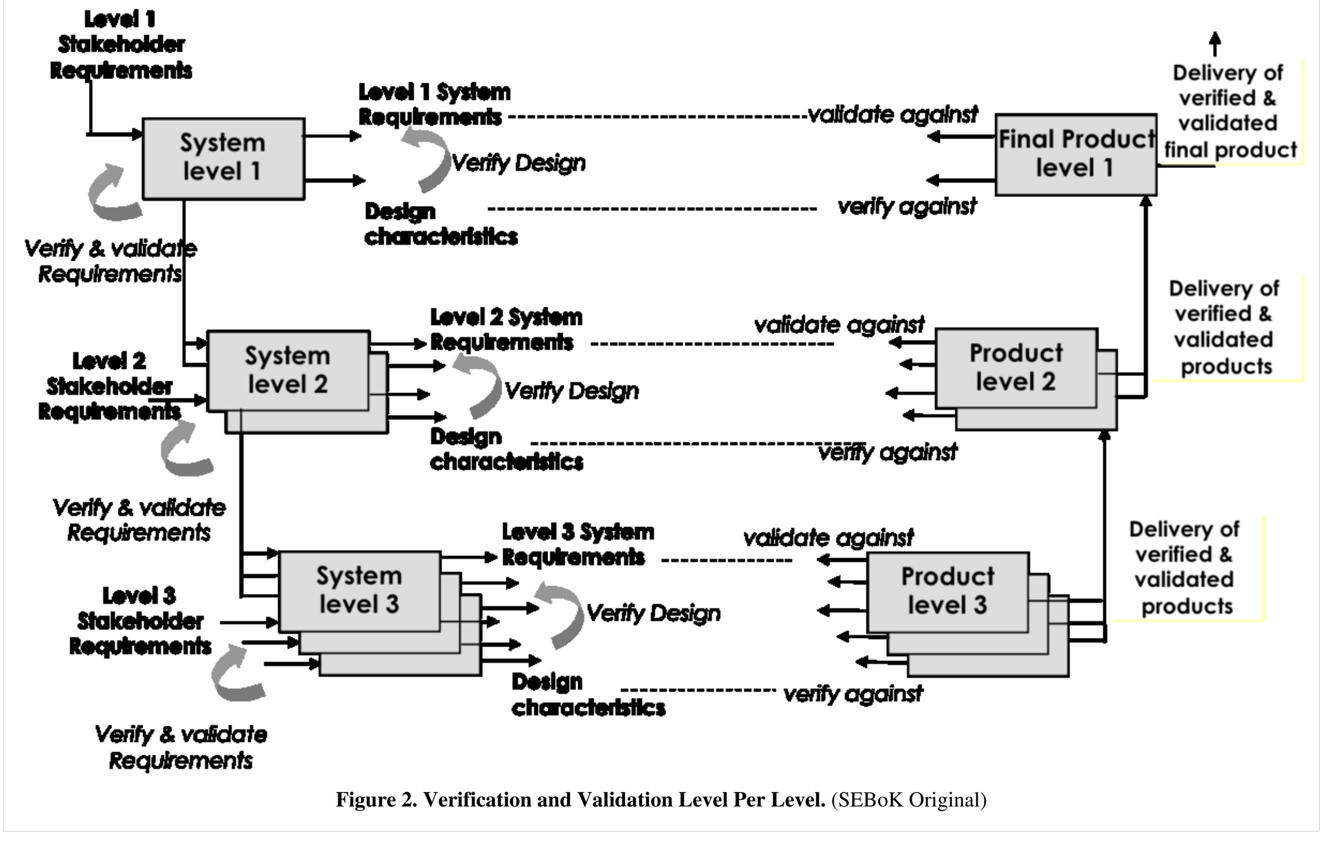

Level 1 Stakeholder Requirements
System level 1
Level 1 System Requirements
validate against
Final Product level 1
Delivery of verified & validated final product
Verify Design
Design characteristics
verify against
Verify & validate Requirements
Level 2 Stakeholder Requirements
System level 2
Level 2 System Requirements
validate against
Product level 2
Delivery of verified & validated products
Verify Design
Design characteristics
verify against
Verify & validate Requirements
Level 3 Stakeholder Requirements
System level 3
Level 3 System Requirements
validate against
Product level 3
Delivery of verified & validated products
Verify Design
Design characteristics
verify against
Verify & validate Requirements

**Figure 2. Verification and Validation Level Per Level.** (SEBoK Original)

As necessary, systems and system elements are partially integrated in subsets in order to limit the number of properties to be verified within a single step. For each level, it is necessary to perform a set of final validation actions to ensure that features stated at preceding levels are not damaged. Moreover, a compliant result obtained in a given environment can turn into a non-compliant result if the environment changes. Thus, as long as the system is not completely integrated and/or doesn't operate in the real operational environment, no result should be regarded as definitive.

## Verification Actions and Validation Actions Inside and Transverse to Levels

Inside each level of system decomposition, verification actions and validation actions are performed during system definition and system realization. This is represented in Figure 3 for the upper levels, and in Figure 4 for the lower levels. Stakeholder requirements definition and operational validation make the link between the two levels of the system decomposition.

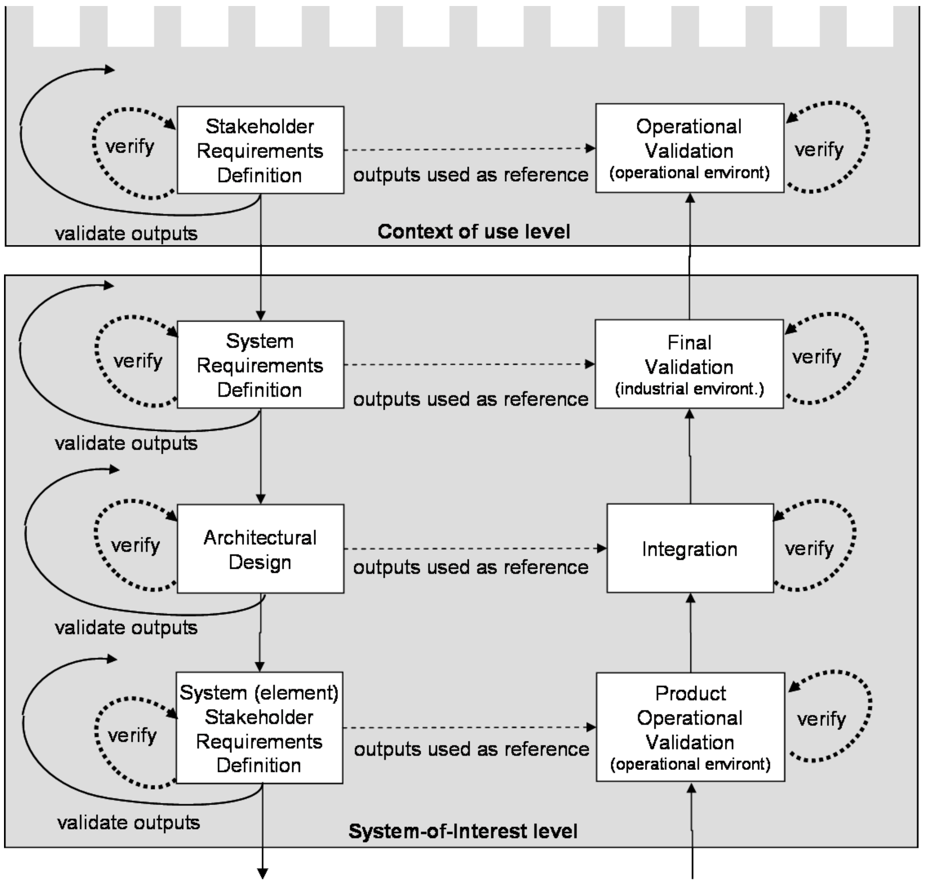

verify
Stakeholder Requirements Definition
outputs used as reference
Operational Validation (operational environt)
verify
validate outputs
Context of use level
verify
System Requirements Definition
outputs used as reference
Final Validation (industrial environt.)
verify
validate outputs
verify
Architectural Design
outputs used as reference
Integration
verify
validate outputs
verify
System (element) Stakeholder Requirements Definition
outputs used as reference
Product Operational Validation (operational environt)
verify
validate outputs
System-of-Interest level

**Figure 3. Verification and Validation Actions in Upper Levels of System Decomposition.** (SEBoK Original)

Operational validation of system element requirements and products makes the link between the two lower levels of the decomposition. See Figure 4 below.

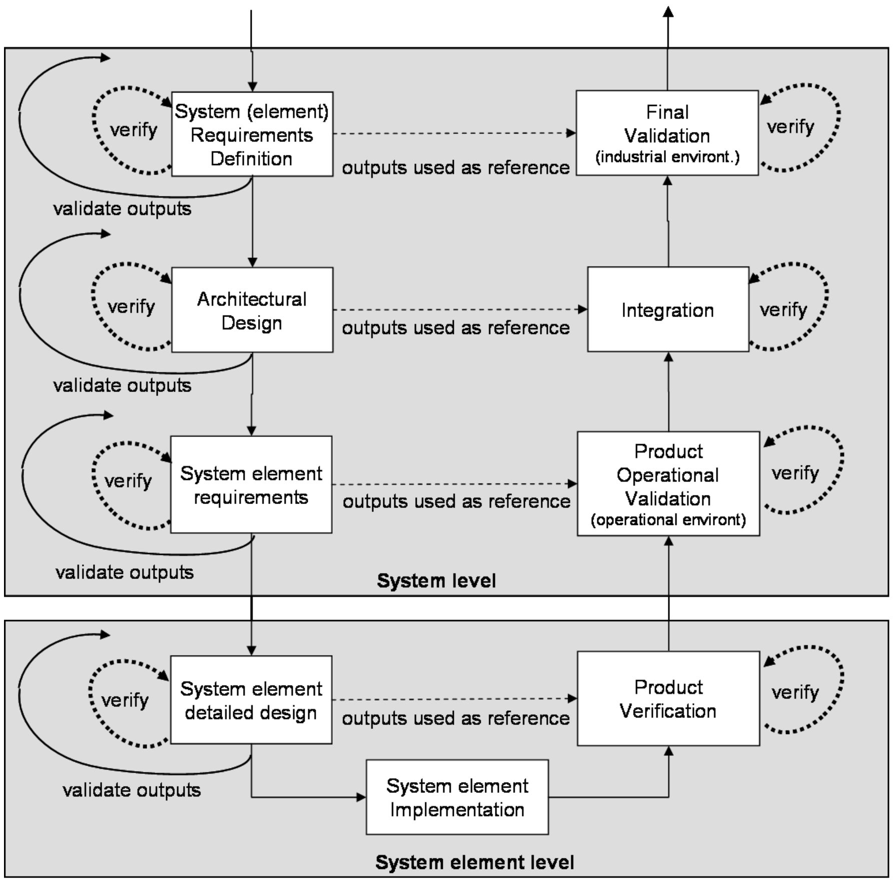

System (element) Requirements Definition
verify
validate outputs
outputs used as reference
Final Validation (industrial environt.)
verify
Architectural Design
verify
validate outputs
outputs used as reference
Integration
verify
System element requirements
verify
validate outputs
outputs used as reference
Product Operational Validation (operational environt)
verify
System level
System element detailed design
verify
validate outputs
outputs used as reference
Product Verification
verify
System element Implementation
System element level

**Figure 4. Verification and Validation Actions in Lower Levels of System Decomposition.** (SEBoK Original)

Note: The last level of system decomposition is dedicated to the realization of system elements and the vocabulary and number of activities may be different from what is seen in Figure 4.

### Verification and Validation Strategy

The difference between verification and validation is especially useful for elaborating on the integration strategy, the verification strategy, and the validation strategy. In fact, the efficiency of system realization is gained by optimizing the three strategies together to form what is often called the verification and validation strategy. This optimization consists of defining and performing the minimum number of verification and validation actions but detecting the maximum number of errors/faults/defects and achieving the maximum level of confidence in the system. The optimization takes into account the risks potentially generated if some verification actions or validation actions are excluded.

## Process Approach

### Purpose and Principles of the Approach

The purpose of the validation process is to provide objective evidence that the services provided by a system in use comply with stakeholder requirements and achieve its intended use in its intended operational environment (ISO/IEC/IEEE 15288 2015). The validation process performs a comparative assessment and confirms that the stakeholder requirements are correctly defined. Where variance is identified, it is recorded to guide future corrective actions. System validation is ratified by stakeholders (ISO/IEC/IEEE 15288 2015).

The validation process demonstrates that the realized end product satisfies its stakeholders' (customers' or other interested parties') expectations within the intended operational environments with validation performed by anticipated operators and/or users (NASA 2007, 1-360). Each system element, system, and the complete SoI are compared against their own applicable requirements (system requirements and stakeholder requirements). This means that the validation process is instantiated as many times as necessary during the global development of the system.

In order to ensure that validation is feasible, the implementation of requirements must be verifiable onto a defined element. It is essential to ensure that requirements are properly written, i.e., quantifiable, measurable, unambiguous, etc. In addition, verification/validation requirements are often written in conjunction with stakeholder and system requirements and provide a method for demonstrating the implementation of each system requirement or stakeholder requirement.

**Generic inputs** are references of requirements applicable to the submitted element. If the element is a system, inputs are system requirements and stakeholder requirements.

**Generic outputs** are the validation plan that includes validation strategy, selected validation actions, validation procedures, validation tools, validated elements or systems, validation reports, issue/trouble reports, and change requests on requirements or on the system.

## Activities of the Process

Major activities and tasks performed during this process include the following:

- Establish a validation strategy (often drafted in a validation plan). This activity is carried out concurrently to system definition activities:
  - Identify the validation scope that is represented by (system and/or stakeholder) requirements; normally, every requirement should be checked as the number of validation actions can be high.
  - Identify constraints according to their origin (technical feasibility, management constraints as cost, time, availability of validation means or qualified personnel, and contractual constraints that are critical to the mission) that limit or increase potential validation actions.
  - Define appropriate verification/validation techniques to be applied, such as inspection, analysis, simulation, review, testing, etc., depending on the best step of the project to perform every validation action according to constraints.
  - Consider a trade-off of what should be validated (scope) while taking into account all constraints or limits and deduce what can be validated objectively; selection of validation actions would be made according to the type of system, objectives of the project, acceptable risks, and constraints.
  - Optimize the validation strategy to define the most appropriate validation technique for every validation action, define necessary validation means (tools, test-benches, personnel, location, and facilities) according to the selected validation technique, schedule the execution of validation actions in the project steps or milestones, and define the configuration of elements submitted to validation actions (this is primarily about testing on physical elements).
- Perform validation actions, including the following tasks:
  - Detail each validation action. In particular, note the expected results, the validation technique to be applied, and the corresponding means necessary (equipment, resources, and qualified personnel).
  - Acquire validation means used during the system definition steps (qualified personnel, modeling tools, mocks-up, simulators, and facilities), then those means used during integration and final and operational steps (qualified personnel, validation tools, measuring equipment, facilities, validation procedures, etc.).
  - Carry out validation procedures at the right time, in the expected environment, with the expected means, tools, and techniques.

- Capture and record results obtained when performing validation actions using validation procedures and means.
- Analyze the obtained results and compare them to the expected results. Decide if they comply acceptably. Record whether the decision and status are compliant or not, and generate validation reports and potential issue/trouble reports, as well as change requests on (system or stakeholder) requirements as necessary.
- Control the process using following tasks:
  - Update the validation plan according to the progress of the project; in particular, planned validation actions can be redefined because of unexpected events.
  - Coordinate validation activities with the project manager regarding the schedule, acquisition of means, personnel, and resources. Coordinate with the designers for issue/trouble/non-conformance reports. Coordinate with the configuration manager for versions of physical elements, design baselines, etc.

## Artifacts and Ontology Elements

This process may create several artifacts, such as:

- a validation plan (contains the validation strategy)
- a validation matrix (contains for each validation action, submitted element, applied technique, step of execution, system block concerned, expected result, obtained result, etc.)
- validation procedures (describe the validation actions to be performed, the validation tools needed, the validation configuration, resources, personnel, schedule, etc.)
- validation reports
- validation tools
- the validated element
- issue, non-conformance, and trouble reports
- change requests on requirements, products, services, and enterprises

This process utilizes the ontology elements of Table 3.

### Table 3. Main Ontology Elements as Handled within Validation. (SEBoK Original)

| Element | Definition |
|---|---|
| | **Attributes (examples)** |
| **Validation Action** | A validation action describes what must be validated (the element as reference), on which element, the expected result, the verification technique to apply, on which level of decomposition. |
| | Identifier, name, description |
| **Validation Procedure** | A validation procedure groups a set of validation actions performed together (as a scenario of tests) in a given validation configuration. |
| | Identifier, name, description, duration, unit of time |
| **Validation Tool** | A validation tool is a device or physical tool used to perform validation procedures (test bench, simulator, cap/stub, launcher, etc.). |
| | Identifier, name, description |
| **Validation Configuration** | A validation configuration groups the physical elements necessary to perform a validation procedure. |
| | Identifier, name, description |

| | |
|---|---|
| **Risk** | An event having a probability of occurrence and a gravity degree on its consequence onto the system mission or on other characteristics (used for technical risk engineering). |
| | Identifier, name, description, status |
| **Rationale** | An argument that provides the justification for the selection of an engineering element. |
| | Identifier, name, description (rationale, reasons for defining a validation action, a validation procedure, for using a validation tool, etc.) |

### Methods and Techniques

The validation techniques are the same as those used for verification, but their purposes are different; verification is used to detect faults/defects, whereas validation is used to prove the satisfaction of (system and/or stakeholder) requirements.

The **validation traceability matrix** is introduced in the stakeholder requirements definition topic. It may also be extended and used to record data, such as a validation actions list, selected validation techniques to validate implementation of every engineering element ( stakeholder and system requirements in particular), expected results, and obtained results when validation actions have been performed. The use of such a matrix enables the development team to ensure that selected stakeholder and system requirements have been checked, or to evaluate the percentage of validation actions completed.

## Practical Considerations

Key pitfalls and good practices related to system validation are described in the next two sections.

### Pitfalls

Some of the key pitfalls encountered in planning and performing system validation are provided in Table 4.

**Table 4. Major Pitfalls with System Validation. (SEBoK Original)**

| Pitfall | Description |
|---|---|
| Start validation at the end of the project | A common mistake is to wait until the system has been entirely integrated and tested (design is qualified) to perform any sort of validation. Validation should occur as early as possible in the [product] life cycle (Martin 1997). |
| Use only testing | Use only testing as a validation technique. Testing requires checking products and services only when they are implemented. Consider other techniques earlier during design; analysis and inspections are cost effective and allow discovering early potential errors, faults, or failures. |
| Stop validation when funding is diminished | Stop the performance of validation actions when budget and/or time are consumed. Prefer using criteria such as coverage rates to end validation activity. |

### Proven Practices

Some good practices gathered from the references are provided in Table 5.

**Table 5. Proven Practices with System Validation. (SEBoK Original)**

| Practice | Description |
|---|---|
| Start Validation Plan Early | It is recommended to start the drafting of the validation plan as soon as the first requirements applicable to the system are known. If the writer of the requirements immediately puts the question to know how to validate whether the future system will answer the requirements, it is possible to:<br>• detect the unverifiable requirements<br>• anticipate, estimate cost, and start the design of validation means (as needed) such as test-benches, simulators<br>• avoid cost overruns and schedule slippages |
| Verifiable Requirements | According to Buede, a requirement is verifiable if a "finite, cost-effective process has been defined to check that the requirement has been attained." (Buede 2009) Generally, this means that each requirement should be quantitative, measurable, unambiguous, understandable, and testable. It is generally much easier and more cost-effective to ensure that requirements meet these criteria while they are being written. Requirement adjustments made after implementation and/or integration are generally much more costly and may have wide-reaching redesign implications. There are several resources which provide guidance on creating appropriate requirements - see the system definition knowledge area, stakeholder requirements, and system requirements topics for additional information. |
| Document Validation Actions | It is important to document both the validation actions performed and the results obtained. This provides accountability regarding the extent to which system, system elements, and subsystems fulfill system requirements and stakeholders' requirements. These data can be used to investigate why the system, system elements, or subsystems do not match the requirements and to detect potential faults/defects. When requirements are met, these data may be reported to organization parties. For example, in a safety critical system, it may be necessary to report the results of safety demonstration to a certification organization. Validation results may be reported to the acquirer for contractual aspects or to internal company for business purpose. |
| Involve Users with Validation | Validation will often involve going back directly to the users to have them perform some sort of acceptance test under their own local conditions. |
| Involve | Often the end users and other relevant stakeholders are involved in the validation process. |

The following are elements that should be considered when practicing any of the activities discussed as a part of system realization:

- Confusing verification and validation is a common issue. Validation demonstrates that the product, service, and/or enterprise as provided, fulfills its intended use, whereas verification addresses whether a local work product properly reflects its specified requirements. Validation actions use the same techniques as the verification actions (e.g., test, analysis, inspection, demonstration, or simulation).
- State who the witnesses will be (for the purpose of collecting the evidence of success), what general steps will be followed, and what special resources are needed, such as instrumentation, special test equipment or facilities, simulators, specific data gathering, or rigorous analysis of demonstration results.
- Identify the test facility, test equipment, any unique resource needs and environmental conditions, required qualifications and test personnel, general steps that will be followed, specific data to be collected, criteria for repeatability of collected data, and methods for analyzing the results.

## References

### Works Cited

Buede, D. M. 2009. *The engineering design of systems: Models and methods*. 2nd ed. Hoboken, NJ: John Wiley & Sons Inc.

INCOSE. 2012. *INCOSE Systems Engineering Handbook*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

NASA. 2007. *Systems Engineering Handbook*. Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105, December 2007.

### Primary References

INCOSE. 2012. *Systems Engineering Handbook*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015.*Systems and software engineering - system life cycle processes*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC 15288:2015.

NASA. 2007. *Systems Engineering Handbook*. Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105, December 2007.

### Additional References

Buede, D.M. 2009. *The engineering design of systems: Models and methods*. 2nd ed. Hoboken, NJ: John Wiley & Sons Inc.

DAU. February 19, 2010. *Defense Acquisition Guidebook (DAG).* Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense.

ECSS. 2009. Systems engineering general requirements. Noordwijk, Netherlands: Requirements and Standards Division, European Cooperation for Space Standardization (ECSS), ECSS-E-ST-10C. 6 March 2009.

MITRE. 2011. "Verification and Validation." *Systems Engineering Guide.* Accessed 11 March 2012 at [[1]].

SAE International. 1996. *Certification considerations for highly-integrated or complex aircraft systems.* Warrendale, PA, USA: SAE International, ARP475.

SEI. 2007. *Capability maturity model integrated (CMMI) for development*, version 1.2, measurement and analysis process area. Pittsburg, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Knowledge Area: System Deployment and Use

## System Deployment and Use

***Lead Authors:*** *Scott Jackson, John Snoderly*, ***Contributing Authors:*** *Garry Roedler*

System deployment and use are critical systems engineering (SE) activities that ensure that the developed system is operationally acceptable and that the responsibility for the effective, efficient, and safe operations of the system is transferred to the owner. Considerations for deployment and use must be included throughout the system life cycle.

### Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- System Deployment
- Operation of the System
- System Maintenance
- Logistics

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

### Overview

**System deployment** involves the transition of the capability to the ultimate end-user, as well as transition of support and maintenance responsibilities to the post-deployment support organization or organizations. It may include a period of reliability demonstration tests and the phasing out of legacy systems that the developed system replaces.

**System use** includes a continual assessment of the operational effectiveness of the deployed system or service, identification of mission threat and operational risk, and performance of the actions required to maintain operational effectiveness or evolve the capability to meet changing needs. Evolution of the operational system may occur with smaller maintenance actions or, if the changes cross an agreed-to threshold (complexity, risk, cost, etc.), may require a formal development project with deliberate planning and SE activities resulting in an enhanced system. As the operational phase is generally the longest in the system life cycle, activities that may occur during operation are allocated between two knowledge areas (KAs): System Deployment and Use and Product and Service Life Management.

The Product and Service Life Management knowledge area (KA) specifically deals with SE activities required for system evolution and end of system life including service life extension (SLE), capability updates, upgrades, and modernization during system operation, and system disposal and retirement. In contrast, the System Deployment and Use KA specifically deals with activities required to ensure that system operation can continue as expected. Planning for system deployment and use should begin early in the SE process to ensure successful transition into operational use.

## System Deployment and Use Fundamentals

System deployment and use includes the processes used to plan for and manage the transition of new or evolved systems and capabilities into operational use and the transition of support responsibilities to the eventual maintenance or support organization. The *use* stage normally represents the longest period of a system life cycle and, hence, generally accounts for the largest portion of the life cycle cost. These activities need to be properly managed in order to evaluate the actual system performance, effectiveness, and cost in its intended environment and within its specified utilization over its life cycle. Included in use fundamentals are the aspects of continuation of personnel training and certification.

As part of deployment/transition activities special conditions that may apply during the eventual decommissioning or disposal of the system are identified and accommodated in life cycle plans and system architectures and designs (See the System Definition KA for additional information). SE leadership ensures the developed system meets specified requirements, that it be used in the intended environment, and that when the system is transitioned into operation, it achieves the users' defined mission capabilities and can be maintained throughout the intended life cycle.

SE ensures that plans and clear criteria for transition into operation are developed and agreed to by relevant stakeholders and that planning is completed for system maintenance and support after the system is deployed. These plans should generally include reasonable accommodation for planned and potential evolution of the system and its eventual removal from operational use (for additional information on evolution and retirement, please see the Product and Service Life Management KA).

## References

None.

---

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# System Deployment

***Lead Authors:*** *Scott Jackson, Brian Gallagher*

As part of system deployment, on-site installation, check-out, integration, and testing must be carried out to ensure that the system is fit to be deployed into the field and/or put into an operational context. *Transfer* is the process that bridges the gap between qualification and use; it deals explicitly with the handoff from development to logistics, operations, maintenance, and support.

## Definition & Purpose

There are many different approaches to transition, or deployment, and many different views on what is included within transition. The SEBoK uses the ISO/IEC/IEEE 15288 definition of transition, as seen below (ISO/IEC/IEEE 15288 2015):

> *[The transition] process installs a verified system, together with relevant enabling systems, e.g., operating system, support system, operator training system, user training system, as defined in agreements. This process is used at each level in the system structure and in each stage to complete the criteria established for exiting the stage.*

Thinking in a linear fashion, the system is transitioned into operation and then would be used and maintained in the operational environment. However, there are other views on transition. For example, the NASA *Systems Engineering Handbook* states that transition can include delivery for end-use as well as delivery of components for integration (NASA 2007). Using this view, transition is the mechanism for moving system components from implementation activities into integration activities. The NASA discussion of transition also implies that transition can include sustainment activities:

> *The act of delivery or moving of a product from the location where the product has been implemented or integrated, as well as verified and validated, to a customer.*

Many systems are deployed using an iterative or evolutionary approach where operationally useful capabilities are developed and deployed incrementally. While these operationally useful capabilities are fully deployed and transitioned into operational use, transition of logistics, maintenance, and support may occur incrementally or be delayed until after the full system capability is delivered.

## Process Approaches

Just as there are multiple views on the definition of transition and deployment, there are also several ways to divide the activities required for transition. For example, the NASA *Systems Engineering Handbook* definition of transition states: *This act can include packaging, handling, storing, moving, transporting, installing, and sustainment activities* (2007). However, the SEBoK includes the topic of *sustainment* as separate from transition; this is instead covered under the maintenance and logistics topics. The International Council on Systems Engineering (INCOSE) views the transition process as two-step: planning and performance. Though there are several processes for deployment and transition, most generally include the following activities:

- **Develop a Deployment/Transition Strategy** - Planning for transition activities would ideally begin early in the SE life cycle, though it is possible to conduct these activities concurrently with realization activities. Planning should generally include some consideration of the common lower-level activities of installation, checkout, integration, and testing. Such activities are crucial to demonstrate that the system and the interfaces with the operational environment can function as intended and meet the contractual system specifications. For these activities to be effectively managed and efficiently implemented, the criteria, responsibility, and procedures for carrying out these activities should be clearly established and agreed upon during the planning phase.

- **Develop Plans for Transitioning Systems** - or system capabilities into operational use and support. Transition plans for the system or incremental system capabilities should be consistent with the overall transition strategy and agreed to by relevant stakeholders. Planning for transition will often include establishing a strategy for support, which may include organic support infrastructures, contractor logistics support, or other sources (Bernard et al. 2005, 1-49). It can also include defining the levels of support to be established. The strategy is important because it drives most of the other transition planning activities, as well as product design considerations. Transition plans should include considerations for coordination with the following activities:
  - **Installation** - Installation generally refers to the activities required to physically instate the system; this will likely include connecting interfaces to other systems such as electrical, computer, or security systems, and may include software interfaces as well. Installation planning should generally document the complexity of the system, the range of environmental conditions expected in the operational environment, any interface specifications, and human factors requirements such as safety. When real-world conditions require changes in the installation requirements, these should be documented and discussed with the relevant stakeholders.
  - **Integration** - Though system integration activities will generally be performed prior to installation, there may be additional steps for integrating the system into its operational setting. Additionally, if the system is being delivered incrementally, there will likely be integration steps associated with the transition (for more information on integration, please see the System Realization knowledge area (KA)).
  - **Verification and Validation (V&V)** - At this stage, V&V for physical, electrical, and mechanical checks may be performed in order to verify that the system has been appropriately installed. Acceptance tests conducted after delivery may become part of this process (for additional information on V&V, please see the System Realization KA). There are several types of acceptance tests which may be used:
  - **On-site Acceptance Test (OSAT)** - This test includes any field acceptance testing and is performed only after the system has successfully been situated in the operational environment. It may consist of functional tests to demonstrate that the system is functioning and performing properly.
    - *Field Acceptance Test* - This test includes flight and sea acceptance tests; it is performed, if applicable, only after the system has successfully passed the OSAT. The purpose of field testing is to demonstrate that the system meets the performance specifications called for in the system specifications in the actual operating environment.
    - *Operational Test and Evaluation (OT&E)* - An OT&E consists of a test series designed to estimate the operational effectiveness of the system.
    - *Evaluate the readiness of the system to transition into operations* - This is based upon the transition criteria identified in the transition plan. These criteria should support an objective evaluation of the system's readiness for transition. The integration, verification, and validation activities associated with transition may be used to gauge whether the system meets transition criteria.
    - *Analyze the results of transition activities throughout and any necessary actions* - As a result of analysis, additional transition activities and actions may be required. The analysis may also identify areas for improvement in future transition activities.

Some common issues that require additional considerations and SE activities are the utilization or replacement of legacy systems. It is also common for an organization to continue testing into the early operational phase. The following activities support these circumstances:

- **System Run-in** - After the successful completion of the various acceptance tests, the system(s) will be handed over to the user or designated post-deployment support organization. The tested system(s) may have to be verified for a stated period (called the system run-in, normally for one to two years) for the adequacy of reliability and maintainability (R&M) and integrated logistics support (ILS) deliverables. R&M are vital system operational characteristics having a dominant impact upon the operational effectiveness, the economy of in-service maintenance support, and the life cycle cost (LCC).

- **Phasing-In/Phasing-Out** - The need for phasing-in will usually be identified during the system definition, when it is clear that the new system entails the replacement of an existing system(s) (for additional information, please see the System Definition KA). These activities should help to minimize disruption to operations and, at the same time, minimize the adverse effect on operational readiness. It is also important that the phasing-in of a new system and the phasing-out of an existing system occur in parallel with the systems activities of the system run-in to maximize resource utilization. Other aspects of phasing-in/phasing-out to be considered include:
  - Proper planning for the phasing out of an existing system (if necessary).
  - For multi-user or complex systems, phase-by-phase introduction of the system according to levels of command, formation hierarchy, etc.
  - Minimum disruption to the current operations of the users.
  - Establishment of a feedback system from users on problems encountered in operation, etc.
  - Disposal process including handling of hazardous items, cost of disposal, approval etc.

## Applicable Methods & Tools

A system may have to undergo reliability demonstration testing (RDT) to ensure that it meets its contractual R&M guarantees. RDT is conducted under actual field conditions, especially for large systems purchased in small quantity. During RDT, the system is operated in the field within stated test duration and all field data are systematically recorded. At the end of the test period, analysis of the RDT data is performed. Data analysis should facilitate determination of system reliability. One possible output of this analysis is shown in Figure 1 below.

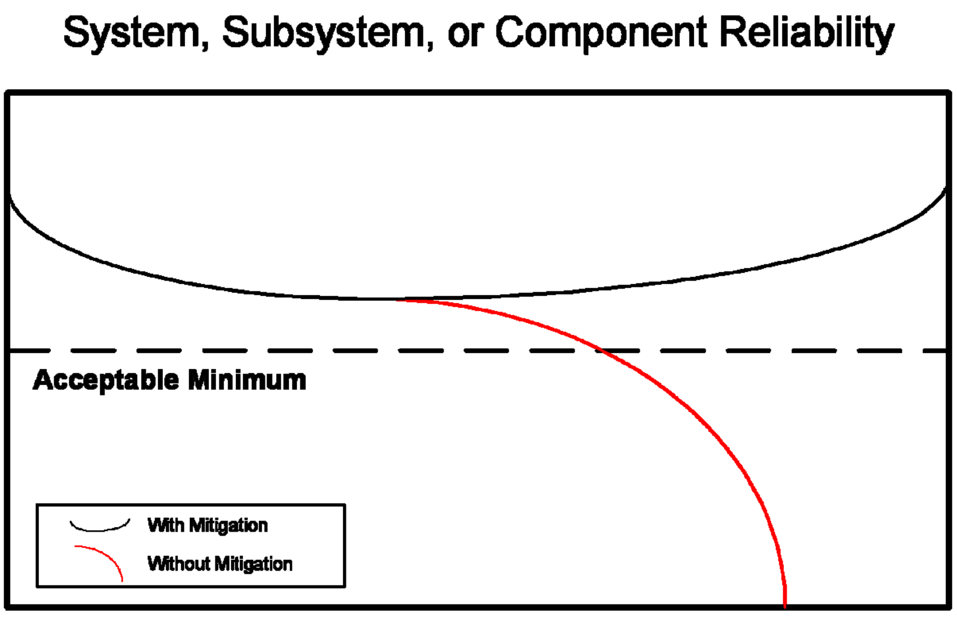


**Figure 1. Notional Reliability Analysis.** (SEBoK Original)

## References

### Works Cited

Bernard, S., B. Gallagher, R. Bate, H. Wilson. 2005. *CMMI® Acquisition Module (CMMI-AM),* version 1.1. Pittsburg, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU). CMU/SEI-2005-TR-011.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC/IEEE 15288:2015.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

### Primary References

INCOSE. 2011. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.1.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering - System Life Cycle Processes.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

NASA. 2007. *Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

### Additional References

None.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Operation of the System

***Lead Authors:*** *Scott Jackson, Brian Gallagher*, **Contributing Author:** *Leopoldo deCardenas*

The role of systems engineering (SE) during the operation of a system consists of ensuring that the system maintains key mission and business functions and is operationally effective. The systems engineer is one of the stakeholders who ensures that maintenance actions and other major changes are performed according to the long-term vision of the system. Both the maintenance actions and any implemented changes must meet the evolving needs of owning and operating stakeholders consistent with the documented and approved architecture. SE considerations will also include the eventual decommissioning or disposal of the system so that the disposal occurs according to disposal/retirement plans. Those plans must take into account and be compliant with relevant laws and regulations (for additional information on disposal or retirement, please see the Product and Service Life Management knowledge area (KA)). When the system-of-interest (SoI) replaces an existing or legacy system, it may be necessary to manage the migration between systems such that stakeholders do not experience a breakdown in services (INCOSE 2012).

## Definition & Purpose

This process assigns personnel to operate the system and monitors the services and operator-system performance. In order to sustain services, it identifies and analyzes operational problems in relation to agreements, stakeholder requirements, and organizational constraints (ISO/IEC/IEEE 2015).

The concept of operations (ConOps) establishes the foundation for initial design specifications according to the long-term vision. It is also possible that pre-planned program improvements (P3I) had been generated based on expected evolving requirements. Throughout the systems life cycle the operation of the system requires the systems engineer to be an active participant in reviews, change management and integrated master schedule activities to ensure the system operations continue to meet the evolving needs of stakeholders, and are consistent with the architecture through the eventual decommissioning or disposal of the system. In the event of decommissioning, a systems engineer must ensure disposal/retirement plans are compliant with relevant laws and regulations (for additional information on disposal or retirement, see the Product and Service Life Management KA).

Two additional areas are of interest to the systems engineer during system operation require special attention. First, it may be determined that a system is at the end of its life cycle, but the cost of replacing the system with a completely new design is too expensive. In this case, there will be intense engineering activities for service life extension program (SLEP). The SLEP solution will take into account obsolescence issues, diminishing manufacturing sources and material shortages (DMSMS), and changes in ConOps. Secondly, in the event that a new SoI is designed and produced as a complete replacement for an existing or legacy system, it will be necessary to manage the migration between systems such that stakeholders do not experience a breakdown in services (INCOSE 2012).

## Process Approaches

During the operational phase, SE activities ensure the system maintains certain operational attributes and usefulness throughout its expected life span. Maintaining operational effectiveness consists of evaluating certain operationally relevant attributes and trends, taking actions to prevent degradation of performance, evolving the system to meet changing mission or business needs (see the Product and Service Life Management KA), and eventually decommissioning the system and disposing of its components. During operation, data would be collected to evaluate the system and determine if changes should be made. It is important to include the process for data collection during operations when considering design and ConOps. In some cases, data may be collected by sensors and reported autonomously. In other cases, operators will identify and report on performance during operations. The systems engineer needs to understand how all data will be collected and presented for further analysis. The systems engineer will be involved in analysis of this data in several areas, including the following:

- Updating training and development of new training as required for operational and support personnel. Training is generally developed early with system design and production and executed during integration and operations. Determination of training updates or changes will be based on evaluation of the operational and support personnel.
- Evaluation of operational effectiveness. Early in the planning phases of a new system or capability, measures of operational effectiveness are established based on mission and business goals. These measures are important during system operation. These attributes are unique for each system and represent characteristics describing the usefulness of the system as defined and agreed to by system stakeholders. Systems engineers monitor and analyze these measurements and recommend actions.
- Failure reporting and corrective actions (FRACA) activities will involve the collection and analysis of data during operations. FRACA data will provide trends involving failures that may require design or component changes. Some failures may also result in safety issues requiring operational modifications until the offending elements under analysis can be corrected. If components or systems must be returned to maintenance facilities for corrective repairs, there will be operational and business impacts due to increased unavailability and unplanned transportation cost.

### Applicable Methods & Tools

Operations manuals generally provide operators the steps and activities required to run the system.

#### Training and Certification

Adequate training must be provided for the operators who are required to operate the system. There are many objectives of training:

- Provide initial training for all operators in order to equip them with the skill and knowledge to operate the system. Ideally, this process will begin prior to system transition and will facilitate delivery of the system. It is important to define the certification standards and required training materials up front (for more information on material supply, please see Logistics).
- Provide continuation training to ensure currency of knowledge.
- Monitor the qualification/certification of the operators to ensure that all personnel operating the system meet the minimum skill requirements and that their currency remains valid.
- Monitor and evaluate the job performance to determine the adequacy of the training program.

## Practical Considerations

The operation process sustains system services by assigning trained personnel to operate the system, as well as by monitoring operator-system performance and monitoring the system performance. In order to sustain services, the operation process identifies and analyzes operational problems in relation to agreements, stakeholder requirements, and organizational constraints. When the system replaces an existing system, it may be necessary to manage the migration between systems such that persistent stakeholders do not experience a breakdown in services.

Results of a successful implementation of the operation process include

- an operation strategy is defined and refined along the way
- services that meet stakeholder requirements are delivered
- approved, corrective action requests are satisfactorily completed
- stakeholder satisfaction is maintained

Outputs of the operation process include

- operational strategy, including staffing and sustainment of enabling systems and materials
- system performance reports (statistics, usage data, and operational cost data)
- system trouble/anomaly reports with recommendations for appropriate action
- operational availability constraints to influence future design and specification of similar systems or reused system elements

Activities of the operation process include

- provide operator training to sustain a pool of operators
- track system performance and account for operational availability
- perform operational analysis
- manage operational support logistics
- document system status and actions taken
- report malfunctions and recommendations for improvement

## References

### Works Cited

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

### Primary References

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis,* 5th Edition. Englewood Cliffs, NJ, USA:Prentice Hall.

Institute of Engineers Singapore. 2009. *Systems Engineering Body of Knowledge*. Provisional version 2.0. Singapore: Institute of Engineers Singapore.

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

### Additional References

None.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# System Maintenance

***Lead Authors:*** *Scott Jackson, Brian Gallagher*, ***Contributing Author:*** *David Dorgan*

System Maintenance planning begins early in the acquisition process with development of a maintenance concept. Maintenance planning is conducted to evolve and establish requirements and tasks to be accomplished for achieving, restoring, and maintaining operational capability for the life of the system. For a system to be sustained throughout its system life cycle, the maintenance process has to be executed concurrently with the operations process (ISO/IEC/IEEE 15288 2015, Clause 6.4.9).

## Overview

The initial requirements for maintenance have to be defined during the stakeholder needs and requirement definition process (Clause 6.4.1) (ISO/IEC/IEEE 15288 2015) and continue to evolve during the development and operation of the system. Considerations include:

- Maximizing system availability to meet the operational requirements. This has to take into account the designed-in reliability and maintainability of the system and resources available.
- Preserving system operating potential through proper planning of system scheduled maintenance. This requires a reliability-centered maintenance strategy that incorporates preventive maintenance in order to preempt failures, thereby extending the mean time between corrective maintenance, as well as enhancing the availability of the system.
- Segmentation of maintenance activities for potential outsourcing of non-critical activities to approved maintenance subcontractors as to optimize scarce technical manpower resources and maintenance/repair turn-around times.
- Harnessing IT technology for maintenance management. This involves rigorous and systematic capturing and tracking of operating and maintenance activities to facilitate analysis and planning.

Maintenance management is concerned with the development and review of maintenance plans, as well as securing and coordinating resources, such as budget, service parts provisioning, and management of supporting tasks (e.g., contract administration, engineering support, and quality assurance). Maintenance planning relies on level of repair analysis (LORA) as a function of the system acquisition process. Initial planning addresses actions and support necessary to ensure a minimum life cycle cost (LCC).

## Process Approaches

The purpose of the maintenance process is to sustain the capability of a system to provide a service. This process monitors the system's capability to deliver services, records problems for analysis, takes corrective, adaptive, perfective, and preventive actions, and confirms restored capability. As a result of the successful implementation of the maintenance process

- a maintenance strategy is developed
- maintenance constraints are provided as inputs to requirements
- replacement system elements are made available
- services meeting stakeholder requirements are sustained
- the need for corrective design changes is reported
- failure and lifetime data is recorded

The project should implement the following activities and tasks in accordance with applicable organization policies and procedures with respect to the maintenance process:

- scheduled servicing, such as daily inspection/checks, servicing, and cleaning

- unscheduled servicing (carrying out fault detection and isolation to the faulty replaceable unit and replacement of the failed unit)
- re-configuration of the system for different roles or functions
- scheduled servicing (higher level scheduled servicing but below depot level)
- unscheduled servicing (carrying out more complicated fault isolation to the faulty replaceable unit and replacement of the failed unit)
- minor modifications
- minor damage repairs
- major scheduled servicing (e.g., overhaul and corrosion treatment)
- major repairs (beyond normal removal and replacement tasks)

The maintenance plan specifies the scheduled servicing tasks and intervals (preventive maintenance) and the unscheduled servicing tasks (adaptive or corrective maintenance). Tasks in the maintenance plan are allocated to the various maintenance agencies. A maintenance allocation chart is developed to tag the maintenance tasks to the appropriate maintenance agencies. These include: in-service or in-house work centers, approved contractors, affiliated maintenance or repair facilities, original equipment manufacturer (OEMs), etc. The maintenance plan also establishes the requirements for the support resources.

Related activities such as resource planning, budgeting, performance monitoring, upgrades, longer term supportability, and sustenance also need to be managed. These activities are being planned, managed, and executed over a longer time horizon and they concern the well being of the system over the entire life cycle.

Proper maintenance of the system (including maintenance-free system designs) relies very much on the availability of support resources, such as support and test equipment (STE), technical data and documentation, personnel, spares, and facilities. These have to be factored in during the acquisition agreement process.

## Training and Certification

Adequate training must be provided for the technical personnel maintaining the system. While initial training may have been provided during the deployment phase, additional personnel may need to be trained to cope with the increased number of systems being fielded, as well as to cater to staff turnover. Timely updates to training materials and trained personnel may be required as part of system upgrades and evolution. It is important to define the certification standards and contract for the training materials as part of the supply agreement.

## Practical Considerations

The organization responsible for maintaining the system should have clear thresholds established to determine whether a change requested by end users, changes to correct latent defects, or changes required to fulfill the evolving mission are within the scope of a maintenance change or require a more formal project to step through the entire systems engineering life-cycle. Evaluation criteria to make such a decision could include cost, schedule, risk, or criticality characteristics.

## References

### Works Cited

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC/IEEE 15288:2015.

### Primary References

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis,* 5th Edition. Upper Saddle River, NJ, USA: Prentice Hall.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense. February 19, 2010.

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Institute of Engineers Singapore. 2009. *Systems Engineering Body of Knowledge,* Provisional version 2.0. Singapore: Institute of Engineers Singapore.

IISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes.*Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers.ISO/IEC/IEEE 15288:2015.

### Additional References

None.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Logistics

***Lead Authors:*** *Scott Jackson, John Snoderly*, ***Contributing Authors:*** *Garry Roedler*

There are several definitions for logistics within systems engineering (SE) and the definition used will determine what activities are considered part of logistics. The SEBoK defines logistics as the science of planning and implementing the acquisition and use of the resources necessary to sustain the operation of a system.

## Overview

The ability to *sustain the operation of a system* is determined by the inherent supportability of the system (a function of design) and the processes used to sustain the functions and capabilities of the system in the context of the end user. Figure 1, below, shows a Defense Acquisition University (DAU) model of the SE aspects for consideration in logistics and logistics planning (DAU 2010).

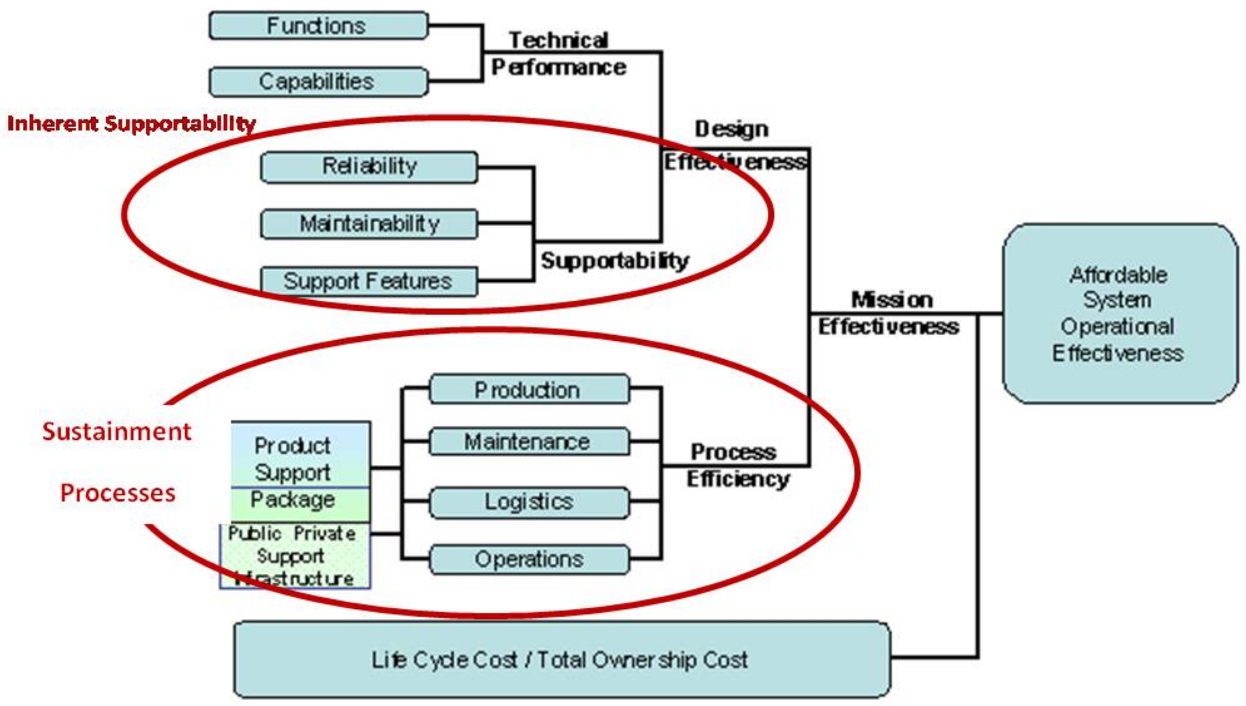


**Figure 1. Affordable System Operational Effectiveness (DAU Guidebook 2010).** Released by Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

## Sustainment Planning

The focus of sustainment planning is to influence the inherent supportability of the system and to plan the sustainment capabilities and processes that will be used to sustain system operations.

### Influence Inherent Supportability (Operational Suitability)

Sustainment influence requires an understanding of the concept of operations (ConOps), system missions, mission profiles, and system capabilities to understand the rationale behind functional and performance priorities. Understanding the rationale paves the way for decisions about necessary tradeoffs between system performance, availability, and life cycle cost (LCC), with impact on the cost effectiveness of system operation, maintenance, and logistics support. There is no single list of sustainment considerations or specific way of grouping them as they are highly inter-related. They range from: compatibility, interoperability, transportability, reliability, maintainability, manpower, human factors, safety, natural environment effects (including occupational health, habitability, see Environmental Engineering); diagnostics & prognostics (including real-time maintenance data collection), and corrosion protection & mitigation. The following are key design considerations:

- **Architecture Considerations** - The focus on openness, modularity, scalability, and upgradeability is critical to implementing an incremental acquisition strategy. In addition, the architecture attributes that expand system flexibility and affordability can pay dividends later when obsolescence and end-of-life issues are resolved through a concerted technology refreshment strategy. Trade-offs are often required relative to the extent each attribute is used.
- **Reliability Considerations**: - Reliability is critical because it contributes to a system's effectiveness as well as its suitability in terms of logistics burden and the cost to fix failures. For each system, there is a level of basic reliability that must be achieved for the system to be considered useful. Reliability is also one of the most critical elements in determining the logistics infrastructure and footprint. Consequently, system reliability should be a primary focus during design (along with system technical performance, functions, and capabilities). The primary objective is to achieve the necessary probability of operational success and minimize the risk of failure within defined availability, cost, schedule, weight, power, and volume constraints. While performing such analyses, trade-offs should be conducted and dependencies should be explored with system maintainability and integrated with the supportability analysis that addresses support event frequency (i.e. reliability), event duration, and event cost. Such a focus will play a significant role in minimizing the necessary logistics footprint, while maximizing system availability.
- **Maintainability Considerations** - The design emphasis on maintainability is to reduce the maintenance burden and supply chain by reducing the time, personnel, tools, test equipment, training, facilities and cost to maintain the system. Maintainability engineering includes the activities, methods, and practices used to design minimal system maintenance requirements (designing out unnecessary and inefficient processes) and associated costs for preventive and corrective maintenance as well as servicing or calibration activities. Maintainability should be a designed-in capability and not an add-on option because good maintenance procedures cannot overcome poor system and equipment maintainability design. The primary objective is to reduce the time it takes for a properly trained maintainer to detect and isolate the failure (coverage and efficiency) and affect repair. Intrinsic factors contributing to maintainability are
  - **Modularity** - Packaging of components such that they can be repaired via remove and replace action vs. on-board repair. Care should be taken not to *over modularize* and trade-offs to evaluate replacement, transportation, and repair costs should be accomplished to determine the most cost effective approach.
  - **Interoperability** - The compatibility of components with standard interface protocols to facilitate rapid repair and enhancement/upgrade through black box technology using common interfaces. Physical interfaces should be designed so that mating between components can only happen correctly.

- **Physical accessibility** - The designed-in structural assurance that components which require more frequent monitoring, checkout, and maintenance can be easily accessed. This is especially important in low observable platforms. Maintenance points should be directly visible and accessible to maintainers, including access for corrosion inspection and mitigation.
- Designs that require *minimum preventative maintenance* including corrosion prevention and mitigation. Emphasis should be on balancing the maintenance requirement over the life cycle with minimal user workload.
- **Embedded training and testing** when it is determined to be the optimal solution from a total ownership cost (TOC) and materiel availability perspective.
- **Human Systems Integration (HSI)** to optimize total system performance and minimize life-cycle costs by designing systems and incorporating technologies that (a) require minimal manpower, (b) provide effective training, (c) can be operated and maintained by users, (d) are suitable (habitable and safe with minimal environmental and occupational health hazards), and (e) are survivable (for both the user and the equipment).

- **Support Considerations** - Support features cannot be easily *added-on* after the design is established. Consequently, supportability should be a high priority early in the program's planning and integral to the system design and development process. Support features cut across reliability, maintainability, and the supply chain to facilitate detection, isolation, and timely repair/replacement of system anomalies. These include features for servicing and other activities necessary for operation and support including resources that contribute to the overall support of the system. Typical supportability features include diagnostics, prognostics (see CBM+ Guidebook), calibration requirements, many HSI issues (e.g. training, safety, HFE, occupational health, etc.), skill levels, documentation, maintenance data collection, compatibility, interoperability, transportability, handling (e.g., lift/hard/tie down points, etc.), packing requirements, facility requirements, accessibility, and other factors that contribute to an optimum environment for sustaining an operational system.

## Planning Sustainment Processes

Process efficiency reflects how well the system can be produced, operated, serviced (including fueling) and maintained. It reflects the degree to which the logistics processes (including the supply chain), infrastructure, and footprint have been balanced to provide an agile, deployable, and operationally effective system.

Achieving process efficiency requires early and continuing emphasis on the various logistics support processes along with the design considerations. The continued emphasis is important because processes present opportunities for improving operational effectiveness even after the *design-in* window has passed via lean-six sigma, supply chain optimization, or other continuous process improvement (CPI) techniques.

## Sustainment Analysis (Product Support Package)

The product support package documents the output of supportability analysis and includes details related to the following twelve elements (links below are to excerpts from (NATO RTO 2001):

- Product/information technology (IT) system/medical system support management (integrated life cycle sustainment planning)
  - product/IT system/medical system support strategies
  - life cycle sustainment planning
  - requirements management
  - total ownership costs (TOC)/life cycle costs (LCC) planning & management
  - Integration and management of product support activities
  - configuration management
  - production & distribution
  - energy, environmental, safety and health (EESH) management
  - policies & guidance

  - risk management
- Design Interface [1]
  - reliability
  - maintainability
  - supportability
  - affordability
  - configuration management
  - safety requirements
  - environmental and hazardous materials (HAZMAT) requirements
  - human systems integration (HSI)
  - calibration
  - anti-tamper
  - habitability
  - disposal
  - legal requirements
- Sustainment Engineering
  - failure reporting, analysis, and corrective action system (FRACAS)
  - value engineering
  - diminishing manufacturing sources and material shortages (DMSMS)
- Supply Support (materiel planning) [2]
- Maintenance Planning [3]
  - reliability centered maintenance (RCM)
  - maintenance concepts
  - levels of maintenance (level of repair analysis)
  - condition-based maintenance
  - prognostics & health management
- Support Equipment [4]
- Technical Data [5]
- Manpower & Personnel [6]
- Training & Training Support [7]
- Facilities & Infrastructure [8]
- Packaging, Handling, Storage, & Transportation [9]
- Computer Resources [10]

## Sustainment Implementation

Once the system becomes operational, the results of sustainment planning efforts need to be implemented. SE supports the execution of the twelve integrated product support elements of a sustainment program that strives to ensure the system meets operational performance requirements in the most cost-effective manner over its total remaining life cycle, as illustrated in Figure 2.

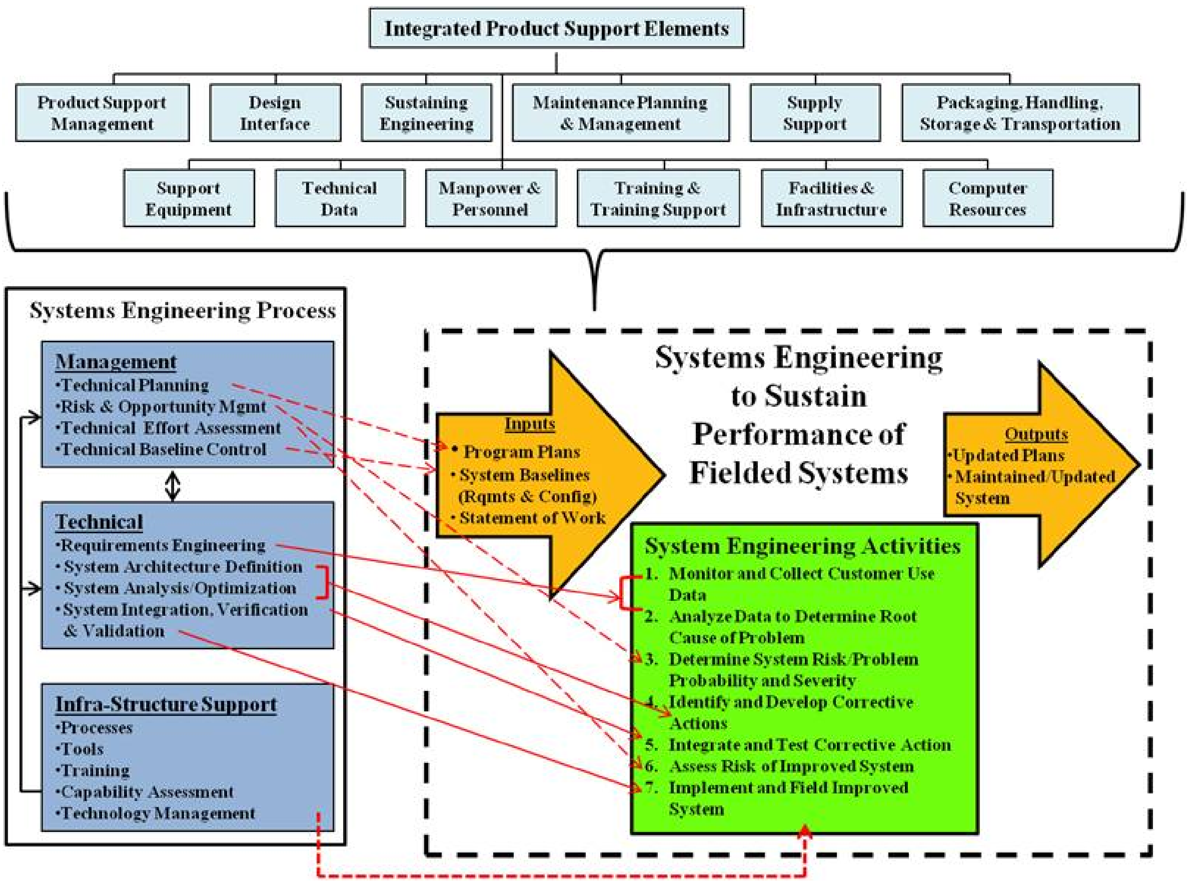

Integrated Product Support Elements
Product Support Management
Design Interface
Sustaining Engineering
Maintenance Planning & Management
Supply Support
Packaging, Handling, Storage & Transportation
Support Equipment
Technical Data
Manpower & Personnel
Training & Training Support
Facilities & Infrastructure
Computer Resources
Systems Engineering Process
Management
•Technical Planning
•Risk & Opportunity Mgmt
•Technical Effort Assessment
•Technical Baseline Control
Technical
•Requirements Engineering
•System Architecture Definition
•System Analysis/Optimization
•System Integration, Verification & Validation
Infra-Structure Support
•Processes
•Tools
•Training
•Capability Assessment
•Technology Management
Systems Engineering to Sustain Performance of Fielded Systems
Inputs
• Program Plans
• System Baselines (Rqmts & Config)
• Statement of Work
Outputs
•Updated Plans
•Maintained/Updated System
System Engineering Activities
1. Monitor and Collect Customer Use Data
2. Analyze Data to Determine Root Cause of Problem
3. Determine System Risk/Problem Probability and Severity
4. Identify and Develop Corrective Actions
5. Integrate and Test Corrective Action
6. Assess Risk of Improved System
7. Implement and Field Improved System

**Figure 2. Sustainment Implementation Illustration (DAU Guidebook 2012).** Released by Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

Once a system is put into use, SE is often required to correct problems that degrade continued use, and/or to add new capabilities to improve product performance in the current or a new environment. In the context of integrated product support, these SE activities correspond to the integrated product support (IPS) element *Sustaining Engineering*. Changes made to fielded systems to correct problems or increase performance should include any necessary adjustments to the IPS elements, and should consider the interrelationships and integration of the elements to maintain the effectiveness of system's support strategy.

The degree of change required to the product support elements varies with the severity of the problem. Minor problems may require a simple adjustment to a maintenance procedure, a change of supplier, a training course modification or a change to a technical manual. In contrast, problems that require system or component redesign may require engineering change proposals and approvals, IPS element trade studies, business case analysis, and updates to the product support strategy. The focus is to correct problems that degrade continued use, regardless of the degree of severity.

Evolutionary systems provide a strategy for acquisition of mature technology; the system delivers capabilities incrementally, planning for future capability enhancements. For these systems a system of systems (SoS) perspective is required to synchronize the primary and sustainment systems.

For more information refer to: *An Enterprise Framework for Operationally Effective System of Systems Design* (Bobinis and Herald 2012.).

## References

### Works Cited

Bobinis, J. and T. Herald. 2012. “An Enterprise Framework for Operationally Effective System of Systems Design.” *Journal of Enterprise Architecture.* 8(2), May 2012. Available at: https://www.mendling.com/publications/JEA12-2.pdf.

DAU. 2010. Defense Acquisition Guidebook (DAG). Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

NATO RTO. 2001. *Logistics Test and Evaluation in Flight Test*. Flight Test Techniques Series – Volume 20. Quebec, Canada: North Atlantic Treaty Organization (NATO) Research and Technology Organization (RTO). RTO-AG-300 Vol. 20, AC/323(SCI-010)TP/38. Table of contents available at: http://ftp.rta.nato.int/public//PubFullText/RTO/AG/RTO-AG-300-V20///AG-300-V20-$$TOC.pdf

### Primary References

Blanchard, B.S. 1998. *Logistics Engineering and Management.* Upper Saddle River, NJ, USA: Prentice Hall.

Blanchard, B. and W. Fabrycky. 2011. *Systems Engineering and Analysis,* 5th Ed. Englewood Cliffs, NJ, USA: Prentice-Hall.

Bobinis, J. and T. Herald. 2012. “An Enterprise Framework for Operationally Effective System of Systems Design.” *Journal of Enterprise Architecture.* 8(2), May 2012. Available at: https://www.mendling.com/publications/JEA12-2.pdf.

Daganzo, C. 2005. *Logistics Systems Analysis,* 4th Edition. New York, NY, USA: Springer.

Fabrycky, W.J. and B.S. Blanchard. 1991. *Life-Cycle Cost and Economic Analysis*. Upper Saddle River, NJ, USA: Prentice-Hall.

Ghiani, G., G. Laporte, and R. Musmanno. 2004. *Introduction to Logistics Systems Planning and Control.* Hoboken, NJ, USA: Wiley-Interscience.

Jones, J.V. 1995. *Integrated Logistics Support Handbook.* New York, NY, USA: McGraw Hill.

### Additional References

Barros, L.L. 1998. "The Optimization of Repair Decision Using Life-Cycle Cost Parameters." *IMA Journal of Management Mathematics.* 9(4): 403.

Berkowitz, D., J.N. Gupta, J.T. Simpson, and J.B. McWilliams. 2005. *Defining and Implementing Performance-Based Logistics in Government*. Washington, DC, USA: Defense Technical Information Center. Accessed 6 Sept 2011. Available at: http://handle.dtic.mil/100.2/ADP018510.

Gajpal, P.P., L.S. Ganesh, and C. Rajendran. 1994. "Criticality Analysis of Spare Parts Using the Analytic Hierarchy Process." *International Journal of Production Economics.* 35(1-3): 293-297.

MITRE. 2011. "Integrated Logistics Support." *Systems Engineering Guide.* Accessed 11 March 2012 at [[11]].

Murthy, D.N.P. and W.R. Blischke. 2000. "Strategic Warranty Management: A Life-Cycle Approach." *Engineering Management.* 47(1): 40-54.

Northrop Grumman Corporation. 2000. *Logistics Systems Engineering*. Accessed 6 Sept 2011. Available at: http://www. northropgrumman. com/ Capabilities/ NavigationSystemsLogisticsSystemsEngineering/ Documents/ nsd_logistics.pdf.

Solomon, R., P.A. Sandborn, and M.G. Pecht. 2000. "Electronic Part Life Cycle Concepts and Obsolescence Forecasting." *IEEE Transactions on Components and Packaging Technologies.* 23(4): 707-717.

Spengler, T. and M. Schroter. 2003. "Strategic Management of Spare Parts in Closed-Loop Supply Chains: A System Dynamics Approach." *Interfaces.* p. 7-17.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

## References

[1] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-12.pdf

[2] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-06.pdf

[3] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-03.pdf

[4] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-05.pdf

[5] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-07.pdf

[6] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-04.pdf

[7] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-08.pdf

[8] http://www.decisionlens.com/docs/WP_Strategic_Facilities_and_Infrastructure_Planning.pdf

[9] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-11.pdf

[10] http://ftp.rta.nato.int/public/PubFullText/RTO/AG/RTO-AG-300-V20/AG-300-V20-09.pdf

[11] http://www.mitre.org/work/systems_engineering/guide/acquisition_systems_engineering/integrated_logistics_support/

# Knowledge Area: Systems Engineering Management

# Systems Engineering Management

***Lead Authors:*** *Ray Madachy, Garry Roedler*

This knowledge area is about managing the resources and assets allocated to perform systems engineering, often in the context of a project or a service, but sometimes in the context of a less well-defined activity. Systems engineering management is distinguished from general project management by its focus on the technical or engineering aspects of a project. SEM also encompasses exploratory research and development (R&D) activities at the enterprise level in commercial or government operations.

## Topics

Each part of the SEBoK is composed of knowledge areas (KAs). Each KA groups topics together around a theme related to the overall subject of the part. This KA contains the following topics:

- Planning
- Assessment and Control
- Risk Management
- Measurement
- Decision Management
- Configuration Management
- Information Management
- Quality Management

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

## Discussion

Implementing systems engineering (SE) requires the coordination of technical and managerial endeavors. Success with the technical is not possible in the absence of the managerial. Management provides the planning, organizational structure, collaborative environment, and program controls to ensure that stakeholder needs are met.

The Venn diagram below provides some context for thinking about SEM. It shows that some functions are managed within the SE function, while others are managed in collaboration with the management of systems implementation and with overall project and systems management.

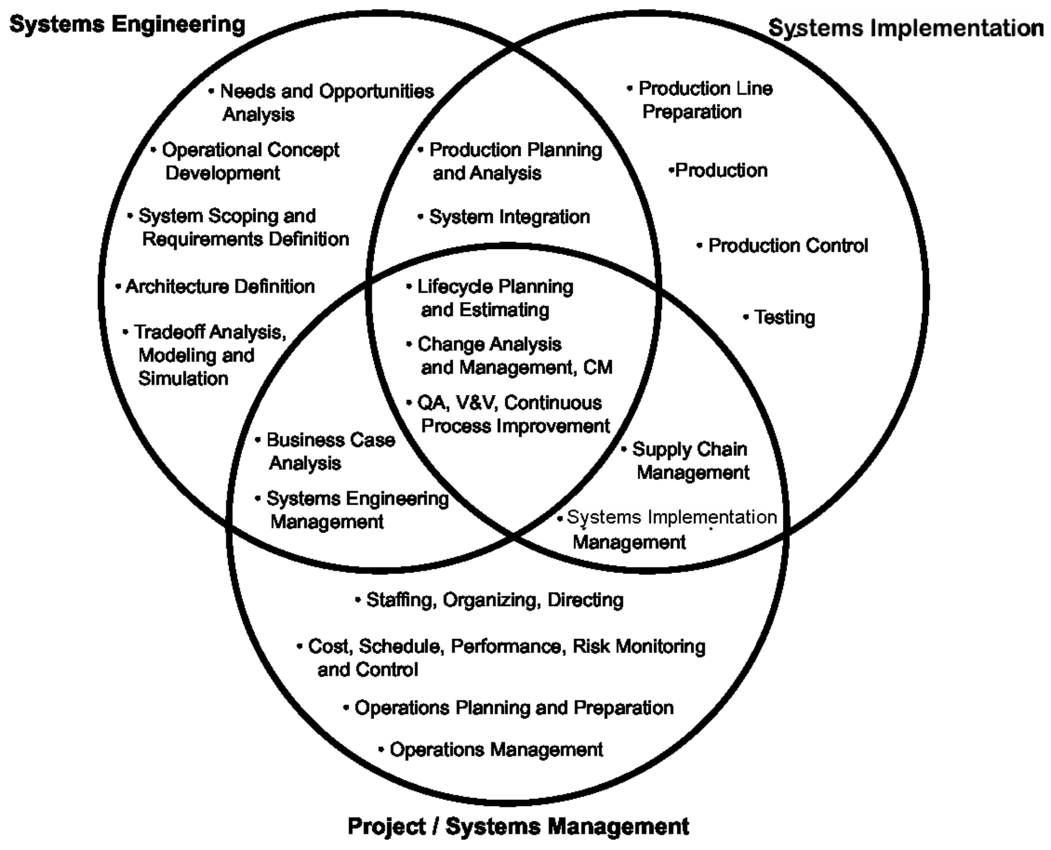

Systems Engineering
Systems Implementation
• Needs and Opportunities Analysis
• Operational Concept Development
• System Scoping and Requirements Definition
• Architecture Definition
• Tradeoff Analysis, Modeling and Simulation
• Production Planning and Analysis
• System Integration
• Production Line Preparation
•Production
• Production Control
• Testing
• Lifecycle Planning and Estimating
• Change Analysis and Management, CM
• QA, V&V, Continuous Process Improvement
• Business Case Analysis
• Systems Engineering Management
• Supply Chain Management
• Systems Implementation Management
• Staffing, Organizing, Directing
• Cost, Schedule, Performance, Risk Monitoring and Control
• Operations Planning and Preparation
• Operations Management
Project / Systems Management

**Figure 1. Systems Engineering Management Boundaries.** (SEBoK Original)

There is no one-size-fits-all way to define the details of where SEM functions are performed. An in-company SE organization does not run its own accounting system, but relies on the corporate management organization for this aspect of SEM. A company performing only SE *does* include the accounting functions as part of SEM. In all cases, the managers of the SE function must be actively involved in the management of all the activities within the SE system boundary, including working out what collaborative arrangements best fit their situation. They must also remain aware of management events in their environment outside the system boundary that may affect their ability to perform. Part 6 of the SEBoK includes relevant knowledge areas for collaborative management, including Systems Engineering and Software Engineering, Systems Engineering and Project Management, Systems Engineering and Industrial Engineering, Systems Engineering and Procurement/Acquisition, and Systems Engineering and Specialty Engineering.

## References

### Works Cited

None.

### Primary References

Blanchard, B.S. 2004. *Systems Engineering Management,* 3rd ed. New York, NY, USA: John Wiley & Sons Inc.

Sage, A.P. and W. Rouse. 2009. *Handbook of Systems Engineering and Management,* 2nd Ed. Hoboken, NJ, USA: John Wiley and Sons.

### Additional References

None.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Planning

***Lead Authors:*** *Ray Madachy, Garry Roedler, Brian Wells*

Planning is an important aspect of systems engineering management (SEM). Systems engineering (SE) planning is performed concurrently and collaboratively with project planning. It involves developing and integrating technical plans to achieve the technical project objectives within the resource constraints and risk thresholds. The planning involves the success-critical stakeholders to ensure that necessary tasks are defined with the right timing in the life cycle in order to manage acceptable risks levels, meet schedules, and avoid costly omissions.

## SE Planning Process Overview

SE planning provides the following elements:

- Definition of the project from a technical perspective.
- Definition or tailoring of engineering processes, practices, methods, and supporting enabling environments to be used to develop products or services, as well as plans for transition and implementation of the products or services, as required by agreements.
- Definition of the technical organizational, personnel, and team functions and responsibilities, as well as all disciplines required during the project life cycle.
- Definition of the appropriate life cycle model or approach for the products or services.
- Definition and timing of technical reviews, product or service assessments, and control mechanisms across the life cycle, including the success criteria such as cost, schedule, and technical performance at identified project milestones.
- Estimation of technical cost and schedule based on the effort needed to meet the requirements; this estimation becomes input to project cost and schedule planning.
- Determination of critical technologies, as well as the associated risks and actions needed to manage and transition these technologies.
- Identification of linkages to other project management efforts.
- Documentation of and commitment to the technical planning.

## Scope

SE planning begins with analyzing the scope of technical work to be performed and gaining an understanding the constraints, risks, and objectives that define and bound the solution space for the product or service. The planning includes estimating the size of the work products, establishing a schedule (or integrating the technical tasks into the project schedule), identification of risks, and negotiating commitments. Iteration of these planning tasks may be necessary to establish a balanced plan with respect to cost, schedule, technical performance, and quality. The planning continues to evolve with each successive life cycle phase of the project (NASA 2007, 1-360; SEI 1995, 12).

SE planning addresses all programmatic and technical elements of the project to ensure a comprehensive and integrated plan for all of the project's technical aspects and should account for the full scope of technical activities, including system development and definition, risk management, quality management, configuration management, measurement, information management, production, verification and testing, integration, validation, and deployment. SE planning integrates all SE functions to ensure that plans, requirements, operational concepts, and architectures are consistent and feasible.

The scope of planning can vary from planning a specific task to developing a major technical plan. The integrated planning effort will determine what level of planning and accompanying documentation is appropriate for the project.

## Integration

The integration of each plan with other higher-level, peer, or subordinate plans is an essential part of SE planning. For the technical effort, the systems engineering management plan (SEMP), also frequently reffered to as the systems engineering plan (SEP), is the highest level technical plan. It is subordinate to the project plan and often has a number of subordinate technical plans providing detail on specific technical focus areas (INCOSE 2011, sec. 5.1.2.2; NASA 2007, appendix J).

In U.S. defense work, the terms SEP and SEMP are not interchangeable. The SEP is a high-level plan that is made before the system acquisition and development begins. It is written by the government customer. The SEMP is the specific development plan written by the developer (or contractor). In this context, intent, and content of these documents are quite different. For example, a SEP will have an acquisition plan that would not be included in a SEMP. Figure 1 below shows the SEMP and integrated plans.

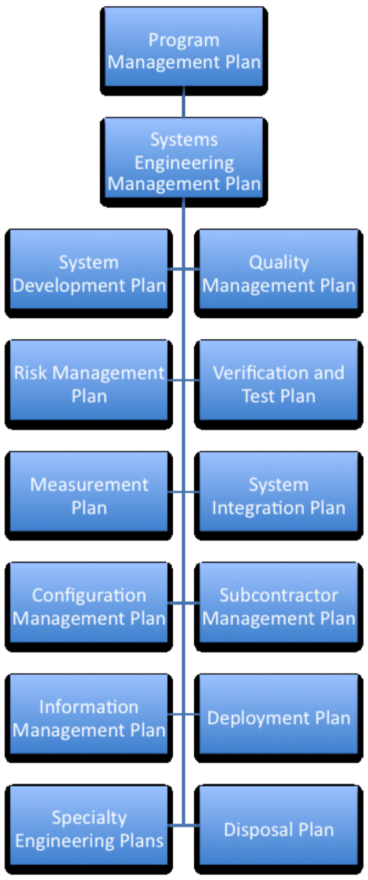

Program Management Plan
Systems Engineering Management Plan
System Development Plan
Quality Management Plan
Risk Management Plan
Verification and Test Plan
Measurement Plan
System Integration Plan
Configuration Management Plan
Subcontractor Management Plan
Information Management Plan
Deployment Plan
Specialty Engineering Plans
Disposal Plan

**Figure 1. SEMP and Integrated Plans.** (SEBoK Original)

Task planning identifies the specific work products, deliverables, and success criteria for systems engineering efforts in support of integrated planning and project objectives. The success criteria are defined in terms of cost, schedule, and technical performance at identified project milestones. Detailed task planning identifies specific resource requirements (e.g., skills, equipment, facilities, and funding) as a function of time and project milestones.

SE planning is accomplished by both the acquirer and supplier and the activities for SE planning are performed in the context of the respective enterprise. The activities establish and identify relevant policies and procedures for managing and executing the project management and technical effort, identifying the management and technical tasks, their interdependencies, risks, and opportunities, and providing estimates of needed resources/budgets. Plans are updated and refined throughout the development process based on status updates and evolving project requirements (SEI 2007).

## Linkages to Other Systems Engineering Management Topics

The project planning process is closely coupled with the measurement, assessment and control, decision management, and risk management processes.

The measurement process provides inputs for estimation models. Estimates and other products from planning are used in decision management. SE assessment and control processes use planning results for setting milestones and assessing progress. Risk management uses the planning cost models, schedule estimates, and uncertainty distributions to support quantitative risk analysis (as desired).

Additionally, planning needs to use the outputs from assessment and control as well as risk management to ensure corrective actions have been accounted for in planning future activities. The planning may need to be updated based on results from technical reviews (from assessment and control) addressing issues pertaining to: measurement, problems that were identified during the performance of risk management activities, or decisions made as a result of the decision management activities (INCOSE 2010, sec. 6.1).

## Practical Considerations

### Pitfalls

Some of the key pitfalls encountered in planning and performing SE planning are listed in Table 1.

**Table 1. Major Pitfalls with Planning. (SEBoK Original)**

| Name | Description |
|---|---|
| Incomplete and Rushed Planning | Inadequate SE planning causes significant adverse impacts on all other engineering activities. Although one may be tempted to save time by rushing the planning, inadequate planning can create additional costs and interfere with the schedule due to planning omissions, lack of detail, lack of integration of efforts, infeasible cost and schedules, etc. |
| Inexperienced Staff | Lack of highly experienced engineering staff members, especially in similar projects, will likely result in inadequate planning. Less experienced engineers are often assigned significant roles in the SE planning; however, they may not have the appropriate judgment to lay out realistic and achievable plans. It is essential to assign the SE planning tasks to those with a good amount of relevant experience. |

### Good Practices

Some good practices gathered from the references are in Table 2.

**Table 2. Proven Practices with Planning. (SEBoK Original)**

| Name | Description |
|---|---|
| Use Multiple Disciplines | Get technical resources from all disciplines involved in the planning process. |
| Early Conflict Resolution | Resolve schedule and resource conflicts early. |
| Task Independence | Tasks should be as independent as possible. |
| Define Interdependencies | Define task interdependencies, using dependency networks or other approaches. |
| Risk Management | Integrate risk management with the SE planning to identify areas that require special attention and/or trades. |
| Management Reserve | The amount of management reserve should be based on the risk associated with the plan. |
| Use Historical Data | Use historical data for estimates and adjust for differences in the project. |
| Consider Lead Times | Identify lead times and ensure that you account for them in the planning (e.g., the development of analytical tools). |
| Update Plans | Prepare to update plans as additional information becomes available or changes are needed. |
| Use IPDTs | An integrated product development team (IPDT) (or integrated product team (IPT)) is often useful to ensure adequate communication across the necessary disciplines, timely integration of all design considerations, as well as integration, testing, and consideration of the full range of risks that need to be addressed. Although there are some issues that need to be managed with them, IPDTs tend to break down the communication and knowledge stovepipes that often exist. |

Additional good practices can be found in the *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS)*, *NASA Systems Engineering Handbook*, the *INCOSE Systems Engineering Handbook*, and *Systems and Software Engineering - Life Cycle Processes - Project Management* (Caltrans and USDOT 2005, 278; NASA December 2007, 1-360, sec. 6.1; INCOSE 2011, sec. 5.1; ISO/IEC/IEEE 2009, Clause 6.1).

## References

### Works Cited

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Reserach & Innovation/U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

DAU. 2010. *Defense Acquisition Guidebook (DAG).* Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense, February 19.

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities.* Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - Life Cycle Processes - Project Management*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16326:2009(E).

NASA. 2007. *NASA Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

SEI. 1995. *A systems engineering capability maturity model.* Version 1.1. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-95-MM-003.

### Primary References

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Reserach & Innovation/U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

DAU. 2010. *Defense Acquisition Guidebook (DAG).* Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense.

INCOSE. 2012. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities.* Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - Life Cycle Processes - Project Management*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16326:2009(E).

NASA. 2007. *NASA Systems Engineering Handbook.* Washington, D.C., USA: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

SEI. 1995. *A Systems Engineering Capability Maturity Model,* version 1.1. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-95-MM-003.

SEI. 2007. *Capability Maturity Model Integrated (CMMI) for Development,* version 1.2, measurement and analysis process area. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

## Additional References

Boehm, B., C. Abts, A.W. Brown, S. Chulani, B.K. Clark, E. Horowitz, R. Madachy, D.J. Reifer, B. Steece. 2000. *Software Cost Estimation with COCOMO II*. Englewood Cliffs, NJ, USA: Prentice Hall

DeMarco, T. and T. Lister. 2003. *Waltzing with Bears; Managing Risks on Software Projects.* New York, NY, USA: Dorset House.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - Life Cycle Processes - Project Management.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16326:2009(E).

Valerdi, R. 2008. *The Constructive Systems Engineering Cost Model (COSYSMO): Quantifying the Costs of Systems Engineering Effort in Complex Systems*. Saarbrücken,Germany: VDM Verlag Dr. Muller

---

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Assessment and Control

***Lead Authors:*** *Ray Madachy, Andy Pickard, Garry Roedler*, ***Contributing Author:*** *Richard Turner*

The purpose of systems engineering assessment and control (SEAC) is to provide adequate visibility into the project's actual technical progress and risks with respect to the technical plans (i.e., systems engineering management plan (SEMP) or systems engineering plan (SEP) and subordinate plans). The visibility allows the project team to take timely preventive action when disruptive trends are recognized or corrective action when performance deviates beyond established thresholds or expected values. SEAC includes preparing for and conducting reviews and audits to monitor performance. The results of the reviews and measurement analyses are used to identify and record findings/discrepancies and may lead to causal analysis and corrective/preventive action plans. Action plans are implemented, tracked, and monitored to closure. (NASA 2007, Section 6.7; SEG-ITS, 2009, Section 3.9.3, 3.9.10; INCOSE, 2010, Clause 6.2; SEI, 2007)

## Systems Engineering Assessment and Control Process Overview

The SEAC process involves determining and initiating the appropriate handling strategies and actions for findings and/or discrepancies that are uncovered in the enterprise, infrastructure, or life cycle activities associated with the project. Analysis of the causes of the findings/discrepancies aids in the determination of appropriate handling strategies. Implementation of approved preventive, corrective, or improvement actions ensures satisfactory completion of the project within planned technical, schedule, and cost objectives. Potential action plans for findings and/or discrepancies are reviewed in the context of the overall set of actions and priorities in order to optimize the benefits to the project and/or organization. Interrelated items are analyzed together to obtain a consistent and cost-effective resolution.

The SEAC process includes the following steps:

- monitor and review technical performance and resource use against plans
- monitor technical risk, escalate significant risks to the project risk register and seek project funding to execute risk mitigation plans
- hold technical reviews and report outcomes at the project reviews
- analyze issues and determine appropriate actions
- manage actions to closure

- hold a post-delivery assessment (also known as a post-project review) to capture knowledge associated with the project (this may be a separate technical assessment or it may be conducted as part of the project assessment and control process).

The following activities are normally conducted as part of a project assessment and control process:

- authorization, release and closure of work
- monitor project performance and resource usage against plan
- monitor project risk and authorize expenditure of project funds to execute risk mitigation plans
- hold project reviews
- analyze issues and determine appropriate actions
- manage actions to closure
- hold a post-delivery assessment (also known as a post-project review) to capture knowledge associated with the project

Examples of major technical reviews used in SEAC are shown in Table 1 from DAU (2010).

**Table 1. Major Technical Review Examples (DAU 2012). Released by Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).**

| Name | Description |
|---|---|
| Alternative Systems Review | A multi-disciplined review to ensure the resulting set of requirements agrees with the customers' needs and expectations. |
| Critical Design Review (CDR) | A multi-disciplined review establishing the initial product baseline to ensure that the system under review has a reasonable expectation of satisfying the requirements of the capability development document within the currently allocated budget and schedule. |
| Functional Configuration Audit | Formal examination of the as-tested characteristics of a configuration item (hardware and software) with the objective of verifying that actual performance complies with design and interface requirements in the functional baseline. |
| In-Service Review | A multi-disciplined product and process assessment that is performed to ensure that the system under review is operationally employed with well-understood and managed risk. |
| Initial Technical Review | A multi-disciplined review that supports a program's initial program objective memorandum submission. |
| Integrated Baseline Review | A joint assessment conducted by the government program manager and the contractor to establish the performance measurement baseline. |
| Operational Test Readiness Review | A multi-disciplined product and process assessment to ensure that the system can proceed into initial operational test and evaluation with a high probability of success, and also that the system is effective and suitable for service introduction. |
| Production Readiness Review (PRR) | The examination of a program to determine if the design is ready for production and if the prime contractor and major subcontractors have accomplished adequate production planning without incurring unacceptable risks that will breach thresholds of schedule, performance, cost, or other established criteria. |
| Physical Configuration Audit | An examination of the actual configuration of an item being produced around the time of the full-rate production decision. |
| Preliminary Design Review (PDR) | A technical assessment establishing the physically allocated baseline to ensure that the system under review has a reasonable expectation of being judged operationally effective and suitable. |
| System Functional Review (SFR) | A multi-disciplined review to ensure that the system's functional baseline is established and has a reasonable expectation of satisfying the requirements of the initial capabilities document or draft capability development document within the currently allocated budget and schedule. |
| System Requirements Review (SRR) | A multi-disciplined review to ensure that the system under review can proceed into initial systems development and that all system requirements and performance requirements derived from the initial capabilities document or draft capability development document are defined and testable, as well as being consistent with cost, schedule, risk, technology readiness, and other system constraints. |

| | |
|---|---|
| System Verification Review (SVR) | A multi-disciplined product and process assessment to ensure the system under review can proceed into low-rate initial production and full-rate production within cost (program budget), schedule (program schedule), risk, and other system constraints. |
| Technology Readiness Assessment | A systematic, metrics-based process that assesses the maturity of critical technology elements, such as sustainment drivers. |
| Test Readiness Review (TRR) | A multi-disciplined review designed to ensure that the subsystem or system under review is ready to proceed into formal testing. |

## Linkages to Other Systems Engineering Management Topics

The SE assessment and control process is closely coupled with the measurement, planning, decision management, and risk management processes. The measurement process provides indicators for comparing actuals to plans. Planning provides estimates and milestones that constitute plans for monitoring as well as the project plan, which uses measurements to monitor progress. Decision management uses the results of project monitoring as decision criteria for making control decisions.

## Practical Considerations

Key pitfalls and good practices related to SEAC are described in the next two sections.

### Pitfalls

Some of the key pitfalls encountered in planning and performing SE assessment and control are shown in Table 2.

**Table 2. Major Pitfalls with Assessment and Control. (SEBoK Original)**

| Name | Description |
|---|---|
| No Measurement | Since the assessment and control activities are highly dependent on insightful measurement information, it is usually ineffective to proceed independently from the measurement efforts - what you get is what you measure. |
| "Something in Time" Culture | Some things are easier to measure than others - for instance, delivery to cost and schedule. Don't focus on these and neglect harder things to measure like quality of the system. Avoid a "something in time" culture where meeting the schedule takes priority over everything else, but what is delivered is not fit for purpose, resulting in the need to rework the project. |
| No Teeth | Make sure that the technical review gates have "teeth". Sometimes the project manager is given authority (or can appeal to someone with authority) to over-ride a gate decision and allow work to proceed, even when the gate has exposed significant issues with the technical quality of the system or associated work products. This is a major risk if the organization is strongly schedule-driven; it can't afford the time to do it right, but somehow it finds the time to do it again (rework). |
| Too Early Baselining | Don't baseline requirements or designs too early. Often there is strong pressure to baseline system requirements and designs before they are fully understood or agreed, in order to start subsystem or component development. This just guarantees high levels of rework. |

### Good Practices

Some good practices gathered from the references are shown in Table 3.

**Table 3. Proven Practices with Assessment and Control. (SEBoK Original)**

| Name | Description |
|---|---|
| Independence | Provide independent (from customer) assessment and recommendations on resources, schedule, technical status, and risk based on experience and trend analysis. |
| Peer Reviews | Use peer reviews to ensure the quality of a product's work before they are submitted for gate review. |
| Accept Uncertainty | Communicate uncertainties in requirements or designs and accept that uncertainty is a normal part of developing a system. |
| Risk Mitigation Plans | Do not penalize a project at gate review if they admit uncertainty in requirements - ask for their risk mitigation plan to manage the uncertainty. |
| Just In-Time Baselining | Baseline requirements and designs only when you need to - when other work is committed based on the stability of the requirement or design. If work must start and the requirement or design is still uncertain, consider how you can build robustness into the system to handle the uncertainty with minimum rework. |
| Communication | Document and communicate status findings and recommendations to stakeholders. |
| Full Visibility | Ensure that action items and action-item status, as well as other key status items, are visible to all project participants. |
| Leverage Previous Root Cause Analysis | When performing root cause analysis, take into account the root cause and resolution data documented in previous related findings/discrepancies. |
| Concurrent Management | Plan and perform assessment and control concurrently with the activities for Measurement and Risk Management. |
| Lessons Learned and Post-Mortems | Hold post-delivery assessments or post-project reviews to capture knowledge associated with the project – e.g., to augment and improve estimation models, lessons learned databases, gate review checklists, etc. |

Additional good practices can be found in INCOSE (2010, Clause 6.2), SEG-ITS (2009, Sections 3.9.3 and 3.9.10), INCOSE (2010, Section 5.2.1.5), and NASA (2007, Section 6.7).

## References

### Works Cited

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research & Innovation/U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

NASA. 2007. *Systems Engineering Handbook*. Washington, DC, USA: National Aeronautics and Space Administration (NASA), December 2007. NASA/SP-2007-6105.

SEI. 2007. "Measurement and Analysis Process Area," in *Capability Maturity Model Integrated (CMMI) for Development,* version 1.2. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

## Primary References

Caltrans and USDOT. 2005. *[[Systems Engineering Guidebook for Intelligent Transportation Systems (ITS)],]* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research & Innovation/U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

NASA. 2007. *Systems Engineering Handbook*. Washington, DC, USA: National Aeronautics and Space Administration (NASA), December 2007. NASA/SP-2007-6105.

SEI. 2007. "Measurement and Analysis Process Area," in *Capability Maturity Model Integrated (CMMI) for Development,* version 1.2. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

## Additional References

ISO/IEC/IEEE. 2009. *ISO/IEC/IEEE 16326|Systems and Software Engineering - Life Cycle Processes - Project Management.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16326:2009(E).

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Risk Management

***Lead Authors:*** *Ed Conrow, Ray Madachy, Garry Roedler,* ***Contributing Author:*** *Richard Turner*

The purpose of risk management is to reduce potential risks to an acceptable level before they occur, throughout the life of the product or project. Risk management is a continuous, forward-looking process that is applied to anticipate and avert risks that may adversely impact the project, and can be considered both a project management and a systems engineering process. A balance must be achieved on each project in terms of overall risk management ownership, implementation, and day-to-day responsibility between these two top-level processes.

For the SEBoK, risk management falls under the umbrella of Systems Engineering Management, though the wider body of risk literature is explored below.

## Risk Management Process Overview

Risk is a measure of the potential inability to achieve overall program objectives within defined cost, schedule, and technical constraints. It has the following two components (DAU 2003a):

1. the probability (or likelihood) of failing to achieve a particular outcome
2. the consequences (or impact) of failing to achieve that outcome

In the domain of catastrophic risk analysis, risk has three components: (1) threat, (2) vulnerability, and (3) consequence (Willis et al. 2005).

Risk management involves defining a risk management strategy, identifying and analyzing risks, handling selected risks, and monitoring the progress in reducing risks to an acceptable level (SEI 2010; DoD 2015; DAU 2003a; DAU 2003b; PMI 2013) (Opportunity and opportunity management is briefly discussed below).

The SE risk management process includes the following activities:

- risk planning
- risk identification
- risk analysis
- risk handling
- risk monitoring

ISO/IEC/IEEE 16085 provides a detailed set of risk management activities and tasks which can be utilized in a risk management process aligned with ISO 31000:2009, Risk management — Principles and Guidelines, and ISO Guide 73:2009,

Risk management — Vocabulary. ISO 9001:2008 standard provides risk-based preventive action requirements in subclause 8.5.3.

The Risk Management Process section of the INCOSE Systems Engineering Handbook: A Guide for Systems Life Cycle Processes and Activities, 4th Edition, provides a comprehensive overview of risk management which is intended to be consistent with the Risk Management Process section of ISO 15288.

## Risk Planning

Risk planning establishes and maintains a strategy for identifying, analyzing, handling, and monitoring risks within the project. The strategy, both the process and its implementation, is documented in a risk management plan (RMP).

The risk management process and its implementation should be tailored to each project and updated as appropriate throughout the life of the project.The RMP should be transmitted in an appropriate means to the project team and key stakeholders.

The risk management strategy includes as necessary the risk management process of all supply chain suppliers and describes how risks from all suppliers will be raised to the next level(s) for incorporation in the project risk process.

The context of the Risk Management process should include a description of stakeholders' perspectives, risk categories, and a description (perhaps by reference) of the technical and managerial objectives, assumptions and constraints. The risk categories include the relevant technical areas of the system and facilitate identification of risks across the life cycle of the system. As noted in ISO 31000 the aim of this step is to generate a comprehensive list of risks based on those events that might create, enhance, prevent, degrade, accelerate or delay the achievement of objectives.

The RMP should contain key risk management information; Conrow (2003) identifies the following as key components of RMP:

- a project summary
- project acquisition and contracting strategies
- key definitions
- a list of key documents
- process steps
- inputs, tools and techniques, and outputs per process step
- linkages between risk management and other project processes
- key ground rules and assumptions
- risk categories
- buyer and seller roles and responsibilities
- organizational and personnel roles and responsibilities

Generally, the level of detail in an RMP is risk-driven, with simple plans for low risk projects and detailed plans for high risk projects.

## Risk Identification

Risk identification is the process of examining the project products, processes, and requirements to identify and document candidate risks. Risk identification should be performed continuously at the individual level as well as through formerly structured events at both regular intervals and following major program changes (e.g., project initiation, re-baselining, change in acquisition phase, etc.).

Conrow (2009) states that systems engineers should use one or more top-level approaches (e.g., work breakdown structure (WBS), key processes evaluation, key requirements evaluation, etc.) and one or more lower-level approaches (e.g., affinity, brainstorming, checklists and taxonomies, examining critical path activities, expert judgment, Ishikawa diagrams, etc.) in risk identification. For example, lower-level checklists and taxonomies exist for software risk identification (Conrow and Shishido 1997, 83-89, p. 84; Boehm 1989, 115-125, Carr et al. 1993, p. A-2) and operational risk identification (Gallagher et al. 2005, p. 4), and have been used on a wide variety of programs. The top and lower-level approaches are essential but there is no single accepted method — all approaches should be examined and used as appropriate.

Candidate risk documentation should include the following items where possible, as identified by Conrow (2003 p.198):

- risk title
- structured risk description
- applicable risk categories
- potential root causes
- relevant historical information
- responsible individual and manager

It is important to use structured risk descriptions such as an *if-then* format: *if* (an event occurs--trigger), *then* (an outcome or affect occurs). Another useful construct is a *condition* (that exists) that leads to a potential *consequence* (outcome) (Gluch 1994). These approaches help the analyst to better think through the potential nature of the risk.

Risk analysis and risk handling activities should only be performed on approved risks to ensure the best use of scarce resources and maintain focus on the correct risks.

## Risk Analysis

Risk analysis is the process of systematically evaluating each identified, approved risk to estimate the probability of occurrence (likelihood) and consequence of occurrence (impact), and then converting the results to a corresponding risk level or rating.

There is no *best* analysis approach for a given risk category. Risk scales and a corresponding matrix, simulations, and probabilistic risk assessments are often used for technical risks, while decision trees, simulations and payoff matrices are used for cost risk; and simulations are used for schedule risk. Risk analysis approaches are sometimes grouped into qualitative and quantitative methods. A structured, repeatable methodology should be used in order to increase analysis accuracy and reduce uncertainty over time.

The most common qualitative method (typically) uses ordinal probability and consequence scales coupled with a risk matrix (also known as a risk cube or mapping matrix) to convert the resulting values to a risk level. Here, one or more probability of occurrence scales, coupled with three consequences of occurrence scales (cost, performance, schedule) are typically used. Mathematical operations should not be performed on ordinal scale values to prevent erroneous results (Conrow 2003, p. 187-364).

Once the risk level for each risk is determined, the risks need to be prioritized. Prioritization is typically performed by risk level (e.g., low, medium, high), risk score (the pair of max (probability), max (consequence) values), and other considerations such as time-frame, frequency of occurrence, and interrelationship with other risks (Conrow 2003, pp. 187-364). An additional prioritization technique is to convert results into an estimated cost, performance, and schedule value (e.g., probability budget consequence). However, the result is only a point estimate and not a distribution of risk.

Widely used quantitative methods include decision trees and the associated expected monetary value analysis (Clemen and Reilly 2001), modeling and simulation (Law 2007; Mun 2010; Vose 2000), payoff matrices (Kerzner 2009, p. 747-751), probabilistic risk assessments (Kumamoto and Henley 1996; NASA 2002), and other techniques. Risk prioritization can directly result from the quantitative methods employed. For quantitative approaches, care is needed in developing the model structure, since the results will only be as good as the accuracy of the structure, coupled with the characteristics of probability estimates or distributions used to model the risks (Law 2007; Evans, Hastings, and Peacock 2011).

If multiple risk facets exist for a given item (e.g., cost risk, schedule risk, and technical risk) the different results should be integrated into a cohesive three-dimensional *picture* of risk. Sensitivity analyses can be applied to both qualitative and quantitative approaches in an attempt to understand how potential variability will affect results. Particular emphasis should be paid to compound risks (e.g., highly coupled technical risks with inadequate fixed budgets and schedules).

## Risk Handling

Risk handling is the process that identifies and selects options and implements the desired option to reduce a risk to an acceptable level, given program constraints (budget, other resources) and objectives (DAU 2003a, 20-23, 70-78).

For a given system-of-interest (SoI), risk handling is primarily performed at two levels. At the system level, the overall ensemble of system risks is initially determined and prioritized and second-level draft risk element plans (REP's) are prepared for handling the risks. For more complex systems, it is important that the REP's at the higher SoI level are kept consistent with the system RMPs at the lower SoI level, and that the top-level RMP preserves continuing risk traceability across the SoI.

The risk handling strategy selected is the combination of the most desirable risk handling option coupled with a suitable implementation approach for that option (Conrow 2003). Risk handling options include assumption, avoidance, control (mitigation), and transfer. All four options should be evaluated and the best one chosen for each risk. An appropriate implementation approach is then chosen for that option. Hybrid strategies can be developed that include more than one risk handling option, but with a single implementation approach. Additional risk handling strategies can also be developed for a given risk and either implemented in parallel with the primary strategy or be made a contingent strategy that is implemented if a particular trigger event occurs during the execution of the primary strategy. Often, this choice is difficult because of uncertainties in the risk probabilities and impacts. In such cases, buying information to reduce risk uncertainty via prototypes, benchmarking, surveying, modeling, etc. will clarify risk handling decisions (Boehm 1981).

### Risk Handling Plans

A risk handling plan (RHP - a REP at the system level), should be developed and implemented for all *high* and *medium* risks and selected *low* risks as warranted.

As identified by Conrow (2003, 365-387), each RHP should include:

- a risk owner and management contacts
- selected option
- implementation approach
- estimated probability and consequence of occurrence levels at the start and conclusion of each activity
- specific measurable exit criteria for each activity
- appropriate metrics
- resources needed to implement the RHP

Metrics included in each RHP should provide an objective means of determining whether the risk handling strategy is on track and whether it needs to be updated. On larger projects these can include earned value, variation in schedule and technical performance measures (TPMs), and changes in risk level vs. time.

The activities present in each RHP should be integrated into the project's integrated master schedule or equivalent; otherwise there will be ineffective risk monitoring and control.

## Risk Monitoring

Risk monitoring is used to evaluate the effectiveness of risk handling activities against established metrics and provide feedback to the other risk management process steps. Risk monitoring results may also provide a basis to update RHPs, develop additional risk handling options and approaches, and re-analyze risks. In some cases, monitoring results may also be used to identify new risks, revise an existing risk with a new facet, or revise some aspects of risk planning (DAU 2003a, p. 20). Some risk monitoring approaches that can be applied include earned value, program metrics, TPMs, schedule analysis, and variations in risk level. Risk monitoring approaches should be updated and evaluated at the same time and WBS level; otherwise, the results may be inconsistent.

## Opportunity and Opportunity Management

In principle, opportunity management is the duality to risk management, with two components: (1) probability of achieving an improved outcome and (2) impact of achieving the outcome. Thus, both should be addressed in risk management planning and execution. In practice, however, a positive opportunity exposure will not match a negative risk exposure in utility space, since the positive utility magnitude of improving an expected outcome is considerably less than the negative utility magnitude of failing to meet an expected outcome (Canada 1971; Kahneman-Tversky 1979). Further, since many opportunity-management initiatives have failed to anticipate serious side effects, all candidate opportunities should be thoroughly evaluated for potential risks to prevent unintended consequences from occurring.

In addition, while opportunities may provide potential benefits for the system or project, each opportunity pursued may have associated risks that detract from the expected benefit. This may reduce the ability to achieve the anticipated effects of the opportunity, in addition to any limitations associated with not pursing an opportunity.

## Linkages to Other Systems Engineering Management Topics

The measurement process provides indicators for risk analysis. Project planning involves the identification of risk and planning for stakeholder involvement. Project assessment and control monitors project risks. Decision management evaluates alternatives for selection and handling of identified and analyzed risks.

## Practical Considerations

Key pitfalls and good practices related to systems engineering risk management are described in the next two sections.

### Pitfalls

Some of the key pitfalls encountered in performing risk management are below in Table 1.

**Table 1. Risk Management Pitfalls. (SEBoK Original)**

| Name | Description |
|---|---|
| Process Over-Reliance | • Over-reliance on the process side of risk management without sufficient attention to human and organizational behavioral considerations. |
| Lack of Continuity | • Failure to implement risk management as a continuous process. Risk management will be ineffective if it's done just to satisfy project reviews or other discrete criteria. (Charette, Dwinnell, and McGarry 2004, 18-24 and Scheinin 2008). |
| Tool and Technique Over-Reliance | • Over-reliance on tools and techniques, with insufficient thought and resources expended on how the process will be implemented and run on a day-to-day basis. |
| Lack of Vigilance | • A comprehensive risk identification will generally not capture all risks; some risks will always escape detection, which reinforces the need for risk identification to be performed continuously. |
| Automatic Mitigation Selection | • Automatically select the risk handling mitigation option, rather than evaluating all four options in an unbiased fashion and choosing the "best" option. |
| Sea of Green | • Tracking progress of the risk handling plan, while the plan itself may not adequately include steps to reduce the risk to an acceptable level. Progress indicators may appear "green" (acceptable) associated with the risk handling plan: budgeting, staffing, organizing, data gathering, model preparation, etc. However, the risk itself may be largely unaffected if the handling strategy and the resulting plan are poorly developed, do not address potential root cause(s), and do not incorporate actions that will effectively resolve the risk. |
| Band-Aid Risk Handling | • Handling risks (e.g., interoperability problems with changes in external systems) by patching each instance, rather than addressing the root cause(s) and reducing the likelihood of future instances. |

## Good Practices

Some good practices gathered from the references are below in Table 2.

**Table 2. Risk Management Good Practices. (SEBoK Original)**

| Name | Description |
|---|---|
| Top Down and Bottom Up | • Risk management should be both "top down" and "bottom up" in order to be effective. The project manager or deputy need to own the process at the top level, but risk management principles should be considered and used by all project personnel. |
| Early Planning | • Include the planning process step in the risk management process. Failure to adequately perform risk planning early in the project phase contributes to ineffective risk management. |
| Risk Analysis Limitations | • Understand the limitations of risk analysis tools and techniques. Risk analysis results should be challenged because considerable input uncertainty and/or potential errors may exist. |
| Robust Risk Handling Strategy | • The risk handling strategy should attempt to reduce both the probability and consequence of occurrence terms. It is also imperative that the resources needed to properly implement the chosen strategy be available in a timely manner, else the risk handling strategy, and the entire risk management process, will be viewed as a "paper tiger." |
| Structured Risk Monitoring | • Risk monitoring should be a structured approach to compare actual vs. anticipated cost, performance, schedule, and risk outcomes associated with implementing the RHP. When ad-hoc or unstructured approaches are used, or when risk level vs. time is the only metric tracked, the resulting risk monitoring usefulness can be greatly reduced. |
| Update Risk Database | • The risk management database (registry) should be updated throughout the course of the program, striking a balance between excessive resources required and insufficient updates performed. Database updates should occur at both a tailored, regular interval and following major program changes. |

## References

### Works Cited

Boehm, B. 1981. *Software Engineering Economics*. Upper Saddle River, NJ, USA: Prentice Hall.

Boehm, B. 1989. *Software Risk Management*. Los Alamitos, CA; Tokyo, Japan: IEEE Computer Society Press: 115-125.

Canada, J.R. 1971. *Intermediate Economic Analysis for Management and Engineering*. Upper Saddle River, NJ, USA: Prentice Hall.

Carr, M., S. Konda, I. Monarch, F. Ulrich, and C. Walker. 1993. *Taxonomy-based risk identification*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-93-TR-6.

Charette, R., L. Dwinnell, and J. McGarry. 2004. "Understanding the roots of process performance failure." *CROSSTALK: The Journal of Defense Software Engineering* (August 2004): 18-24.

Clemen, R., and T. Reilly. 2001. *Making hard decisions*. Boston, MA, USA: Duxbury.

Conrow, E. 2003. *Effective Risk Management: Some Keys to Success,* 2nd ed. Reston, VA, USA: American Institute of Aeronautics and Astronautics (AIAA).

Conrow, E. 2008. "Risk analysis for space systems." Paper presented at Space Systems Engineering and Risk Management Symposium, 27-29 February, 2008, Los Angeles, CA, USA.

Conrow, E. and P. Shishido. 1997. "Implementing risk management on software intensive projects." IEEE *Software.* 14(3) (May/June 1997): 83-9.

DAU. 2003a. *Risk Management Guide for DoD Acquisition: Fifth Edition,* version 2. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU) Press.

DAU. 2003b. *U.S. Department of Defense extension to: A guide to the project management body of knowledge (PMBOK(R) guide), first edition*. Version 1. 1st ed. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU) Press.

DoD. 2015. *Risk, Issue, and Opportunity Management Guide for Defense Acquisition Programs*. Washington, DC, USA: Office of the Deputy Assistant Secretary of Defense for Systems Engineering/Department of Defense.

Evans, M., N. Hastings, and B. Peacock. 2000. *Statistical Distributions,* 3rd ed. New York, NY, USA: Wiley-Interscience.

Forbes, C., M. Evans, N. Hastings, and B. Peacock. 2011. “Statistical Distributions,” 4th ed. New York, NY, USA.

Gallagher, B., P. Case, R. Creel, S. Kushner, and R. Williams. 2005. *A taxonomy of operational risk*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-2005-TN-036.

Gluch, P. 1994. *A Construct for Describing Software Development Risks*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-94-TR-14.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Kerzner, H. 2009. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling.* 10th ed. Hoboken, NJ, USA: John Wiley & Sons.

Kahneman, D., and A. Tversky. 1979. "Prospect theory: An analysis of decision under risk." *Econometrica.* 47(2) (Mar., 1979): 263-292.

Kumamoto, H. and E. Henley. 1996. *Probabilistic Risk Assessment and Management for Engineers and Scientists,* 2nd ed. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) Press.

Law, A. 2007. *Simulation Modeling and Analysis,* 4th ed. New York, NY, USA: McGraw Hill.

Mun, J. 2010. *Modeling Risk,* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

NASA. 2002. *Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners,* version 1.1. Washington, DC, USA: Office of Safety and Mission Assurance/National Aeronautics and Space Administration (NASA).

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Scheinin, W. 2008. "Start Early and Often: The Need for Persistent Risk Management in the Early Acquisition Phases." Paper presented at Space Systems Engineering and Risk Management Symposium, 27-29 February 2008, Los Angeles, CA, USA.

SEI. 2010. *Capability Maturity Model Integrated (CMMI) for Development,* version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Vose, D. 2000. *Quantitative Risk Analysis,* 2nd ed. New York, NY, USA: John Wiley & Sons.

Willis, H.H., A.R. Morral, T.K. Kelly, and J.J. Medby. 2005. *Estimating Terrorism Risk.* Santa Monica, CA, USA: The RAND Corporation, MG-388.

## Primary References

Boehm, B. 1981. *Software Engineering Economics.* Upper Saddle River, NJ, USA:Prentice Hall.

Boehm, B. 1989. *Software Risk Management.* Los Alamitos, CA; Tokyo, Japan: IEEE Computer Society Press, p. 115-125.

Conrow, E.H. 2003. *Effective Risk Management: Some Keys to Success,* 2nd ed. Reston, VA, USA: American Institute of Aeronautics and Astronautics (AIAA).

DoD. 2015. Risk, Issue, and Opportunity Management Guide for Defense Acquisition Programs. Washington, DC, USA: Office of the Deputy Assistant Secretary of Defense for Systems Engineering/Department of Defense.

SEI. 2010. *Capability Maturity Model Integrated (CMMI) for Development,* version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

## Additional References

Canada, J.R. 1971. *Intermediate Economic Analysis for Management and Engineering*. Upper Saddle River, NJ, USA: Prentice Hall.

Carr, M., S. Konda, I. Monarch, F. Ulrich, and C. Walker. 1993. *Taxonomy-based risk identification*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-93-TR-6.

Charette, R. 1990. *Application Strategies for Risk Management*. New York, NY, USA: McGraw-Hill.

Charette, R. 1989. *Software Engineering Risk Analysis and Management.* New York, NY, USA: McGraw-Hill (MultiScience Press).

Charette, R., L. Dwinnell, and J. McGarry. 2004. "Understanding the roots of process performance failure." *CROSSTALK: The Journal of Defense Software Engineering* (August 2004): 18-24.

Clemen, R., and T. Reilly. 2001. *Making hard decisions*. Boston, MA, USA: Duxbury.

Conrow, E. 2010. "Space program schedule change probability distributions." Paper presented at American Institute of Aeronautics and Astronautics (AIAA) Space 2010, 1 September 2010, Anaheim, CA, USA.

Conrow, E. 2009. "Tailoring risk management to increase effectiveness on your project." Presentation to the Project Management Institute, Los Angeles Chapter, 16 April, 2009, Los Angeles, CA.

Conrow, E. 2008. "Risk analysis for space systems." Paper presented at Space Systems Engineering and Risk Management Symposium, 27-29 February, 2008, Los Angeles, CA, USA.

Conrow, E. and P. Shishido. 1997. "Implementing risk management on software intensive projects." IEEE *Software.* 14(3) (May/June 1997): 83-9.

DAU. 2003a. *Risk Management Guide for DoD Acquisition: Fifth Edition.* Version 2. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU) Press.

DAU. 2003b. *U.S. Department of Defense extension to: A guide to the project management body of knowledge (PMBOK(R) guide),* 1st ed. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU) Press.

Dorofee, A., J. Walker, C. Alberts, R. Higuera, R. Murphy, and R. Williams (eds). 1996. *Continuous Risk Management Guidebook.* Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU).

Gallagher, B., P. Case, R. Creel, S. Kushner, and R. Williams. 2005. *A taxonomy of operational risk*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-2005-TN-036.

Gluch, P. 1994. *A Construct for Describing Software Development Risks*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU), CMU/SEI-94-TR-14.

Haimes, Y.Y. 2009. *Risk Modeling, Assessment, and Management*. Hoboken, NJ, USA: John Wiley & Sons, Inc.

Hall, E. 1998. *Managing Risk: Methods for Software Systems Development.* New York, NY, USA: Addison Wesley Professional.

INCOSE. 2015. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 4. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2014-001-04.

ISO. 2009. *Risk Management—Principles and Guidelines*. Geneva, Switzerland: International Organization for Standardization (ISO), ISO 31000:2009.

ISO/IEC. 2009. *Risk Management—Risk Assessment Techniques*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 31010:2009.

ISO/IEC/IEEE. 2006. *Systems and Software Engineering - Risk Management*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 16085.

ISO. 2003. *Space Systems - Risk Management.* Geneva, Switzerland: International Organization for Standardization (ISO), ISO 17666:2003.

Jones, C. 1994. *Assessment and Control of Software Risks.* Upper Saddle River, NJ, USA: Prentice-Hall.

Kahneman, D. and A. Tversky. 1979. "Prospect theory: An analysis of decision under risk." *Econometrica.* 47(2) (Mar., 1979): 263-292.

Kerzner, H. 2009. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling,* 10th ed. Hoboken, NJ: John Wiley & Sons.

Kumamoto, H., and E. Henley. 1996. *Probabilistic Risk Assessment and Management for Engineers and Scientists,* 2nd ed. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) Press.

Law, A. 2007. *Simulation Modeling and Analysis,* 4th ed. New York, NY, USA: McGraw Hill.

MITRE. 2012. *Systems Engineering Guide to Risk Management.* Available online: http://www.mitre.org/work/systems_engineering/guide/acquisition_systems_engineering/risk_management/. Accessed on July 7, 2012. Page last updated on May 8, 2012.

Mun, J. 2010. *Modeling Risk,* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

NASA. 2002. *Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners,* version 1.1. Washington, DC, USA: Office of Safety and Mission Assurance/National Aeronautics and Space Administration (NASA).

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Scheinin, W. 2008. "Start Early and Often: The Need for Persistent Risk Management in the Early Acquisition Phases." Paper presented at Space Systems Engineering and Risk Management Symposium, 27-29 February 2008, Los Angeles, CA, USA.

USAF. 2005. *SMC systems engineering primer & handbook: Concepts, processes, and techniques,* 3rd ed. Los Angeles, CA, USA: Space & Missile Systems Center/U.S. Air Force (USAF).

USAF. 2014. "SMC Risk Management Process Guide. *Version 2. Los Angeles, CA, USA: Space & Missile Systems Center/U.S. Air Force (USAF).*

Vose, D. 2000. *Quantitative Risk Analysis.* 2nd ed. New York, NY, USA: John Wiley & Sons.

Willis, H.H., A.R. Morral, T.K. Kelly, and J.J. Medby. 2005. *Estimating Terrorism Risk*. Santa Monica, CA, USA: The RAND Corporation, MG-388.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Measurement

***Lead Authors:*** *Garry Roedler, Ray Madachy*

Measurement and the accompanying analysis are fundamental elements of systems engineering (SE) and technical management. SE measurement provides information relating to the products developed, services provided, and processes implemented to support effective management of the processes and to objectively evaluate product or service quality. Measurement supports realistic planning, provides insight into actual performance, and facilitates assessment of suitable actions (Roedler and Jones 2005, 1-65; Frenz et al. 2010).

Appropriate measures and indicators are essential inputs to tradeoff analyses to balance cost, schedule, and technical objectives. Periodic analysis of the relationships between measurement results and review of the requirements and attributes of the system provides insights that help to identify issues early, when they can be resolved with less impact. Historical data, together with project or organizational context information, forms the basis for the predictive models and methods that should be used.

## Fundamental Concepts

The discussion of measurement in this article is based on some fundamental concepts. Roedler et al. (2005, 1-65) states three key SE measurement concepts that are paraphrased here:

1. **SE measurement is a consistent but flexible process** tailored to the unique information needs and characteristics of a particular project or organization and revised as information needs change.
2. **Decision makers must understand what is being measured.** Key decision-makers must be able to connect *what is being measured* to *what they need to know* and *what decisions they need to make* as part of a closed-loop, feedback control process (Frenz et al. 2010).
3. **Measurement must be used to be effective.**

## Measurement Process Overview

The measurement process as presented here consists of four activities from Practical Software and Systems Measurement (PSM) (2011) and described in (ISO/IEC/IEEE 15939; McGarry et al. 2002):

1. establish and sustain commitment
2. plan measurement
3. perform measurement
4. evaluate measurement

This approach has been the basis for establishing a common process across the software and systems engineering communities. This measurement approach has been adopted by the Capability Maturity Model Integration (CMMI) measurement and analysis process area (SEI 2006, 10), as well as by international systems and software engineering standards (ISO/IEC/IEEE 15939; ISO/IEC/IEEE 15288, 1). The International Council on Systems Engineering (INCOSE) Measurement Working Group has also adopted this measurement approach for several of their measurement assets, such as the INCOSE SE Measurement Primer (Frenz et al. 2010) and Technical Measurement Guide (Roedler and Jones 2005). This approach has provided a consistent treatment of measurement that allows the engineering community to communicate more effectively about measurement. The process is illustrated in Figure 1 from Roedler and Jones (2005) and McGarry et al. (2002).

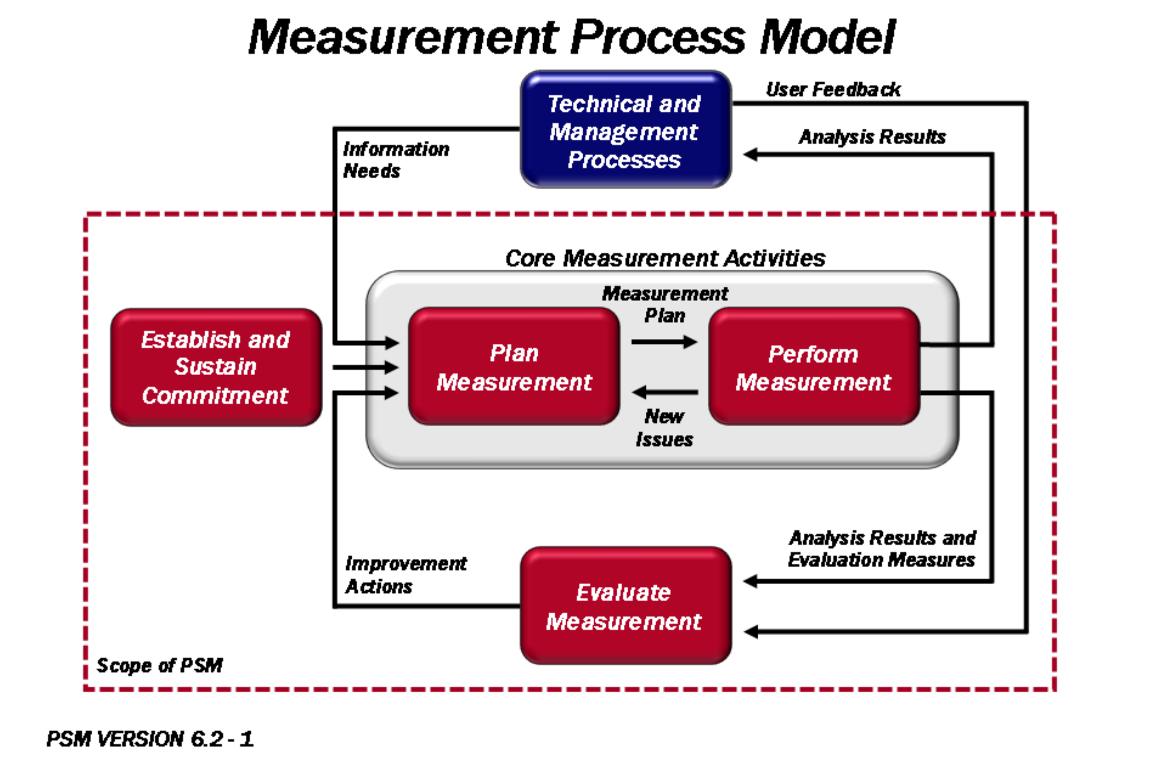


**Figure 1. Four Key Measurement Process Activities (PSM 2011).** Reprinted with permission of Practical Software and Systems Measurement (PSM [1]). All other rights are reserved by the copyright owner.

## Establish and Sustain Commitment

This activity focuses on establishing the resources, training, and tools to implement a measurement process and ensure that there is a management commitment to use the information that is produced. Refer to PSM (August 18, 2011) and SPC (2011) for additional detail.

## Plan Measurement

This activity focuses on defining measures that provide insight into project or organization information needs. This includes identifying what the decision-makers need to know and when they need to know it, relaying these information needs to those entities in a manner that can be measured, and identifying, prioritizing, selecting, and specifying measures based on project and organization processes (Jones 2003, 15-19). This activity also identifies the reporting format, forums, and target audience for the information provided by the measures.

Here are a few widely used approaches to identify the information needs and derive associated measures, where each can be focused on identifying measures that are needed for SE management:

- The PSM approach, which uses a set of information categories, measurable concepts, and candidate measures to aid the user in determining relevant information needs and the characteristics of those needs on which to focus (PSM August 18, 2011).
- The (GQM) approach, which identifies explicit measurement goals. Each goal is decomposed into several questions that help in the selection of measures that address the question and provide insight into the goal achievement (Park, Goethert, and Florac 1996).

- Software Productivity Center’s (SPC's) 8-step Metrics Program, which also includes stating the goals and defining measures needed to gain insight for achieving the goals (SPC 2011).

The following are good sources for candidate measures that address information needs and measurable concepts/questions:

- PSM Web Site (PSM 2011)
- PSM Guide, Version 4.0, Chapters 3 and 5 (PSM 2000)
- SE Leading Indicators Guide, Version 2.0, Section 3 (Roedler et al. 2010)
- Technical Measurement Guide, Version 1.0, Section 10 (Roedler and Jones 2005, 1-65)
- Safety Measurement (PSM White Paper), Version 3.0, Section 3.4 (Murdoch 2006, 60)
- Security Measurement (PSM White Paper), Version 3.0, Section 7 (Murdoch 2006, 67)
- Measuring Systems Interoperability, Section 5 and Appendix C (Kasunic and Anderson 2004)
- Measurement for Process Improvement (PSM Technical Report), version 1.0, Appendix E (Statz 2005)

The INCOSE *SE Measurement Primer* (Frenz et al. 2010) provides a list of attributes of a good measure with definitions for each attribute; these attributes include *relevance, completeness, timeliness, simplicity, cost effectiveness, repeatability, and accuracy.* Evaluating candidate measures against these attributes can help assure the selection of more effective measures.

The details of each measure need to be unambiguously defined and documented. Templates for the specification of measures and indicators are available on the PSM website (2011) and in Goethert and Siviy (2004).

## Perform Measurement

This activity focuses on the collection and preparation of measurement data, measurement analysis, and the presentation of the results to inform decision makers. The preparation of the measurement data includes verification, normalization, and aggregation of the data, as applicable. Analysis includes estimation, feasibility analysis of plans, and performance analysis of actual data against plans.

The quality of the measurement results is dependent on the collection and preparation of valid, accurate, and unbiased data. Data verification, validation, preparation, and analysis techniques are discussed in PSM (2011) and SEI (2010). Per TL 9000, *Quality Management System Guidance*, *The analysis step should integrate quantitative measurement results and other qualitative project information, in order to provide managers the feedback needed for effective decision making* (QuEST Forum 2012, 5-10). This provides richer information that gives the users the broader picture and puts the information in the appropriate context.

There is a significant body of guidance available on good ways to present quantitative information. Edward Tufte has several books focused on the visualization of information, including *The Visual Display of Quantitative Information* (Tufte 2001).

Other resources that contain further information pertaining to understanding and using measurement results include

- PSM (2011)
- ISO/IEC/IEEE 15939, clauses 4.3.3 and 4.3.4
- Roedler and Jones (2005), sections 6.4, 7.2, and 7.3

## Evaluate Measurement

This activity involves the analysis of information that explains the periodic evaluation and improvement of the measurement process and specific measures. One objective is to ensure that the measures continue to align with the business goals and information needs, as well as provide useful insight. This activity should also evaluate the SE measurement activities, resources, and infrastructure to make sure it supports the needs of the project and organization. Refer to PSM (2011) and *Practical Software Measurement: Objective Information for Decision Makers* (McGarry et al. 2002) for additional detail.

## Systems Engineering Leading Indicators

Leading indicators are aimed at providing predictive insight that pertains to an information need. A SE leading indicator is *a measure for evaluating the effectiveness of a how a specific activity is applied on a project in a manner that provides information about impacts that are likely to affect the system performance objectives* (Roedler et al. 2010). Leading indicators may be individual measures or collections of measures and associated analysis that provide future systems engineering performance insight throughout the life cycle of the system; they *support the effective management of systems engineering by providing visibility into expected project performance and potential future states* (Roedler et al. 2010).

As shown in Figure 2, a leading indicator is composed of characteristics, a condition, and a predicted behavior. The characteristics and conditions are analyzed on a periodic or as-needed basis to predict behavior within a given confidence level and within an accepted time range into the future. More information is also provided by Roedler et al. (2010).

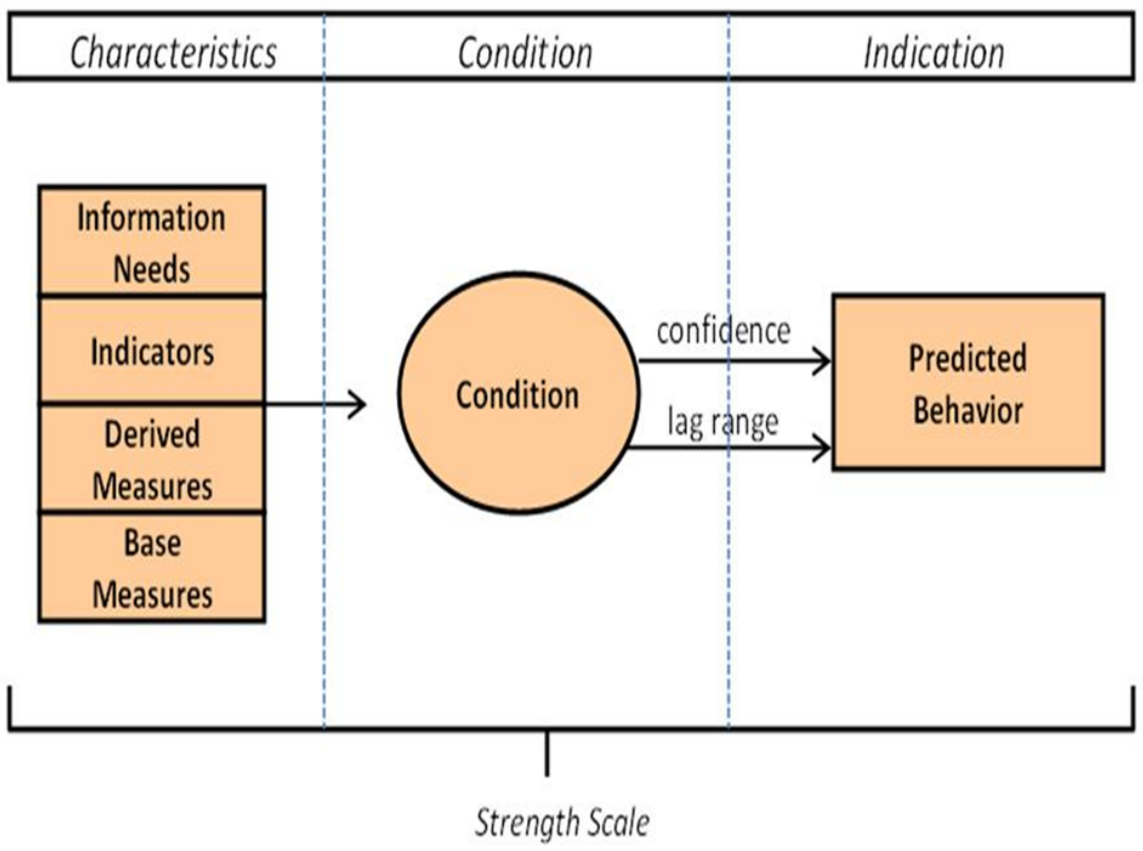


**Figure 2. Composition of a Leading Indicator (Roedler et al. 2010).** Reprinted with permission of the International Council on Systems Engineering (INCOSE [2]) and Practical Software and Systems Measurement (PSM [1]). All other rights are reserved by the copyright owner.

## Technical Measurement

Technical measurement is the set of measurement activities used to provide information about progress in the definition and development of the technical solution, ongoing assessment of the associated risks and issues, and the likelihood of meeting the critical objectives of the acquirer. This insight helps an engineer make better decisions throughout the life cycle of a system and increase the probability of delivering a technical solution that meets both the specified requirements and the mission needs. The insight is also used in trade-off decisions when performance is not within the thresholds or goals.

Technical measurement includes measures of effectiveness (MOEs), measures of performance (MOPs), and technical performance measures (TPMs) (Roedler and Jones 2005, 1-65). The relationships between these types of technical measures are shown in Figure 3 and explained in the reference for Figure 3. Using the measurement process described above, technical measurement can be planned early in the life cycle and then performed throughout the life cycle with increasing levels of fidelity as the technical solution is developed, facilitating predictive insight and preventive or corrective actions. More information about technical measurement can be found in the *NASA Systems Engineering Handbook*, *System Analysis, Design, Development: Concepts, Principles, and Practices*, and the *Systems Engineering Leading Indicators Guide* (NASA December 2007, 1-360, Section 6.7.2.2; Wasson 2006, Chapter 34; Roedler and Jones 2005).

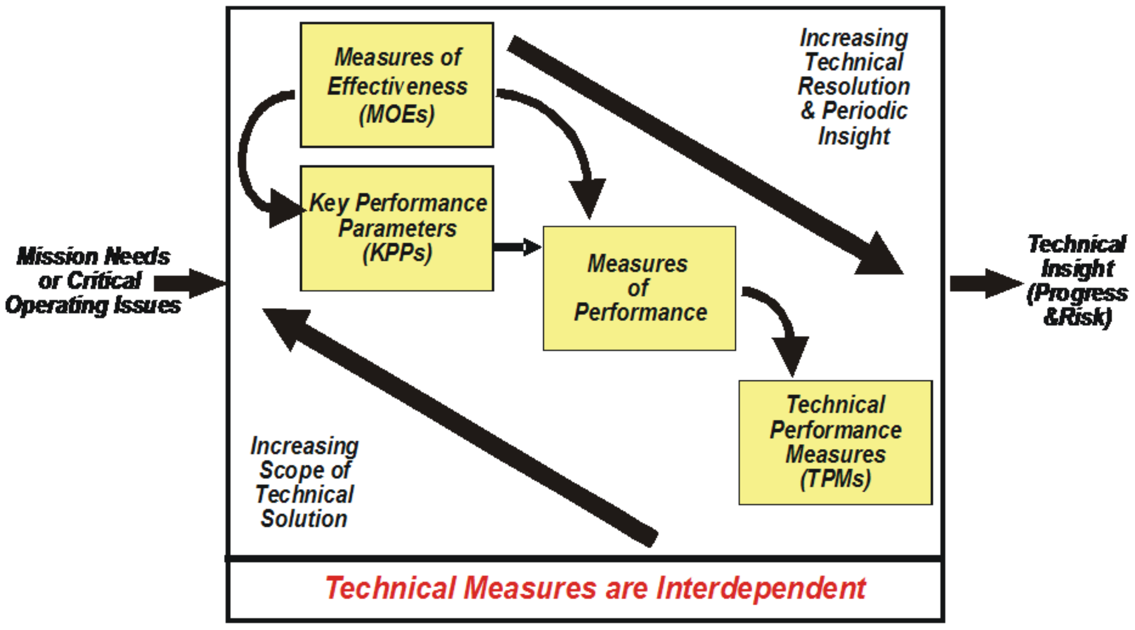

Measures of Effectiveness (MOEs)
Key Performance Parameters (KPPs)
Measures of Performance
Technical Performance Measures (TPMs)
Increasing Technical Resolution & Periodic Insight
Increasing Scope of Technical Solution
Mission Needs or Critical Operating Issues
Technical Insight (Progress &Risk)
Technical Measures are Interdependent

**Figure 3. Relationship of the Technical Measures (Roedler et al 2010).** Reprinted with permission of the International Council on Systems Engineering (INCOSE [1]) and Practical Software and Systems Measurement (PSM [1]). All other rights are reserved by the copyright owner.

## Service Measurement

The same measurement activities can be applied for service measurement; however, the context and measures will be different. Service providers have a need to balance efficiency and effectiveness, which may be opposing objectives. Good service measures are outcome-based, focus on elements important to the customer (e.g., service availability, reliability, performance, etc.), and provide timely, forward-looking information.

For services, the terms critical success factors (CSF) and key performance indicators (KPI) are used often when discussing measurement. CSFs are the key elements of the service or service infrastructure that are most important to achieve the business objectives. KPIs are specific values or characteristics measured to assess achievement of those objectives.

More information about service measurement can be found in the *Service Design* and *Continual Service Improvement* volumes of BMP (2010, 1). More information on service SE can be found in the Service Systems Engineering article.

## Linkages to Other Systems Engineering Management Topics

SE measurement has linkages to other SEM topics. The following are a few key linkages adapted from Roedler and Jones (2005):

- Planning – SE measurement provides the historical data and supports the estimation for, and feasibility analysis of, the plans for realistic planning.
- Assessment and Control – SE measurement provides the objective information needed to perform the assessment and determination of appropriate control actions. The use of leading indicators allows for early assessment and control actions that identify risks and/or provide insight to allow early treatment of risks to minimize potential impacts.
- Risk Management – SE risk management identifies the information needs that can impact project and organizational performance. SE measurement data helps to quantify risks and subsequently provides information about whether risks have been successfully managed.
- Decision Management – SE Measurement results inform decision making by providing objective insight.

## Practical Considerations

Key pitfalls and good practices related to SE measurement are described in the next two sections.

### Pitfalls

Some of the key pitfalls encountered in planning and performing SE Measurement are provided in Table 1.

**Table 1. Measurement Pitfalls. (SEBoK Original)**

| Name | Description |
|---|---|
| Golden Measures | • Looking for the one measure or small set of measures that applies to all projects.<br>• No one-size-fits-all measure or measurement set exists.<br>• Each project has unique information needs (e.g., objectives, risks, and issues).<br>• The one exception is that, in some cases with consistent product lines, processes, and information needs, a small core set of measures may be defined for use across an organization. |
| Single-Pass Perspective | • Viewing measurement as a single-pass activity.<br>• To be effective, measurement needs to be performed continuously, including the periodic identification and prioritization of information needs and associated measures. |
| Unknown Information Need | • Performing measurement activities without the understanding of why the measures are needed and what information they provide.<br>• This can lead to wasted effort. |
| Inappropriate Usage | • Using measurement inappropriately, such as measuring the performance of individuals or making interpretations without context information.<br>• This can lead to bias in the results or incorrect interpretations. |

### Good Practices

Some good practices gathered from the references are provided in Table 2.

**Table 2. Measurement Good Practices. (SEBoK Original)**

| Name | Description |
|---|---|
| Periodic Review | • Regularly review each measure collected. |
| Action Driven | • Measurement by itself does not control or improve process performance.<br>• Measurement results should be provided to decision makers for appropriate action. |
| Integration into Project Processes | • SE Measurement should be integrated into the project as part of the ongoing project business rhythm.<br>• Data should be collected as processes are performed, not recreated as an afterthought. |
| Timely Information | • Information should be obtained early enough to allow necessary action to control or treat risks, adjust tactics and strategies, etc.<br>• When such actions are not successful, measurement results need to help decision-makers determine contingency actions or correct problems. |
| Relevance to Decision Makers | • Successful measurement requires the communication of meaningful information to the decision-makers.<br>• Results should be presented in the decision-makers' preferred format.<br>• Allows accurate and expeditious interpretation of the results. |
| Data Availability | • Decisions can rarely wait for a complete or perfect set of data, so measurement information often needs to be derived from analysis of the best available data, complemented by real-time events and qualitative insight (including experience). |
| Historical Data | • Use historical data as the basis of plans, measure what is planned versus what is achieved, archive actual achieved results, and use archived data as a historical basis for the next planning effort. |
| Information Model | • The information model defined in ISO/IEC/IEEE (2007) provides a means to link the entities that are measured to the associated measures and the identified information need, and also describes how the measures are converted into indicators that provide insight to decision-makers. |

Additional information can be found in the *Systems Engineering Measurement Primer*, Section 4.2 (Frenz et al. 2010), and INCOSE *Systems Engineering Handbook*, Section 5.7.1.5 (2012).

## References

### Works Cited

Frenz, P., G. Roedler, D.J. Gantzer, P. Baxter. 2010. *Systems Engineering Measurement Primer: A Basic Introduction to Measurement Concepts and Use for Systems Engineering.* Version 2.0. San Diego, CA: International Council on System Engineering (INCOSE). INCOSE-TP-2010-005-02. Accessed April 13, 2015 at http://www.incose.org/ProductsPublications/techpublications/PrimerMeasurement.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2007. *Systems and software engineering - Measurement process*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC/IEEE 15939:2007.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Kasunic, M. and W. Anderson. 2004. *Measuring Systems Interoperability: Challenges and Opportunities.* Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

McGarry, J., D. Card, C. Jones, B. Layman, E. Clark, J. Dean, F. Hall. 2002. *Practical Software Measurement: Objective Information for Decision Makers*. Boston, MA, USA: Addison-Wesley.

NASA. 2007. *Systems Engineering Handbook.* Washington, DC, USA: National Aeronautics and Space Administration (NASA), December 2007. NASA/SP-2007-6105.

Park, R.E., W.B. Goethert, and W.A. Florac. 1996. *Goal-Driven Software Measurement – A Guidebook*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU), CMU/SEI-96-BH-002.

PSM. 2011. "Practical Software and Systems Measurement." Accessed August 18, 2011. Available at: http://www.psmsc.com/.

PSM. 2000. *Practical Software and Systems Measurement (PSM) Guide,* version 4.0c. Practical Software and System Measurement Support Center. Available at: http://www.psmsc.com/PSMGuide.asp.

PSM Safety & Security TWG. 2006. *Safety Measurement,* version 3.0. Practical Software and Systems Measurement. Available at: http://www.psmsc.com/Downloads/TechnologyPapers/SafetyWhitePaper_v3.0.pdf.

PSM Safety & Security TWG. 2006. *Security Measurement,* version 3.0. Practical Software and Systems Measurement. Available at: http://www.psmsc.com/Downloads/TechnologyPapers/SecurityWhitePaper_v3.0.pdf.

QuEST Forum. 2012. *Quality Management System (QMS) Measurements Handbook,* Release 5.0. Plano, TX, USA: Quest Forum.

Roedler, G., D. Rhodes, C. Jones, and H. Schimmoller. 2010. *Systems Engineering Leading Indicators Guide,* version 2.0. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2005-001-03.

Roedler, G. and C. Jones. 2005. *Technical Measurement Guide,* version 1.0. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-020-01.

SEI. 2010. "Measurement and Analysis Process Area" in *Capability Maturity Model Integrated (CMMI) for Development*, version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Software Productivity Center, Inc. 2011. Software Productivity Center web site. August 20, 2011. Available at: http://www.spc.ca/.

Statz, J. et al. 2005. *Measurement for Process Improvement,* version 1.0. York, UK: Practical Software and Systems Measurement (PSM).

Tufte, E. 2006. *The Visual Display of Quantitative Information.* Cheshire, CT, USA: Graphics Press.

Wasson, C. 2005. *System Analysis, Design, Development: Concepts, Principles, and Practices*. Hoboken, NJ, USA: John Wiley and Sons.

## Primary References

Frenz, P., G. Roedler, D.J. Gantzer, P. Baxter. 2010. *Systems Engineering Measurement Primer: A Basic Introduction to Measurement Concepts and Use for Systems Engineering.* Version 2.0. San Diego, CA: International Council on System Engineering (INCOSE). INCOSE-TP-2010-005-02. Accessed April 13, 2015 at http://www.incose.org/ProductsPublications/techpublications/PrimerMeasurement.

ISO/IEC/IEEE. 2007. *Systems and Software Engineering - Measurement Process*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC/IEEE 15939:2007.

PSM. 2000. *Practical Software and Systems Measurement (PSM) Guide,* version 4.0c. Practical Software and System Measurement Support Center. Available at: http://www.psmsc.com.

Roedler, G., D. Rhodes, C. Jones, and H. Schimmoller. 2010. *Systems Engineering Leading Indicators Guide,* version 2.0. San Diego, CA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2005-001-03.

Roedler, G. and C. Jones. 2005. *Technical Measurement Guide,* version 1.0. San Diego, CA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-020-01.

## Additional References

Kasunic, M. and W. Anderson. 2004. *Measuring Systems Interoperability: Challenges and Opportunities.* Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

McGarry, J. et al. 2002. *Practical Software Measurement: Objective Information for Decision Makers*. Boston, MA, USA: Addison-Wesley.

NASA. 2007. *NASA Systems Engineering Handbook.* Washington, DC, USA: National Aeronautics and Space Administration (NASA), December 2007. NASA/SP-2007-6105.

Park, Goethert, and Florac. 1996. *Goal-Driven Software Measurement – A Guidebook*. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU), CMU/SEI-96-BH-002.

PSM. 2011. "Practical Software and Systems Measurement." Accessed August 18, 2011. Available at: http://www. psmsc.com/.

PSM Safety & Security TWG. 2006. *Safety Measurement,* version 3.0. Practical Software and Systems Measurement. Available at: http://www.psmsc.com/Downloads/TechnologyPapers/SafetyWhitePaper_v3.0. pdf.

PSM Safety & Security TWG. 2006. *Security Measurement,* version 3.0. Practical Software and Systems Measurement. Available at: http://www.psmsc.com/Downloads/TechnologyPapers/SecurityWhitePaper_v3.0. pdf.

SEI. 2010. "Measurement and Analysis Process Area" in *Capability Maturity Model Integrated (CMMI) for Development*, version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

Software Productivity Center, Inc. 2011. Software Productivity Center web site. August 20, 2011. Available at: http:/ /www.spc.ca/.

Statz, J. 2005. *Measurement for Process Improvement,* version 1.0. York, UK: Practical Software and Systems Measurement (PSM).

Tufte, E. 2006. *The Visual Display of Quantitative Information.* Cheshire, CT, USA: Graphics Press.

Wasson, C. 2005. *System Analysis, Design, Development: Concepts, Principles, and Practices*. Hoboken, NJ, USA: John Wiley and Sons.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

## References

[1] http://www.psmsc.com
[2] http://www.incose.com

# Decision Management

***Lead Author:*** *Ray Madachy*, **Contributing Authors:** *Garry Roedler, Greg Parnell*

Many systems engineering decisions are difficult because they include numerous stakeholders, multiple competing objectives, substantial uncertainty, and significant consequences. In these cases, good decision making requires a formal decision management process. The purpose of the decision management process is:

> "…to provide a structured, analytical framework for objectively identifying, characterizing and evaluating a set of alternatives for a decision at any point in the life cycle and select the most beneficial course of action."(ISO/IEC/IEEE 15288)

Decision situations (opportunities) are commonly encountered throughout a system's lifecycle. The decision management method most commonly employed by systems engineers is the trade study. Trade studies aim to define, measure, and assess shareholder and stakeholder value to facilitate the decision maker's search for an alternative that represents the best balance of competing objectives. By providing techniques for decomposing a trade decision into logical segments and then synthesizing the parts into a coherent whole, a decision management process allows the decision maker to work within human cognitive limits without oversimplifying the problem. Furthermore, by decomposing the overall decision problem, experts can provide assessments of alternatives in their area of expertise.

## Decision Management Process

The decision analysis process is depicted in Figure 1 below. The decision management process is based on several best practices, including:

- Utilizing sound mathematical technique of decision analysis for trade studies. Parnell (2009) provided a list of decision analysis concepts and techniques.
- Developing one master decision model, followed by its refinement, update, and use, as required for trade studies throughout the system life cycle.
- Using Value-Focused Thinking (Keeney 1992) to create better alternatives.
- Identifying uncertainty and assessing risks for each decision.

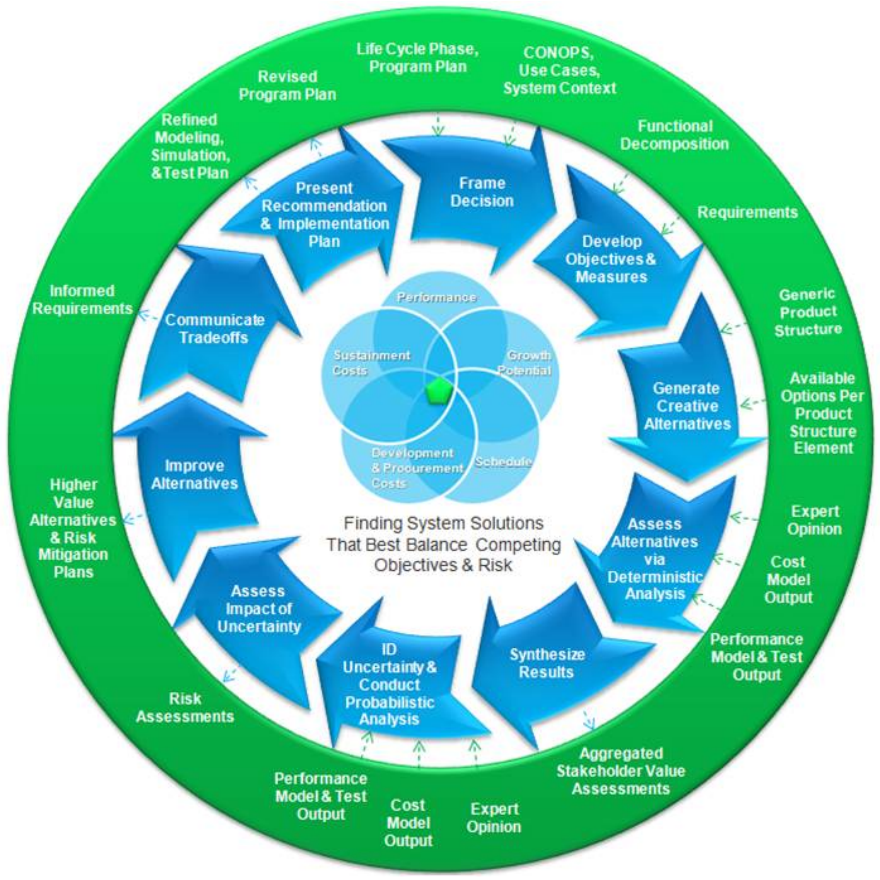

Life Cycle Phase, Program Plan
CONOPS, Use Cases, System Context
Functional Decomposition
Requirements
Generic Product Structure
Available Options Per Product Structure Element
Expert Opinion
Cost Model Output
Performance Model & Test Output
Aggregated Stakeholder Value Assessments
Expert Opinion
Cost Model Output
Performance Model & Test Output
Risk Assessments
Higher Value Alternatives & Risk Mitigation Plans
Informed Requirements
Refined Modeling, Simulation, & Test Plan
Revised Program Plan
Frame Decision
Develop Objectives & Measures
Generate Creative Alternatives
Assess Alternatives via Deterministic Analysis
Synthesize Results
ID Uncertainty & Conduct Probabilistic Analysis
Assess Impact of Uncertainty
Improve Alternatives
Communicate Tradeoffs
Present Recommendation & Implementation Plan
Performance
Growth Potential
Schedule
Development & Procurement Costs
Sustainment Costs
Finding System Solutions That Best Balance Competing Objectives & Risk

**Figure 1. Decision Management Process (INCOSE DAWG 2013).** Permission granted by Matthew Cilli who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

The center of the diagram shows the five trade space objectives (listed clockwise): Performance, Growth Potential, Schedule, Development & Procurement Costs, and Sustainment Costs . The ten blue arrows represent the decision management process activities and the white text within the green ring represents SE process elements. Interactions are represented by the small, dotted green or blue arrows. The decision analysis process is an iterative process. A hypothetical UAV decision problem is used to illustrate each of the activities in the following sections.

## Framing and Tailoring the Decision

To ensure the decision team fully understands the decision context, the analyst should describe the system baseline, boundaries and interfaces. The decision context includes: the system definition, the life cycle stage, decision milestones, a list of decision makers and stakeholders, and available resources. The best practice is to identify a decision problem statement that defines the decision in terms of the system life cycle.

## Developing Objectives and Measures

Defining how an important decision will be made is difficult. As Keeney (2002) puts it:

> *Most important decisions involve multiple objectives, and usually with multiple-objective decisions, you can't have it all. You will have to accept less achievement in terms of some objectives in order to achieve more on other objectives. But how much less would you accept to achieve how much more?*

The first step is to develop objectives and measures using interviews and focus groups with subject matter experts (SMEs) and stakeholders. For systems engineering trade-off analyses, stakeholder value often includes competing objectives of performance, development schedule, unit cost, support costs, and growth potential. For corporate decisions, shareholder value would also be added to this list. For performance, a functional decomposition can help generate a thorough set of potential objectives. Test this initial list of fundamental objectives by checking that each fundamental objective is essential and controllable and that the set of objectives is complete, non-redundant, concise, specific, and understandable (Edwards et al. 2007). Figure 2 provides an example of an objectives hierarchy.

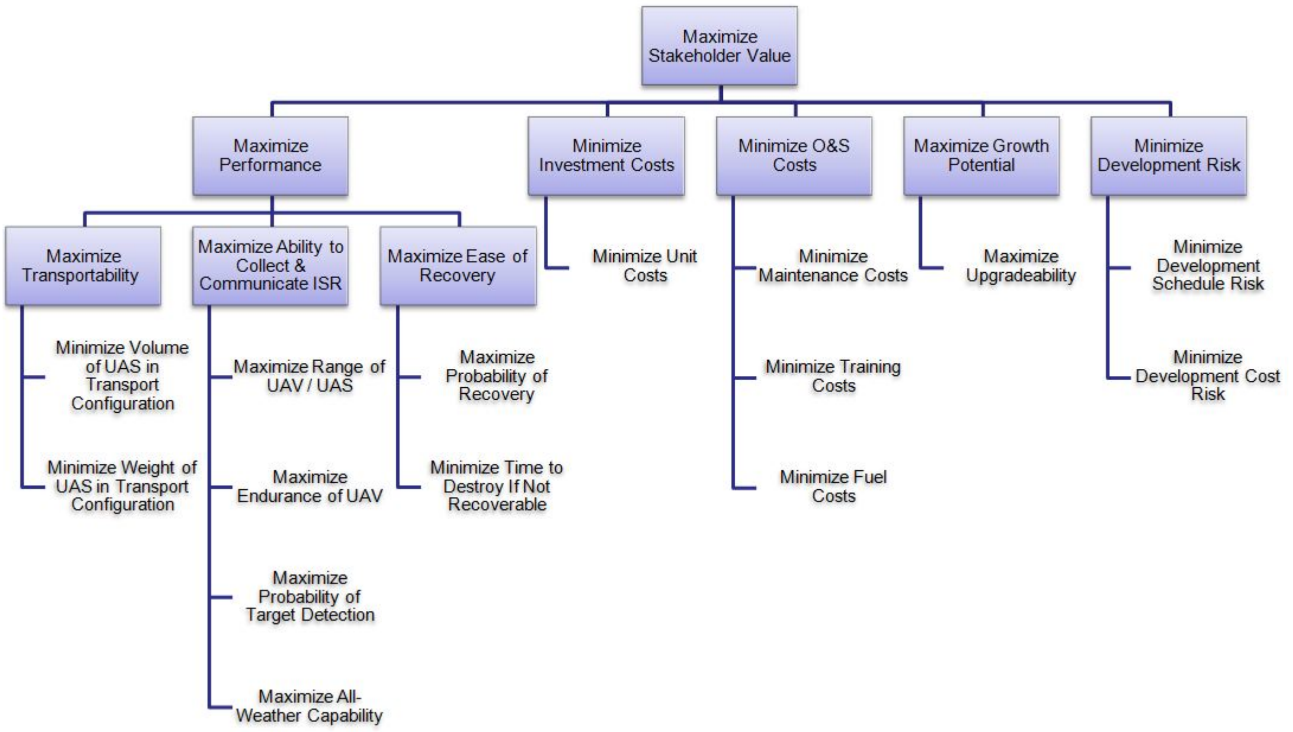

Maximize Stakeholder Value
Maximize Performance
Minimize Investment Costs
Minimize O&S Costs
Maximize Growth Potential
Minimize Development Risk
Maximize Transportability
Maximize Ability to Collect & Communicate ISR
Maximize Ease of Recovery
Minimize Unit Costs
Minimize Maintenance Costs
Maximize Upgradeability
Minimize Development Schedule Risk
Minimize Volume of UAS in Transport Configuration
Maximize Range of UAV / UAS
Maximize Probability of Recovery
Minimize Training Costs
Minimize Development Cost Risk
Minimize Weight of UAS in Transport Configuration
Maximize Endurance of UAV
Minimize Time to Destroy If Not Recoverable
Minimize Fuel Costs
Maximize Probability of Target Detection
Maximize All-Weather Capability

**Figure 2. Fundamental Objectives Hierarchy (INCOSE DAWG 2013).** Permission granted by Matthew Cilli who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

For each objective, a measure must be defined to assess the value of each alternative for that objective. A measure (attribute, criterion, and metric) must be unambiguous, comprehensive, direct, operational, and understandable (Keeney & Gregory 2005). A defining feature of multi-objective decision analysis is the transformation from measure space to value space. This transformation is performed by a value function which shows returns to scale on the measure range. When creating a value function, the walk-away point on the measure scale (x-axis) must be ascertained and mapped to a 0 value on the value scale (y-axis). A walk-away point is the measure score where regardless of how well an alternative performs in other measures, the decision maker will walk away from the alternative. He or she does this through working with the user, finding the measure score beyond, at which point an alternative provides no additional value, and labeling it "stretch goal" (ideal) and then mapping it to 100 (or 1 and 10) on the value scale (y-axis). Figure 3 provides the most common value curve shapes. The rationale for the shape of the value functions should be documented for traceability and defensibility (Parnell et al. 2011).

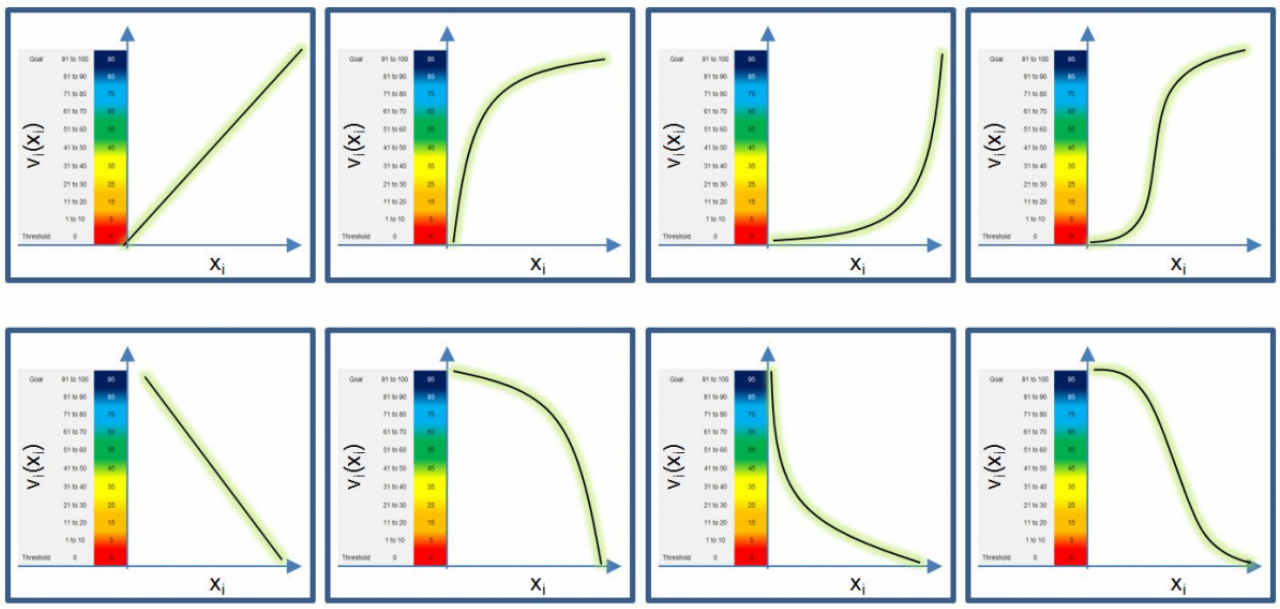
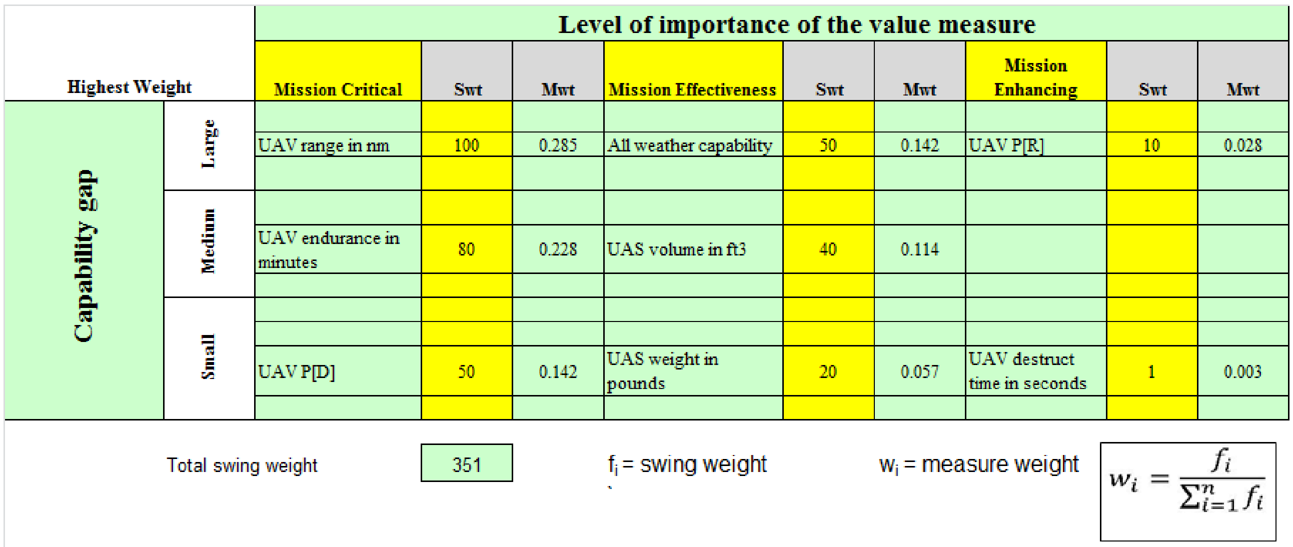


**Figure 3. Value Function Examples (INCOSE DAWG 2013).** Permission granted by Matthew Cilli who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

The mathematics of multiple objective decision analysis (MODA) requires that the weights depend on importance of the measure and the range of the measure (walk away to stretch goal). A useful tool for determining priority weighting is the swing weight matrix (Parnell et al. 2011). For each measure, consider its importance through determining whether the measure corresponds to a defining, critical, or enabling function and consider the gap between the current capability and the desired capability; finally, put the name of the measure in the appropriate cell of the matrix (Figure 4). The highest priority weighting is placed in the upper-left corner and assigned an unnormalized weight of 100. The unnormalized weights are monotonically decreasing to the right and down the matrix. Swing weights are then assessed by comparing them to the most important value measure or another assessed measure. The swing weights are normalized to sum to one for the additive value model used to calculate value in a subsequent section.



| | | Level of importance of the value measure | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Highest Weight | | Mission Critical | Swt | Mwt | Mission Effectiveness | Swt | Mwt | Mission Enhancing | Swt | Mwt |
| Capability gap | Large | UAV range in nm | 100 | 0.285 | All weather capability | 50 | 0.142 | UAV P[R] | 10 | 0.028 |
| | Medium | UAV endurance in minutes | 80 | 0.228 | UAS volume in ft3 | 40 | 0.114 | | | |
| | Small | UAV P[D] | 50 | 0.142 | UAS weight in pounds | 20 | 0.057 | UAV destruct time in seconds | 1 | 0.003 |

Total swing weight 351

$f_i$ = swing weight

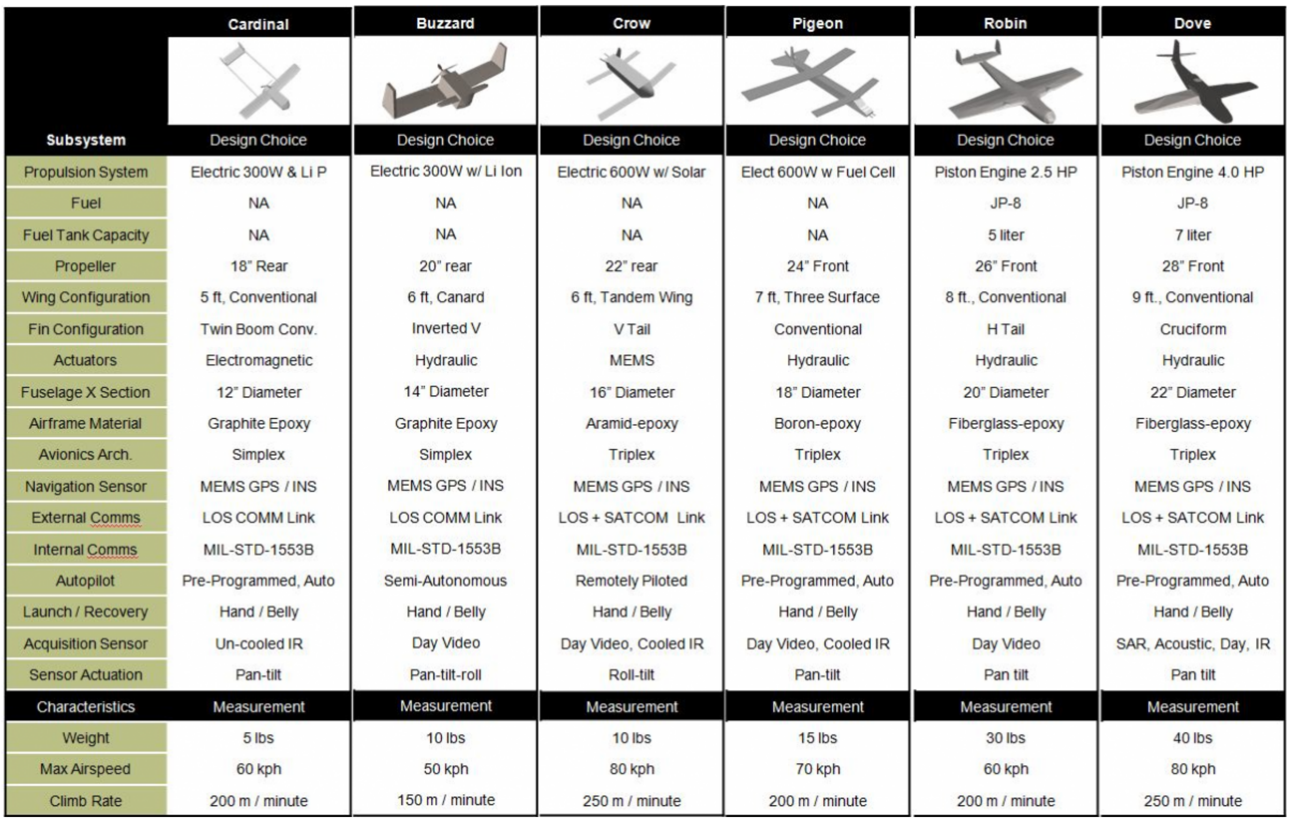
$w_i$ = measure weight

$$w_i = \frac{f_i}{\sum_{i=1}^{n} f_i}$$

**Figure 4. Swing Weight Matrix (INCOSE DAWG 2013).** Permission granted by Gregory Parnell who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

## Generating Creative Alternatives

To help generate a creative and comprehensive set of alternatives that span the decision space, consider developing an alternative generation table (also called a morphological box) (Buede, 2009; Parnell et al. 2011). It is a best practice to establish a meaningful product structure for the system and to be reported in all decision presentations (Figure 5).

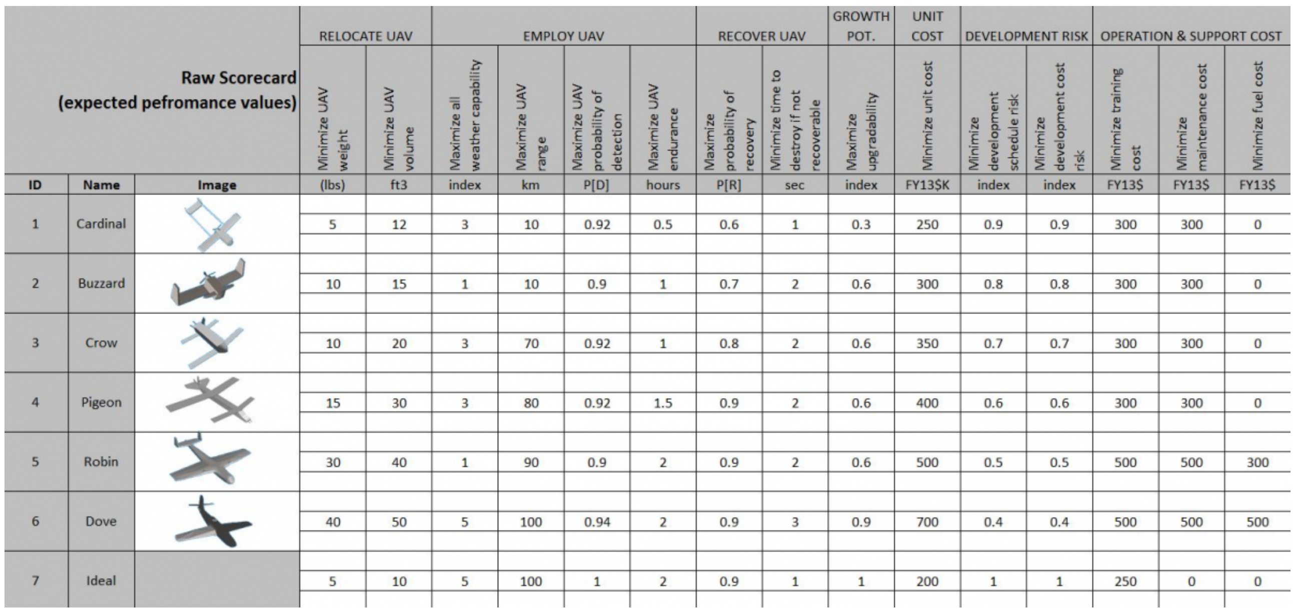


| | Cardinal | Buzzard | Crow | Pigeon | Robin | Dove |
|---|---|---|---|---|---|---|
| **Subsystem** | Design Choice | Design Choice | Design Choice | Design Choice | Design Choice | Design Choice |
| Propulsion System | Electric 300W & Li P | Electric 300W w/ Li Ion | Electric 600W w/ Solar | Elect 600W w Fuel Cell | Piston Engine 2.5 HP | Piston Engine 4.0 HP |
| Fuel | NA | NA | NA | NA | JP-8 | JP-8 |
| Fuel Tank Capacity | NA | NA | NA | NA | 5 liter | 7 liter |
| Propeller | 18" Rear | 20" rear | 22" rear | 24" Front | 26" Front | 28" Front |
| Wing Configuration | 5 ft, Conventional | 6 ft, Canard | 6 ft, Tandem Wing | 7 ft, Three Surface | 8 ft., Conventional | 9 ft., Conventional |
| Fin Configuration | Twin Boom Conv. | Inverted V | V Tail | Conventional | H Tail | Cruciform |
| Actuators | Electromagnetic | Hydraulic | MEMS | Hydraulic | Hydraulic | Hydraulic |
| Fuselage X Section | 12" Diameter | 14" Diameter | 16" Diameter | 18" Diameter | 20" Diameter | 22" Diameter |
| Airframe Material | Graphite Epoxy | Graphite Epoxy | Aramid-epoxy | Boron-epoxy | Fiberglass-epoxy | Fiberglass-epoxy |
| Avionics Arch. | Simplex | Simplex | Triplex | Triplex | Triplex | Triplex |
| Navigation Sensor | MEMS GPS / INS | MEMS GPS / INS | MEMS GPS / INS | MEMS GPS / INS | MEMS GPS / INS | MEMS GPS / INS |
| External Comms | LOS COMM Link | LOS COMM Link | LOS + SATCOM Link | LOS + SATCOM Link | LOS + SATCOM Link | LOS + SATCOM Link |
| Internal Comms | MIL-STD-1553B | MIL-STD-1553B | MIL-STD-1553B | MIL-STD-1553B | MIL-STD-1553B | MIL-STD-1553B |
| Autopilot | Pre-Programmed, Auto | Semi-Autonomous | Remotely Piloted | Pre-Programmed, Auto | Pre-Programmed, Auto | Pre-Programmed, Auto |
| Launch / Recovery | Hand / Belly | Hand / Belly | Hand / Belly | Hand / Belly | Hand / Belly | Hand / Belly |
| Acquisition Sensor | Un-cooled IR | Day Video | Day Video, Cooled IR | Day Video, Cooled IR | Day Video | SAR, Acoustic, Day, IR |
| Sensor Actuation | Pan-tilt | Pan-tilt-roll | Roll-tilt | Pan-tilt | Pan tilt | Pan tilt |
| **Characteristics** | Measurement | Measurement | Measurement | Measurement | Measurement | Measurement |
| Weight | 5 lbs | 10 lbs | 10 lbs | 15 lbs | 30 lbs | 40 lbs |
| Max Airspeed | 60 kph | 50 kph | 80 kph | 70 kph | 60 kph | 80 kph |
| Climb Rate | 200 m / minute | 150 m / minute | 250 m / minute | 200 m / minute | 200 m / minute | 250 m / minute |

**Figure 5. Descriptions of Alternatives (INCOSE DAWG 2013).** Permission granted by Matthew Cilli who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

## Assessing Alternatives via Deterministic Analysis

With objectives and measures established and alternatives having been defined, the decision team should engage SMEs, equipped with operational data, test data, simulations, models, and expert knowledge. Scores are best captured on scoring sheets for each alternative/measure combination which document the source and rationale. Figure 6 provides a summary of the scores.



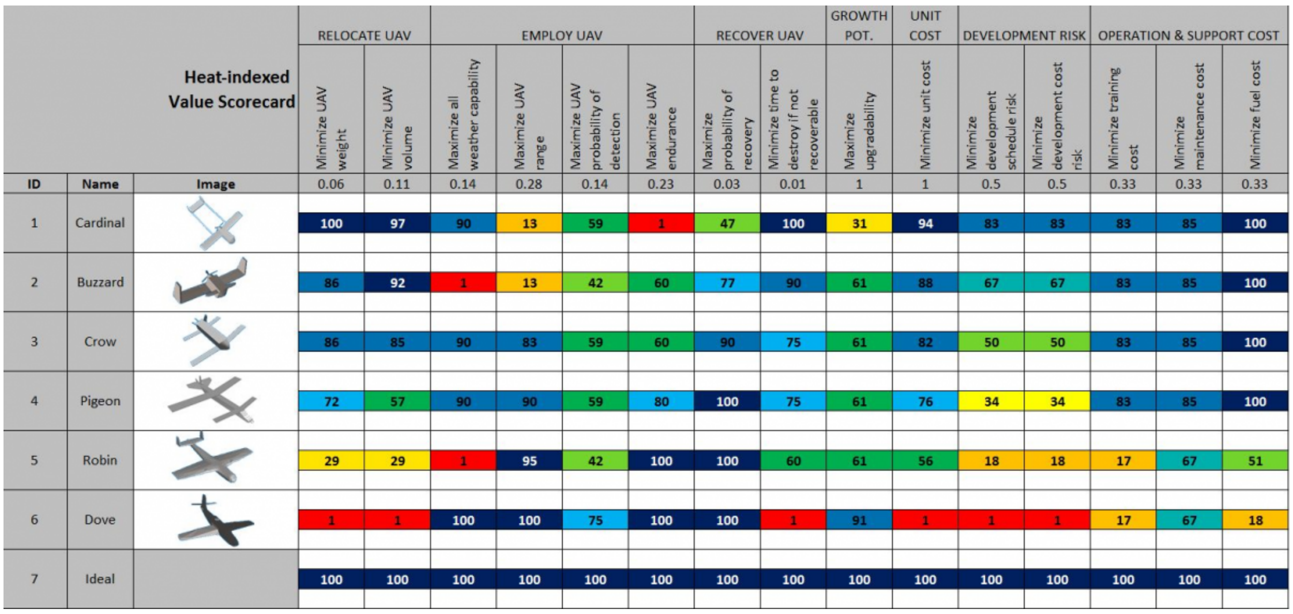
| Raw Scorecard (expected pefromance values) | | | RELOCATE UAV | | EMPLOY UAV | | | | RECOVER UAV | | GROWTH POT. | UNIT COST | DEVELOPMENT RISK | | OPERATION & SUPPORT COST | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Minimize UAV weight | Minimize UAV volume | Maximize all weather capability | Maximize UAV range | Maximize UAV probability of detection | Maximize UAV endurance | Maximize probability of recovery | Minimize time to destroy if not recoverable | Maximize upgradability | Minimize unit cost | Minimize development schedule risk | Minimize development cost risk | Minimize training cost | Minimize maintenance cost | Minimize fuel cost |
| ID | Name | Image | (lbs) | ft3 | index | km | P[D] | hours | P[R] | sec | index | FY13$K | index | index | FY13$ | FY13$ | FY13$ |
| 1 | Cardinal | | 5 | 12 | 3 | 10 | 0.92 | 0.5 | 0.6 | 1 | 0.3 | 250 | 0.9 | 0.9 | 300 | 300 | 0 |
| 2 | Buzzard | | 10 | 15 | 1 | 10 | 0.9 | 1 | 0.7 | 2 | 0.6 | 300 | 0.8 | 0.8 | 300 | 300 | 0 |
| 3 | Crow | | 10 | 20 | 3 | 70 | 0.92 | 1 | 0.8 | 2 | 0.6 | 350 | 0.7 | 0.7 | 300 | 300 | 0 |
| 4 | Pigeon | | 15 | 30 | 3 | 80 | 0.92 | 1.5 | 0.9 | 2 | 0.6 | 400 | 0.6 | 0.6 | 300 | 300 | 0 |
| 5 | Robin | | 30 | 40 | 1 | 90 | 0.9 | 2 | 0.9 | 2 | 0.6 | 500 | 0.5 | 0.5 | 500 | 500 | 300 |
| 6 | Dove | | 40 | 50 | 5 | 100 | 0.94 | 2 | 0.9 | 3 | 0.9 | 700 | 0.4 | 0.4 | 500 | 500 | 500 |
| 7 | Ideal | | 5 | 10 | 5 | 100 | 1 | 2 | 0.9 | 1 | 1 | 200 | 1 | 1 | 250 | 0 | 0 |

**Figure 6. Alternative Scores (INCOSE DAWG 2013).** Permission granted by Richard Swanson who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

Note that in addition to identified alternatives, the score matrix includes a row for the ideal alternative. The ideal is a tool for value-focused thinking, which will be covered later.

## Synthesizing Results

Next, one can transform the scores into a value table, by using the value functions developed previously. A color heat map can be useful to visualize value tradeoffs between alternatives and identify where alternatives need improvement (Figure 7).



| | | Heat-indexed Value Scorecard | RELOCATE UAV | | EMPLOY UAV | | | | RECOVER UAV | | GROWTH POT. | UNIT COST | DEVELOPMENT RISK | | OPERATION & SUPPORT COST | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Minimize UAV weight | Minimize UAV volume | Maximize all weather capability | Maximize UAV range | Maximize UAV probability of detection | Maximize UAV endurance | Maximize probability of recovery | Minimize time to destroy if not recoverable | Maximize upgradability | Minimize unit cost | Minimize development schedule risk | Minimize development cost risk | Minimize training cost | Minimize maintenance cost | Minimize fuel cost |
| ID | Name | Image | 0.06 | 0.11 | 0.14 | 0.28 | 0.14 | 0.23 | 0.03 | 0.01 | 1 | 1 | 0.5 | 0.5 | 0.33 | 0.33 | 0.33 |
| 1 | Cardinal | | 100 | 97 | 90 | 13 | 59 | 1 | 47 | 100 | 31 | 94 | 83 | 83 | 83 | 85 | 100 |
| 2 | Buzzard | | 86 | 92 | 1 | 13 | 42 | 60 | 77 | 90 | 61 | 88 | 67 | 67 | 83 | 85 | 100 |
| 3 | Crow | | 86 | 85 | 90 | 83 | 59 | 60 | 90 | 75 | 61 | 82 | 50 | 50 | 83 | 85 | 100 |
| 4 | Pigeon | | 72 | 57 | 90 | 90 | 59 | 80 | 100 | 75 | 61 | 76 | 34 | 34 | 83 | 85 | 100 |
| 5 | Robin | | 29 | 29 | 1 | 95 | 42 | 100 | 100 | 60 | 61 | 56 | 18 | 18 | 17 | 67 | 51 |
| 6 | Dove | | 1 | 1 | 100 | 100 | 75 | 100 | 100 | 1 | 91 | 1 | 1 | 1 | 17 | 67 | 18 |
| 7 | Ideal | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |

**Figure 7. Value Scorecard with Heat Map (INCOSE DAWG 2013).** Permission granted by Richard Swanson who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

The additive value model uses the following equation to calculate each alternative's value:

$$v(x) = \sum_{i=1}^{n} w_i v_i(x_i)$$

where

$v(x)$ is the alternative's value

$i = 1\ to\ n$ is the number of the measure

$x_i$ is the alternative's score on the $i^{th}$ measure

$v_i(x_i)$ is the single dimensional value of a score of $x_i$

$w_i$ is the weight of the $i^{th}$ measure

and $\sum_{i=1}^{n} w_i = 1$ (all weights sum to one)

The value component chart (Figure 8) shows the total value and the weighted value measure contribution of each alternative (Parnell et al. 2011).

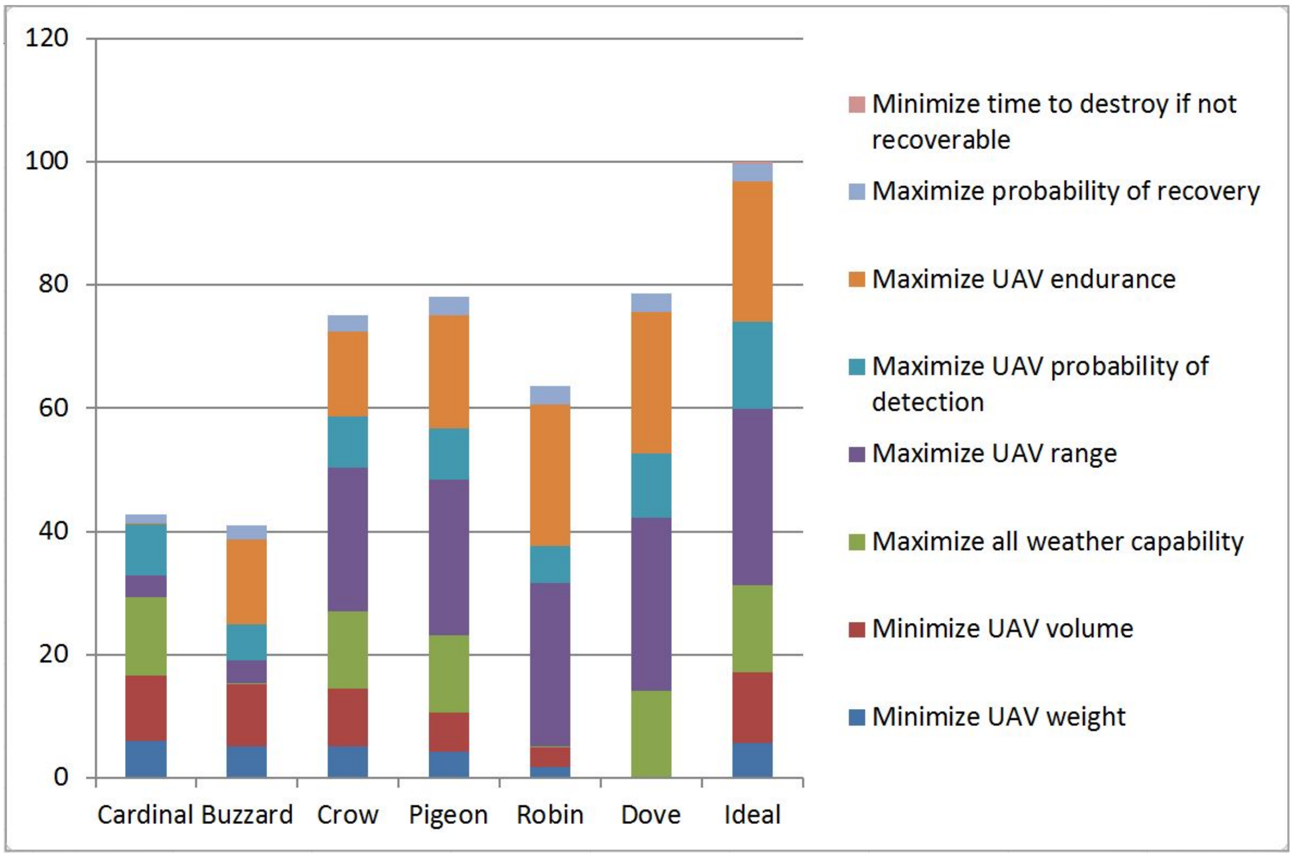


**Figure 8. Value Component Graph (INCOSE DAWG 2013).** Permission granted by Richard Swanson who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

The heart of a decision management process for system engineering trade off analysis is the ability to assess all dimensions of shareholder and stakeholder value. The stakeholder value scatter plot in Figure 9 shows five dimensions: unit cost, performance, development risk, growth potential, and operation and support costs for all alternatives.

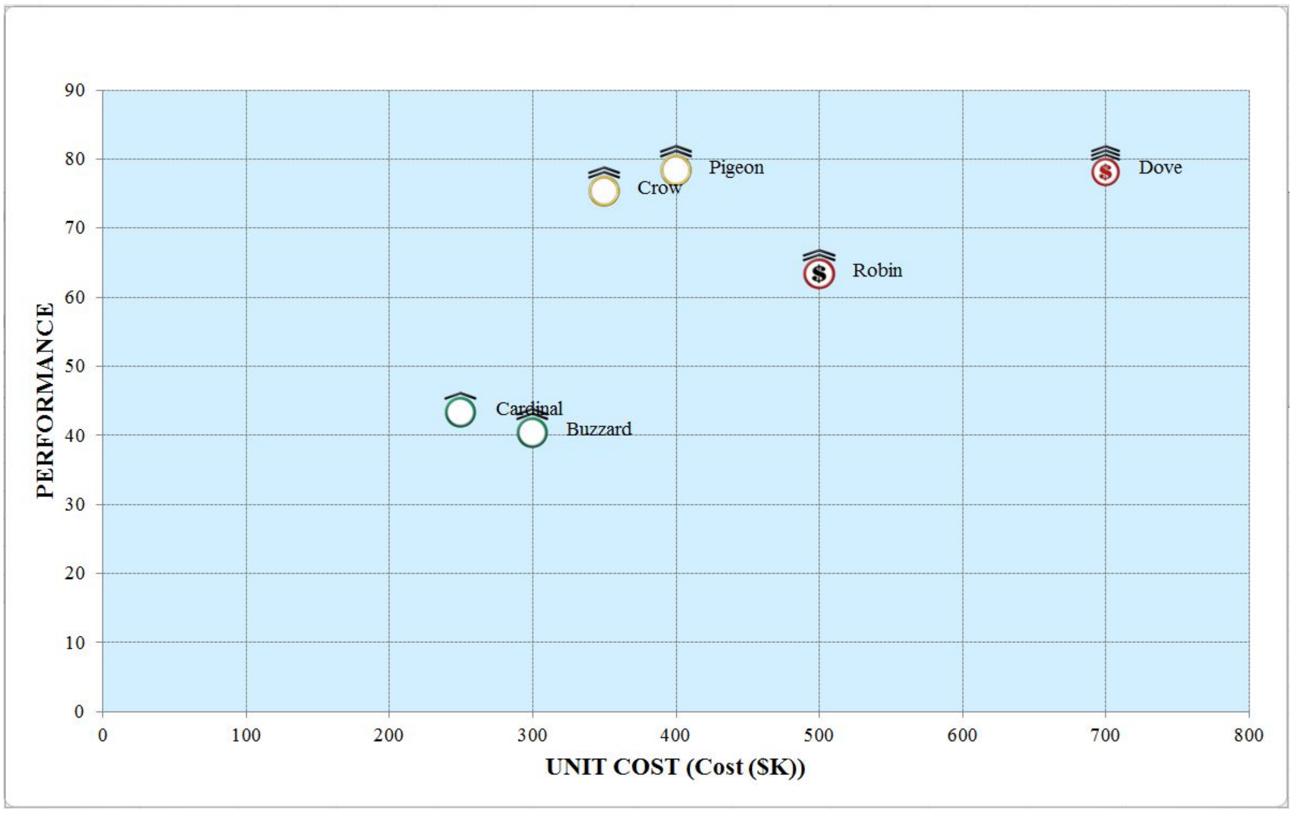


**Figure 9. Example of a Stakeholder Value Scatterplot (INCOSE DAWG 2013).** Permission granted by Richard Swanson who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

Each system alternative is represented by a scatter plot marker (Figure 9). An alternative's unit cost and performance value are indicated by x and y positions respectively. An alternative's development risk is indicated by the color of the marker (green = low, yellow= medium, red = high), while the growth potential is shown as the number of hats above the circular marker (1 hat = low, 2 hats = moderate, 3 hats = high).

## Identifying Uncertainty and Conducting Probabilistic Analysis

As part of the assessment, the SME should discuss the potential uncertainty of the independent variables. The independent variables are the variables that impact one or more scores; the scores that are independent scores. Many times the SME can assess an upper, nominal, and lower bound by assuming low, moderate, and high performance. Using this data, a Monte Carlo Simulation summarizes the impact of the uncertainties and can identify the uncertainties that have the most impact on the decision.

## Accessing Impact of Uncertainty - Analyzing Risk and Sensitivity

Decision analysis uses many forms of sensitivity analysis including line diagrams, tornado diagrams, waterfall diagrams and several uncertainty analyses including Monte Carlo Simulation, decision trees, and influence diagrams (Parnell et al. 2013). A line diagram is used to show the sensitivity to the swing weight judgment (Parnell et al. 2011). Figure 10 shows the results of a Monte Carlo Simulation of performance value.

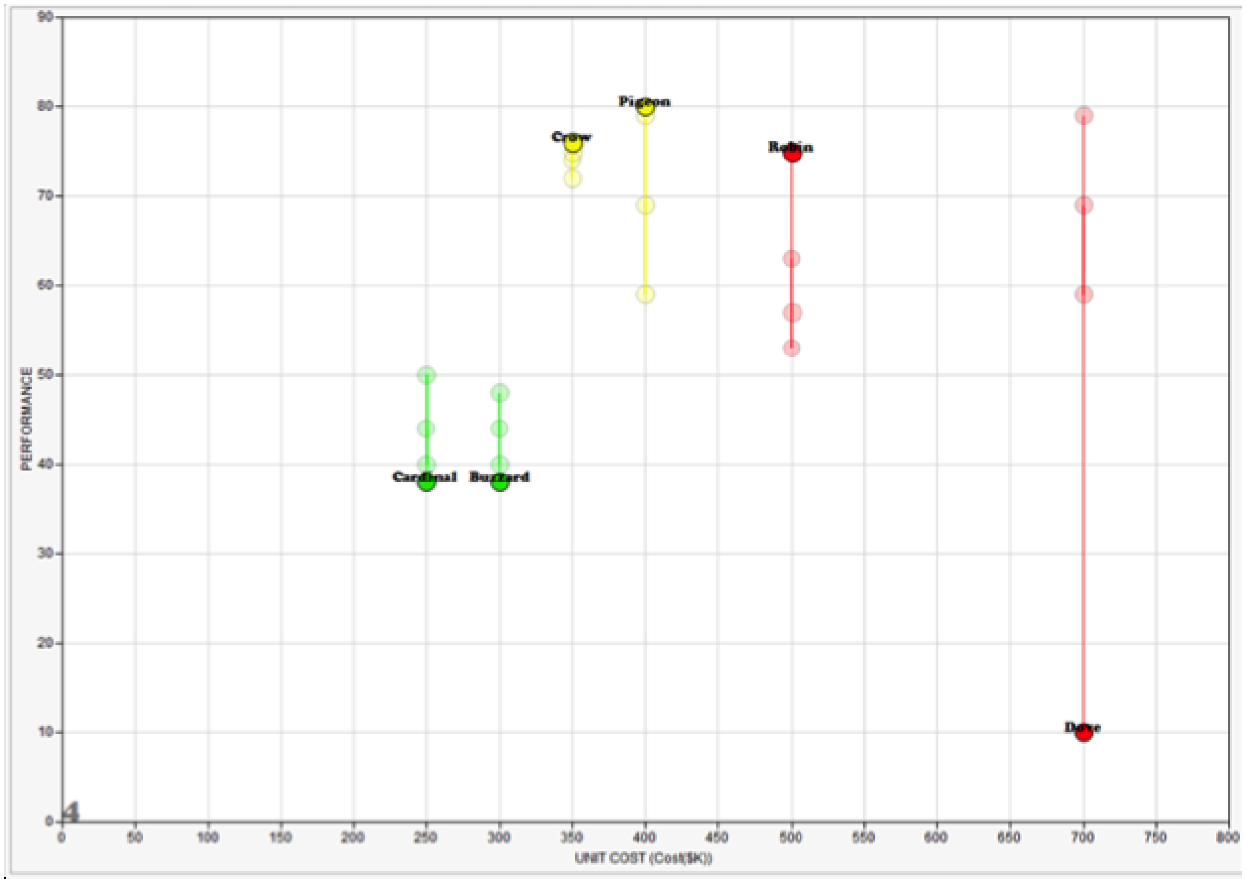


**Figure 10. Uncertainty on Performance Value from Monte Carlo Simulation (INCOSE DAWG 2013).** Permission granted by Matthew Cilli who prepared image for the INCOSE Decision Analysis Working Group (DAWG). All other rights are reserved by the copyright owner.

## Improving Alternatives

Mining the data generated for the alternatives will likely reveal opportunities to modify some design choices to claim untapped value and/or reduce risk. Taking advantage of initial findings to generate new and creative alternatives starts the process of transforming the decision process from "alternative-focused thinking" to "value-focused thinking" (Keeney 1993).

## Communicating Tradeoffs

This is the point in the process where the decision analysis team identifies key observations about tradeoffs and the important uncertainties and risks.

### Presenting Recommendations and Implementing Action Plan

It is often helpful to describe the recommendation(s) in the form of a clearly-worded, actionable task-list in order to increase the likelihood of the decision implementation. Reports are important for historical traceability and future decisions. Take the time and effort to create a comprehensive, high-quality report detailing study findings and supporting rationale. Consider static paper reports augmented with dynamic hyper-linked e-reports.

## References

### Works Cited

Buede, D.M. 2009. *The engineering design of systems: Models and methods*. 2nd ed. Hoboken, NJ: John Wiley & Sons Inc.

Edwards, W., R.F. Miles Jr., and D. Von Winterfeldt. 2007. *Advances In Decision Analysis: From Foundations to Applications.* New York, NY: Cambridge University Press.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Keeney, R.L. and H. Raiffa H. 1976. *Decisions with Multiple Objectives - Preferences and Value Tradeoffs.* New York, NY: Wiley.

Keeney, R.L. 1992. *Value-Focused Thinking: A Path to Creative Decision-Making.* Cambridge, MA: Harvard University Press.

Keeney, R.L. 1993. "Creativity in MS/OR: Value-focused thinking—Creativity directed toward decision making." *Interfaces*, 23(3), p.62–67.

Parnell, G.S. 2009. "Decision Analysis in One Chart," *Decision Line, Newsletter of the Decision Sciences Institute*. May 2009.

Parnell, G.S., P.J. Driscoll, and D.L Henderson (eds). 2011. *Decision Making for Systems Engineering and Management*, 2nd ed. Wiley Series in Systems Engineering. Hoboken, NJ: Wiley & Sons Inc.

Parnell, G.S., T. Bresnick, S. Tani, and E. Johnson. 2013. *Handbook of Decision Analysis.* Hoboken, NJ: Wiley & Sons.

### Primary References

Buede, D.M. 2004. "On Trade Studies." Proceedings of the 14th Annual International Council on Systems Engineering International Symposium, 20-24 June, 2004, Toulouse, France.

Keeney, R.L. 2004. "Making Better Decision Makers." *Decision Analysis*, 1(4), pp.193–204.

Keeney, R.L. & R.S. Gregory. 2005. "Selecting Attributes to Measure the Achievement of Objectives". *Operations Research*, 53(1), pp.1–11.

Kirkwood, C.W. 1996. *Strategic Decision Making: Multiobjective Decision Analysis with Spreadsheets.* Belmont, California: Duxbury Press.

## Additional References

Buede, D.M. and R.W. Choisser. 1992. "Providing an Analytic Structure for Key System Design Choices." *Journal of Multi-Criteria Decision Analysis*, 1(1), pp.17–27.

Felix, A. 2004. "Standard Approach to Trade Studies." Proceedings of the International Council on Systems Engineering (INCOSE) Mid-Atlantic Regional Conference, November 2-4 2004, Arlington, VA.

Felix, A. 2005. "How the Pro-Active Program (Project) Manager Uses a Systems Engineer's Trade Study as a Management Tool, and not just a Decision Making Process." Proceedings of the International Council on Systems Engineering (INCOSE) International Symposium, July 10-15, 2005, Rochester, NY.

Miller, G.A. 1956. "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information." *Psychological Review*, 63(2), p.81.

Ross, A.M. and D.E. Hastings. 2005. "Tradespace Exploration Paradigm." Proceedings of the International Council on Systems Engineering (INCOSE) International Symposium, July 10-15, 2005, Rochester, NY.

Sproles, N. 2002. "Formulating Measures of Effectiveness." *Systems Engineering", 5(4), p. 253-263.*

Silletto, H. 2005. "Some Really Useful Principles: A new look at the scope and boundaries of systems engineering." Proceedings of the International Council on Systems Engineering (INCOSE) International Symposium, July 10-15, 2005, Rochester, NY.

Ullman, D.G. and B.P. Spiegel. 2006. "Trade Studies with Uncertain Information." Proceedings of the International Council on Systems Engineering (INCOSE) International Symposium, July 9-13, 2006, Orlando, FL.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Configuration Management

***Lead Authors:*** *Ray Madachy, John Snoderly, Garry Roedler*, ***Contributing Author:*** *Rick Adcock*

The purpose of configuration management (CM) is to establish and maintain the integrity of all of the identified outputs of a project or process and make them available to concerned parties (ISO/IEC/IEEE 2015). Unmanaged changes to system artifacts (such as those associated with plans, requirements, design, software, hardware, testing, and documentation) can lead to problems that persist throughout the system life cycle. Hence, one primary objective of CM is to manage and control the change to such artifacts.

## Configuration Management Process Overview

CM is the discipline of identifying and formalizing the functional and physical characteristics of a configuration item at discrete points in the product evolution for the purpose of maintaining the integrity of the product system and controlling changes to the baseline. The baseline for a project contains all of the technical requirements and related cost and schedule requirements that are sufficiently mature to be accepted and placed under change control by the project manager. The project baseline consists of two parts: the technical baseline and the business baseline. The systems engineer is responsible for managing the technical baseline and ensuring that it is consistent with the costs and schedules in the business baseline. Typically, the project control office manages the business baseline.

The ANSI/GEIA EIA-649-A standard presents CM from the viewpoint that configuration management practices are employed because they make good business sense rather than because requirements are imposed by an external customer (ANSI/GEIA 2005). The standard discusses CM principles and practices from an enterprise view; it does not prescribe which CM activities individual organizations or teams within the enterprise should perform. Each enterprise assigns responsibilities in accordance with its own management policy. See also the Implementation Guide for Configuration Management, which supports and provides further information on this standard (ANSI/GEIA October 2005).

Effective CM depends on the establishment, maintenance, and implementation of an effective process. The CM process should include, but is not limited to, the following activities:

- identification and involvement of relevant stakeholders
- setting of CM goals and expected outcomes
- identification and description of CM tasks
- assignment of responsibility and authority for performing the CM process tasks
- establishment of procedures for monitoring and control of the CM process
- measurement and assessment of the CM process effectiveness

As a minimum the CM process should incorporate and detail the following tasks (SEI 2010):

- identifying the configuration of selected work products that compose the baselines at given points in time
- controlling changes to configuration items
- building or providing specifications to build work products from the configuration management system
- maintaining the integrity of baselines
- providing accurate status and current configuration data to developers, end users, and customers

Figure 1 below shows the primary functions of systems CM.

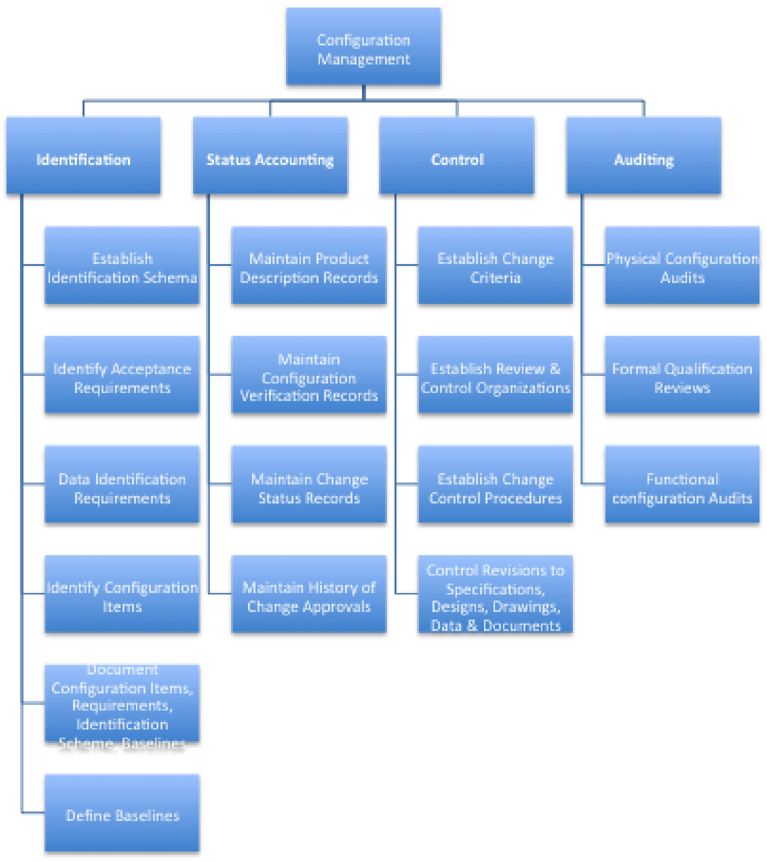

Configuration Management
Identification
Status Accounting
Control
Auditing
Establish Identification Schema
Identify Acceptance Requirements
Data Identification Requirements
Identify Configuration Items
Document Configuration Items, Requirements, Identification Scheme, Baselines
Define Baselines
Maintain Product Description Records
Maintain Configuration Verification Records
Maintain Change Status Records
Maintain History of Change Approvals
Establish Change Criteria
Establish Review & Control Organizations
Establish Change Control Procedures
Control Revisions to Specifications, Designs, Drawings, Data & Documents
Physical Configuration Audits
Formal Qualification Reviews
Functional configuration Audits

**Figure 1. Configuration Management Functions.** (SEBoK Original)

### Planning

The CM plan must be developed in consideration of the organizational context and culture; it must adhere to or incorporate applicable policies, procedures, and standards and it must accommodate acquisition and subcontractor situations. A CM plan details and schedules the tasks to be performed as part of the CM process including: configuration identification, change control, configuration status accounting, configuration auditing, and release management and delivery.

### Configuration Identification

This activity is focused on identifying the configuration items which will be managed and controlled under a CM process. The identification activity involves establishing a procedure for labeling items and their versions. The labeling provides a context for each item within the system configuration and shows the relationship between system items.

### Establishing Baseline

Configuration items are typically assembled into a baseline which specifies how a system will be viewed for the purposes of management, control, and evaluation. This baseline is fixed at a specific point in time in the system life cycle and represents the current approved configuration. It generally can only be changed through formal change procedures.

## Change Control

A disciplined change control process is critical for systems engineering. A generalized change control process in response to an engineering change proposal (ECP) is shown in Figure 2 below, which is adapted from Systems Engineering and Analysis (Blanchard and Fabrycky 1999).

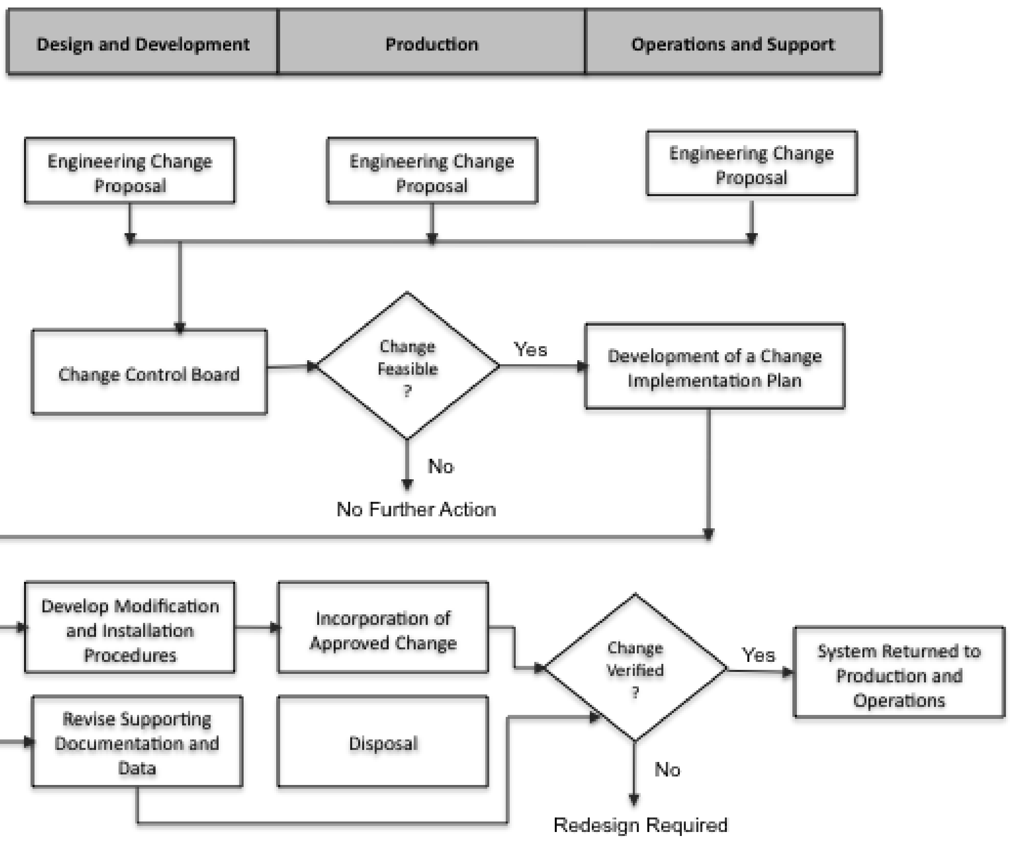

Design and Development
Production
Operations and Support
Engineering Change Proposal
Engineering Change Proposal
Engineering Change Proposal
Change Control Board
Change Feasible ?
Yes
No
No Further Action
Development of a Change Implementation Plan
Develop Modification and Installation Procedures
Incorporation of Approved Change
Revise Supporting Documentation and Data
Disposal
Change Verified ?
Yes
System Returned to Production and Operations
No
Redesign Required

**Figure 2. Configuration Change Control Process.** (SEBoK Original)

## Configuration Auditing

Audits are independent evaluations of the current stat us of configuration items and determine conformance of the configuration activities to the CM process. Adherence to applicable CM plans, regulations, and standards, is typically assessed during audits.

## Constraints and Guidance

Constraints affecting and guiding the CM process come from a number of sources. Policies, procedures, and standards set forth at corporate or other organizational levels might influence or constrain the design and implementation of the CM process. Also, the contract with an acquirer or supplier may contain provisions affecting the CM process. The system life cycle process adopted and the tools, methods, and other processes used in system development can affect the CM process (Bourque and Fairley 2014). There are a variety of sources for guidance on the development of a CM process. These include the ISO standards on system life cycle processes (ISO/IEC/IEEE 15288 2015) and configuration management guidelines (ISO 10007 2003), as well as the *Guide to The Software Engineering Body of Knowledge* (SWEBOK) (Bourque and Fairley 2014), and the CMMI for Development (SEI 2010).

### Organizational Issues

Successful CM planning, management, and implementation requires an understanding of the organizational context for the design and implementation of the CM process and why constraints are placed upon it. To plan a CM process for a project, it is necessary to understand the organizational context and the relationships among the organizational elements. CM interacts with other organizational elements, which may be structured in a number of ways. Although the responsibility for performing certain CM tasks might be assigned to other parts of the organization, the overall responsibility for CM often rests with a distinct organizational element or designated individual (Bourque and Fairley 2014).

### Measurement

In order to carry out certain CM functions, such as status accounting and auditing, as well as to monitor and assess the effectiveness of CM processes, it is necessary to measure and collect data related to CM activities and system artifacts. CM libraries and automated report tools provide convenient access and facilitation of data collection. Examples of metrics include the size of documentation artifacts, number of change requests, mean time to change to a configuration item, and rework costs.

### Tools

CM employs a variety of tools to support the process, for example:

- library management
- tracking and change management
- version management
- release management

The INCOSE Tools Database Working Group (INCOSE TDWG 2010) maintains an extensive list of tools including configuration management.

## Linkages to Other Systems Engineering Management Topics

Configuration management is involved in the management and control of artifacts produced and modified throughout the system life cycle in all areas of system definition, system realization, system deployment and use, and product and service life management. This includes CM application to the artifacts of all the other management processes (plans, analyses, reports, statuses, etc.).

## Practical Considerations

Key pitfalls and good practices related to systems engineering CM are described in the next two sections.

### Pitfalls

Some of the key pitfalls encountered in planning and performing CM are in Table 1.

**Table 1. Configuration Management Pitfalls. (SEBoK Original)**

| Name | Description |
|---|---|
| Shallow Visibility | • Not involving all affected disciplines in the change control process. |
| Poor Tailoring | • Inadequate CM tailoring to adapt to the project scale, number of subsystems, etc. |
| Limited CM Perspective | • Not considering and integrating the CM processes of all contributing organizations including COTS vendors and subcontractors. |

## Good Practices

Some good practices gathered from the references are provided in Table 2 below.

**Table 2. Configuration Management Good Practices. (SEBoK Original)**

| Name | Description |
|---|---|
| Cross-Functional CM | • Implement cross-functional communication and CM processes for software, hardware, firmware, data, or other types of items as appropriate. |
| Full Lifecycle Perspective | • Plan for integrated CM through the life cycle. Do not assume that it will just happen as part of the program. |
| CM Planning | • Processes are documented in a single, comprehensive CM plan early in the project. The plan should be a (systems) CM plan.<br>• Include tools selected and used. |
| Requirements Traceability | • Initiate requirements traceability at the start of the CM activity. |
| CCB Hierarchy | • Use a hierarchy of configuration control boards commensurate with the program elements. |
| Consistent Identification | • Software CI and hardware CI use consistent identification schemes. |
| CM Automation | • Configuration status accounting should be as automated as possible. |

Additional good practices can be found in ISO/IEC/IEEE (2009, Clause 6.4) and INCOSE (2010, sec. 5.4.1.5).

## References

### Works Cited

ANSI/GEIA. 2005. Implementation Guide for Configuration Management. Arlington, VA, USA: American National Standards Institute/Government Electronics & Information Technology Association, GEIA-HB-649. October 2005.

Blanchard, B.S. and W J. Fabrycky. 2005. *Systems Engineering and Analysis,* 4th ed. Prentice-hall international series in industrial and systems engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK).* Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org

ISO. 2003. *Quality Management Systems – Guidelines for Configuration Management*. Geneva, Switzerland: International Organization for Standardization (ISO), ISO 10007:2003.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering-- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2015

SEI. 2010. *Capability Maturity Model Integrated (CMMI) for Development,* version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

## Primary References

ANSI/GEIA. 2005. *Implementation Guide for Configuration Management*. Arlington, VA, USA: American National Standards Institute/Government Electronics & Information Technology Association, GEIA-HB-649. October 2005.

GEIA. 2004. *GEIA Consensus Standard for Data Management*. Arlington, VA, USA: Government Electronics & Information Technology Association, GEIA-859.

ISO. 2003. *Quality Management Systems – Guidelines for Configuration Management*. Geneva, Switzerland: International Organization for Standardization (ISO), ISO 10007:2003.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering-- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2015

SEI. 2010. *Capability Maturity Model Integrated (CMMI) for Development,* version 1.3. Pittsburgh, PA, USA: Software Engineering Institute (SEI)/Carnegie Mellon University (CMU).

## Additional References

INCOSE Tools database working group (TDWG). in International Council on Systems Engineering (INCOSE) [database online]. San Diego, CA, USA, 2010. Accessed April 13, 2015. Available at: http://www.incose.org/docs/default-source/wgcharters/tools-database.pdf.

INCOSE. 2008. "INCOSE measurement tools survey." in International Council on Systems Engineering (INCOSE) [database online]. San Diego, CA, USA, 2008.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - Life Cycle Processes - Project Management*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16326:2009(E).

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Information Management

***Lead Authors:*** *Andy Pickard, Garry Roedler*, ***Contributing Authors:*** *Ray Madachy*

The information management (IM) process is a set of activities associated with the collection and management of information from one or more sources and the distribution of that information to one or more audiences. Information, in its most restricted technical sense, is an ordered sequence of symbols that record or transmit a message. The key idea is that information is a collection of facts that is organized in such a way that they have additional value beyond the value of the facts themselves. The systems engineer is both the generator and recipient of information products; thus, the systems engineer has a vital stake in the success of the development and use of the IM process and IM systems.

## Overview

Information can exist in many forms in an organization; some information is related to a specific system development program and some is held at an enterprise level and made available to programs as required. It may be held in electronic format or in physical form (for instance, paper drawings or documents, microfiche or other photographic records).

The IM process includes a set of interrelated activities associated with information systems, systems of systems (SoS), architectures, services, nested hardware/platforms, and people. Fundamentally, this process is a set of activities that are concerned with improvements in a variety of human problem-solving endeavors. This includes the design, development, and use of technologically based systems and processes that enhance the efficiency and effectiveness of information and associated knowledge in a variety of strategic/business, tactical, and operational situations.

Management refers to the organization of and control over process activities associated with the structure, processing, and delivery of information. For example, the organizational structure must have management processes capable of managing this information throughout the information life cycle regardless of source or format (e.g., data, paper documents, electronic documents, audio, video, etc.) for delivery through multiple channels that may include cell phones and web interfaces.

A computer-based IM system is an organized combination of people, hardware, software, communication networks, and the data resources that collect, transform, and disseminate information in an organization. From the perspective of the systems engineer, the IM process is a cycle of inter-related information activities to be planned for, designed, and coordinated. Numerous life cycle development process models exist. Figure 1 below is a high-level process model that emphasizes the role of systems engineering (SE) in IM.

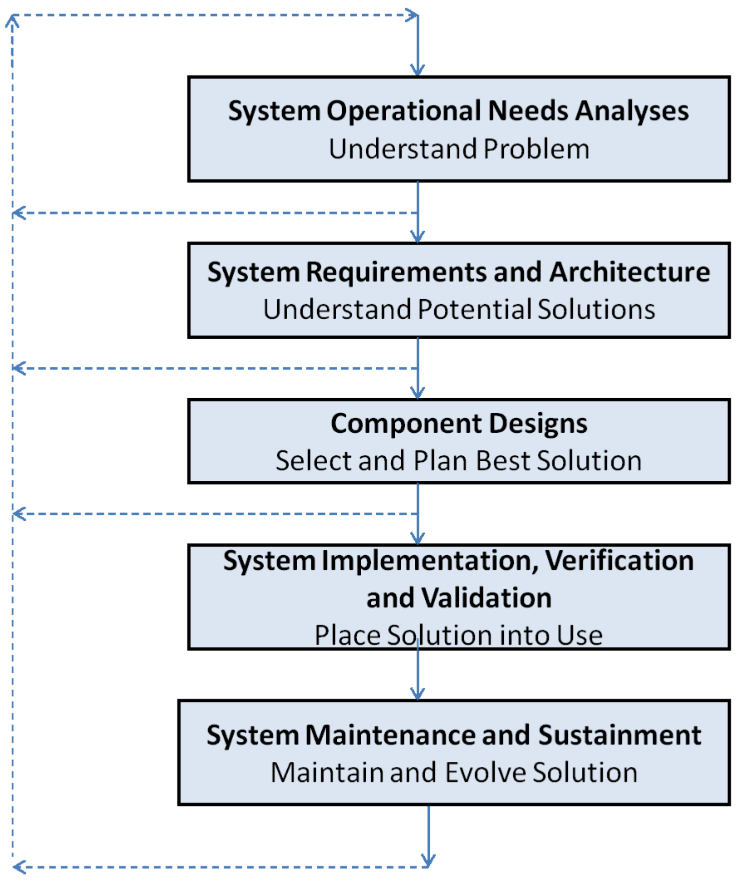

System Operational Needs Analyses
Understand Problem
System Requirements and Architecture
Understand Potential Solutions
Component Designs
Select and Plan Best Solution
System Implementation, Verification and Validation
Place Solution into Use
System Maintenance and Sustainment
Maintain and Evolve Solution

**Figure 1. Life Cycle Information Management Process Development Model.**
(SEBoK Original)

The SE function in the development of an IM system is concerned with several rate-limiting architecture and design variables, (e.g., information sharing, quality, security, efficiency, compliance, etc.) that should be considered up-front in the life cycle development process. Each of these variables can be subdivided into architecture and design considerations for the information system-of-interest (SoI). For example, quality can be viewed in terms of data validity, consistency, and comprehensiveness. Figure 2 provides an overview of information management considerations.

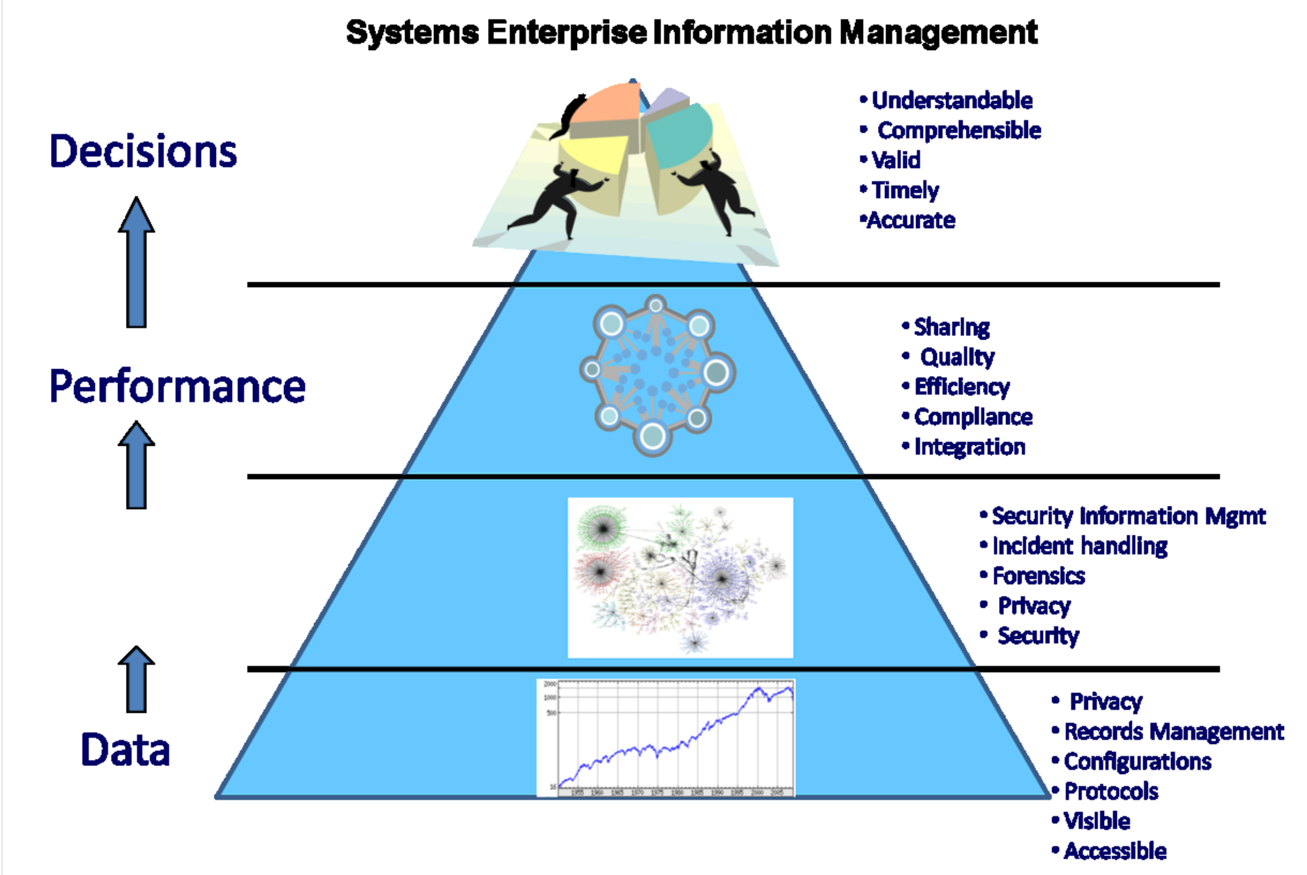


**Figure 2. Information Management Architecture and Design Considerations.** (SEBoK Original)

The effective and efficient employment of IM systems should solve business needs. These needs can center on several business objectives, such as efficiency, effectiveness, competitiveness, or profitability. From a business enterprise perspective, the systems engineer may be involved in several activities that support the development of IM systems, such as strategic planning, analyses of technology/business trends, development of applications, understanding operational disciplines, resource control techniques, and assessment of organization structures.

The IM process ensures that necessary information is created, stored, retained, protected, managed, and made easily available to those with a need and who are permitted access. It also ensures that information is disposed of when it is no longer relevant.

## The Information Management Process

To quote from ISO/IEC/IEEE 15288 (2015):

> *The purpose of the Information Management Process is to generate, obtain, confirm, transform, retain, retrieve, disseminate and dispose of information, to designated stakeholders…*
>
> *Information management plans, executes, and controls the provision of information to designated stakeholders that is unambiguous, complete, verifiable, consistent, modifiable, traceable, and presentable.*

The first step in the IM process is to plan IM. The output of this step is the IM strategy or plan. The second step is to perform IM. The outputs of this step are the creation, population and maintenance of one or more information repositories, together with the creation and dissemination of information management reports.

## Plan Information Management

Issues that should be considered when creating the IM strategy/plan include:

- Scope
  - What information has to be managed?
  - How long will the information need to be retained?
  - Is a system data dictionary is required to be able to "tag" information for ease of search and retrieval?
  - Will the media that will be used for the management of the information be physical, electronic, or both?
  - Have a work in progress (WIP) and formally released information already been considered when establishing data repositories?
- Constraints
  - What level of configuration control must be applied to the information?
  - Are there any regulatory requirements relating to the management of information for this project, including export control requirements?
  - Are there any customer requirements or agreements relating to the management of project information?
  - Are there any industry standards relating to the management of project information?
  - Are there any organization/enterprise directives, procedures, or standards relating to the management of project information?
  - Are there any project directives, procedures, or standards relating to the management of project information?
- Control/Security
  - Who is allowed access to the information? This could include people working on the project, other members of the organization/enterprise, customers, partners, suppliers and regulatory authorities.
  - Are there requirements to protect the information from unauthorized access? This could include intellectual property (IP) rights that have to be respected - for instance, if information from suppliers is to be stored and there is the possibility of a supplier gaining access to information belonging to a competitor who is also a supplier for the project.
  - What data repository or repositories are to be used?
    - Has the volume of information to be stored been considered when selecting repositories?
    - Has speed of access and search been considered when selecting repositories?
  - If electronic information is to be stored, what file formats are allowed?
  - Have requirements been defined to ensure that the information being stored is valid?
  - Have requirements been defined to ensure that information is disposed of correctly when it is no longer required to be stored, or when it is no longer valid? For instance, has a review period been defined for each piece of information?
- Life Cycle
  - If electronic information is to be stored for a long time, how will it be "future-proofed?" For instance, are neutral file formats available, or will copies of the software that created or used the information be retained?
  - Have disaster recovery requirements been considered – e.g., if a server holding electronic information is destroyed, are there back-up copies of the information? Are the back-up copies regularly accessed to show that information recovery is flawless?
  - Is there a formal requirement to archive designated information for compliance with legal (including regulatory), audit, and information retention requirements? If so, has an archive and archiving method been defined?
  - Some information may not be required to be stored (e.g., the results files for analyses when the information occupies a large volume and can be regenerated by the analysis tool and the input file). However, if the cost to re-generate the information is high, consider doing a cost/benefit analysis for storage versus regeneration.

### Perform Information Management

Issues that should be considered when performing information management include:

- Is the information valid (is it traceable to the information management strategy/plan and the list of information to be managed)?
- Has the workflow for review and approval of information been defined to transfer information from "work in progress" to "released?"
- Are the correct configuration management requirements being applied to the information? Has the information been baselined?
- Have the correct "tags" been applied to the information to allow for easy search and retrieval?
- Have the correct access rules been applied to the information? Can users access the information that they are permitted to access, and only this information?
- If required, has the information been translated into a neutral file format prior to storage?
- Has a review date been set for assessing the continued validity of the information?
- Has the workflow for review and removal of unwanted, invalid, or unverifiable information (as defined in organization/enterprise policy, project policy, security or intellectual property requirements) been defined?
- Has the information been backed up and has the backup recovery system been tested?
- Has designated information been archived in compliance with legal (including regulatory), audit, and information retention requirements?
- Does the IM system satisfy defined performance requirements - for instance, speed of access, availability, and searchability?

## Linkages to Other Systems Engineering Management Topics

The systems engineering IM process is closely coupled with the system definition, planning, and CM processes. The requirements for IM are elicited from stakeholders as part of the system definition process. What/when information is to be stored in the systems engineering lifecycle is defined in the planning process and configuration control requirements are defined in the CM process.

## Practical Considerations

Key pitfalls and good practices related to IM are described in the next two sections.

### Pitfalls

Some of the key pitfalls encountered in planning and performing IM are provided in Table 1:

**Table 1. Information Management Pitfalls. (SEBoK Original)**

| Pitfall Name | Pitfall Description |
|---|---|
| No Data Dictionary | • Not defining a data dictionary for the project may result in inconsistencies in naming conventions for information and proliferation of meta-data "tags", which reduces the accuracy and completeness of searches for information and adding search time performance. |
| No Metadata | • Not "tagging" information with metadata or tagging inconsistently may result in searches being based on metadata tags, which are ineffective and can overlook key information. |
| No Back-Up Verification | • Not checking that information can be retrieved effectively from a back-up repository when access to the back-up is needed may result in one discovering that the back-up information is corrupted or not accessible. |
| Access Obsolescence | • This refers to saving information in an electronic format which eventually ceases to be accessible and not retaining a working copy of the obsolete software to be able to access the information. |

| | |
|---|---|
| Inadequate Long-Term Retention | • This refers to the error of archiving information on an electronic medium that does not have the required durability to be readable through the required retention life of the information and not regularly accessing and re-archiving the information. |
| Inadequate Validity Maintenance | • Not checking the continued validity of information results in outdated or incorrect information being retained and used. |

## Good Practices

Some good practices gathered from the references are provided in Table 2:

**Table 2. Information Management Good Practices. (SEBoK Original)**

| Good Practice Name | Good Practice Description |
|---|---|
| Guidance | • The DAMA Guide to the Data Management Body of Knowledge provides an excellent, detailed overview of IM at both the project and enterprise level. |
| Information as an Asset | • Recognize that information is a strategic asset for the organization and needs to be managed and protected. |
| Information Storage Capacity | • Plan for the organization's information repository storage capacity to need to double every 12 to 18 months. |
| Effective Information Access | • Information that sits in a repository adds no value. It only adds value when it is used, so the right people need to be able to access the right information easily and quickly. |
| Data Modeling | • Invest time and effort in designing data models that are consistent with the underlying structure and information needs of the organization. |
| Quality Management | • The cost impact of using poor quality information can be enormous. Be rigorous about managing the quality of information. |
| Information Repository Design | • The impact of managing information poorly can also be enormous (e.g., violating intellectual property or export control rules).<br>• Make sure that these requirements are captured and implemented in the information repository, and that all users of the repository are aware of the rules that they need to follow and the penalties for infringement. |

## References

### Works Cited

ISO/IEC/IEEE. 2008. *Systems and Software Engineering - System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation/International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2008.

Mosley, M. (ed.). 2009. *The DAMA Guide to the Data Management Body of Knowledge (DAMA-DMBOK Guide).* Bradley Beach, NJ, USA: Technics Publications.

### Primary References

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2015.

Mosley, M. (ed.). 2009. *The DAMA Guide to the Data Management Body of Knowledge (DAMA-DMBOK Guide)*. Bradley Beach, NJ, USA: Technics Publications.

Redman, T. 2008. *Data Driven: Profiting from Your Most Important Business Asset*. Cambridge, MA, USA: Harvard Business Press.

### Additional References

None.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Quality Management

***Lead Authors:*** *Quong Wang, Massood Towhidnejad*, ***Contributing Authors:*** *Dick Fairley, Garry Roedler*

Whether a systems engineer delivers a product, a service, or an enterprise, the deliverable should meet the needs of the customer and be fit for use. Such a deliverable is said to be of high quality. The process to assure high quality is called quality management.

## Overview

Over the past 80 years, a *quality movement* has emerged to enable organizations to produce high quality deliverables. This movement has gone through four stages:

1. **Acceptance Sampling** was developed to apply statistical tests to assist in the decision of whether or not to accept a lot of material based on a random sample of its content.
2. **Statistical Process Control** (SPC) was developed to determine if production processes were stable. Instead of necessarily measuring products, processes are measured instead. Processes that departed from a state of statistical control were far more likely to develop low quality deliverables.
3. **Design for Quality** focused on designing processes that were robust against causes of variation, reducing the likelihood that a process would go out of control, and accordingly reducing the monitoring requirements.
4. **Six sigma** methods are applied the tools and power of statistical thinking to improve other aspects of the organization.

## Definitions

The American Society for Quality [1] provides the following definitions:

- Acceptance Sampling involves the inspection of a sample to decide whether to accept the entire lot. There are two types of sampling:
  - In attributes sampling, the presence or absence of a characteristic is noted in each of the units inspected.
  - In variables sampling, the numerical magnitude of a characteristic is measured and recorded for each inspected unit. This involves reference to a continuous scale of some kind.
- SPC is the application of statistical techniques to control a process. It is often used interchangeably with the term "statistical quality control."
- Quality is a subjective term for which each person or sector has its own definition. In technical usage, quality can have two meanings:
  - The characteristics of a product or service that bear on its ability to satisfy stated or implied needs.

- A product or service free of deficiencies. According to Joseph Juran, quality means “fitness for use.” According to Philip Crosby, it means "conformance to requirements."
- Six Sigma is a method that provides organizations with tools to improve the capability of their business processes. This increase in performance and decrease in process variation leads to defect reduction and improvement in profits, employee morale, and quality of products or services. Six Sigma quality is a term generally used to indicate a process is well controlled (±6 s from the centerline in a control chart).

## Quality Attributes

Quality attributes, also known as quality factors, quality characteristics, or non-functional requirements, are a set of system functional and non-functional requirements that are used to evaluate the system performance. There are a large number of system quality attributes identified in the literature (e.g. MSDN 2010, Barbacci et al. 1995). Depending on the type of the system being considered, some of these attributes are more prominent than others. Ideally, a system would be optimized for all the quality attributes that are important to the stakeholders, but this is an impossible task. Therefore, it is important to conduct a trade-off analysis to identify the relationship between the attributes and establish whether a change in one attribute would positively or negatively affect any other attributes. An example of such trade-off is shown in Table 1 below. (See SEBoK discussion on specialty engineering for additional information on quality attributes.)

**Table 1. Attribute Trade Off. (SEBoK Original)**

| | Flexibility | Maintainability | Reliability |
|---|---|---|---|
| **Flexibility** | | + | - |
| **Maintainability** | + | | + |
| **Reliability** | - | + | |

Finding the right set of quality attributes is the first step in quality control and management. In order to achieve high quality, quality must be measured, monitored, managed, and improved on. Therefore, in order to increase the overall system quality, it is necessary to:

- identify and prioritize the quality attributes
- identify the metrics that can be used for these attributes
- measure and monitor the attributes
- validate the measurements
- analyze the result of those measurements
- establish processes and procedures that result in improved system quality, based on the analysis.

### Quality Attributes for Products

Quality attributes for a product focus on the conformance to the specifications for the product; frequently these are manufacturing specifications. Examples include physical characteristics (length, weight, finish, capacity, etc.) being inside a given tolerance range. The physical characteristics can be related to the function of the product or to aesthetic qualities.

A single product may have a vector of quality attributes of high dimension as well as an associated region in which the vector is expected to be. Often the quality is summarized by saying the item is "in compliance" (if the vector is in the acceptable region) or "defective" (if the vector is outside the acceptable region).

### Quality Attributes for Services

Quality of services plays a major role in customer satisfaction, which is the measurement of the overall system quality. Services can be divided into two major categories: primary and secondary. The city public transportation system, the U.S. postal service, or the medical services provided by a hospital are all examples of primary services. Services that provide help to a customer are secondary services, which are typically referred to as a *customer service*. Identifying the appropriate quality attributes is critical in the quality management of services. Some examples of service quality attributes include: affordability, availability, dependability, efficiency, predictability, reliability, responsiveness, safety, security, usability, etc. Again, depending on the type of the service, some of these attributes are more prominent than the others.

For example, in the case of services that are provided by the hospital, one may potentially be more interested in the availability, reliability, and responsiveness than the security (typically hospitals are assumed to be safe) and the affordability (typically insurance covers the majority of the cost). Of course, if the patient does not have a good insurance coverage, then the importance of affordability will increase (de Knoning, 2006).

### Quality Attributes for Enterprises

An enterprise typically refers to a large, complex set of interconnected entities that includes people, technologies, processes, financial, and physical elements. Clearly, a typical enterprise has a number of internal and external stakeholders, and as a result there are a large number of quality attributes that will define its quality. Identifying the right set of attributes is typically more challenging in such a complex system. An example of an enterprise is the air traffic management system that is mainly responsible for the safe and efficient operation of the civil aviation within a country or collection of countries. There are many stakeholders that are concerned about the overall quality of the system, some example of these stakeholders and some of the primary quality attributes that they are concerned with are identified in Table 2.

**Table 2. Enterprise Stakeholders and their Quality Attributes. (SEBoK Original)**

| Stakeholders | Primary Quality Attributes |
|---|---|
| Passengers | Safety, affordability, and reliability |
| Airlines | Adaptability, efficiency, and profitability |
| Air Traffic Controller | Safety, reliability, and usability |
| Hardware & Software Developers | Reliability, fault tolerance, and maintainability |
| Government/Regulatory Agency | Safety, reliability, affordability, etc. |

### Measuring Quality Attributes

Quality cannot be achieved if it cannot be measured. The Measurement System Analysis (MSA) (Wheeler and Lynday 1989) is a set of measuring instruments that provide an adequate capability for a team to conduct appropriate measurements in order to monitor and control quality. The MSA is a collection of:

- **Tools** - measuring instruments, calibration, etc.
- **Processes** - testing and measuring methods, set of specifications, etc.
- **Procedures** - policies and procedures and methodologies that are defined by the company and/or regulatory agency
- **People** - personnel (managers, testers, analysis, etc.) who are involved in the measurement activities
- **Environment** - both environmental setting and physical setting that best simulate the operational environment and/or the best setting to get the most accurate measurements

Once the quality attributes are identified and prioritized, then the MSA supports the monitor and control of overall system quality.

Additional details about measurement are presented in the measurement article.

## Quality Management Strategies

### Acceptance Sampling

In acceptance sampling many examples of a product are presented for delivery. The consumer samples from the lot and each member of the sample is then categorized as either *acceptable* or *unacceptable* based on an attribute (attribute sampling) or measured against one or more metrics (variable sampling). Based on the measurements, an inference is made as to whether the lot meets the customer requirements.

There are four possible outcomes of the sampling of a lot, as shown in Table 3.

**Table 3. Truth Table - Outcomes of Acceptance Sampling. (SEBoK Original)**

| | **Lot Meets Requirement** | **Lot Fails Requirement** |
|---|---|---|
| **Sample Passes Test** | No error | Consumer risk |
| **Sample Fails Test** | Producer risk | No error |

A sample acceptance plan balances the risk of error between the producer and consumer. Detailed ANSI/ISO/ASQ standards describe how this allocation is performed (ANSI/ISO/ASQ A3534-2-1993: *Statistics—Vocabulary and Symbols—Statistical Quality Control*).

### Statistical Process Control

SPC is a method that was invented by Walter A. Shewhart (1931) that adopts statistical thinking to monitor and control the behaviors and performances of a process. It involves using statistical analysis techniques as tools in appropriate ways, such as providing an estimate of the variation in the performance of a process, investigating the causes of this variation, and offering the engineer the means to recognize when the process is not performing as it should based on the data. (Mary et al. 2006, 441). In this context, *performance* is measured by how well the process is performed.

The theory of quality management emphasizes managing processes by fact and maintaining systematic improvement. All product developments are a series of interconnected processes that have variation in their results. Understanding variation with SPC technology can help the process executors understand the facts of their processes and find the improvement opportunities from a systematic view.

Control charts are common tools in SPC. The control chart is also called the Shewhart 3-sigma chart. It consists of 3 limit lines: the center line, which is the mean of statistical samples, and the upper and lower control limit lines, which are calculated using the mean and standard deviation of statistical samples. The observed data points or their statistical values are drawn in the chart with time or other sequence orders. Upper and lower control limits indicate the thresholds at which the process output will be considered as *unlikely*. There are two sources of process variation. One is common cause variation, which is due to inherent interaction among process components. Another is assignable cause, which is due to events that are not part of the normal process. SPC stresses bringing a process into a state of statistical control, where only common cause variation exists, and keeping it in control. A control chart is used to distinguish between variation in a process resulting from common causes and assignable causes.

If the process is in control, and if standard assumptions are met, points will demonstrate a normal distribution around the control limit. Any points outside either of the limits, or in systematic patterns, imply a new source of variation would be introduced. A new variation means increased quality cost. Additional types of control charts exist, including: cumulative sum charts that detect small, persistent step change model departures and moving average charts, which use different possible weighting schemes to detect persistent changes (Hawkins and Olwell 1996).

### Design for Quality

Variation in the inputs to a process usually results in variation in the outputs. Processes can be designed, however, to be robust against variation in the inputs. Response surface experimental design and analysis is the statistical technique that is used to assist in determining the sensitivity of the process to variations in the input. Such an approach was pioneered by Taguchi.

### Six Sigma

Six sigma methodology (Pyzdek and Keller, 2009) is a set of tools to improve the quality of business processes; in particular, to improve performance and reduce variation. Six sigma methods were pioneered by Motorola and came into wide acceptance after they were championed by General Electric.

Problems resulting in variation are addressed by six sigma projects, which follow a five-stage process:

1. **Define** the problem, the stakeholders, and the goals.
2. **Measure** key aspects and collect relevant data.
3. **Analyze** the data to determine cause-effect relationships.
4. **Improve** the current process or **design** a new process.
5. **Control** the future state or **verify** the design.

These steps are known as **DMAIC** for existing processes and **DMADV** for new processes. A variant of six sigma is called lean six sigma wherein the emphasis is on improving or maintaining quality while driving out waste.

### Standards

Primary standards for quality management are maintained by ISO, principally the IS0 9000 series [2]. The ISO standards provide requirements for the quality management systems of a wide range of enterprises, without specifying how the standards are to be met. The key requirement is that the system must be audited. ISO standards have world-wide acceptance.

In the United States, the Malcolm Baldridge National Quality Award presents up to three awards in six categories: manufacturing, service company, small business, education, health care, and nonprofit. The Baldridge Criteria [3] have become de facto standards for assessing the quality performance of organizations.

## References

### Works Cited

Barbacci, M., M.H. Klein, T.A. Longstaff, and C.B. Weinstock. 1995. *Quality Attributes.* Pittsburgh, PA, USA: Software Engineering Institute/Carnegie Melon University. CMU/SEI-95-TR-021.

Chrissis, M.B., M. Konrad, and S. Shrum. 2006. *CMMI for Development: Guidelines for Process Integration and Product Improvement,* 2nd ed. Boston, MA, USA: Addison Wesley.

Evans, J. and W. Lindsay. 2010. *Managing for Quality and Performance Excellence.* Florence, KY, USA: Cengage Southwestern.

Juran, J.M. 1992. *Juran on Quality by Design: The New Steps for Planning Quality into Goods and Services.* New York, NY, USA: The Free Press.

Koning, H. de, J.P.S. Verver, J. van den Heuvel, S. Bisgaard, R.J.M.M. Does. 2006. "Lean Six Sigma in Healthcare." *Journal for Healthcare Quality.* 28(2) pp 4-11. MSDN. 2010. "Chapter 16: Quality Attributes," in *Microsoft*

*Application Architecture Guide*, 2nd Edition. Microsoft Software Developer Network, Microsoft Corporation. Accessed August 31, 2012. Available online at http://msdn.microsoft.com/en-us/library/ff650706.

Moen, R.D., T.W. Nolan, and L.P. Provost. 1991. *Quality Improvement through Planned Experimentation.* New York, NY, USA: McGraw-Hill.

Pyzdek, T. and P.A. Keller. 2009. *The Six Sigma Handbook,* 3rd ed. New York, NY: McGraw-Hill.

Shewhart, W.A. 1931. *Economic Control of Manufactured Product*. New York, NY, USA: Van Nostrand.

Wheeler, D.J. and R.W. Lyday. 1989. *Evaluating the Measurement Process,* 2nd ed. Knoxville, TN, USA: SPC Press.

### Primary References

Chrissis, M.B, M. Konrad, S. Shrum. 2011. *CMMI for Development: Guidelines for Process Integration and Product Improvement*, 3rd ed. Boston, MA, USA: Addison-Wesley Professional.

Evans, J. and W. Lindsay. 2010. *Managing for Quality and Performance Excellence.* Florence, KY, USA: Cengage Southwestern.

Juran, J.M. 1992. *Juran on Quality by Design: The New Steps for Planning Quality into Goods and Services.* New York, NY, USA: The Free Press.

Moen, R.D., T.W. Nolan, and L.P. Provost. 1991. *Quality Improvement through Planned Experimentation.* New York, NY, USA: McGraw-Hill.

Pyzdek, T. and P.A. Keller. 2009. *The Six Sigma Handbook,* 3rd ed. New York, NY, USA: McGraw-Hill.

Wheeler, D.J. and R.W. Lyday. 1989. *Evaluating the Measurement Process,* 2nd ed. Knoxville, TN, USA: SPC Press.

### Additional References

Hawkins, D. and D.H. Olwell. 1996. *Cumulative Sum Charts and Charting for Quality Improvement*. New York, NY, USA: Springer.

---

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

## References

[1] http://asq.org/glossary/index.html

[2] http://www.iso.org/iso/iso_catalogue/management_and_leadership_standards/quality_management.htm

[3] http://www.nist.gov/baldrige/publications/criteria.cfm

# Knowledge Area: Product and Service Life Management

## Product and Service Life Management

***Contributing Author:*** *William Stiffler*

Product and service life management deals with the overall life cycle planning and support of a system. The life of a product or service spans a considerably longer period of time than the time required to design and develop the system. Systems engineers need to understand and apply the principles of life management throughout the life cycle of the system. (See Life Cycle Models for a general discussion of life cycles.) Specifically, this knowledge area (KA) focuses on changes to a system after deployment, including extension, modernization, disposal, and retirement.

### Topics

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Service Life Extension
- Capability Updates, Upgrades, and Modernization
- Disposal and Retirement

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

### Overview

Product and service life management is also referred to as *system sustainment*. Sustainment involves the supportability of operational systems from the initial procurement to disposal. Sustainment is a key task for systems engineering that influences product and service performance and support costs for the entire life of the program.

Sustainment activities include: design for maintainability, application of built-in test, diagnostics, prognostics and other condition-based maintenance techniques, implementation of logistics footprint reduction strategies, identification of technology insertion opportunities, identification of operations and support cost reduction opportunities, and monitoring of key support metrics. Life cycle sustainment plans should be created for large, complex systems (DAU 2010). Product and service life management applies to both commercial systems (e.g. energy generation and distribution systems, information management systems, the Internet, and health industries) and government systems (e.g. defense systems, transportation systems, water-handling systems, and government services).

It is critical that the planning for system life management occur during the requirements phase of system development. (See System Requirements and System Definition). The requirements phase includes the analysis of life cycle cost alternatives, as well as gaining the understanding of how the system will be sustained and modified once it is operational.

The body of knowledge associated with product and service life management includes the following areas:

1. Service Life Extension - Systems engineers need to understand the principles of service life extension, the challenges that occur during system modifications, and issues involved with the disposal and retirement after a

system has reached the end of its useful life.

2. Modernization and Upgrades - Managing service life extension uses the engineering change management process with an understanding of the design life constraints of the system. Modernizing existing legacy systems requires special attention and understanding of the legacy requirements and the importance of having a complete inventory of all the system interfaces and technical drawings.
3. Disposal and Retirement - Disposal and retirement of a product after reaching its useful life requires attention to environmental concerns, special handling of hazardous waste, and concurrent operation of a replacement system as the existing system is being retired.

## Principles and Standards

The principles of product and service life management apply to different types of systems and domains. The type of system (commercial or government) should be used to select the correct body of knowledge and best practices that exist in different domains. For example, U.S. military systems would rely on sustainment references and best practices from the Department of Defense (DoD) (e.g., military services, Defense Acquisition University (DAU), etc.) and military standardization bodies (e.g., the American Institute of Aeronautics and Astronautics (AIAA), the Society of Automotive Engineers (SAE), the Society of Logistics Engineers (SOLE), the Open Geospatial Consortium (OGC), etc.).

Commercial aviation, power distribution, transportation, water-handling systems, the Internet, and health industries would rely on system life management references and best practices from a combination of government agencies, local municipalities, and commercial standardization bodies and associations (e.g., in the U.S.- the Department of Transportation (DOT), State of Michigan, International Organization for Standardization (ISO), Institute of Electrical and Electronics Engineers (IEEE), International Council on Systems Engineering (INCOSE), etc.).

Some standardization bodies have developed system life management practices that bridge both military and commercial systems (e.g., INCOSE, SOLE, ISO, IEEE, etc.). There are multiple commercial associations involved with defining engineering policies, best practices, and requirements for commercial product and service life management. Each commercial association has a specific focus for the market or domain area where the product is used. Examples of such commercial associations in the U.S. include: American Society of Hospital Engineering (ASHE); Association of Computing Machinery (ACM); American Society of Mechanical Engineers (ASME); American Society for Testing & Materials (ASTM) International; National Association of Home Builders (NAHB); and Internet Society (ISOC), including Internet Engineering Task Force (IETF), and SAE.

In addition, there are several specific resources which provide useful information on product and service life management:

- The *INCOSE Systems Engineering Handbook*, version 3.2.2, identifies several relevant points regarding product and service life management (2011).
- The *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS)*, version 1.1, provides guidance on product changes and system retirement (Caltrans and USDOT 2005).
- *Systems Engineering and Analysis* emphasizes design for supportability and provides a framework for product and service supportability and planning for system retirement (Blanchard and Fabrycky 2006).
- *Modernizing Legacy Systems* identifies strategies for product and service modernization (Seacord, Plakosh, and Lewis 2003).
- "Logistics and Materiel Readiness" (http://www.acq.osd.mil/log/ [1]) provides online policies, procedures, and planning references for product service life extension, modernization, and retirement (OUSD(AT&L) 2011).
- *A Multidisciplinary Framework for Resilience to Disasters and Disruptions* provides insight into architecting a system for extended service life (Jackson 2007).

## Good Practices

Major pitfalls associated with systems engineering (SE) after the deployment of products and services can be avoided if the systems engineer:

- Recognizes that the systems engineering process does not stop when the product or service becomes operational.
- Understands that certain life management functions and organizations, especially in the post-delivery phase of the life cycle, are part of the systems engineering process.
- Identifies that modifications need to comply with the system requirements.
- Considers that the users must be able to continue the maintenance activities drawn up during the system requirement phase after an upgrade or modification to the system is made.
- Accounts for changing user requirements over the system life cycle.
- Adapts the support concepts drawn up during development throughout the system life cycle.
- Applies engineering change management to the total system.

Not addressing these areas of concern early in development and throughout the product or service's life cycle can have dire consequences.

## References

### Works Cited

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis,* 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research & Innovation/U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

DAU. 2010. "Acquisition Community Connection (ACC): Where the DoD AT&L workforce meets to share knowledge." Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/US Department of Defense (DoD). https://acc.dau.mil.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Jackson. 2007. "A Multidisciplinary Framework for Resilience to Disasters and Disruptions." *Journal of Integrated Design and Process Science.* 11(2).

OUSD(AT&L). 2011. "Logistics and Materiel Readiness On-line policies, procedures, and planning references." Arlington, VA, USA: Office of the Under Secretary of Defense for Aquisition, Transportation and Logistics (OUSD(AT&L)). http://www.acq.osd.mil/log/.

Seacord, R.C., D. Plakosh, and G.A. Lewis. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Boston, MA, USA: Pearson Education.

## Primary References

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis,* 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* ver 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research and Innovation and U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Jackson, S. 2007. "A Multidisciplinary Framework for Resilience to Disasters and Disruptions." *Journal of Integrated Design and Process Science.* 11(2).

OUSD(AT&L). 2011. "Logistics and Materiel Readiness On-line policies, procedures, and planning references." Arlington, VA, USA: Office of the Under Secretary of Defense for Acquisition, Transportation and Logistics (OUSD(AT&L). http://www.acq.osd.mil/log/.

Seacord, R.C., D. Plakosh, and G.A. Lewis. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Boston, MA, USA: Pearson Education.

## Additional References

Blanchard, B.S. 2010. *Logistics engineering and management,* 5th ed. Englewood Cliffs, NJ, USA: Prentice Hall: 341-342.

Braunstein, A. 2007. "Balancing Hardware End-of-Life Costs and Responsibilities." Westport, CT, USA: Experture Group, ETS 07-12-18.

Brown, M., R. Weyers, and M. Sprinkel. 2006. "Service Life Extension of Virginia Bridge Decks afforded by Epoxy-Coated Reinforcement." *Journal of ASTM International (JAI).* 3(2): 13.

DLA. 2010. "Defense logistics agency disposition services." In Defense Logistics Agency (DLA)/U.S. Department of Defense [database online]. Battle Creek, MI, USA, accessed June 19 2010: 5. Available at: http://www.dtc.dla. mil.

EPA. 2010. "Wastes In U.S. Environmental Protection Agency (EPA)." Washington, D.C. Available at: http:// www.epa.gov/epawaste/index.htm.

Finlayson, B. and B. Herdlick. 2008. *Systems Engineering of Deployed Systems.* Baltimore, MD, USA: Johns Hopkins University: 28.

FSA. 2010. "Template for 'System Retirement Plan' and 'System Disposal Plan'." In Federal Student Aid (FSA)/U.S. Department of Eduation (DoEd). Washington, DC, USA. Accessed August 5, 2010. Available at: http:// federalstudentaid.ed.gov/business/lcm.html.

Gehring, G., D. Lindemuth, and W.T. Young. 2004. "Break Reduction/Life extension Program for CAST and Ductile Iron Water Mains." Paper presented at NO-DIG 2004, Conference of the North American Society for Trenchless Technology (NASTT), March 22-24, New Orleans, LA, USA.

Hovinga, M.N., and G.J. Nakoneczny. 2000. "Standard Recommendations for Pressure Part Inspection during a Boiler Life Extension Program." Paper presented at ICOLM (International Conference on Life Management and Life Extension of Power Plant), May, Xi'an, P.R. China.

IEC. 2007. *Obsolescence Management - Application Guide*, ed 1.0. Geneva, Switzerland: International Electrotechnical Commission, IEC 62302.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Ihii, K., C.F. Eubanks, and P. Di Marco. 1994. “Design for Product Retirement and Material Life-Cycle.” *Materials & Design.* 15(4): 225-33.

INCOSE UK Chapter. 2010. *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook,* version 3.2, issue 1.0. Foresgate, UK: International Council on Systems Engineering (INCOSE) UK Chapter: 10, 13, 23.

Institute of Engineers Singapore. 2009. *Systems Engineering Body of Knowledge, provisional,* version 2.0. Singapore.

Jackson, S. 1997. *Systems Engineering for Commercial Aircraft.* Surrey, UK: Ashgate Publishing, Ltd.

Koopman, P. 1999. “Life Cycle Considerations.” Pittsburgh, PA, USA: Carnegie Mellon. Accessed August 5, 2010. Available at: http://www.ece.cmu.edu/~koopman/des_s99/life_cycle/index.html.

L3 Communications. 2010. “Service Life Extension Program (SLEP).” Newport News, VA, USA: L3 Communications, Flight International Aviation LLC.

Livingston, H. 2010. “GEB1: Diminishing Manufacturing Sources and Material Shortages (DMSMS) Management Practices.” McClellan, CA, USA: Defense MicroElectronics Activity (DMEA)/U.S. Department of Defense (DoD).

Minneapolis-St. Paul Chapter of SOLE. 2003. "Systems Engineering in Systems Deployment and Retirement, presented to INCOSE." Minneapolis-St. Paul, MN, USA: International Society of Logistics (SOLE), Minneapolis-St. Paul Chapter.

NAS. 2006. *National Airspace System (NAS) System Engineering Manual,* version 3.1 (volumes 1-3). Washington, D.C.: Air Traffic Organization (ATO)/U.S. Federal Aviation Administration (FAA), NAS SEM 3.1.

NASA. 2007. *Systems Engineering Handbook.* Washington, DC, USA: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105, December 2007.

Nguyen, L. 2006. “Adapting the Vee Model to Accomplish Systems Engineering on Change Projects.” Paper presented at 9th Annual National Defense Industrial Association (NDIA) Systems Engineering Conference, San Diego, CA, USA.

Office of Natural Gas and Oil Technology. 1999. *Reservoir LIFE Extension Program: Encouraging Production of Remaining Oil and Gas.* Washington, DC, USA: U.S. Department of Energy (DoE).

Paks Nuclear Power Plant. 2010. “Paks Nuclear Power Plant: Service Life Extension.” In Paks Nuclear Power Plant, Ltd.. Hungary, accessed August 5, 2010. Available at: http://paksnuclearpowerplant.com/service-life-extension.

Ryen, E. 2008. *Overview of the Systems Engineering Process.* Bismarck, ND, USA: North Dakota Department of Transportation (NDDOT).

SAE International. 2010. “Standards: Commercial Vehicle--Maintenance and Aftermarket.” Warrendale, PA, USA: Society of Automotive Engineers (SAE) International.

SAE International. 2010. “Standards: Maintenance and Aftermarket.” Warrendale, PA, USA: Society of Automotive Engineers (SAE) International.

Sukamto, S. 2003. “Plant Aging and Life Extension Program at Arun LNG Plant Lhokseumawe, North Aceh, Indonesia.” Paper presented at 22nd Annual World Gas Conference, June 1-5, Tokyo, Japan.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

## References

[1] http://www.acq.osd.mil/log/

# Service Life Extension

***Lead Author:*** *William Stiffler*, ***Contributing Author:*** *Brian Wells*

Product and service life extension involves continued use of a product and/or service beyond its original design life. Product and service life extension involves assessing the risks and the life cycle cost (LCC) of continuing the use of the product or service versus the cost of a replacement system.

Service life extension (SLE) emphasizes reliability upgrades and component replacement or rebuilding of the system to delay the system’s entry into *wear-out* status due to issues such as expensive sustainment, reliability, safety, and/or performance requirements that can no longer be met. The goal is typically to return the system to as new a condition as possible while remaining consistent with the economic constraints of the program.

SLE is regarded as an environmentally friendly way to relieve rampant waste by prolonging the useful life of retiring products and preventing them from being discarded too early when they still have unused value. However, challenged by fast-changing technology and physical deterioration, a major concern in planning a product SLE is considering to what degree a product or service is fit to have its life extended.

## Topic Overview

SLE is typically required in the following circumstances:

- The system no longer meets the system performance or reliability requirements.
- The cost of operation and maintenance exceeds the cost of SLE, or the available budgets.
- Parts are no longer available for repair and maintenance.
- Operation of the system violates rules or regulations, such as environmental or safety regulations.
- Parts of the system are about to reach their operations life limits, which will result in the issue listed above occurring.

It is best if systems engineers use a proactive approach that predicts ahead, so that SLE can be accomplished before the system fails to meet its requirements and before the operations and support costs rise above acceptable limits.

Key factors that must be considered by the systems engineer during service life extension include:

- current life cycle costs of the system
- design life and expected remaining useful life of the system
- software maintenance
- configuration management
- warranty policy
- availability of parts, subsystems, and manufacturing sources
- availability of system documentation to support life extension

System design life is a major consideration for SLE. System design life parameters are established early on during the system design phase and include key assumptions involving safety limits and material life. Safety limits and the properties of material aging are critical to defining system life extension. Jackson emphasizes the importance of architecting for system resiliency in increasing system life. He also points out that a system can be architected to withstand internal and external disruptions (2007, 91-108). Systems that age through use, such as aircraft, bridges, and nuclear power plants, require periodic inspection to ascertain the degree of aging and fatigue. The results of inspections determine the need for actions to extend the product life (Elliot, Chen, and Swanekamp 1998, sec. 6.5).

Software maintenance is a critical aspect of SLE. The legacy system may include multiple computer resources that have been in operation for a period of many years and have essential functions that must not be disrupted during the upgrade or integration process. Typically, legacy systems include a computer resource or application software program that continues to be used because the cost of replacing or redesigning it is prohibitive. The Software Engineering Institute (SEI) has addressed the need for SLE of software products and services and provides useful guidance in the online library for Software Product Lines (SEI 2010, 1). (See Systems Engineering and Software Engineering for additional discussion of software engineering (SwE) factors to consider.)

Systems engineers have found that service life can be extended through the proper selection of materials. For example, transportation system elements such as highway bridges and rail systems are being designed for extended service life by using special epoxy-coated steel (Brown, Weyers, and Spinkel 2006, 13). Diminishing manufacturing sources and diminishing suppliers need to be addressed early in the SLE process. Livingston (2010) in *Diminishing Manufacturing Sources and Material Shortages (DMSMS) Management Practices* provides a method for addressing product life extension when the sources of supply are an issue. He addresses the product life cycle model and describes a variety of methods that can be applied during system design to minimize the impact of future component obsolescence issues.

During product and service life extension, it is often necessary to revisit and challenge the assumptions behind any previous life cycle cost analysis (and constituent analyses) to evaluate their continued validity and/or applicability early in the process.

## Application to Product Systems

Product life extension requires an analysis of the LCC associated with continued use of the existing product versus the cost of a replacement product. In the INCOSE Systems Engineering Handbook, Chapter 3.3 points out that the support stage includes service life extension (2012). Chapter 7 provides a framework to determine if a product's life should be extended (INCOSE 2012). In Systems Engineering and Analysis, Chapter 17 provides an LCC methodology and emphasizes the analysis of different alternatives before deciding on product life extension (Blanchard and Fabrycky 2011).

For military systems, service life extension is considered a subset of modification or modernization. Military systems use well-developed and detailed guidance for SLE programs (SLEP). The Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics (OSD AT&L) provides an online reference for policies, procedures, planning guidance, and whitepapers for military product service life extension (DAU 2011). Continuous military system modernization is a process by which state-of-the-art technologies are inserted continuously into weapon systems to increase reliability, lower sustainment costs, and increase the war fighting capability of a military system to meet evolving customer requirements throughout an indefinite service life.

Aircraft service life can be extended by reducing the dynamic loads which lead to structural fatigue. The Boeing B-52 military aircraft and the Boeing 737 commercial aircraft are prime examples of system life extension. The B-52 was first fielded in 1955 and the Boeing 737 has been fielded since 1967; both aircraft are still in use today.

For nuclear reactors, system safety is the most important precondition for service life extension. System safety must be maintained while extending the service life (Paks 2010). Built-in tests, automated fault reporting and prognostics, analysis of failure modes, and the detection of early signs of wear and aging may be applied to predict the time when maintenance actions will be required to extend the service life of the product. (For additional discussion, see Safety Engineering.)

## Application to Service Systems

For systems that provide services to a larger consumer base, SLE involves continued delivery of the service without disrupting consumer use. This involves capital investment and financial planning, as well as a phased deployment of changes. Examples of these concepts can be seen in transportation systems, water treatment facilities, energy generation and delivery systems, and the health care industry. As new technologies are introduced, service delivery can be improved while reducing LCC's. Service systems must continuously assess delivery costs based upon the use of newer technologies.

Water handling systems provide a good example of a service system that undergoes life extension. Water handling systems have been in existence since early civilization. Since water handling systems are in use as long as a site is occupied (e.g., the Roman aqueducts) and upgrades are required as the population expands, such systems are a good example of "systems that live forever." For example, there are still U.S. water systems that use a few wooden pipes since there has been no reason to replace them. Water system life extension must deal with the issue of water quality and the capacity for future users (Mays 2000). Water quality requirements can be further understood from the AWWA Manuals of Water Supply Practices (AWWA 2010).

## Application to Enterprises

SLE of a large enterprise, such as the National Astronautics and Space Administration's (NASA) national space transportation system, involves SLE on the elements of the enterprise, such as the space vehicle (shuttle), ground processing systems for launch operations and mission control, and space-based communication systems that support space vehicle tracking and status monitoring. SLE of an enterprise requires a holistic look across the entire enterprise. A balanced approach is required to address the cost of operating older system components versus the cost required to implement service life improvements.

Large enterprise systems, such as oil and natural gas reservoirs which span broad geographical areas, can use advanced technology to increase their service life. The economic extraction of oil and natural gas resources from previously established reservoirs can extend their system life. One such life extension method is to pump special liquids or gases into the reservoir to push the remaining oil or natural gas to the surface for extraction (Office of Natural Gas & Oil Technology 1999).

## Other Topics

Commercial product developers have been required to retain information for extended periods of time after the last operational product or unit leaves active service (for up to twenty years). Regulatory requirements should be considered when extending service life (INCOSE 2012).

## Practical Considerations

The cost associated with life extension is one of the main inputs in the decision to extend service life of a product or a service. The cost of SLE must be compared to the cost of developing and deploying a new system, as well as the functional utility the user will obtain from each of the alternatives. It is often the case that the funding required for SLE of large complex systems is spread over several fiscal planning cycles and is therefore subject to changes in attitude by the elected officials that appropriate the funding.

The challenges with upgrading a system while it is still being used, which is often the case with SLE, must be understood and planned to avoid serious disruptions to the services the systems provide.

Any SLE must also consider the obsolescence of the systems parts, (e.g., software, amount of system redesign that is required to eliminate the obsolete parts, etc.).

## References

### Works Cited

AWWA. 2010. “AWWA Manuals of Water Supply Practices.” In American Water Works Association (AWWA). Denver, CO. Accessed August 5, 2010. Available at: http:/ / www. awwa. org/ Resources/ standards. cfm?ItemNumber=47829&navItemNumber=47834.

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Brown, M., R. Weyers, and M. Sprinkel. 2006. “Service Life Extension of Virginia Bridge Decks afforded by Epoxy-Coated Reinforcement.” *Journal of ASTM International (JAI),* 3(2): 13.

DAU. 2010. “Acquisition community connection (ACC): Where the DoD AT&L workforce meets to share knowledge.” In Defense Acquisition University (DAU)/US Department of Defense (DoD). Ft. Belvoir, VA, USA, accessed August 5, 2010. https://acc.dau.mil/.

Elliot, T., K. Chen, and R.C. Swanekamp. 1998. "Section 6.5" in *Standard Handbook of Powerplant Engineering.* New York, NY, USA: McGraw Hill.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Jackson. 2007. “A Multidisciplinary Framework for Resilience to Disasters and Disruptions.” *Journal of Integrated Design and Process Science.* 11(2).

Livingston, H. 2010. “GEB1: Diminishing Manufacturing Sources and Material Shortages (DMSMS) Management Practices.” McClellan, CA: Defense MicroElectronics Activity (DMEA)/U.S. Department of Defense (DoD).

Mays, L. (ed). 2000. "Chapter 3" in *Water Distribution Systems Handbook.* New York, NY, USA: McGraw-Hill Book Company.

Office of Natural Gas and Oil Technology. 1999. *Reservoir LIFE Extension Program: Encouraging Production of Remaining Oil and Gas.* Washington, DC, USA: U.S. Department of Energy (DoE).

Paks Nuclear Power Plant. 2010. “Paks Nuclear Power Plant: Service Life Extension.” In Paks Nuclear Power Plant, Ltd. Hungary, accessed August 5, 2010. Available at: http://paksnuclearpowerplant.com/service-life-extension.

SEI. 2010. “Software Engineering Institute.” In Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU). Pittsburgh, PA, accessed August 5, 2010. http://www.sei.cmu.edu.

### Primary References

Blanchard, B.S., and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research and Innovation and U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Jackson. 2007. “A Multidisciplinary Framework for Resilience to Disasters and Disruptions.” *Journal of Integrated Design and Process Science.* 11(2).

OUSD(AT&L). 2011. “Logistics and Materiel Readiness On-line policies, procedures, and planning references.” Arlington, VA, USA: Office of the Under Secretary of Defense for Acquisition, Transportation and Logistics

(OUSD(AT&L). http://www.acq.osd.mil/log/.

Seacord, R.C., D. Plakosh, and G.A. Lewis. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Boston, MA, USA: Pearson Education.

## Additional References

AWWA. 2010. “AWWA Manuals of Water Supply Practices.” In American Water Works Association (AWWA). Denver, CO, USA. Accessed August 5, 2010. Available at: http:/ / www. awwa. org/ Resources/ standards. cfm?ItemNumber=47829&navItemNumber=47834.

Blanchard, B.S. 2010. *Logistics engineering and management*, 5th ed. Englewood Cliffs, NJ, USA: Prentice Hall, 341-342.

Braunstein, A. 2007. “Balancing Hardware End-of-Life Costs and Responsibilities.” Westport, CT, USA: Experture Group, ETS 07-12-18.

Brown, M., R. Weyers, and M. Sprinkel. 2006. “Service Life Extension of Virginia Bridge Decks afforded by Epoxy-Coated Reinforcement.” *Journal of ASTM International (JAI),* 3(2): 13.

Caltrans and USDOT. 2005. *Systems engineering guidebook for ITS,* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research & Innovation/U.S. Department of Transportation (USDOT), SEG for ITS 1.1: 278, 101-103, 107.

Casetta, E. 2001. *Transportation Systems Engineering: Theory and methods*. New York, NY, USA: Kluwer Publishers Academic, Springer.

DAU. 2010. “Acquisition community connection (ACC): Where the DoD AT&L workforce meets to share knowledge.” In Defense Acquisition University (DAU)/US Department of Defense (DoD). Ft. Belvoir, VA, USA, accessed August 5, 2010. https://acc.dau.mil/.

DLA. 2010. “Defense logistics agency disposition services.” In Defense Logistics Agency (DLA)/U.S. Department of Defense [database online]. Battle Creek, MI, USA, accessed June 19 2010: 5. Available at: http://www.dtc.dla. mil.

Elliot, T., K. Chen, and R.C. Swanekamp. 1998. "Section 6.5" in *Standard Handbook of Powerplant Engineering.* New York, NY, USA: McGraw Hill.

FAA. 2006. "Section 4.1" in “Systems Engineering Manual.” Washington, DC, USA: US Federal Aviation Administration (FAA).

FCC. 2009. “Radio and Television Broadcast Rules.” Washington, DC, USA: US Federal Communications Commission (FCC), 47 CFR Part 73, FCC Rule 09-19: 11299-11318.

Finlayson, B. and B. Herdlick. 2008. *Systems Engineering of Deployed Systems.* Baltimore, MD, USA: Johns Hopkins University: 28.

Gehring, G., D. Lindemuth, and W.T. Young. 2004. “Break Reduction/Life extension Program for CAST and Ductile Iron Water Mains.” Paper presented at NO-DIG 2004, Conference of the North American Society for Trenchless Technology (NASTT), March 22-24, New Orleans, LA, USA.

Hovinga, M.N. and G.J. Nakoneczny. 2000. “Standard Recommendations for Pressure Part Inspection during a Boiler Life Extension Program.” Paper presented at ICOLM (International Conference on Life Management and Life Extension of Power Plant), May, Xi’an, P.R. China.

IEC. 2007. *Obsolescence Management - Application Guide*, ed 1.0. Geneva, Switzerland: International Electrotechnical Commission, IEC 62302.

IEEE. 2010. *IEEE Standard Framework for Reliability Prediction of Hardware*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE STD 1413.

IEEE. 1998. *IEEE Standard Reliability Program for the Development and Production of Electronic Systems and Equipment*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE STD 1332.

IEEE. 2008. *IEEE Recommended practice on Software Reliability*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE STD 1633.

IEEE 2005. *IEEE Standard for Software Configuration Management Plans*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE STD 828.

IEEE. 2010. IEEE Standard Framework for Reliability Prediction of Hardware. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE STD 1413.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Ihii, K., C.F. Eubanks, and P. Di Marco. 1994. "Design for Product Retirement and Material Life-Cycle." *Materials & Design.* 15(4): 225-33.

INCOSE UK Chapter. 2010. *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook,* version 3.2, issue 1.0. Foresgate, UK: International Council on Systems Engineering (INCOSE) UK Chapter: 10, 13, 23.

Institute of Engineers Singapore. 2009. "Systems Engineering Body of Knowledge, provisional," version 2.0. Singapore.

ISO/IEC. 2003. "Industrial Automation Systems Integration-Integration of Life-Cycle Data for Process Plants including Oil, Gas Production Facilities." Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC).

ISO/IEC. 1997. "Systems Engineering for Commercial Aircraft." Surrey, UK: Ashgate Publishing Ltd.

Koopman, P. 1999. "Life Cycle Considerations." In Carnegie-Mellon University (CMU). Pittsburgh, PA, USA, accessed August 5, 2010. Available at: http://www.ece.cmu.edu/~koopman/des_s99/life_cycle/index.html.

L3 Communications. 2010. "Service Life Extension Program (SLEP)." Newport News, VA, USA: L3 Communications, Flight International Aviation LLC.

Livingston, H. 2010. "GEB1: Diminishing Manufacturing Sources and Material Shortages (DMSMS) Management Practices." McClellan, CA: Defense MicroElectronics Activity (DMEA)/U.S. Department of Defense (DoD).

Mays, L. (ed). 2000. "Chapter 3" in *Water Distribution Systems Handbook.* New York, NY, USA: McGraw-Hill Book Company.

MDIT. 2008. *System Maintenance Guidebook (SMG),* version 1.1: A companion to the systems engineering methodology (SEM) of the state unified information technology environment (SUITE). MI, USA: Michigan Department of Information Technology (MDIT), DOE G 200: 38.

NAS. 2006. *National Airspace System (NAS) System Engineering Manual,* version 3.1 (volumes 1-3). Washington, D.C., USA: Air Traffic Organization (ATO)/U.S. Federal Aviation Administration (FAA), NAS SEM 3.1.

NASA. 2007. *Systems Engineering Handbook.* Washington, DC, USA: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105, December 2007.

Office of Natural Gas and Oil Technology. 1999. *Reservoir LIFE Extension Program: Encouraging Production of Remaining Oil and Gas.* Washington, DC, USA: U.S. Department of Energy (DoE).

Paks Nuclear Power Plant. 2010. "Paks Nuclear Power Plant: Service Life Extension." In Paks Nuclear Power Plant, Ltd. Hungary, accessed August 5, 2010. Available at: http://paksnuclearpowerplant.com/service-life-extension.

Reason, J. 1997. *Managing the Risks of Organizational Accident.* Aldershot, UK: Ashgate.

Ryen, E. 2008. *Overview of the Systems Engineering Process.* Bismarck, ND, USA: North Dakota Department of Transportation (NDDOT).

SAE International. 2010. “Standards: Automotive--Maintenance and Aftermarket.” Warrendale, PA: Society of Automotive Engineers (SAE) International.

Schafer, D.L. 2003. “Keeping Pace With Technology Advances When Funding Resources Are Diminished.” Paper presented at Auto Test Con. IEEE Systems Readiness Technology Conference, Anaheim, CA, USA: 584.

SEI. 2010. “Software Engineering Institute.” In Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU). Pittsburgh, PA, accessed August 5, 2010. http://www.sei.cmu.edu.

SOLE. 2009. “Applications Divisons.” In The International Society of Logistics (SOLE). Hyattsville, MD, USA, accessed August 5, 2010. http://www.sole.org/appdiv.asp.

Sukamto, S. 2003. “Plant Aging and Life Extension Program at Arun LNG Plant Lhokseumawe, North Aceh, Indonesia.” Paper presented at 22nd Annual World Gas Conference, June 1-5, Tokyo, Japan.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Capability Updates, Upgrades, and Modernization

***Lead Authors:*** *William Stiffler*, ***Contributing Author:*** *Brian Wells*

Modernization and upgrades involve changing the product or service to include new functions and interfaces, improve system performance, and/or improve system supportability. The logistic support of a product or service reaches a point in its life where system modernization is required to resolve supportability problems and to reduce operational costs. The *INCOSE Systems Engineering Handbook* (INCOSE 2012) and *Systems Engineering and Analysis* (Blanchard and Fabrycky 2005) both stress the importance of using life cycle costs (LCC) when determining if a product or service should be modernized. Systems can be modernized in the field or returned to a depot or factory for modification.

Design for system modernization and upgrade is an important part of the system engineering process and should be considered as part of the early requirements and design activities. Engineering change proposals (ECPs) are used to initiate updates and modifications to the original system. Product and service upgrades can include new technology insertion, removing old equipment, or adding new equipment. Form, fit, function, and interface (F3I) is an important principle for upgrades where backward compatibility is a requirement.

## Topic Overview

Product and service modernization involves the same systems engineering (SE) processes and principles that are employed during the upfront design, development, integration, and testing. The primary differences between product and service modernization are the various constraints imposed by the existing system architecture, design, and components. Modernizing a legacy system requires a detailed understanding of the product or service prior to making any changes. The constraints and the existence of design and test artifacts make it necessary for the systems engineers performing modernization to tailor the traditional development processes to fit the situation.

Product and service modernization occurs for many reasons, including the following:

1. The system or one of its subsystems is experiencing reduced performance, safety, or reliability.
2. A customer or other stakeholder desires a new capability for the system.

3. Some system components may be experiencing obsolescence, including the lack of spare parts.
4. New uses for the system require modification to add capabilities not built into the originally deployed system.

The first three reasons above are discussed in more detail in *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook.* (INCOSE UK Chapter 2010).

The UK chapter of the INCOSE developed *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook* (INCOSE UK Chapter 2010). This guidance document applies to any system for which multiple systems are produced. These systems may be buildings, transmission networks, aircraft, automobiles or military vehicles, trains, naval vessels, and mass transit systems.

Government and military products provide a comprehensive body of knowledge for system modernization and updates. Key references have been developed by the defense industry and can be particular to their needs.

Key factors and questions that must be considered by the systems engineer when making modifications and upgrades to a product or service include:

- type of system (space, air, ground, maritime, and safety critical)
- missions and scenarios of expected operational usage
- policy and legal requirements that are imposed by certain agencies or business markets
- product or service LCCs
- electromagnetic spectrum usage expected, including change in RF emissions
- system original equipment manufacturer (OEM) and key suppliers, and availability of parts and subsystems
- understanding and documenting the functions, interfaces, and performance requirements, including environmental testing and validation
- system integration challenges posed by the prevalence of system-of-systems solutions and corresponding interoperability issues between legacy, modified, and new systems
- amount of regression testing to be performed on the existing software

Key processes and procedures that should be considered during product and service modernization include:

- legislative policy adherence review and certification
- safety critical review
- engineering change management and configuration control
- analysis of alternatives
- warranty and product return process implementation
- availability of manufacturing and supplier sources and products

## Application to Product Systems

Product modernization involves understanding and managing a list of product deficiencies, prioritizing change requests, and handling customer issues associated with product usage. The *INCOSE Systems Engineering Handbook* emphasizes the use of Failure Modes, Effects, and Criticality Analysis (FMECA) to understand the root causes of product failures and provide the basis for making any product changes.

Product modernization uses the engineering change management principle of change control boards to review and implement product changes and improvements. The U.S. Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics (OUSD AT&L) provides an online reference for product modernization and the use of an ECP to document planned product or service modernization efforts.

Product modernization and upgrades require the use of system documentation. A key part of the product change process is to change the supporting system documentation functions, interfaces, modes, performance requirements, and limitations. Both INCOSE (2012) and Blanchard and Fabrycky (2005) stress the importance of understanding the intended usage of the product or service documented in the form of a concept of operations.

If system documentation is not available, reverse engineering is required to capture the proper “as is configuration” of the system and to gain understanding of system behavior prior to making any changes. Seacord, Plakosh, and Lewis's *Modernizing Legacy Systems* (2003) explains the importance of documenting the existing architecture of a system, including documenting the software architecture prior to making any changes. Chapter 5 of Seacord, Plakosh, and Lewis provides a framework for understanding and documenting a legacy system (2003). The authors point out that the product or service software will undergo a transformation during modernization and upgrades. Chapter 5 introduces a horseshoe model that includes functional transformation, code transformation, and architecture transformation (Seacord, Plakosh, and Lewis 2005).

During system verification and validation (after product change), it is important to perform regression testing on the portions of the system that were not modified to confirm that upgrades did not impact the existing functions and behaviors of the system. The degree and amount of regression testing depends on the type of change made to the system and whether the upgrade includes any changes to those functions or interfaces involved with system safety. INCOSE (2012) recommends the use of a requirements verification traceability matrix to assist the systems engineer during regression testing.

It is important to consider changes to the system support environment. Change may require modification or additions to the system test equipment and other support elements such as packaging and transportation.

Some commercial products contain components and subsystems where modernization activities cannot be performed. An example of these types of commercial systems can be seen by looking at consumer electronics, such as radios and computer components. The purchase price of these commercial systems is low enough that upgrades are not economical and are considered cost prohibitive.

## Application to Service Systems

Service system modernization may require regulatory changes to allow the use of new technologies and new materials. Service system modernization requires backward compatibility to previous provided service capability during the period of change. Service system modernization also generally spans large geographical areas, requiring a phase-based change and implementation strategy. Transportation systems, such as highways, provide service to many different types of consumers and span large geographical areas. Modernization of transportation systems often requires reverse engineering prior to making changes to understand how traffic monitoring devices such as metering, cameras, and toll tags interface with the rest of the system. The California Department of Transportation's (CDOT's) Systems Engineering Guidebook for Intelligent Transportation Systems (ITS) (2005) adds reverse engineering to the process steps for system upgrade. In addition, this reference points out the need to maintain system integrity and defines integrity to include the accurate documentation of the system's functional, performance, and physical requirements in the form of requirements, design, and support specifications.

Software modernization is discussed in the *Guide to the Software Engineering Body of Knowledge (SWEBOK)* (Bourque and Fairley, 2014).

## Application to Enterprises

Enterprise system modernization must consider the location of the modification and the conditions under which the work will be performed. The largest challenge is implementing the changes while the system remains operational. In these cases, disruption of ongoing operations is a serious risk. For some systems, the transition between the old and new configuration is particularly important and must be carefully planned.

Enterprise system modernization may require coordination of changes across international boundaries. Enterprise modifications normally occur at a lower level of the system hierarchy. Change in requirements at the system level would normally constitute a new system or a new model of a system.

The *INCOSE UK Chapter Supplementary Guidance* (2010) discusses the change to the architecture of the system. In cases where a component is added or changed, this change will constitute a change to the architecture. As an example, the global positioning system (GPS) is an enterprise system implemented by the United States military but used by both commercial and government consumers worldwide. Modernization may involve changes to only a certain segment of the enterprise, such as the ground user segment to reduce size, weight, and power. Modernization may only occur in certain geographical areas of operation. For example, the air transportation system consists of multiple countries and governing bodies dispersed over the entire world. Changes can occur locally or can require coordination and integration world-wide.

## Other Topics

### The Vee Model for Modifications

Figure 1 below illustrates how the standard Vee model would be applied to a system modification. This Vee model is for the entire system; the key point is that if a modification is being initiated at a lower level of the system hierarchy, the Vee model must be entered at that level as shown in the figure. The figure shows three entry points to the Vee model. As the *INCOSE UK Chapter Supplementary Guidance* (2010) points out, the Vee model may be entered multiple times during the life of the system.

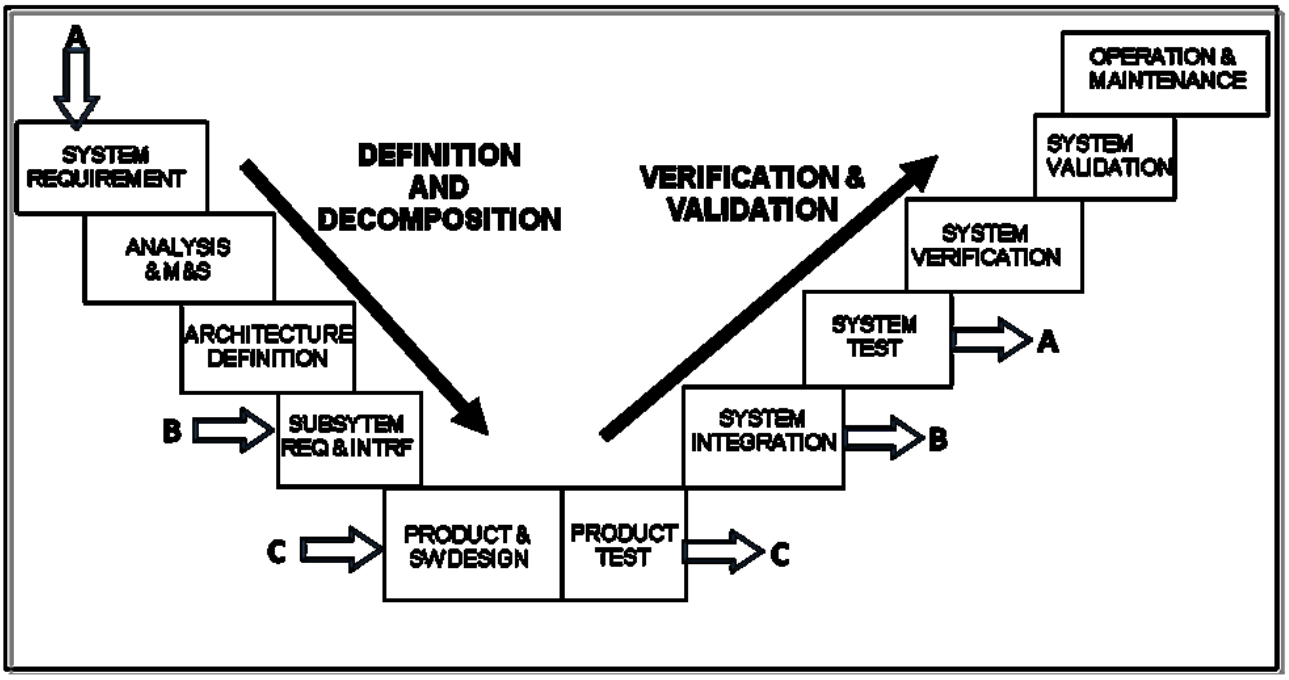


**Figure 1. The Vee Model for Modifications at the Three Different Levels.** (SEBoK Original)

A change to the system that does not change the system capabilities but does change the requirements and design of a subsystem that may be introduced into the process at point B on the Vee model (see Figure 1). Changes of this type could provide a new subsystem, such as a computer system, that meets the system-level requirements but has differences from the original, which necessitates modifications to the lower-level requirements and design, such as changing disk memory to solid state memory. The process for implementing changes starting at this point has been described by Nguyen (2006). Modification introduced at points B or C (in Figure 1) necessitate flowing the requirements upward through their "parent" requirements to the system-level requirements.

There are many cases where the change to a system needs to be introduced at the lowest levels of the architectural hierarchy; here, the entry point to the process is at point C on the Vee model. These cases are typically related to obsolete parts caused by changes in technology or due to reliability issues with subsystems and parts chosen for the original design. A change at this level should be F3I compatible so that none of the higher-level requirements are affected. The systems engineer must ensure there is no impact at the higher levels; when this does occur, it must be immediately identified and worked out with the customer and the other stakeholders.

In "Life extension of Civil Infrastructural works - a systems engineering approach" van der Laan (2008) provides a maintenance process that interacts with the system engineering process, represented by the Vee model. His life extension (or modernization) process model includes reverse engineering to obtain the system definition necessary for the modernization process. Consideration of the total lifecycle of the system will result in the capture of all of the records necessary for later upgrade; however, for many reasons, the systems engineer will find that the necessary information has not been captured or maintained.

## Practical Considerations

As pointed out by the *INCOSE UK Chapter Supplementary Guidance* (2010) there may be multiple modifications to a system in its lifetime. Often these modifications occur concurrently. This situation requires special attention and there are two methods for managing it. The first is called the "block" method. This means that a group of systems are in the process of being modified simultaneously and will be deployed together as a group at a specific time. This method is meant to ensure that at the end state, all the modifications have been coordinated and integrated so there are no conflicts and no non-compliance issues with the system-level requirements. The second method is called continuous integration and is meant to occur concurrently with the block method. Information management systems provide an example of a commercial system where multiple changes can occur concurrently. The information management system hardware and network modernization will cause the system software to undergo changes. Software release management is used to coordinate the proper timing for the distribution of system software changes to end-users (Michigan Department of Information Technology, 2008).

### Application of Commercial-Off-the-Shelf Components

Currently, a prominent consideration is the use of commercial-off-the-shelf (COTS) components. The application of COTS subsystems, components, and technologies to system life management provides a combination of advantages and risks. The first advantage is the inherent technological advancements that come with COTS components. COTS components continue to evolve toward a higher degree of functional integration. They provide increased functionality, while shrinking in physical size. The other advantage to using COTS components is that they typically have a lower cost.

The risks associated with using COTS during system life management involve component obsolescence and changes to system interfaces. Commercial market forces drive some components to obsolescence within two years or less. Application of COTS requires careful consideration to form factor and interface (physical and electrical).

## References

### Works Cited

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK).* Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research and Innovation and U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

INCOSE UK Chapter. 2010. *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook,* version 3.2, issue 1.0. Foresgate, UK: International Council on Systems Engineering (INCOSE) UK Chapter: 10, 13, 23.

MDIT. 2008. *System Maintenance Guidebook (SMG),* version 1.1: A companion to the systems engineering methodology (SEM) of the state unified information technology environment (SUITE). MI, USA: Michigan Department of Information Technology (MDIT), DOE G 200: 38.

Nguyen, L. 2006. “Adapting the Vee Model to Accomplish Systems Engineering on Change Projects.” Paper presented at 9th Annual National Defense Industrial Association (NDIA) Systems Engineering Conference, San Diego, CA, USA.

OUSD(AT&L). 2012. “On-line policies, procedures, and planning references.” Office of the Under Secretary of Defense for Acquisition, Transportation and Logistics, US Department of Defense (DoD). Accessed on August 30, 2012. Available at: http://www.acq.osd.mil/log/

Seacord, R.C., D. Plakosh, and G.A. Lewis. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Boston, MA, USA: Pearson Education.

van der Laan, J. 2008. “Life extension of Civil Infrastructural works - a systems engineering approach.” Proceedings of the 18th annual International Symposium of the International Council on Systems Engineering, Utrecht, the Netherlands.

### Primary References

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research and Innovation and U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Jackson. 2007. “A Multidisciplinary Framework for Resilience to Disasters and Disruptions.” *Journal of Integrated Design and Process Science.* 11(2).

OUSD(AT&L). 2011. “Logistics and Materiel Readiness On-line policies, procedures, and planning references.” Arlington, VA, USA: Office of the Under Secretary of Defense for Acquisition, Transportation and Logistics

(OUSD(AT&L). http://www.acq.osd.mil/log/.

Seacord, R.C., D. Plakosh, and G.A. Lewis. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Boston, MA, USA: Pearson Education.

## Additional References

Braunstein, A. 2007. “Balancing Hardware End-of-Life Costs and Responsibilities.” Westport, CT, USA: Experture Group, ETS 07-12-18.

Casetta, E. 2001. *Transportation Systems Engineering: Theory and methods*. New York, NY, USA: Kluwer Publishers Academic, Springer.

DAU. 2010. “Acquisition community connection (ACC): Where the DoD AT&L workforce meets to share knowledge.” In Defense Acquisition University (DAU)/US Department of Defense (DoD). Ft. Belvoir, VA, USA, accessed August 5, 2010. https://acc.dau.mil/.

Elliot, T., K. Chen, and R.C. Swanekamp. 1998. "Section 6.5" in *Standard Handbook of Powerplant Engineering.* New York, NY, USA: McGraw Hill.

FAA. 2006. "Section 4.1" in “Systems Engineering Manual.” Washington, DC, USA: US Federal Aviation Administration (FAA).

FCC. 2009. “Radio and Television Broadcast Rules.” Washington, DC, USA: US Federal Communications Commission (FCC), 47 CFR Part 73, FCC Rule 09-19: 11299-11318.

Finlayson, B. and B. Herdlick. 2008. *Systems Engineering of Deployed Systems.* Baltimore, MD, USA: Johns Hopkins University: 28.

IEC. 2007. *Obsolescence Management - Application Guide*, ed 1.0. Geneva, Switzerland: International Electrotechnical Commission, IEC 62302.

IEEE. 2010. *IEEE Standard Framework for Reliability Prediction of Hardware*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE 1413.

IEEE. 1998. *IEEE Standard Reliability Program for the Development and Production of Electronic Systems and Equipment*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE 1332.

IEEE. 2008. *IEEE Recommended practice on Software Reliability*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE 1633.

IEEE. 2005. *IEEE Standard for Software Configuration Management Plans*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE 828.

INCOSE. 2010. “In-service systems working group.” San Diego, CA, USA: International Council on Systems Engineering (INCOSE).

INCOSE UK Chapter. 2010. *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook, version 3.2*, issue 1.0. Foresgate, UK: International Council on Systems Engineering (INCOSE) UK Chapter: 10, 13, 23.

Institute of Engineers Singapore. 2009. “Systems Engineering Body of Knowledge, provisional,” version 2.0. Singapore.

ISO/IEC. 2003. “Industrial Automation Systems Integration-Integration of Life-Cycle Data for Process Plants including Oil, Gas Production Facilities.” Geneva, Switzerland: International Organization for Standardization (ISO)/International Electro technical Commission (IEC).

Jackson, S. 2007. “A Multidisciplinary Framework for Resilience to Disasters and Disruptions.” *Journal of Design and Process Science.* 11(2): 91-108, 110.

Jackson, S. 1997. *Systems Engineering for Commercial Aircraft.* Surrey, UK: Ashgate Publishing, Ltd.

Koopman, P. 1999. “Life Cycle Considerations.” In Carnegie-Mellon University (CMU). Pittsburgh, PA, USA, accessed August 5, 2010. Available at: http://www.ece.cmu.edu/~koopman/des_s99/life_cycle/index.html.

Livingston, H. 2010. “GEB1: Diminishing Manufacturing Sources and Material Shortages (DMSMS) Management Practices.” McClellan, CA, USA: Defense MicroElectronics Activity (DMEA)/U.S. Department of Defense (DoD).

Mays, L. (ed). 2000. "Chapter 3" in *Water Distribution Systems Handbook.* New York, NY, USA: McGraw-Hill Book Company.

MDIT. 2008. *System Maintenance Guidebook (SMG),* version 1.1: A companion to the systems engineering methdology (SEM) of the state unified information technology environment (SUITE). MI, USA: Michigan Department of Information Technology (MDIT), DOE G 200: 38.

NAS. 2006. *National Airspace System (NAS) System Engineering Manual,* version 3.1 (volumes 1-3). Washington, D.C., USA: Air Traffic Organization (ATO)/U.S. Federal Aviation Administration (FAA), NAS SEM 3.1.

NASA. 2007. *Systems Engineering Handbook.* Washington, DC, USA: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105, December 2007.

Nguyen, L. 2006. “Adapting the Vee Model to Accomplish Systems Engineering on Change Projects.” Paper presented at 9th Annual National Defense Industrial Association (NDIA) Systems Engineering Conference, San Diego, CA, USA.

Reason, J. 1997. *Managing the Risks of Organizational Accident.* Aldershot, UK: Ashgate.

Ryen, E. 2008. *Overview of the Systems Engineering Process.* Bismarck, ND, USA: North Dakota Department of Transpofration (NDDOT).

SAE International. 2010. “Standards: Automotive--Maintenance and Aftermarket.” Warrendale, PA, USA: Society of Automotive Engineers (SAE) International.

Schafer, D.L. 2003. “Keeping Pace With Technology Advances When Funding Resources Are Diminished.” Paper presented at Auto Test Con. IEEE Systems Readiness Technology Conference, Anaheim, CA, USA: 584.

SEI. 2010. “Software Engineering Institute.” In Software Engineering Institute (SEI)/Carnegie-Mellon University (CMU). Pittsburgh, PA, USA, accessed August 5, 2010. http://www.sei.cmu.edu.

SOLE. 2009. “Applications Divisons.” In The International Society of Logistics (SOLE). Hyattsville, MD, USA, accessed August 5, 2010. http://www.sole.org/appdiv.asp.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Disposal and Retirement

***Lead Author:*** *Brian Wells*

Product or service disposal and retirement is an important part of system life management. At some point, any deployed system will become one of the following: uneconomical to maintain; obsolete; or unrepairable. A comprehensive systems engineering process includes an anticipated equipment phase-out period and takes disposal into account in the design and life cycle cost assessment. (See other knowledge areas in Part 3 for discussion on life cycle metrics and assessment.)

A public focus on sustaining a clean environment encourages contemporary systems engineering (SE) design to consider recycling, reuse, and responsible disposal techniques. (See Environmental Engineering for additional discussion.)

## Topic Overview

According to the INCOSE Systems Engineering Handbook (2012), “The purpose of the disposal process is to remove a system element from the operation environment with the intent of permanently terminating its use; and to deal with any hazardous or toxic materials or waste products in accordance with the applicable guidance, policy, regulation, and statutes." In addition to technological and economical factors, the system-of-interest (SoI) must be compatible, acceptable, and ultimately address the design of a system for the environment in terms of ecological, political, and social considerations.

Ecological considerations associated with system disposal or retirement are of prime importance. The most concerning problems associated with waste management include

- Air Pollution and Control,
- Water Pollution and Control,
- Noise Pollution and Control,
- Radiation, and
- Solid Waste.

In the US, the Environmental Protection Agency (EPA) and the Occupational Safety and Health Administration (OSHA) govern disposal and retirement of commercial systems. Similar organizations perform this function in other countries. OSHA addresses hazardous materials under the 1910-119A List of Highly Hazardous Chemicals, Toxics, and Reactives (OSHA 2010). System disposal and retirement spans both commercial and government developed products and services. While both the commercial and government sectors have common goals, methods differ during the disposition of materials associated with military systems.

US DoD Directive 4160.21-M, *Defense Material Disposition Manual* (1997) outlines the requirements of the Federal Property Management Regulation (FPMR) and other laws and regulations as appropriate regarding the disposition of excess, surplus, and foreign excess personal property (FEPP). Military system disposal activities must be compliant with EPA and OSHA requirements.

## Application to Product Systems

Product system retirement may include system disposal activities or preservation activities (e.g., mothballing) if there is a chance the system may be called upon for use at a later time.

Systems Engineering and Analysis has several chapters that discuss the topics of design for goals such as “green engineering,” reliability, maintainability, logistics, supportability, producibility, disposability, and sustainability. Chapter 16 provides a succinct discussion of green engineering considerations and ecology-based manufacturing. Chapter 17 discusses life cycle costing and the inclusion of system disposal and retirement costs (Blanchard and Fabrycky 2011).

Some disposal of a system's components occurs during the system’s operational life. This happens when the components fail and are replaced. As a result, the tasks and resources needed to remove them from the system need to be planned well before a demand for disposal exists.

Transportation of failed items, handling equipment, special training requirements for personnel, facilities, technical procedures, technical documentation updates, hazardous material (HAZMAT) remediation, all associated costs, and reclamation or salvage value for precious metals and recyclable components are important considerations during system planning. Phase-out and disposal planning addresses when disposal should take place, the economic feasibility of the disposal methods used, and what the effects on the inventory and support infrastructure, safety, environmental requirements, and impact to the environment will be (Blanchard 2010).

Disposal is the least efficient and least desirable alternative for the processing of waste material (Finlayson and Herdlick 2008).

The EPA collects information regarding the generation, management and final disposition of hazardous wastes regulated under the Resource Conservation and Recovery Act of 1976 (RCRA). EPA waste management regulations are codified at 40 C.F.R. parts 239-282. Regulations regarding management of hazardous wastes begin at 40 C.F.R. part 260. Most states have enacted laws and promulgated regulations that are at least as stringent as federal regulations.

Due to the extensive tracking of the life of hazardous waste, the overall process has become known as the "cradle-to-grave system". Stringent bookkeeping and reporting requirements have been levied on generators, transporters, and operators of treatment, storage, and disposal facilities that handle hazardous waste.

Unfortunately, disposability has a lower priority compared to other activities associated with product development. This is due to the fact that typically, the disposal process is viewed as an external activity to the entity that is in custody of the system at the time. Reasons behind this view include:

- There is no direct revenue associated with the disposal process and the majority of the cost associated with the disposal process is initially hidden.
- Typically, someone outside of SE performs the disposal activities. For example, neither a car manufacturer nor the car's first buyer may be concerned about a car's disposal since the car will usually be sold before disposal.

The European Union’s Registration, Evaluation, Authorization, and Restriction of Chemicals (REACH) regulation requires manufacturers and importers of chemicals and products to register and disclose substances in products when specific thresholds and criteria are met (European Parliament 2007). The European Chemicals Agency (ECHA) manages REACH processes. Numerous substances will be added to the list of substances already restricted under European legislation; a new regulation emerged when the Restriction on Hazardous Substances (RoHS) in electrical and electronic equipment was adopted in 2003.

Requirements for substance use and availability are changing across the globe. Identifying the use of materials in the supply chain that may face restriction is an important part of system life management. System disposal and retirement requires upfront planning and the development of a disposal plan to manage the activities. An important consideration during system retirement is the proper planning required to update the facilities needed to support the system during retirement, as explained in the *California Department of Transportation Systems Engineering*

*Guidebook* (2005).

Disposal needs to take into account environmental and personal risks associated with the decommissioning of a system and all hazardous materials need to be accounted for. The decommissioning of a nuclear power plant is a prime example of hazardous material control and exemplifies the need for properly handling and transporting residual materials resulting from the retirement of certain systems.

The US Defense Logistics Agency (DLA) is the lead military agency responsible for providing guidance for worldwide reuse, recycling, and disposal of military products. A critical responsibility of the military services and defense agencies is demilitarization prior to disposal.

## Application to Service Systems

An important consideration during service system retirement or disposal is the proper continuation of services for the consumers of the system. As an existing service system is decommissioned, a plan should be adopted to bring new systems online that operate in parallel of the existing system so that service interruption is kept to a minimum. This parallel operation needs to be carefully scheduled and can occur over a significant period of time.

Examples of parallel operation include phasing-in new Air Traffic Control (ATC) systems (FAA 2006), the migration from analog television to new digital television modulation (FCC 2009), the transition to Internet protocol version 6 (IPv6), maintaining water handling systems, and maintaining large commercial transportation systems, such as rail and shipping vessels.

The *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS)* provides planning guidance for the retirement and replacement of large transportation systems. Chapter 4.7 identifies several factors which can shorten the useful life of a transportation system and lead to early retirement, such as the lack of proper documentation, the lack of effective configuration management processes, and the lack of an adequate operations and maintenance budget (Caltrans, and USDOT 2005).

## Application to Enterprises

The disposal and retirement of large enterprise systems requires a phased approach, with capital planning being implemented in stages. As in the case of service systems, an enterprise system's disposal and retirement require parallel operation of the replacement system along with the existing (older) system to prevent loss of functionality for the user.

## Other Topics

See the OSHA standard (1996) and EPA (2010) website for references that provide listings of hazardous materials. See the DLA Disposal Services website [1] for disposal services sites and additional information on hazardous materials.

## Practical Considerations

A prime objective of systems engineering is to design a product or service such that its components can be recycled after the system has been retired. The recycling process should not cause any detrimental effects to the environment.

One of the latest movements in the industry is green engineering. According to the EPA, green engineering is the design, commercialization, and use of processes and products that are technically and economically feasible while minimizing

- the generation of pollutants at the source; and
- the risks to human health and the environment.

See Environmental Engineering for additional information.

## References

### Works Cited

Blanchard, B. S. 2010. *Logistics Engineering and Management*, 5th ed. Englewood Cliffs, NJ: Prentice Hall, 341-342.

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* ver 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research and Innovation and U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

DoD. 1997. *Defense Materiel Disposition Manual.* Arlington, VA, USA: US Department of Defense, DoD 4160.21 -M.

DLA. 2010. “Defense logistics agency disposition services.” In Defense Logistics Agency (DLA)/U.S. Department of Defense [database online]. Battle Creek, MI, USA, accessed June 19 2010: 5. Available at: http://www.dtc.dla. mil.

ECHA. 2010. “European Chemicals Agency (ECHA).” In European Chemicals Agency (ECHA). Helsinki, Finland. Available at: http://echa.europa.edu/home_en.asp.

EPA. 2010. “Wastes In U.S. Environmental Protection Agency (EPA)." Washington, D.C. Available at: http:// www.epa.gov/epawaste/index.htm.

European Parliament. 2007. Regulation (EC) no 1907/2006 of the european parliament and of the council of 18 december 2006 concerning the registration, evaluation, authorisation and restriction of chemicals (REACH), establishing a european chemicals agency, amending directive 1999/45/EC and repealing council regulation (EEC) no 793/93 and commission regulation (EC) no 1488/94 as well as council directive 76/769/EEC and commission directives 91/155/EEC, 93/67/EEC, 93/105/EC and 2000/21/EC. Official Journal of the European Union 29 (5): 136/3,136/280.

FAA. 2006. "Section 4.1" in “Systems Engineering Manual.” Washington, DC, USA: US Federal Aviation Administration (FAA).

FCC. 2009. “Radio and Television Broadcast Rules.” Washington, DC, USA: US Federal Communications Commission (FCC), 47 CFR Part 73, FCC Rule 09-19: 11299-11318.

Finlayson, B. and B. Herdlick. 2008. *Systems Engineering of Deployed Systems.* Baltimore, MD, USA: Johns Hopkins University: 28.

OSHA. 1996. “Hazardous Materials: Appendix A: List of Highly Hazardous Chemicals, Toxics and Reactives.” Washington, DC, USA: Occupational Safety and Health Administration (OSHA)/U.S. Department of Labor (DoL), 1910.119(a).

*Resource Conservation and Recovery Act of 1976*, 42 U.S.C. §6901 et seq. (1976)

### Primary References

Blanchard, B.S. and W.J. Fabrycky. 2011. *Systems Engineering and Analysis*, 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Caltrans and USDOT. 2005. *Systems Engineering Guidebook for Intelligent Transportation Systems (ITS),* ver 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Research and Innovation and U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE),

INCOSE-TP-2003-002-03.2.2.

Jackson, S. 2007. “A Multidisciplinary Framework for Resilience to Disasters and Disruptions.” *Journal of Integrated Design and Process Science.* 11(2).

OUSD(AT&L). 2011. “Logistics and Materiel Readiness On-line policies, procedures, and planning references.” Arlington, VA, USA: Office of the Under Secretary of Defense for Aquisition, Transportation and Logistics (OUSD(AT&L). http://www.acq.osd.mil/log/.

Seacord, R.C., D. Plakosh, and G.A. Lewis. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Boston, MA, USA: Pearson Education.

## Additional References

Blanchard, B. S. 2010. *Logistics Engineering and Management*, 5th ed. Englewood Cliffs, NJ: Prentice Hall, 341-342.

Casetta, E. 2001. *Transportation Systems Engineering: Theory and methods*. New York, NY, USA: Kluwer Publishers Academic, Springer.

DAU. 2010. “Acquisition community connection (ACC): Where the DoD AT&L workforce meets to share knowledge.” In Defense Acquisition University (DAU)/US Department of Defense (DoD). Ft. Belvoir, VA, USA, accessed August 5, 2010. https://acc.dau.mil/.

DLA. 2010. “Defense logistics agency disposition services.” In Defense Logistics Agency (DLA)/U.S. Department of Defense [database online]. Battle Creek, MI, USA, accessed June 19 2010: 5. Available at: http://www.dtc.dla.mil.

ECHA. 2010. “European Chemicals Agency (ECHA).” In European Chemicals Agency (ECHA). Helsinki, Finland. Available at: http://echa.europa.edu/home_en.asp.

Elliot, T., K. Chen, and R.C. Swanekamp. 1998. "Section 6.5" in *Standard Handbook of Powerplant Engineering.* New York, NY, USA: McGraw Hill.

EPA. 2010. “Wastes In U.S. Environmental Protection Agency (EPA)." Washington, D.C. Available at: http://www.epa.gov/epawaste/index.htm.

European Parliament. 2007. Regulation (EC) no 1907/2006 of the european parliament and of the council of 18 december 2006 concerning the registration, evaluation, authorisation and restriction of chemicals (REACH), establishing a european chemicals agency, amending directive 1999/45/EC and repealing council regulation (EEC) no 793/93 and commission regulation (EC) no 1488/94 as well as council directive 76/769/EEC and commission directives 91/155/EEC, 93/67/EEC, 93/105/EC and 2000/21/EC. Official Journal of the European Union 29 (5): 136/3,136/280.

FAA. 2006. "Section 4.1" in “Systems Engineering Manual.” Washington, DC, USA: US Federal Aviation Administration (FAA).

FCC. 2009. “Radio and Television Broadcast Rules.” Washington, DC, USA: US Federal Communications Commission (FCC), 47 CFR Part 73, FCC Rule 09-19: 11299-11318.

Finlayson, B. and B. Herdlick. 2008. *Systems Engineering of Deployed Systems.* Baltimore, MD, USA: Johns Hopkins University: 28.

FSA. 2010. “Template for 'System Retirement Plan' and 'System Disposal Plan'.” In Federal Student Aid (FSA)/U.S. Department of Eduation (DoEd). Washington, DC, USA. Accessed August 5, 2010. Available at: http://federalstudentaid.ed.gov/business/lcm.html.

IEEE 2005. *IEEE Standard for Software Configuration Management Plans*. New York, NY, USA: Institute of Electrical and Electronics Engineers (IEEE), IEEE 828.

Ihii, K., C.F. Eubanks, and P. Di Marco. 1994. “Design for Product Retirement and Material Life-Cycle.” *Materials & Design.* 15(4): 225-33.

INCOSE. 2010. “In-service systems working group.” San Diego, CA, USA: International Council on Systems Engineering (INCOSE).

INCOSE UK Chapter. 2010. *Applying Systems Engineering to In-Service Systems: Supplementary Guidance to the INCOSE Systems Engineering Handbook, version 3.2*, issue 1.0. Foresgate, UK: International Council on Systems Engineering (INCOSE) UK Chapter: 10, 13, 23.

Institute of Engineers Singapore. 2009. “Systems Engineering Body of Knowledge, provisional,” version 2.0. Singapore: Institute of Engineers Singapore.

Mays, L. (ed). 2000. "Chapter 3" in *Water Distribution Systems Handbook.* New York, NY, USA: McGraw-Hill Book Company.

MDIT. 2008. *System Maintenance Guidebook (SMG),* version 1.1: A companion to the systems engineering methdology (SEM) of the state unified information technology environment (SUITE). MI, USA: Michigan Department of Information Technology (MDIT), DOE G 200: 38.

Minneapolis-St. Paul Chapter of SOLE. 2003. "Systems Engineering in Systems Deployment and Retirement, presented to INCOSE." Minneapolis-St. Paul, MN, USA: International Society of Logistics (SOLE), Minneapolis-St. Paul Chapter.

NAS. 2006. *National Airspace System (NAS) System Engineering Manual,* version 3.1 (volumes 1-3). Washington, D.C.: Air Traffic Organization (ATO)/U.S. Federal Aviation Administration (FAA), NAS SEM 3.1.

NASA. 2007. *Systems Engineering Handbook.* Washington, DC, USA: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105, December 2007.

OSHA. 1996. “Hazardous Materials: Appendix A: List of Highly Hazardous Chemicals, Toxics and Reactives.” Washington, DC, USA: Occupational Safety and Health Administration (OSHA)/U.S. Department of Labor (DoL), 1910.119(a).

Ryen, E. 2008. *Overview of the Systems Engineering Process.* Bismarck, ND, USA: North Dakota Department of Transpofration (NDDOT).

SAE International. 2010. “Standards: Automotive--Maintenance and Aftermarket.” Warrendale, PA: Society of Automotive Engineers (SAE) International.

Schafer, D.L. 2003. “Keeping Pace With Technology Advances When Funding Resources Are Diminished.” Paper presented at Auto Test Con. IEEE Systems Readiness Technology Conference, Anaheim, CA, USA: 584.

SOLE. 2009. “Applications Divisons.” In The International Society of Logistics (SOLE). Hyattsville, MD, USA, accessed August 5, 2010. http://www.sole.org/appdiv.asp.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

## References

[1] http://www.dtc.dla.mil

# Knowledge Area: Systems Engineering Standards

## Systems Engineering Standards

***Lead Authors:*** *Garry Roedler, Chuck Calvano*

This knowledge area (KA) focuses on the standards and technical protocols that are relevant to systems engineering. It looks at the types of standards, some of the key standards, and the alignment efforts to achieve a consistent set of standards. It then compares some of the standards, and surveys the application of the standards. Note that many of these standards have been used as references throughout Part 3.

### Topics

Each part of the SEBoK is divided into KA's, which are groupings of information with a related theme. The KA's in turn are divided into topics. This KA contains the following topics:

- Relevant Standards
- Alignment and Comparison of the Standards
- Application of Systems Engineering Standards

See the article Matrix of Implementation Examples for a mapping of case studies and vignettes included in Part 7 to topics covered in Part 3.

### References

None.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Relevant Standards

***Lead Authors:*** *Garry Roedler, Ken Zemrowski, Chuck Calvano*, ***Contributing Author:*** *Sanford Friedenthal*

There are a multitude of standards across a number of standards development organizations (SDOs) that are related to systems engineering and systems domains. This topic examines the types of standards and provides a summary of the relevant standards for systems engineering (SE).

## Standards Taxonomies and Types of Standards

There are many types of standards that focus on different aspects of SE. Thus, it can be helpful to have a taxonomy that classifies the types of standards and the objective of each type. Table 1 provides the types of the current standards and a description of the types. Refer to the Modeling Standards for a list of relevant system modeling standards.

**Table 1. Types of Systems Engineering Standards. (SEBoK Original)**

| Standard Type | Description of Type |
|---|---|
| Concepts and Terminology | • Defines the terminology and describes the concepts of a specific domain. |
| Process | • Elaborates a specific process, giving normative requirements for the essential elements of the process. It may give guidance to the requirements. |
| Requirements | • Describes the requirements for something.<br>• Most often used for actions, activities, or practices and not objects (see specifications). |
| Procedure (Practice, Activity) | • A specific procedure. Instructions or requirements on how to do something.<br>• Could be a description of best practices.<br>• Sometimes guidance and sometimes normative. |
| Guidance | • Usually an interpretation and guidance of a published standard. |
| Management System | • Requirements for management. |
| Specification | • Specifies the form, attributes, or properties of a subject artifact.<br>• Usually an object and usually normative. |
| Reference Model | • A reference model or collection of specifications of which a reference model is composed. |
| Process Reference Model (PRM) | • A collection of processes necessary and sufficient to achieve a nominated business outcome. |
| Process Assessment Model (PAM) | • Requirements and guidance for assessing attributes of nominated processes or attributes of a nominated collection of processes. |
| Guide to Body of Knowledge (BOK) | • Collects and describes the current body of knowledge in a domain, or guidance to the body of knowledge. |

## Systems Engineering Related Standards

### Summary of Systems Engineering Related Standards

Table 2 contains a summary of SE related standards. This table does not include all SE related standards, as there are many are focused on a specific domain, sector, or user group (e.g., it does not include standards from a specific government agenda). The table does include standards that are considered to be widely applicable systems engineering and systems life cycle management system life cycle processes, such as ISO/IEC/IEEE 15288 (2015). Where available, there is a link to the official abstract for the standard.

## Table 2. Summary of Systems Engineering Standards. (SEBoK Original)

| Document ID | Document Title | Organization |
|---|---|---|
| ISO/IEC/IEEE 15288 [1] | Systems and Software Engineering - System Life Cycle Processes | ISO/IEC/IEEE |
| ISO/IEC/IEEE 24765 [2] | Systems and Software Engineering - Systems and Software Engineering Vocabulary | ISO/IEC/IEEE |
| ISO/IEC/IEEE 42010 [3] | Systems and Software Engineering - Architecture Description | ISO/IEC/IEEE |
| ISO/IEC 26702 [4] / IEEE 1220 [5] | Management of the Systems Engineering Process | ISO/IEC/IEEE |
| ISO/IEC/IEEE 29148 [6] | Systems and Software Engineering - Requirements Engineering | ISO/IEC/IEEE |
| ISO/IEC/IEEE 16085 [7] | Systems and Software Engineering - Risk Management | ISO/IEC/IEEE |
| ISO/IEC/IEEE 15939 [8] | Systems and Software Engineering - Measurement Process | ISO/IEC/IEEE |
| ISO/IEC/IEEE 16326 [9] | Systems and Software Engineering - Project Management | ISO/IEC/IEEE |
| prEN9277 [10] | Programme management - Guide for the management of Systems Engineering | CEN |
| EIA 632 [11] | Engineering of a System | TechAmerica |
| ISO 9001:2008 [12] | Quality Management Systems - Requirements | ISO TC 176 |
| EIA-649-B [13] | National Consensus Standard for Configuration Management | TechAmerica |
| ISO/IEC/IEEE TR 24748-1 [14] | Systems and Software Engineering - Guide to Life Cycle Management | ISO/IEC/IEEE |
| ISO/IEC/IEEE TR 24748-2 [15] | Systems and Software Engineering - Guide To The Application of ISO/IEC 15288:2008 | ISO/IEC/IEEE |
| ISO/IEC/IEEE CD 24748-4 [16] | Systems and Software Engineering - Application and management of the systems engineering process | ISO/IEC/IEEE |
| ISO/IEC DTR 16337 [17] | Systems Engineering Handbook (INCOSE) | ISO/IEC/INCOSE |
| ISO/IEC/IEEE 15289:2011 [18] | Systems and Software Engineering - Content of Life-Cycle Information Products (Documentation) | ISO/IEC/IEEE |
| ISO/IEC/IEEE 15026-1:2010 [19] | Systems and Software Engineering - System and Software Assurance – Part 1: Concepts And Vocabulary | ISO/IEC/IEEE |
| ISO/IEC/IEEE 15026-2:2011 [20] | Systems and Software Engineering - System and Software Assurance – Part 2: Assurance Case | ISO/IEC/IEEE |
| ISO/IEC/IEEE 15026-3:2011 [21] | Systems and Software Engineering - System and Software Assurance – Part 3: Integrity Levels | ISO/IEC/IEEE |
| ISO/IEC/IEEE 15026-4:2012 [22] | Systems and Software Engineering - System And Software Assurance – Part 4: Assurance in the Life Cycle | ISO/IEC/IEEE JTC 1 |
| ISO/IEC TR 90005:2008 [23] | Guidelines for the Application of ISO 9001 to Systems Life Cycle Processes | ISO/IEC JTC 1 |
| ISO 10303-233:2012 [24] | Systems Engineering Data Interchange Standard | ISO TC 184 |
| ECSS-E-ST-10C [25] | Systems Engineering General Requirements | ECSS |
| ECSS-E-ST-10-02 [26] | Space Engineering - Verification {Note - standard is canceled} | ECSS |
| ECSS-E-ST-10-06 [27] | Space Engineering - Technical Requirements Specification | ECSS |
| ECSS-E-ST-10-24 [28] | Space Engineering - Interface Control | ECSS |

| | | |
|---|---|---|
| ECSS-M-ST-10 [29] | Space Project Management - Project Planning and Implementation | ECSS |
| ECSS-M-ST-40 [30] | Space Project Management - Configuration and Information Management | ECSS |
| ECSS-M-00-03 [31] | Space Project Management - Risk Management | ECSS |
| ISO 31000:2009 [32] | Risk Management - Principles and Guidelines | ISO |
| ISO 31010:2009 [33] | Risk Management - Risk Assessment Techniques | ISO |
| ISO 19439:2006 [34] | Enterprise Integration - Framework for Enterprise Modeling | ISO |
| ISO 15704:2000 [35] | Requirements for Enterprise - Reference Architectures and Methodologies | ISO |
| EIA 748 [36] | Earned Value Management System | TechAmerica |

## Breadth and Level of Detail of Key Systems Engineering Related Standards

Figure 1 shows the level of detail and the coverage of the life cycle for some key standards or groups of standards.

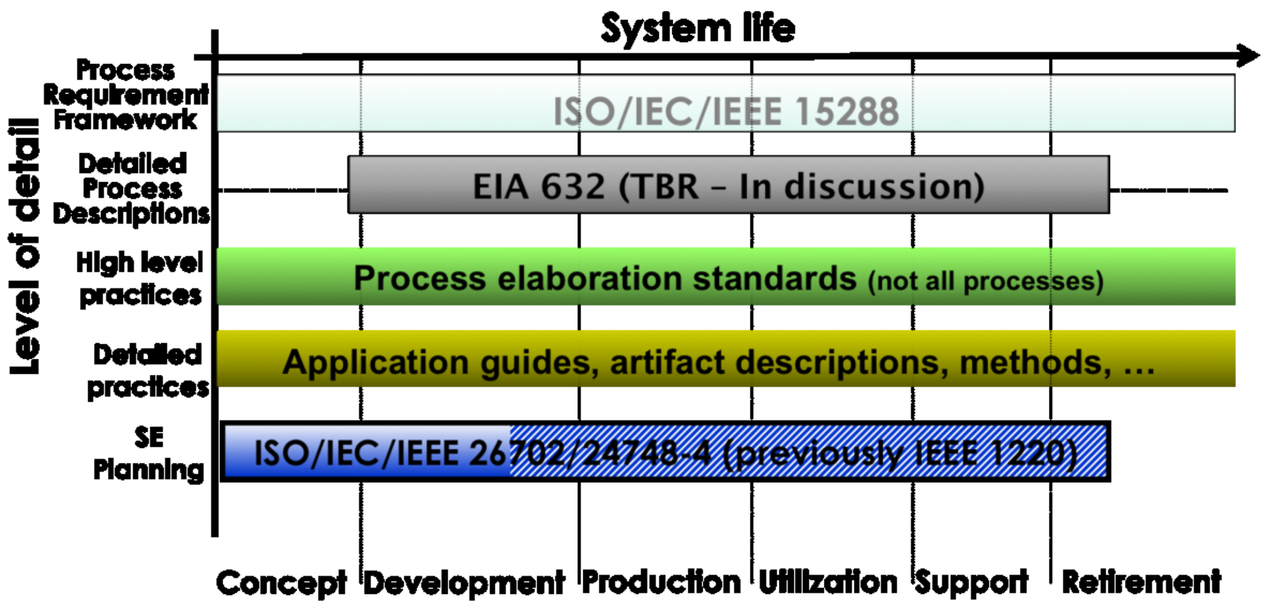

System life
Level of detail
Process Requirement Framework
Detailed Process Descriptions
High level practices
Detailed practices
SE Planning
ISO/IEC/IEEE 15288
EIA 632 (TBR – In discussion)
Process elaboration standards (not all processes)
Application guides, artifact descriptions, methods, ...
ISO/IEC/IEEE 26702/24748-4 (previously IEEE 1220)
Concept
Development
Production
Utilization
Support
Retirement

**Figure 1. Breadth and Depth of Key SE Related Standards (Adapted from Roedler 2011).** Reprinted with permission of Garry Roedler. All other rights are reserved by the copyright owner.

## Practical Considerations

Key pitfalls and good practices related to systems engineering standards are described in the next two sections.

### Pitfalls

Some of the key pitfalls encountered in the selection and use of SE standards are provided in Table 3.

**Table 3. Pitfalls in Using Systems Engineering Standards. (SEBoK Original)**

| Pitfall Name | Pitfall Description |
|---|---|
| Turnkey Process Provision | • Expecting the standard to fully provide your SE processes without any elaboration or tailoring. |
| No Need for Knowledge | • Expecting that the standard can be used without any functional or domain knowledge since the standard is the product of collective industry knowledge. |
| No Process Integration | • Lack of integrating the standards requirements with the organization or project processes. |

## Good Practices

Some good practices as gathered from the references and provided in Table 4.

**Table 4. Good Practices in Using Systems Engineering Standards. (SEBoK Original)**

| Good Practice Name | Good Practice Description |
|---|---|
| Tailor for Business Needs | • Tailor the standard within conformance requirements to best meet business needs. |
| Integration into Project | • Requirements of the standard should be integrated into the project via processes or product/service requirements. |

## References

### Works Cited

Roedler, G. 2010. "An Overview of ISO/IEC/IEEE 15288, System Life Cycle Processes." Proceedings of the 4th Asian Pacific Council on Systems Engineering (APCOSE) Conference, 4-6 October 2010, Keelung, Taiwan.

Roedler, G. 2011. "Towards Integrated Systems and Software Engineering Standards." National Defense Industrial Association (NDIA) Conference, San Diego, CA, USA.

### Primary References

ANSI/EIA. 2003. *Processes for Engineering a System.* Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA), ANSI/EIA 632-1998.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - System and Software Engineering Vocabulary.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 24765:2009.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Architecture description.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 42010.

ISO/IEC/IEEE. 2011. *Systems and software engineering - Requirements engineering.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission/Institute of Electrical and Electronics Engineers (IEEE), (IEC), ISO/IEC/IEEE 29148.

Roedler, G. 2010. *An Overview of ISO/IEC/IEEE 15288, System Life Cycle Processes.* Proceedings of the 4th Asian Pacific Council on Systems Engineering (APCOSE) Conference, 4-6 October 2010, Keelung, Taiwan.

### Additional References

ISO. 2003. *Space Systems - Risk Management.* Geneva, Switzerland: International Organization for Standardization (ISO), ISO 17666:2003.

ISO. 2009. *Risk Management - Principles and Guidelines.* Geneva, Switzerland: International Organization for Standardization (ISO), ISO 31000:2009.

ISO/IEC. 2009. *Risk Management - Risk Assessment Techniques.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 31010:2009.

ISO/IEC/IEEE. 2006. *Systems and Software Engineering - Risk Management.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16085.

ISO/IEC/IEEE. 2007. *Systems and Software Engineering - Measurement Process.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 15939.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - Project Management.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 16326.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - System and Software Assurance, Part 1: Concepts and definitions.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15026-1.

ISO/IEC/IEEE. 2010. *Systems and Software Engineering - System and Software Assurance, Part 2: Assurance case.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15026-2.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Content of Life-Cycle Information Products (documentation).* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15289.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - System and Software Assurance, Part 3: Integrity Levels.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15026-3.

---

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

## References

[1] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43564
[2] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50518
[3] http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=50508
[4] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43693
[5] http://standards.ieee.org/findstds/standard/1220-2005.html
[6] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=45171
[7] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=40723
[8] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=44344
[9] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=41977
[10] http://infostore.saiglobal.com/store/details.aspx?ProductID=1615031
[11] http://www.techstreet.com/products/1145585
[12] http://www.iso.org/iso/catalogue_detail?csnumber=46486
[13] http://www.techstreet.com/products/1800866
[14] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50502
[15] http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=54994

[16] http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=56887
[17] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56186
[18] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54388
[19] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50520
[20] http://www.iso.org/iso/catalogue_detail.htm?csnumber=52926
[21] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57107
[22] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=59927
[23] http://www.iso.org/iso/catalogue_detail?csnumber=41553
[24] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=55257
[25] http://www.inpe.br/twiki/pub/Main/GerenciamentoProjetosEspaciais/ECSS-E-ST-10C(6March2009).pdf
[26] http://www.everyspec.com/ESA/ECSS-E-10-02A_14991/
[27] http://www.inpe.br/twiki/pub/Main/GerenciamentoProjetosEspaciais/ECSS-E-ST-10-06C6March2009.pdf
[28] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCsQFjAA&url=http%3A%2F%2Fwww.ecss.nl%2Fforums%2Fecss%2Fdispatch.cgi%2Fhome%2FshowFile%2F100807%2Fd20130924080101%2FNo%2FECSS-E-ST-10-24C_DIR1(24September2013).doc&ei=Ji5cUrt0woLbBZ2qgOgN&usg=AFQjCNHNB_u3X71aMcFAeiiN2PAZuxGGmQ&bvm=bv.53899372,d.b2I
[29] http://www.everyspec.com/ESA/ECSS-M-ST-10C_REV-1_47763/
[30] http://www.ecss.nl/forums/ecss/dispatch.cgi/standards/showFile/100665/d20080802121136/No/ECSS-M-ST-80C(31July2008).doc
[31] http://www.everyspec.com/ESA/ecss-m-00-03a_2569/
[32] http://www.iso.org/iso/catalogue_detail?csnumber=43170
[33] http://www.iso.org/iso/catalogue_detail?csnumber=51073
[34] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=33833
[35] http://www.iso.org/iso/catalogue_detail.htm?csnumber=28777
[36] http://www.techstreet.com/products/1854970

# Alignment and Comparison of the Standards

***Contributing Authors:*** *Daniel Robbins, Rick Adcock*

Over the past decade, a number of the standards development organizations (SDOs) and other industry associations have been working collaboratively to align the systems engineering (SE) and software engineering (SwE) standards. The objective is to have a set of standards that can easily be used concurrently within both engineering disciplines, due to the disparity that often lies within their use of common terminology and concepts.

## Problem

There has been a lack of integration both within and across SDOs. This has led to SE and SwE standards that use different terminology, process sets, process structures, levels of prescription, and audiences. These differences have been both between systems and software, and to some extent, within each. The problem has been exacerbated, in whole or part, by competing standards (Roedler 2010).

### Cause

The cause of this problem includes several factors, as follows (Roedler 2010):

- culture - “we’re different”, “not invented here” , etc.
- organizational - different teams, committees, etc.
- competition - many SDO's
- domains - focused, narrow view often doesn’t look beyond the domain for commonality

### Impact

The impact of this problem includes the following (Roedler 2010):

- Less effective or efficient processes that are not focused on leveraging commonalities. This causes redundancy and has resulted in incompatibilities and inconsistencies between the standards making it difficult to concurrently use them together.
- Less effective solutions that are not focused on a common approach to solve a problem or need.
- Obstacle for communicating (at all levels), working in integrated teams, and leveraging resources.
- Stove-piping due to the incompatibilities, inconsistencies, and lack of leveraging commonalities.

## Objective of Alignment

The objective is to make the standards more usable together by achieving the following (Roedler 2010):

- common vocabulary
- single, integrated process set
- single process structure
- jointly planned level of prescription
- suitable across the audiences
- accounts for considerations in a wide range of domains and applications

## Alignment of Systems Engineering Standards

### Approach

A collaborative effort has been in place for the past decade that includes ISO/IEC JTC1/SC7 (Information Technology, Systems and Software Engineering), the IEEE Computer Society, the International Council on Systems Engineering (INCOSE), and others. A collaborative process is being used to align the standards. This process is built around a foundational set of vocabulary, process definition conventions, and life cycle management concepts provided in ISO/IEC/IEEE 24765 (2009) (*Systems and software engineering vocabulary*), ISO/IEC TR 24774 (2010) (*Guidelines for process description*), and ISO/IEC/IEEE TR 24748-1 (2001) (*Guide to Life Cycle Management*), respectively. At the heart of the approach is the alignment of the ISO/IEC/IEEE 15288 (2015) (system life cycle processes) and ISO/IEC/IEEE 12207 (2008) (*Software life cycle processes*), which provide the top level process framework for life cycle management of systems and software. This enables concurrent and consistent use of the standards to support both systems and software life cycle management on a single project. The approach includes the development or revision of a set of lower level supporting standards and technical reports for elaboration of specific processes, description of practices for specific purposes (e.g., systems/software assurance), description of artifacts, and guidance for the application of the standards.

## Past Accomplishments

Significant progress has been made towards the alignment objectives for the groups discussed above. Figure 1 shows a May 2011 snapshot of the status of the standards that are being aligned. In addition, four of the standards shown as "in-process" are complete, but waiting for final publication. The set of standards span ISO/IEC, IEEE, INCOSE, and the Project Management Institute (PMI). This figure depicts the standards in one of many possible taxonomies.

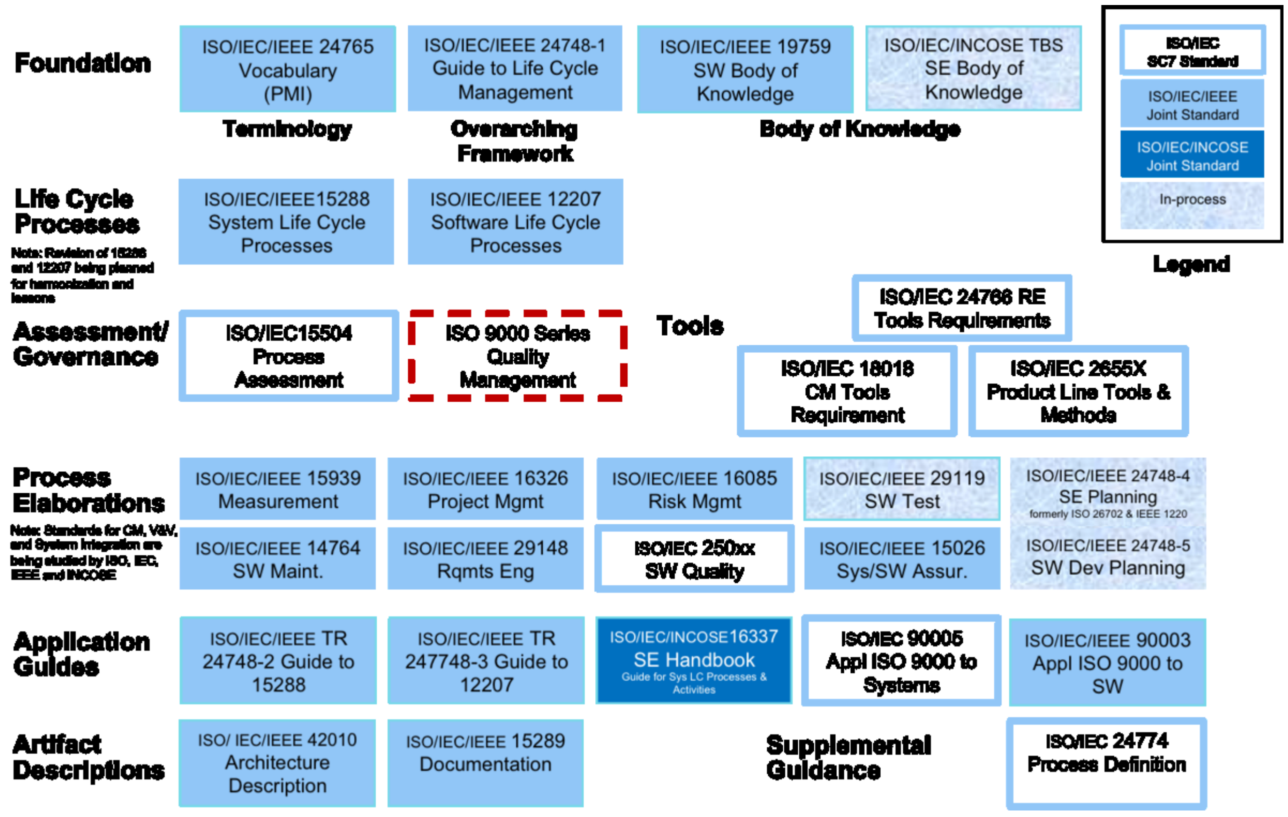

Foundation
ISO/IEC/IEEE 24765 Vocabulary (PMI)
ISO/IEC/IEEE 24748-1 Guide to Life Cycle Management
ISO/IEC/IEEE 19759 SW Body of Knowledge
ISO/IEC/INCOSE TBS SE Body of Knowledge
Terminology
Overarching Framework
Body of Knowledge
ISO/IEC SC7 Standard
ISO/IEC/IEEE Joint Standard
ISO/IEC/INCOSE Joint Standard
In-process
Legend
Life Cycle Processes
Note: Revision of 15288 and 12207 being planned for harmonization and lessons
ISO/IEC/IEEE15288 System Life Cycle Processes
ISO/IEC/IEEE 12207 Software Life Cycle Processes
Assessment/ Governance
ISO/IEC15504 Process Assessment
ISO 9000 Series Quality Management
Tools
ISO/IEC 24766 RE Tools Requirements
ISO/IEC 18018 CM Tools Requirement
ISO/IEC 2655X Product Line Tools & Methods
Process Elaborations
Note: Standards for CM, V&V, and System Integration are being studied by ISO, IEC, IEEE and INCOSE
ISO/IEC/IEEE 15939 Measurement
ISO/IEC/IEEE 16326 Project Mgmt
ISO/IEC/IEEE 16085 Risk Mgmt
ISO/IEC/IEEE 29119 SW Test
ISO/IEC/IEEE 24748-4 SE Planning formerly ISO 26702 & IEEE 1220
ISO/IEC/IEEE 14764 SW Maint.
ISO/IEC/IEEE 29148 Rqmts Eng
ISO/IEC 250xx SW Quality
ISO/IEC/IEEE 15026 Sys/SW Assur.
ISO/IEC/IEEE 24748-5 SW Dev Planning
Application Guides
ISO/IEC/IEEE TR 24748-2 Guide to 15288
ISO/IEC/IEEE TR 247748-3 Guide to 12207
ISO/IEC/INCOSE16337 SE Handbook Guide for Sys LC Processes & Activities
ISO/IEC 90005 Appl ISO 9000 to Systems
ISO/IEC/IEEE 90003 Appl ISO 9000 to SW
Artifact Descriptions
ISO/ IEC/IEEE 42010 Architecture Description
ISO/IEC/IEEE 15289 Documentation
Supplemental Guidance
ISO/IEC 24774 Process Definition

**Figure 1. Current Alignment/Integration Status (Adapted from Roedler 2011).** Reprinted with permission of Garry Roedler. All other rights are reserved by the copyright owner.

## Current Efforts

A Life Cycle Process Harmonization Advisory Group has been evaluating the current standards for systems and software engineering. The objective of the group is to provide a set of recommendations for further harmonization of the industry standards. Specifically, its charter includes:

- Performing an architectural analysis and recommend a framework for an integrated set of process standards in software and IT systems domains.
- Making recommendations regarding the future content, structure, and relationships of ISO/IEC 12207 (2008), ISO/IEC 15288 (2015) and their guides, as well as other related SC7 documents.

To support the development of the recommendations, process modeling of ISO/IEC/IEEE 15288 (2015) and ISO/IEC/IEEE 12207 (2008) has been performed and analyzed for consistency, completeness/gaps, and opportunities. In addition, analysis from other working groups, technical liaisons, and users of the standards has been collected. The output of this effort will be a harmonization strategy, set of recommendations for specific standards, and timing/sequencing recommendations (Roedler 2011).

Additionally, as the industry continues to consider harmonization needs of these standards, the collaboration has grown to include the work of the organizations and projects shown in Figure 2. These organizations are working towards the goal of completing a complementary and supplementary set of systems engineering resources that use the same terminology, principles, concepts, practices, and processes and can be used concurrently without issues.

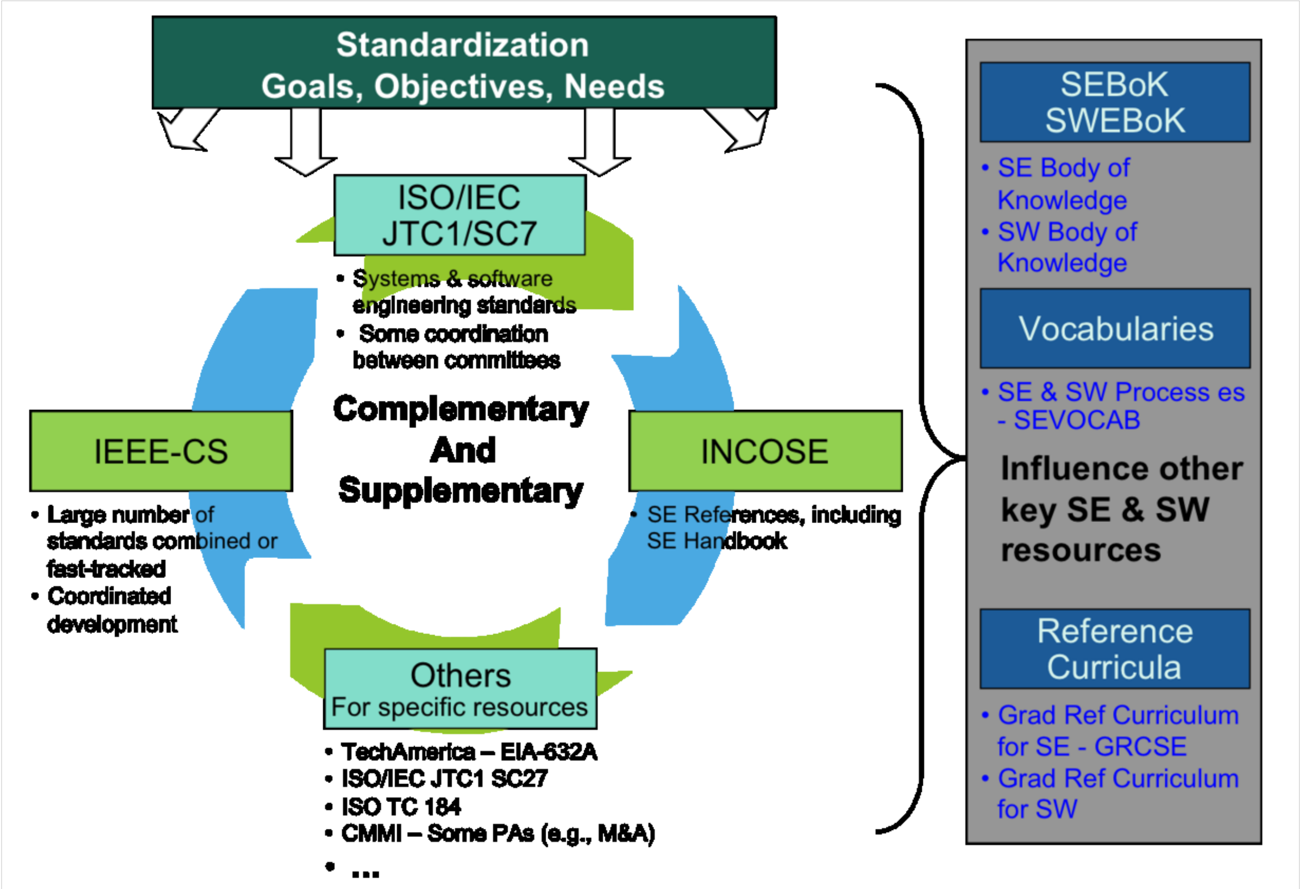

Standardization
Goals, Objectives, Needs
ISO/IEC
JTC1/SC7
• Systems & software engineering standards
• Some coordination between committees
Complementary
And
Supplementary
IEEE-CS
• Large number of standards combined or fast-tracked
• Coordinated development
INCOSE
• SE References, including SE Handbook
Others
For specific resources
• TechAmerica – EIA-632A
• ISO/IEC JTC1 SC27
• ISO TC 184
• CMMI – Some PAs (e.g., M&A)
• ...
SEBoK
SWEBoK
• SE Body of Knowledge
• SW Body of Knowledge
Vocabularies
• SE & SW Process es - SEVOCAB
Influence other key SE & SW resources
Reference Curricula
• Grad Ref Curriculum for SE - GRCSE
• Grad Ref Curriculum for SW

**Figure 2. Growing Industry Collaboration.** (SEBoK Original)

## Comparison of Systems Engineering Standards

See Figure 1 located in the Relevant Standards article to see the breadth and level of detail for many of the SE related standards. Since EIA 632 (2003) (*Engineering of a System*) is currently in revision, a comparison of ISO/IEC/IEEE 15288 (2015) (*System life cycle processes*) and EIA 632 will be deferred until the revision is complete.

Figure 3 shows a comparison of the 3-part technical reports that provide life cycle management guidance. Part 1 is focused on the provision of common terminology and concepts that apply to both systems and software. Part 2 provides guidance that directly supports ISO/IEC/IEEE 15288 (2015) that is specific to systems. And Part 3 provides guidance that directly supports ISO/IEC/IEEE 12207 (2008) that is specific to software (Roedler 2010).

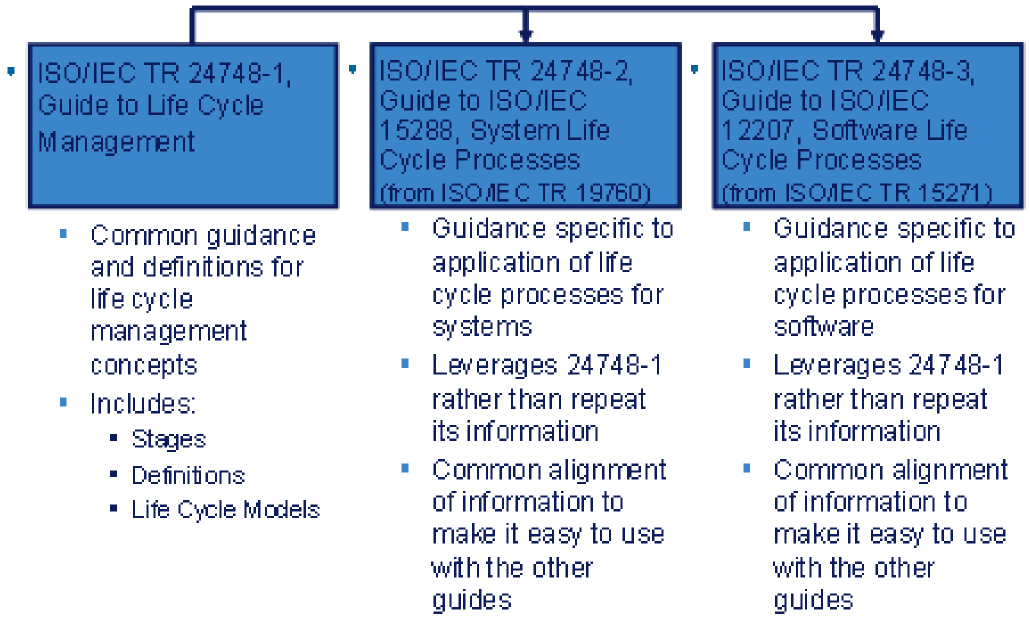

ISO/IEC TR 24748-1, Guide to Life Cycle Management
Common guidance and definitions for life cycle management concepts
Includes:
Stages
Definitions
Life Cycle Models
ISO/IEC TR 24748-2, Guide to ISO/IEC 15288, System Life Cycle Processes (from ISO/IEC TR 19760)
Guidance specific to application of life cycle processes for systems
Leverages 24748-1 rather than repeat its information
Common alignment of information to make it easy to use with the other guides
ISO/IEC TR 24748-3, Guide to ISO/IEC 12207, Software Life Cycle Processes (from ISO/IEC TR 15271)
Guidance specific to application of life cycle processes for software
Leverages 24748-1 rather than repeat its information
Common alignment of information to make it easy to use with the other guides

**Figure 3. Standards Alignment Results as of May 2011 (Roedler 2011).** Reprinted with permission of Garry Roedler. All other rights are reserved by the copyright owner.

## Practical Considerations

Key pitfalls and good practices related to systems engineering standards are described in the Relevant Standards article.

There are also instances in which standards groups for program management, safety, or other disciplines create standards on topics addressed within systems engineering but use different terminology, culture, etc. One such example is risk management, which has been dealt with by many professional societies from a number of perspectives.

Systems engineers must also be aware of the standards that govern the specialty disciplines that support systems engineering, as discussed in Part 6.

## References

### Works Cited

ANSI/EIA. 2003. Processes for Engineering a System. Philadelphia, PA, USA: American National Standards Institute (ANSI)/Electronic Industries Association (EIA). ANSI/EIA 632-1998.

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

ISO/IEC. 2010. *Systems and software engineering -- Life cycle management -- Guidelines for process description*. Geneva, Switzerland: International Organisation for Standardisation/International Electrotechnical Commissions. ISO/IEC TR 24774:2010.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - System and Software Engineering Vocabulary (SEVocab)*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/ Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 24765:2009.

ISO/IEC/IEEE. 2011. "Part 1: Guide for life cycle management," in *Systems and software engineering--life cycle management.*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/ Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE TR 24748-1:2010.

ISO/IEEE. 2008. Systems and Software Engineering — Software Life Cycle Processes. Geneva, Switzerland: International Organization for Standards (ISO)/Institute of Electrical & Electronics Engineers (IEEE) Computer Society, ISO/IEEE 12207:2008(E).

Roedler, G. 2010. *An Overview of ISO/IEC/IEEE 15288, System Life Cycle Processes.* Asian Pacific Council on Systems Engineering (APCOSE) Conference.

Roedler, G. 2011. "Towards Integrated Systems and Software Engineering Standards." National Defense Industrial Association (NDIA) Conference, San Diego, CA, USA.

## Primary References

Roedler, G. 2010. "An Overview of ISO/IEC/IEEE 15288, System Life Cycle Processes." Asian Pacific Council on Systems Engineering (APCOSE) Conference.

## Additional References

ISO. 2003. *Space Systems - Risk Management.* Geneva, Switzerland: International Organization for Standardization (ISO), ISO 17666:2003.

ISO. 2009. *Risk Management—Principles and Guidelines.* Geneva, Switzerland: International Organization for Standardization (ISO), ISO 31000:2009.

ISO/IEC. 2009. *Risk Management—Risk Assessment Techniques]].* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 31010:2009.

ISO/IEC/IEEE. 2006. *Systems and Software Engineering - Risk Management.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 16085.

ISO/IEC/IEEE. 2007. *Systems and Software Engineering - Measurement Process.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 15939.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - Project Management]].* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 16326.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - System and Software Assurance, Part 1: Concepts and definitions.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15026-1.

ISO/IEC/IEEE. 2009. *Systems and Software Engineering - System and Software Engineering Vocabulary (SEVocab).* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/ Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 24765:2009.

ISO/IEC/IEEE. 2010. *Systems and Software Engineering - System and Software Assurance, Part 2: Assurance case.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15026-2.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - Content of Life-Cycle Information Products (documentation).* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15289.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering - System and Software Assurance, Part 3: Integrity Levels.* Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE). ISO/IEC/IEEE 15026-3.

< Previous Article | Parent Article | Next Article >

**SEBoK v. 2.1, released 31 October 2019**

# Application of Systems Engineering Standards

***Lead Authors:*** *Bud Lawson, Heidi Davidz*, ***Contributing Authors:*** *Garry Roedler*

There are many systems engineering standards that have evolved over time, as indicated in Relevant Standards. In particular, there are standards that can have an influence on organizations and their projects as indicated in Figure 1 (below). Some pitfalls and good practices in utilizing standards are also identified in the article on relevant standards. In this article, several additional factors related to the utilization of the standards in systems engineering (SE) are presented.

## Standards and their Utilization

A standard is an agreed upon, repeatable way of doing something. It is a published document that contains a technical specification or other precise criteria designed to be used consistently as a rule, guideline, or definition. Standards help to make life simpler and to increase the reliability and the effectiveness of many goods and services we use. Standards are created by bringing together the experience and expertise of all interested parties, such as the producers, sellers, buyers, users, and regulators of a particular material, product, process, or service.

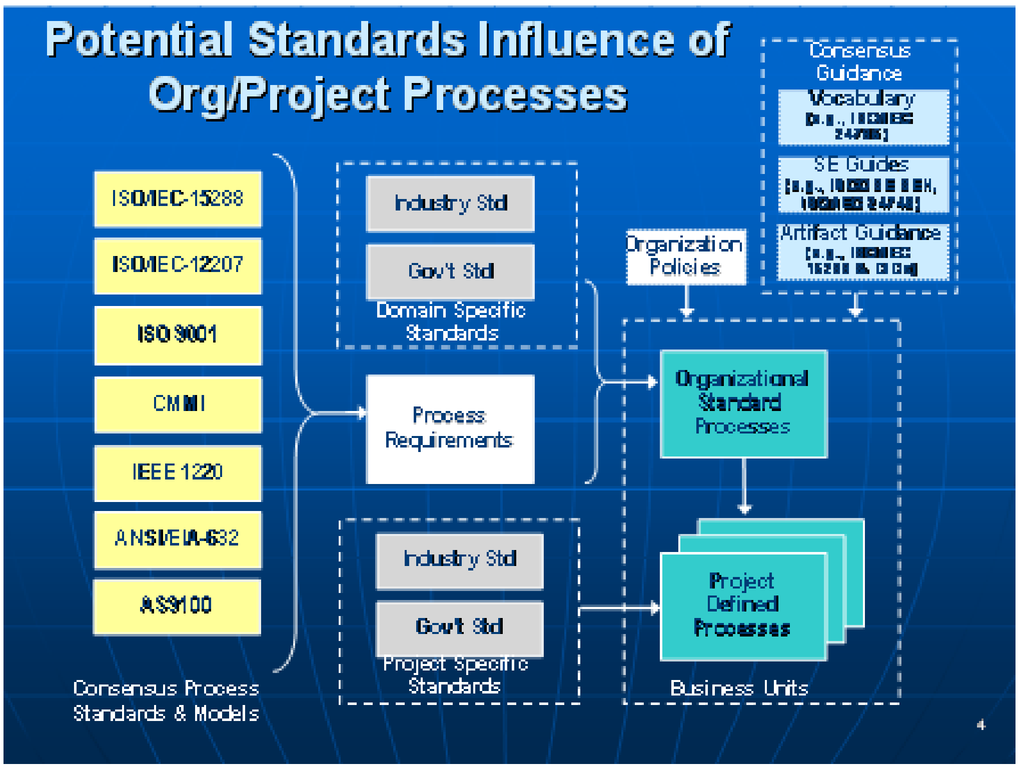

Potential Standards Influence of Org/Project Processes
Consensus Guidance
Vocabulary [e.g., ISO/IEC 24765]
SE Guides
Artifact Guidance
ISO/IEC-15288
ISO/IEC-12207
ISO 9001
CMMI
IEEE 1220
ANSI/EIA-632
AS9100
Consensus Process Standards & Models
Industry Std
Gov't Std
Domain Specific Standards
Process Requirements
Industry Std
Gov't Std
Project Specific Standards
Organization Policies
Organizational Standard Processes
Project Defined Processes
Business Units

**Figure 1. Potential Standards Influence of Organization and Project Processes (Adapted from Roedler 2011).** Reprinted with permission of Garry Roedler. All other rights are reserved by the copyright owner.

Standards are designed for voluntary use and do not impose any regulations. However, laws and regulations may address certain standards and may make compliance with them compulsory.

Further, organizations and their enterprises may choose to use standards as a means of providing uniformity in their operations and/or the products and services that they produce. The standard becomes a part of the corporate culture. In this regard, it is interesting to note that the ISO/IEC/15288 15288 (2015) standard has provided such guidance and has provided a strong framework for systems engineers as well as systems engineering and business management, as forecast earlier by Arnold and Lawson (2004).

ISO directives [1] state the following:

> *A standard does not in itself impose any obligation upon anyone to follow it. However, such an obligation may be imposed, for example, by legislation or by a contract. In order to be able to claim compliance with a standard, the user (of the standard) needs to be able to identify the requirements he is obliged to satisfy. The user needs also to be able to distinguish these requirements from other provisions where a certain freedom of choice is possible. Clear rules for the use of verbal forms (including modal auxiliaries) are therefore essential.*

## Requirements, Recommendations, and Permissions

In order to provide specificity, standards employ verb forms that convey requirements, recommendations, and permissions. For example, the ISO directives specify the following verb usages:

- The word *shall* indicates requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted.
- The word *should* indicates that among several possibilities, one is recommended as particularly suitable without mentioning or excluding others, or that a certain course of action is preferred, but not necessarily required, or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.
- The word *may* indicates a course of action permissible within the limits of the standard.

The directive also indicates that standards should avoid the use of *will, must,* and other imperatives.

## Certification, Conformance, and Compliance

In the context of the management system standards (ISO 9001:2000 and ISO 9001:2008 or ISO 14001:2004), *certification* refers to the issuing of written assurance (the certificate) by an independent external body that it has audited a management system and verified that it conforms to the requirements specified in the standard.

Typically, other more specific systems engineering standards are not the subject of certification. They are self-imposed in order to improve uniformity of organization and enterprise operations or to improve the quality of products and services. Alternatively, they may be dictated by legislation, policy, or as part of a formal agreement between an acquirer and a supplier.

Conformance testing, or type testing, is testing to determine whether a product or system meets some specified standard that has been developed for efficiency or interoperability. To aid in this, many test procedures and test setups have been developed either by the standard's maintainers or by external organizations, such as the Underwriters Laboratory (UL), specifically for testing conformity to standards.

Conformance testing is often performed by external organizations, which is sometimes the standards body itself, to give greater guarantees of compliance. Products tested in such a manner are then advertised as being certified by that external organization as complying with the standard. Service providers, equipment manufacturers, and equipment suppliers rely on this data to ensure quality of service (QoS) through this conformance process.

## Tailoring of Standards

Since the SE standards provide guidelines, they are most often tailored to fit the needs of organizations and their enterprises in their operations and/or for the products and services that they provide, as well as to provide agreement in a contract. Tailoring is a process described in an annex to the ISO/IEC/IEEE 15288 (2015) standard.

The ISO/IEC/IEEE 15288 (2015) addresses the issues of conformance, compliance, and tailoring as follows:

- Full conformance, or a claim of full conformance first declares the set of processes for which conformance is claimed. Full conformance is achieved by demonstrating that all of the requirements of the declared set of processes have been satisfied using the outcomes as evidence.
- Tailored conformance is an international standard that used as a basis for establishing a set of processes that do not qualify for full conformance; the clauses of this international standard are selected or modified in accordance with the tailoring process.
- The tailored text, for which tailored conformance is claimed, is declared. Tailored conformance is achieved by demonstrating that requirements for the processes, as tailored, have been satisfied using the outcomes as evidence.
- When the standard is used to help develop an agreement between an acquirer and a supplier, clauses of the standard can be selected for incorporation in the agreement with or without modification. In this case, it is more appropriate for the acquirer and supplier to claim compliance with the agreement than conformance with the

standard.

- Any organization (e.g., a national organization, industrial association, or company) imposing the standard as a condition of trade should specify and make public the minimum set of required processes, activities, and tasks, which constitute a supplier's conformance with the standard.

## References

### Works Cited

Arnold, S., and H. Lawson. 2004. "Viewing systems from a business management perspective." *Systems Engineering*, 7 (3): 229.

ISO. 2008. *Quality management systems -- Requirements.* Geneva, Switzerland: International Organisation for Standardisation. ISO 9001:2008.

ISO. 2004. *Environmental management systems -- Requirements with guidance for use.* Geneva, Switzerland: International Organisation for Standardisation. ISO 14001:2004

ISO/IEC/IEEE. 2015. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation / International Electrotechnical Commissions / Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

Roedler, G. 2010. "An Overview of ISO/IEC/IEEE 15288, System Life Cycle Processes. Asian Pacific Council on Systems Engineering." Asia-Pacific Council on Systems Engineering (APCOSE) Conference, Keelung, Taiwan.

Roedler, G. 2011. "Towards Integrated Systems and Software Engineering Standards." National Defense Industrial Association (NDIA) Conference, San Diego, CA, USA.

### Primary References

Roedler, G. 2010. "An Overview of ISO/IEC/IEEE 15288, System Life Cycle Processes." Proceedings of the 4th Asian Pacific Council on Systems Engineering (APCOSE) Conference, 4-6 October 2010, Keelung, Taiwan.

### Additional References

None.

---

< Previous Article | Parent Article | Next Article (Part 4) >

**SEBoK v. 2.1, released 31 October 2019**

## References

[1] http://www.iso.org/directives

# Article Sources and Contributors

**Letter from the Editor** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57764 *Contributors*: Apyster, Bkcase, Cnielsen, Dholwell, Eleach, Kguillemette, Radcock, Rcloutier, Smenck2, Wikiexpert

**BKCASE Governance and Editorial Board** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57626 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Kguillemette, Radcock, Rcloutier, Smenck2

**Acknowledgements and Release History** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57682 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Ddori, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Kguillemette, Nicole.hutchison, Radcock, Rcloutier, Smenck2, Smurawski, Wikiexpert

**Cite the SEBoK** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57684 *Contributors*: Apyster, Bkcase, Cnielsen, Dholwell, Kguillemette, Smenck2

**Bkcase Wiki:Copyright** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57613 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Kguillemette, Radcock, Smenck2, Wikiexpert

**SEBoK Table of Contents** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57803 *Contributors*: Apyster, Araher, Bkcase, Blenard, Cnielsen, Daniel Robbins, Dhenry, Dholwell, Kguillemette, Nicole.hutchison, Smenck2, Thilburn, WikiWorks, Wikiexpert

**Systems Engineering and Management** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57467 *Contributors*: Afaisandier, Apyster, Bkcase, Blawson, Cnielsen, Dcarey, Dfairley, Dhenry, Dholwell, Eleach, Groedler, Janthony, Jgercken, Mhaas, Radcock, Rmadachy, Sfriedenthal, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**Introduction to Life Cycle Processes** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57125 *Contributors*: Bkcase, Mhaas, Radcock, Sfriedenthal, WikiWorks753

**Generic Life Cycle Model** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57126 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Kforsberg, Mhaas, Radcock, Rturner, Skmackin, Smurawski, WikiWorks753, Wikiexpert, Zamoses

**Applying Life Cycle Processes** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57127 *Contributors*: Bkcase, Mhaas, Radcock, WikiWorks753

**Life Cycle Processes and Enterprise Need** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57128 *Contributors*: Bkcase, Mhaas, Radcock, WikiWorks753

**Life Cycle Models** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57129 *Contributors*: Afaisandier, Apyster, Asquires, Bkcase, Dcarey, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Kforsberg, Mhaas, Radcock, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

**System Life Cycle Process Drivers and Choices** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57466 *Contributors*: Apyster, Bkcase, Cnielsen, Dcarey, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Kforsberg, Mhaas, Radcock, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**System Life Cycle Process Models: Vee** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57316 *Contributors*: Apyster, Bkcase, Cnielsen, Dfairley, Dhenry, Dholwell, Janthony, Jgercken, Kforsberg, Mhaas, Rmadachy, Rmalove, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**System Life Cycle Process Models: Iterative** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57455 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Jgercken, Kforsberg, Mhaas, Rmalove, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert

**Integration of Process and Product Models** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57134 *Contributors*: Apyster, Bkcase, Blawson, Dhenry, Dholwell, Jgercken, Kforsberg, Mhaas, Rmalove, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

**Lean Engineering** *Source*: https://www.sebokwiki.org/d/index.php?oldid=56978 *Contributors*: Bkcase, Dhenry, Dholwell, Eleach, Mhaas, Smenck2, WikiWorks753, Wikiexpert

**Concept Definition** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57135 *Contributors*: Bkcase, Dhenry, Dholwell, Eleach, Groedler, Mhaas, Radcock, Rmalove, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert

**Business or Mission Analysis** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57136 *Contributors*: Afaisandier, Asquires, Bkcase, Dhenry, Dholwell, Groedler, Janthony, Jgercken, Mhaas, Radcock, Rmalove, Rturner, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

**Stakeholder Needs and Requirements** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57398 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Eleach, Groedler, Janthony, Mhaas, Radcock, Rmalove, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert

**System Definition** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57470 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Eleach, Groedler, Jgercken, Mhaas, Radcock, Rmalove, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**System Requirements** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57512 *Contributors*: Afaisandier, Apyster, Asofer, Asquires, Bkcase, Cjones, Dhenry, Dholwell, Eleach, Groedler, Jgercken, Mhaas, Radcock, Rmalove, Rturner, Skmackin, Smenck2, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**System Architecture** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57332 *Contributors*: Afaisandier, Bkcase, Mhaas, Radcock, WikiWorks, WikiWorks753

**Logical Architecture Model Development** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57141 *Contributors*: Afaisandier, Apyster, Asquires, Bkcase, Dhenry, Dholwell, Eleach, Groedler, Jgercken, Mhaas, Radcock, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

**Physical Architecture Model Development** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57142 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Eleach, Mhaas, Radcock, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert

**System Design** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57300 *Contributors*: Afaisandier, Bkcase, Mhaas, Radcock, WikiWorks, WikiWorks753

**System Analysis** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57397 *Contributors*: Afaisandier, Asquires, Bkcase, Dhenry, Dholwell, Jgercken, Mhaas, Radcock, Rmadachy, Rmalove, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**System Realization** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57462 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Eleach, Jgercken, Jsnoderly, Mhaas, Radcock, Rmalove, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**System Implementation** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57430 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Eleach, Jgercken, Jsnoderly, Mhaas, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**System Integration** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57419 *Contributors*: Afaisandier, Bkcase, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Jsnoderly, Mhaas, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**System Verification** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57414 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Eleach, Jgercken, Jsnoderly, Kguillemette, Mhaas, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**System Validation** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57302 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Eleach, Mhaas, Radcock, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert

**System Deployment and Use** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57364 *Contributors*: Bkcase, Dhenry, Dholwell, Eleach, Groedler, Jgercken, Jsnoderly, Mhaas, Sjackson, Skmackin, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**System Deployment** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57287 *Contributors*: Apyster, Bgallagher, Bkcase, Dhenry, Dholwell, Eleach, Jgercken, Mhaas, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**Operation of the System** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57273 *Contributors*: Apyster, Bgallagher, Bkcase, Dhenry, Dholwell, Eleach, Jgercken, Ldecardenas, Mhaas, Sjackson, Skmackin, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**System Maintenance** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57347 *Contributors*: Bgallagher, Bkcase, Ddorgan, Dhenry, Dholwell, Eleach, Jgercken, Mhaas, Sjackson, Skmackin, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**Logistics** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57154 *Contributors*: Bkcase, Dhenry, Dholwell, Eleach, Groedler, Jgercken, Jsnoderly, Mhaas, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

**Systems Engineering Management** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57374 *Contributors*: Araher, Bkcase, Dhenry, Dholwell, Groedler, Jgercken, Mhaas, Rmadachy, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**Planning** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57156 *Contributors*: Apyster, Bkcase, Bwells, Dhenry, Dholwell, Eleach, Groedler, Janthony, Jgercken, Mhaas, Rmadachy, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**Assessment and Control** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57157 *Contributors*: Apickard, Apyster, Bkcase, Dhenry, Dholwell, Eleach, Groedler, Jgercken, Mhaas, Rmadachy, Rturner, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

**Risk Management** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57158 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Econrow, Eleach, Groedler, Jgercken, Kguillemette, Mhaas, Rmadachy, Rturner, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**Measurement** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57161 *Contributors*: Apyster, Bkcase, Cjones, Dhenry, Dholwell, Eleach, Groedler, Jgercken, Mhaas, Rcarson, Rmadachy, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

**Decision Management** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57270 *Contributors*: Asquires, Bkcase, Dhenry, Dholwell, Eleach, Gparnell, Groedler, Jgercken, Mhaas, Rmadachy, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

**Configuration Management** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57163 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Eleach, Groedler, Jgercken, Jsnoderly, Mhaas, Radcock, Rmadachy, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

**Information Management** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57164 *Contributors*: Apickard, Bkcase, Dhenry, Dholwell, Eleach, Groedler, Jgercken, Mhaas, Rmadachy, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

**Quality Management** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57309 *Contributors*: Bkcase, Dfairley, Dhenry, Dholwell, Eleach, Groedler, Jgercken, Mhaas, Mtowhid, Qwang, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**Product and Service Life Management** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57272 *Contributors*: Apyster, Bkcase, Bstiffler, Dhenry, Dholwell, Eleach, Jgercken, Mhaas, Skmackin, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**Service Life Extension** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57371 *Contributors*: Asquires, Bkcase, Bstiffler, Bwells, Dhenry, Dholwell, Eleach, Jgercken, Mhaas, Skmackin, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

**Capability Updates, Upgrades, and Modernization** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57286 *Contributors*: Asquires, Bkcase, Bstiffler, Bwells, Dhenry, Dholwell, Eleach, Jgercken, Mhaas, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

**Disposal and Retirement** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57170 *Contributors*: Asquires, Bkcase, Bstiffler, Bwells, Dhenry, Dholwell, Jgercken, Mhaas, Rmalove, Skmackin, Smurawski, WikiWorks753, Wikiexpert, Zamoses

**Systems Engineering Standards** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57517 *Contributors*: Bkcase, Ccalvano, Dhenry, Dholwell, Eleach, Groedler, Jgercken, Mhaas, Skmackin, Smurawski, WikiWorks, Wikiexpert, Zamoses

**Relevant Standards** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57173 *Contributors*: Bkcase, Ccalvano, Dhenry, Dholwell, Eleach, Groedler, Janthony, Jgercken, Kguillemette, Kzemrowski, Mhaas, Sfriedenthal, Skmackin, Smenck2, Smurawski, WikiWorks, Wikiexpert, Zamoses

**Alignment and Comparison of the Standards** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57174 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Eleach, Groedler, Hdavidz, Janthony, Jgercken, Mhaas, Skmackin, Smenck2, Smurawski, Wikiexpert, Zamoses

**Application of Systems Engineering Standards** *Source*: https://www.sebokwiki.org/d/index.php?oldid=57175 *Contributors*: Bkcase, Blawson, Dhenry, Dholwell, Eleach, Groedler, Hdavidz, Janthony, Jgercken, Mhaas, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

# Image Sources, Licenses and Contributors

**File:Rob_Sm for Web.jpg** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Rob_Sm_for_Web.jpg *License*: unknown *Contributors*: Rcloutier

**File:RobSignature2.jpeg** *Source*: https://www.sebokwiki.org/d/index.php?title=File:RobSignature2.jpeg *License*: unknown *Contributors*: Bkcase

**File:Hutchison profilephoto.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Hutchison_profilephoto.png *License*: unknown *Contributors*: -

**File:SEBoK Navigation Management.PNG** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoK_Navigation_Management.PNG *License*: unknown *Contributors*: Bkcase

**File:Mapping_of_tech_topics_SEBoK_with_ISO_IEC_15288techPro_060612.jpg** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Mapping_of_tech_topics_SEBoK_with_ISO_IEC_15288techPro_060612.jpg *License*: unknown *Contributors*: Bkcase, Smenck2, Smurawski

**File:062211_BL_Paradigm.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:062211_BL_Paradigm.png *License*: unknown *Contributors*: Bkcase, Smenck2, Smurawski

**File:Fig_1_A_generic_life_cycle_KF.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig_1_A_generic_life_cycle_KF.png *License*: unknown *Contributors*: Bkcase

**File:SE Hump diagram.PNG** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SE_Hump_diagram.PNG *License*: unknown *Contributors*: Bkcase

**File:Ex_Itera_of_processes_related_to_Sys_Def_AF_052312.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Ex_Itera_of_processes_related_to_Sys_Def_AF_052312.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:JS_Figure_1.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:JS_Figure_1.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Hierarchical_decomposition_of_a_system-of-interest_060612.jpg** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Hierarchical_decomposition_of_a_system-of-interest_060612.jpg *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Recursion_of_processes_on_layers_060612.jpg** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Recursion_of_processes_on_layers_060612.jpg *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Needs-to-requirements-ryan.PNG** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Needs-to-requirements-ryan.PNG *License*: unknown *Contributors*: Bkcase

**File:Fig 1 Primary models of incremental and evolutionary development KF.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig_1_Primary_models_of_incremental_and_evolutionary_development_KF.png *License*: unknown *Contributors*: Bkcase, Smenck2, Smurawski

**File:KF_EvolutionaryConcurrentChange.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_EvolutionaryConcurrentChange.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:KF_VeeModel_Left.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_VeeModel_Left.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Fig_2_Life_Cycle_Stages_Vee_KF.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig_2_Life_Cycle_Stages_Vee_KF.png *License*: unknown *Contributors*: Bkcase

**File:JS_Figure_2.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:JS_Figure_2.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:KF_GenericStageStructure.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_GenericStageStructure.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Comparisons_of_life_cycle_models.PNG** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Comparisons_of_life_cycle_models.PNG *License*: unknown *Contributors*: Janthony, Smurawski

**File:KF_SchedulingDevelopment.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_SchedulingDevelopment.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:KF_ILSSystemLifeCycle.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_ILSSystemLifeCycle.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:KF_VeeModel_Right.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_VeeModel_Right.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:KF_IncrementalDevelopment_Multiple.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_IncrementalDevelopment_Multiple.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Incremental_Development_with_a_single_delivery.PNG** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Incremental_Development_with_a_single_delivery.PNG *License*: unknown *Contributors*: Janthony, Smurawski

**File:Evolutionary_Generic_Model.PNG** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Evolutionary_Generic_Model.PNG *License*: unknown *Contributors*: Janthony, Smurawski

**File:KF_EvolutionComponents_Orbiter.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_EvolutionComponents_Orbiter.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:KF_IncrementalBuildCycles.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_IncrementalBuildCycles.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:KF_IncrementalCommitmentSpiral.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_IncrementalCommitmentSpiral.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:KF_Phase_GenericIncremental.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_Phase_GenericIncremental.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:KF_ICSMActivityCategories.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_ICSMActivityCategories.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:KF_FeasibilityEvidenceDescription.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:KF_FeasibilityEvidenceDescription.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Tale_of_Two_Implementations_Schwaber.jpg** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Tale_of_Two_Implementations_Schwaber.jpg *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Example_of_Architected_Agile_Process_Replacement_070912.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Example_of_Architected_Agile_Process_Replacement_070912.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Generic_(T)_Stage_Structure_of_System_Life_Cycle_Models_(Lawson_2010,_Figure_6-2).png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Generic_(T)_Stage_Structure_of_System_Life_Cycle_Models_(Lawson_2010,_Figure_6-2).png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:@@BKCASE_Wiki_Section_2.5_Fig_2a_PDF_110820.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:@@BKCASE_Wiki_Section_2.5_Fig_2a_PDF_110820.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:@@BKCASE_Wiki_Section_2.5_Fig_2b_PDF_110820.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:@@BKCASE_Wiki_Section_2.5_Fig_2b_PDF_110820.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Figure_3._Spiral_Model_support_for_Process_Models,_Product_Models....png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Figure_3._Spiral_Model_support_for_Process_Models,_Product_Models....png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:T-Model_for_Software_System_(Lawson_2010,_Figure_6-3).png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:T-Model_for_Software_System_(Lawson_2010,_Figure_6-3).png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Iteration_through_Life_Cycle_Stages_(Lawson_2010,_Figure_6-4).png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Iteration_through_Life_Cycle_Stages_(Lawson_2010,_Figure_6-4).png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:@@BKCASE_Sect_2.5_Fig_6A.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:@@BKCASE_Sect_2.5_Fig_6A.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:@@BKCASE_Sect_2.5_Fig_6B.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:@@BKCASE_Sect_2.5_Fig_6B.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:QQBKCASE_Sect_2.5_Fig_6C.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:QQBKCASE_Sect_2.5_Fig_6C.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Enterprise_Strategy_and_Concept_Development.PNG** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Enterprise_Strategy_and_Concept_Development.PNG *License*: unknown *Contributors*: Smenck2, Smurawski

**File:SEBoKv05_KA-SystDef_Cycle_of_needs.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystDef_Cycle_of_needs.png *License*: unknown *Contributors*: Bkcase, Smenck2, Smurawski

**File:SEBoKv05_KA-SystDef_Top-down_development_of_design_and_requirements.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystDef_Top-down_development_of_design_and_requirements.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Recursive_Instantiation_of_Def_Process_AF_071112.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Recursive_Instantiation_of_Def_Process_AF_071112.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:SEBoKv075_KA-SystDef_Progressive_Approach_for_Designing.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv075_KA-SystDef_Progressive_Approach_for_Designing.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:SEBoKv075_KA-SystDef_Complete_Interface_Representation.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv075_KA-SystDef_Complete_Interface_Representation.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Decomposition_of_Functions_AF_071112(2).png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Decomposition_of_Functions_AF_071112(2).png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Illustration_of_a_scenario_(eFFBD)_AF_071112.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Illustration_of_a_scenario_(eFFBD)_AF_071112.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Illustration_of_a_scenario_Activity_Diagram_AF_071112.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Illustration_of_a_scenario_Activity_Diagram_AF_071112.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:SEBoKv075_KA-SystDef_Scenario_of_Operational_Modes.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv075_KA-SystDef_Scenario_of_Operational_Modes.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:SEBoKv05 KA-SystDef Temporal and decision hierarchy levels.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystDef_Temporal_and_decision_hierarchy_levels.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:SEBoKv075_KA-SystDef_Limited_nb_in_decomposition.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv075_KA-SystDef_Limited_nb_in_decomposition.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:SEBoKv075_KA-SystDef_Encapsulation.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv075_KA-SystDef_Encapsulation.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:JS_Figure_3.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:JS_Figure_3.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:SEBoKv05_KA-SystRealiz_how_outputs_of_Definition_relate_to_Implementation.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystRealiz_how_outputs_of_Definition_relate_to_Implementation.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:JS_Figure_5.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:JS_Figure_5.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:SEBoKv05_KA-SystRealiz_Implementation_relationships.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystRealiz_Implementation_relationships.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Limits_of_integration_activities.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Limits_of_integration_activities.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:SEBoKv05_KA-SystRealiz_Integration_relationships.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystRealiz_Integration_relationships.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:JS_Figure_9.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:JS_Figure_9.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Definition_and_usage_of_a_Verification_Action.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Definition_and_usage_of_a_Verification_Action.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Definition_and_usage_of_a_Validation_Action.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Definition_and_usage_of_a_Validation_Action.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Verification_and_Validation_level_per_level.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Verification_and_Validation_level_per_level.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:SEBoKv05_KA-SystRealiz_V&V_Actions_upper_levels.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystRealiz_V&V_Actions_upper_levels.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:SEBoKv05_KA-SystRealiz_V&V_Actions_lower_levels.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SystRealiz_V&V_Actions_lower_levels.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:052611_SJ_notional_reliability_analysis.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:052611_SJ_notional_reliability_analysis.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Affordable_Sys_Ops_Effect_DAU_GB_Roedler.jpg** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Affordable_Sys_Ops_Effect_DAU_GB_Roedler.jpg *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Sustainment_Implementation_Illustration_Logistics_Roedler.jpg** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Sustainment_Implementation_Illustration_Logistics_Roedler.jpg *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Scope_BoundariesSE_PM_SM.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Scope_BoundariesSE_PM_SM.png *License*: unknown *Contributors*: Bkcase, Smenck2, Smurawski

**File:semp_and_integrated_plans.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Semp_and_integrated_plans.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Measurement_Process_Model-Figure_1.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Measurement_Process_Model-Figure_1.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Composition_of_Leading_Indicator-Figure_2.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Composition_of_Leading_Indicator-Figure_2.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Technical_Measures_Relationship-Figure_3.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Technical_Measures_Relationship-Figure_3.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Decision_Mgt_Process_DM.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Decision_Mgt_Process_DM.png *License*: unknown *Contributors*: Smenck2

**File:Fund_Obj_Hierarchy_DM.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fund_Obj_Hierarchy_DM.png *License*: unknown *Contributors*: Smenck2

**File:Value_Function_Example_DM.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Value_Function_Example_DM.png *License*: unknown *Contributors*: Smenck2

**File:Swing_Weight_Matrix_DM.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Swing_Weight_Matrix_DM.png *License*: unknown *Contributors*: Smenck2

**File:Descript_of_Alt_DM.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Descript_of_Alt_DM.png *License*: unknown *Contributors*: Smenck2

**File:ALT_Scores_DM.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:ALT_Scores_DM.png *License*: unknown *Contributors*: Smenck2

**File:Value_Scorecard_w_Heat_Map_DM.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Value_Scorecard_w_Heat_Map_DM.png *License*: unknown *Contributors*: Smenck2

**File:Eq_1.jpg** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Eq_1.jpg *License*: unknown *Contributors*: Smenck2

**File:Eq_2.jpg** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Eq_2.jpg *License*: unknown *Contributors*: Smenck2

**File:Value_Comp_Graph_DM.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Value_Comp_Graph_DM.png *License*: unknown *Contributors*: Smenck2

**File:Ex_Stakeholder_Value_Scat_DM.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Ex_Stakeholder_Value_Scat_DM.png *License*: unknown *Contributors*: Smenck2

**File:Uncertainty_on_Perf_Value_from_Monte_DM.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Uncertainty_on_Perf_Value_from_Monte_DM.png *License*: unknown *Contributors*: Smenck2

**File:Cm_functions.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Cm_functions.png *License*: unknown *Contributors*: Bkcase, Smenck2, Smurawski

**File:Cm_change_control_process.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Cm_change_control_process.png *License*: unknown *Contributors*: Bkcase, Smenck2, Smurawski

**File:InfoMgtProcess_Fig1.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:InfoMgtProcess_Fig1.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:ArchitectureDesignConsid_Fig2.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:ArchitectureDesignConsid_Fig2.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:052411_BSBW_The_Vee_Model.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:052411_BSBW_The_Vee_Model.png *License*: unknown *Contributors*: Smenck2, Smurawski

**File:Breadth_and_Depth_of_Key_SE_Related_Standards.PNG** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Breadth_and_Depth_of_Key_SE_Related_Standards.PNG *License*: unknown *Contributors*: Bkcase, Janthony, Smurawski

**File:Standards_Alignment_Results_as_of_May_2011.PNG** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Standards_Alignment_Results_as_of_May_2011.PNG *License*: unknown *Contributors*: Bkcase, Janthony, Smurawski

**File:Fig_3_Growing_Industry_Collaboration_GR.png** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Fig_3_Growing_Industry_Collaboration_GR.png *License*: unknown *Contributors*: Bkcase, Smurawski

**File:Standards_Alignment_Results_as_of_May_2011a.PNG** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Standards_Alignment_Results_as_of_May_2011a.PNG *License*: unknown *Contributors*: Janthony, Smurawski

**File:Potential_Standards_Influence_of_Organization_and_Project_Processes.PNG** *Source*: https://www.sebokwiki.org/d/index.php?title=File:Potential_Standards_Influence_of_Organization_and_Project_Processes.PNG *License*: unknown *Contributors*: Janthony, Smurawski