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

Part 2

Guide to the Systems Engineering Body of Knowledge

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

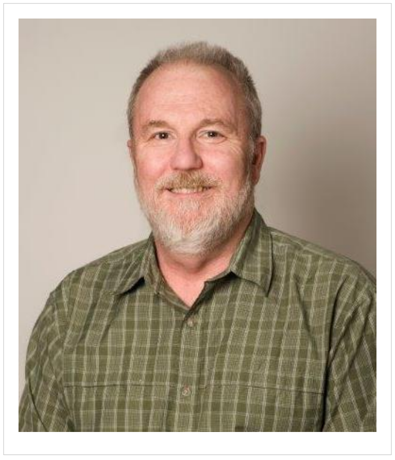
Past INCOSE governors Bill Miller, Kevin Forsberg, David Newbern, David Walden, Courtney Wright, Dave Olwell, Ken Nidiffer, Richard Fairley, Massood Towhidnejad, and John Keppler. The governors would also like to acknowledge John Keppler, IEEE Computer Society, who has been instrumental in helping the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

Part 2: Foundations of Systems Engineering

Foundations of Systems Engineering

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Scott Jackson, Janet Singer, Duane Hybertson*

Part 2 of the Guide to the SE Body of Knowledge (SEBoK) is a guide to foundational knowledge which is relevant or useful to systems engineering (SE).

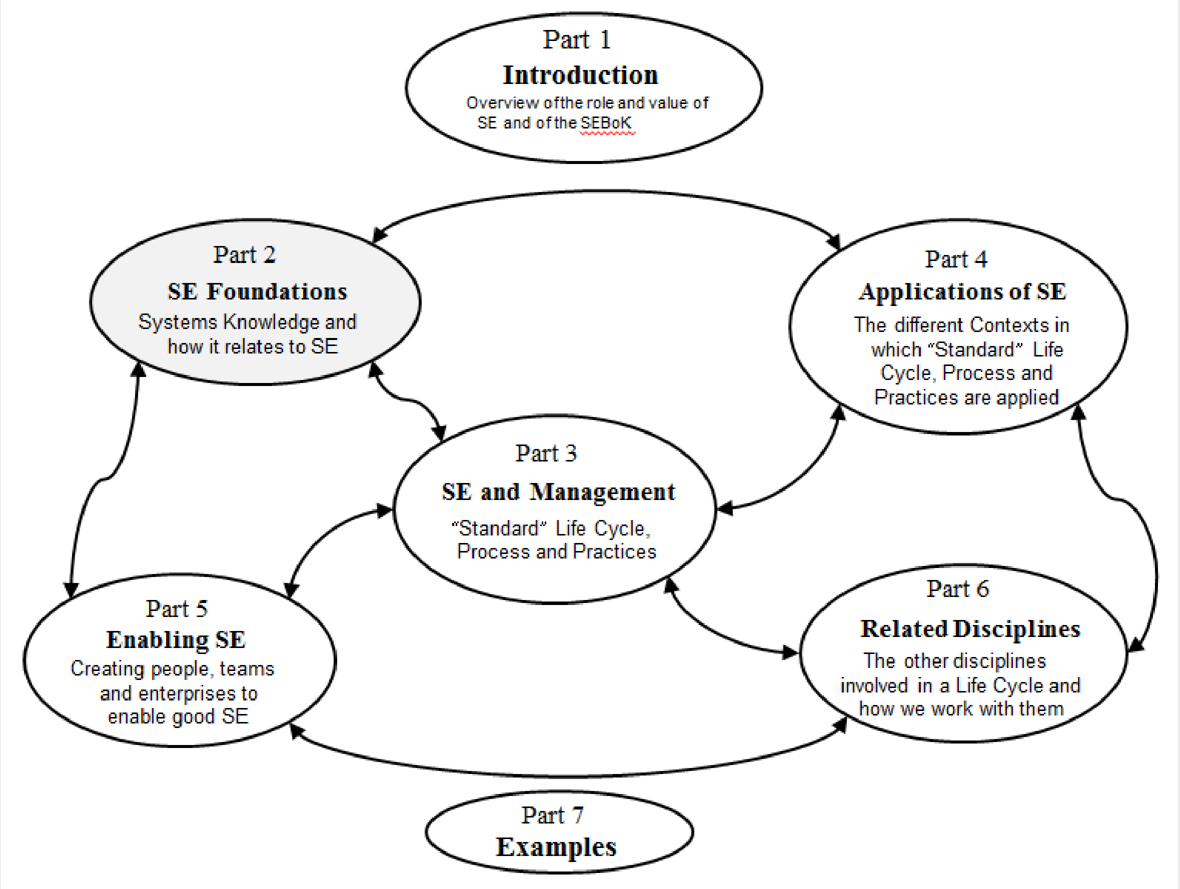

Part 1
Introduction
Overview of the role and value of SE and of the SEBoK
Part 2
SE Foundations
Systems Knowledge and how it relates to SE
Part 4
Applications of SE
The different Contexts in which "Standard" Life Cycle, Process and Practices are applied
Part 3
SE and Management
"Standard" Life Cycle, 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 2 in context (SEBoK Original). For more detail seeStructure of the SEBoK

This knowledge is included in the SEBoK firstly to help systems engineers benefit from an understanding of the foundations of their discipline, and to provide them with access to some of the theories and practices of systems science and other fields of systems practice. Including this wider integrative systems science context in the SEBoK should also help to make SE knowledge more accessible to a wider audience outside of its traditional domains.

Knowledge Areas in Part 2

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. Part 2 contains the following KAs:

- Systems Fundamentals
- Systems Approach Applied to Engineered Systems
- Systems Science
- Systems Thinking
- Representing Systems with Models

Introduction

Most systems engineers are practitioners, applying processes and methods that have been developed and evolved over decades. SE is a pragmatic approach, inherently interdisciplinary, yet specialized. Systems engineers usually work within a specific domain, using processes and methods that are tailored to their domain’s unique problems, constraints, risks and opportunities. These processes and methods have evolved to capture domain experts’ knowledge regarding the best approach to applying SE the particular domain.

Specific domains in which systems approaches are used and adapted include:

- Technology products, integrating multiple engineering disciplines
- Information-rich systems, e.g. command & control, air traffic management etc.
- Platforms, e.g. aircraft, civil airliners, cars, trains, etc.
- Organizational and enterprise systems, which may be focused on delivering service or capability
- Civil engineering/infrastructure systems, e.g. roads networks, bridges, builds, communications networks, etc.

The specific skill-sets for each domain, and the kinds and scales of system it considers, may be quite different. However, there are certain underlying unifying systems principles that can improve the effectiveness of the systems approach in any domain. In particular, shared knowledge of systems principles and terminology will enable communication and improve system engineers’ ability to integrate complex systems that span traditional domain boundaries (Sillitto 2012). This integrated approach is increasingly needed to solve today’s complex system challenges, but as these different communities come together they may find that assumptions underpinning their world-views are not shared.

General Systems Engineering Foundations

To bridge the gap between different domains and communities of practice, it is important to first establish a well-grounded definition of the “intellectual foundations of systems engineering”, as well as a common language to describe the relevant concepts and paradigms. An integrated systems approach for solving complex problems needs to combine elements of systems theories and systems approaches to practice. This may range from the technical-systems focus that has been dominant in systems engineering to the learning-systems focus of social systems intervention. An integrated systems approach needs to provide a framework and language that allow different communities, with highly divergent world-views and skill sets, to work together for a common purpose.

The SEBoK as a whole aims to provide principles and concepts which can be used to support all potential applications of systems engineering, and which can be easily translated to any particular application by the reader. Often the published knowledge related to systems engineering has been developed from particular application areas, typically combinations of applications like defense, transport, or medical, business models such as government, commercial or voluntary or technology domains such as mechanical, electrical or cyber. In publishing it authors will make some effort to **Specialize** it into knowledge which can be applied across related applications.

In the SEBoK we are looking to find or create **General** descriptions of SE knowledge. A general description should cover all applications of systems engineering and should include an explanation of the special cases it covers and how it applied to them. The generalization of knowledge can be informal, providing coverage of the most common specializations or being the domains current best understanding of the general case. A truly general description must be based upon stronger theoretical considerations and be in some sense proven to predict and cover all special cases. Knowledge described in the SEBoK will usually be informally generalized knowledge, with any specific knowledge being identified as such and related to the general as appropriate.

The INCOSE Vision 2025 includes an aim for systems engineering to be become a discipline with a formally defined theoretical basis. Such a general theory of SE would be largely included in SEBoK Part 2. The current SEBoK part 2 does not include such a theory. It provides generalized descriptions of foundational knowledge which has a pragmatic value to help describe and improve the current and future practice of systems engineering. We would expect any emerging general theory of systems engineering to draw from and expand these foundations. As such a theory is defined it will be included in Part 2 of the SEBoK.

The Systems Praxis Framework

The term **"systems praxis"** refers to the entire intellectual and practical endeavor for creating holistic solutions to today's complex system challenges. Praxis is defined as "translating an idea into action" (Wordnet 2012) and suggests that the best holistic approach to a given complex challenge may require integrating appropriate theory and appropriate practice from a wide variety of sources. Systems praxis requires many communities to work together. To work together we must first communicate; and to communicate, we must first connect.

A framework for unifying systems praxis was developed by members of International Council on Systems Engineering (INCOSE) and International Society for the System Sciences (ISSS) (International Federation for Systems Research (IFSR) 2012)) as the first step towards a "common language for systems praxis". This **Systems Praxis Framework** is included here because it represents current thinking on the foundations and common language of systems engineering, making the concepts and principles of systems thinking and practice accessible to anyone applying a systems approach to engineered system problems. This framework and thinking have been used to help organize the guide to systems knowledge in the SEBoK.

The diagram below shows the flows and interconnections among elements of a "knowledge ecosystem" of systems theory and practice.

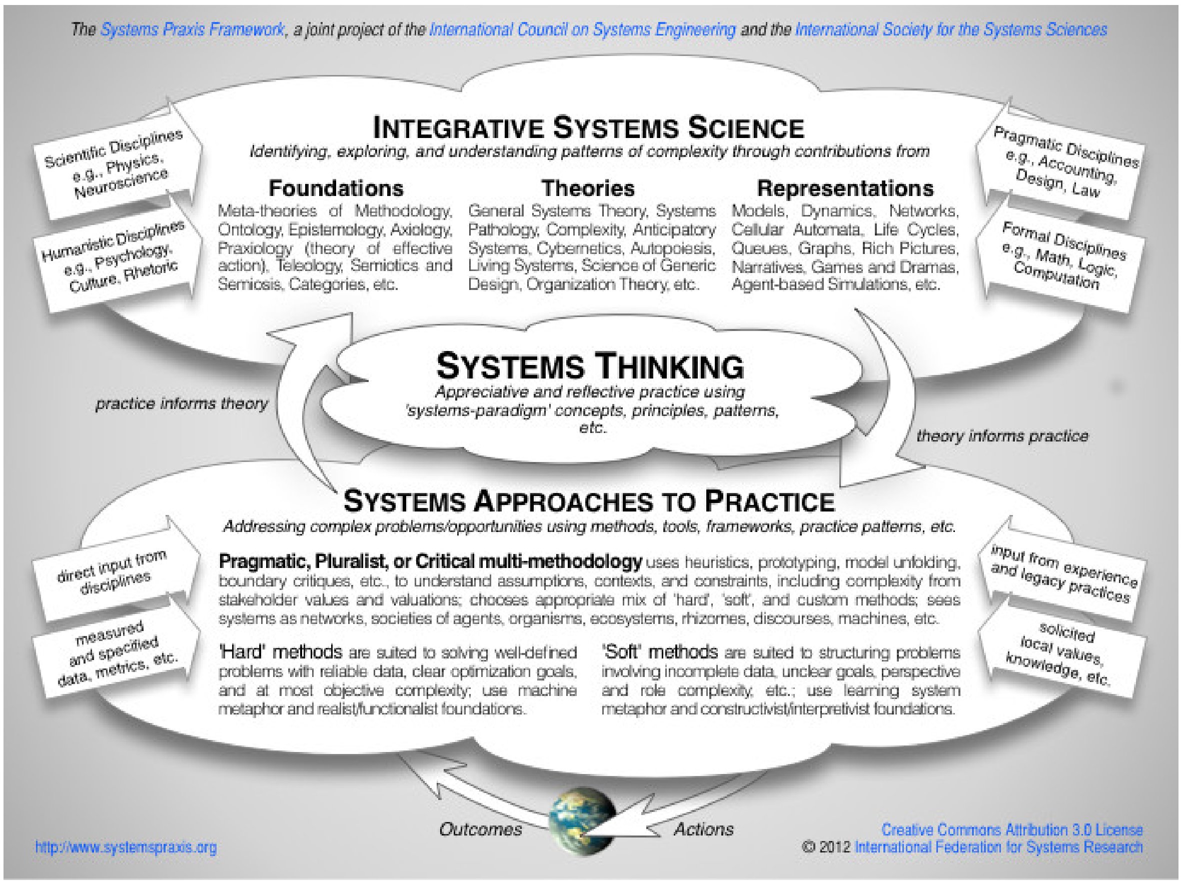

The Systems Praxis Framework, a joint project of the International Council on Systems Engineering and the International Society for the Systems Sciences
INTEGRATIVE SYSTEMS SCIENCE
Identifying, exploring, and understanding patterns of complexity through contributions from
Foundations
Meta-theories of Methodology, Ontology, Epistemology, Axiology, Praxiology (theory of effective action), Teleology, Semiotics and Semiosis, Categories, etc.
Theories
General Systems Theory, Systems Pathology, Complexity, Anticipatory Systems, Cybernetics, Autopoiesis, Living Systems, Science of Generic Design, Organization Theory, etc.
Representations
Models, Dynamics, Networks, Cellular Automata, Life Cycles, Queues, Graphs, Rich Pictures, Narratives, Games and Dramas, Agent-based Simulations, etc.
Scientific Disciplines e.g., Physics, Neuroscience
Humanistic Disciplines e.g., Psychology, Culture, Rhetoric
Pragmatic Disciplines e.g., Accounting, Design, Law
Formal Disciplines e.g., Math, Logic, Computation
SYSTEMS THINKING
Appreciative and reflective practice using 'systems-paradigm' concepts, principles, patterns, etc.
practice informs theory
theory informs practice
SYSTEMS APPROACHES TO PRACTICE
Addressing complex problems/opportunities using methods, tools, frameworks, practice patterns, etc.
Pragmatic, Pluralist, or Critical multi-methodology uses heuristics, prototyping, model unfolding, boundary critiques, etc., to understand assumptions, contexts, and constraints, including complexity from stakeholder values and valuations; chooses appropriate mix of 'hard', 'soft', and custom methods; sees systems as networks, societies of agents, organisms, ecosystems, rhizomes, discourses, machines, etc.
'Hard' methods are suited to solving well-defined problems with reliable data, clear optimization goals, and at most objective complexity; use machine metaphor and realist/functionalist foundations.
'Soft' methods are suited to structuring problems involving incomplete data, unclear goals, perspective and role complexity, etc.; use learning system metaphor and constructivist/interpretivist foundations.
direct input from disciplines
measured and specified data, metrics, etc.
input from experience and legacy practices
solicited local values, knowledge, etc.
Outcomes
Actions
http://www.systemspraxis.org
Creative Commons Attribution 3.0 License
© 2012 International Federation for Systems Research

Figure 1. The Systems Praxis Framework, Developed as a Joint Project of INCOSE and ISSS. (© 2012 International Federation for Systems Research) Released under Creative Commons Attribution 3.0 License. Source is available at http://systemspraxis.org/framework.pdf.

In this framework, the following elements are connected:

Systems Thinking is the core integrative element of the framework. It binds the foundations, theories and representations of systems science together with the hard, soft and pragmatic approaches of systems practice. In systems praxis, as in any practical discipline underpinned by science, there is constant interplay between theories and practice, with theory informing practice and outcomes from practice informing theory. Systems thinking is the ongoing activity of assessing and appreciating the system context, and guiding appropriate adaptation, throughout the praxis cycle.

Integrative Systems Science has a very wide scope and is grouped into three broad areas:

- **Foundations**, which help to organize knowledge and promote learning and discovery including: meta-theories of methodology, ontology, epistemology, axiology, praxiology (theory of effective action), teleology, semiotics & semiosis, category theory, etc.
- **Theories** pertaining to systems are abstracted from domains and specialties, so as to be universally applicable: general system theory, systems pathology, complexity, anticipatory systems, cybernetics, autopoiesis, living systems, science of generic design, organization theory, etc.
- **Representations** and corresponding theories describe, explore, analyze, and make predictions about systems and their wider contexts, whether in terms of models, dynamics, networks, cellular automata, life cycles, queues, graphs, rich pictures, narratives, games and dramas, agent-based simulations, etc.

Systems Approaches to Practice aim to act on real world experiences to produce desired outcomes without adverse, unintended consequences; ergo, practice needs to draw on the wide range of knowledge appropriate to the system-of-interest and its wider context. No one branch of systems science or practice provides a satisfactory explanation for all aspects of a typical system “problematique”; therefore, a more pragmatic approach is needed. Traditional systems approaches are often described to be either hard or soft:

- **Hard** approaches are suited to solving well-defined problems with reliable data and clear goals, using analytical methods and quantitative techniques. Strongly influenced by “machine” metaphors, they focus on technical systems, objective complexity, and optimization to achieve desired combinations of emergent properties. They are based on “realist” and “functionalist” foundations and worldview.
- **Soft** approaches are suited to structuring problems involving incomplete data, unclear goals, and open inquiries, using a “learning system” metaphor, focus on communication, intersubjective complexity, interpretations and roles, and draw on subjective and “humanist” philosophies with constructivist and interpretivist foundations.

Pragmatic (pluralist or critical) approaches judiciously select an appropriate set of tools and patterns that will give sufficient and appropriate insights to manage the issue at hand, by applying multiple methodologies drawn from different foundations as appropriate to the situation. Heuristics, boundary critiques, model unfolding, etc, enable the understanding of assumptions, contexts, and constraints, including complexity due to different stakeholders’ values and valuations. An appropriate mix of “hard”, “soft”, and custom methods draws on both systems and domain-specific traditions. Systems may be viewed as networks, societies of agents, organisms, ecosystems, rhizomes, discourses, machines, etc.

The set of “clouds” that collectively represents systems praxis is part of a wider ecosystem of knowledge, learning, and action. Successful integration with this wider ecosystem is the key to success with real world systems. Systems science is augmented by “hard” scientific disciplines, such as physics and neuroscience, and by formal disciplines, such as mathematics, logic and computation. It is both enhanced by, and used in, humanistic disciplines, such as psychology, culture, and rhetoric, and pragmatic disciplines, such as accounting, design, and law. Systems practice depends on measured data and specified metrics relevant to the problem situation and domain, the solicitation of local values and knowledge, and the pragmatic integration of experience, legacy practices, and discipline knowledge.

In summary, **Integrative Systems Science** allows us to identify, explore, and understand patterns of complexity through contributions from the foundations, theories, and representations of systems science and other disciplines relevant to the “problematique”. **Systems Approaches to Practice** address complex problems and opportunities using methods, tools, frameworks, patterns, etc., drawn from the knowledge of integrative systems science, while the observation of the results of systems practice enhances the body of theory. **Systems Thinking** binds the two together through appreciative and reflective practice using systems concepts, principles, patterns, etc.

Scope of Part 2

Part 2 of the SEBoK contains a guide to knowledge about systems, which is relevant to a better understanding of SE. It does not try to capture all of this systems knowledge here; rather, it provides an overview of a number of key aspects of systems theory and practice especially relevant to SE.

The organization of knowledge in Part 2 is based around the Praxis Framework discussed above (IFSR 2012). The need to develop a clear guide to the underpinning knowledge of SE is one of the motivations behind the praxis framework. It is expected that the coverage of systems knowledge will be significantly increased in future versions of the SEBoK as this work progresses.

The following diagram summarizes the way in which the knowledge in SEBoK Part 2 is organized.

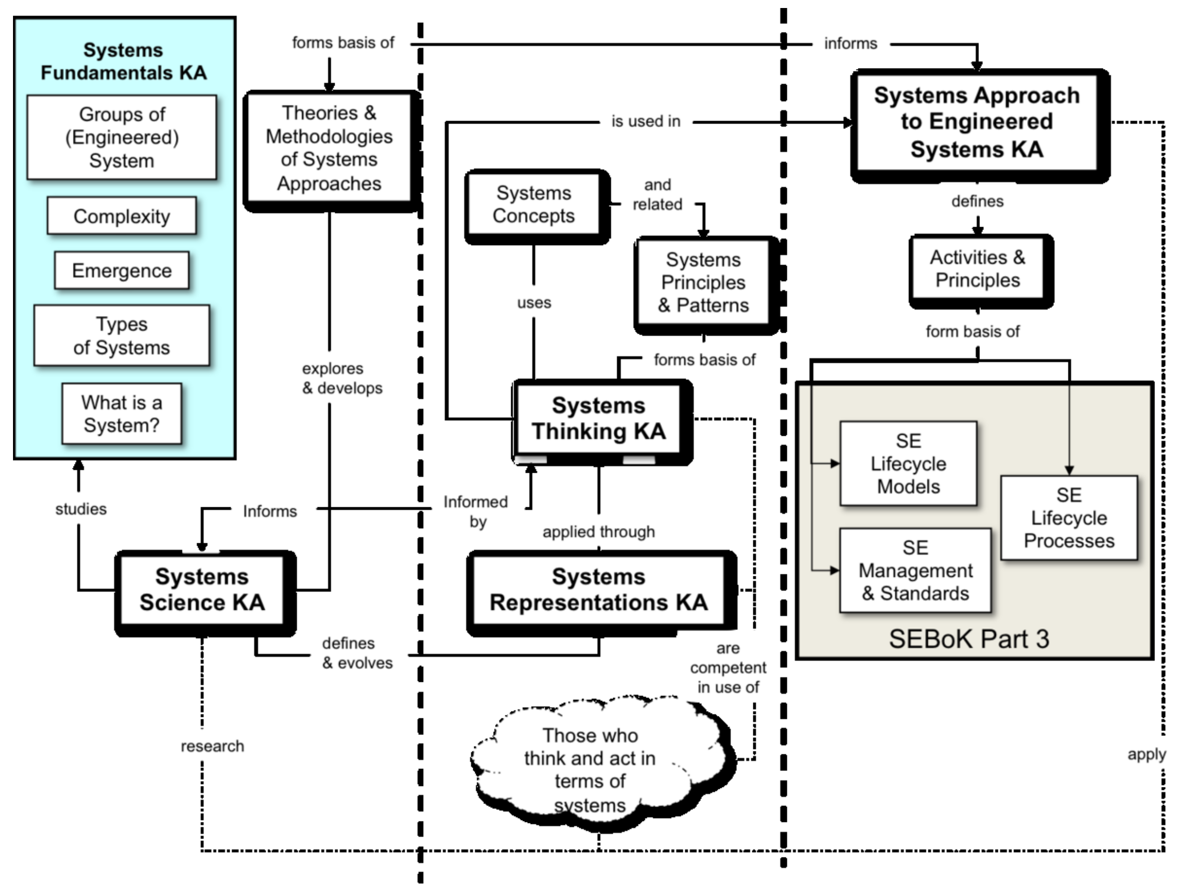


Figure 2. The Relationships between Key Systems Ideas and SE. (SEBoK Original)

The diagram is divided into five sections, each describing how systems knowledge is treated in the SEBoK.

1. The Systems Fundamentals Knowledge Area considers the question “What is a System?” It explores the wide range of system definitions and considers open systems, system types, groupings of systems, complexity, and emergence. All of these ideas are particularly relevant to engineered systems and to the groupings of such systems associated with the systems approach applied to engineered systems (i.e. product system, service system, enterprise system and system of systems).
2. The Systems Approach Applied to Engineered Systems Knowledge Area defines a structured approach to problem/opportunity discovery, exploration, and resolution, that can be applied to all engineered systems. The approach is based on systems thinking and utilizes appropriate elements of system approaches and representations. This KA provides principles that map directly to SE practice.
3. The Systems Science Knowledge Area presents some influential movements in systems science, including the chronological development of systems knowledge and underlying theories behind some of the approaches taken in applying systems science to real problems.
4. The Systems Thinking Knowledge Area describes key concepts, principles and patterns shared across systems research and practice.
5. The Representing Systems with Models Knowledge Area considers the key role that abstract models play in both the development of system theories and the application of systems approaches.

Systems thinking is a fundamental paradigm describing a way of looking at the world. People who think and act in a systems way are essential to the success of both the research and practice of system disciplines. In particular, individuals who have an awareness and/or active involvements in both research and practice of system disciplines are needed to help integrate these closely related activities.

The knowledge presented in this part of the SEBoK has been organized into these areas to facilitate understanding; the intention is to present a rounded picture of research and practice based on system knowledge. These knowledge areas should be seen together as a “system of ideas” for connecting research, understanding, and practice, based on system knowledge which underpins a wide range of scientific, management, and engineering disciplines and applies to all types of domains.

References

Works Cited

IFSR. 2012. *The Systems Praxis Framework, developed as a joint project of INCOSE and ISSS*. Vienna, Austria: International Federation for Systems Research (IFSR). Source is available at http://systemspraxis.org/framework.pdf.

Sillitto, H G, 2012. "Integrating Systems Science, Systems Thinking, and Systems Engineering: understanding the differences and exploiting the synergies", Proceedings of the 22nd INCOSE International Symposium, 9-12 July, 2012, Rome, Italy.

Wordnet. 2012. “Praxis.” Accessed 4/16/2013 at http:/ / wordnetweb. princeton. edu/ perl/ webwn?s=praxis& sub=Search+WordNet&o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&h=

Primary References

Bertalanffy, L., von. 1968. *General System Theory: Foundations, Development, Applications*, rev. ed. New York, NY, USA: Braziller.

Checkland, P. B. 1999. *Systems Thinking, Systems Practice.* Chichester, UK: John Wiley & Sons.

Additional References

Blanchard, B., and Fabrycky, W. 2010. *Systems Engineering and Analysis*, (5th edition). Saddle River, NJ, USA: Prentice Hall.

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

Martin J, Bendz J, Chroust G, Hybertson D, Lawson H, Martin R, Sillitto H, Singer J, Singer M, Takaku T. “Towards a Common Language for Systems Praxis”, proceedings of the 23rd INCOSE International Symposium, Philadelphia, June 2013.

MITRE Corporation. 2011. *Systems Engineering Guide: Comprehensive Viewpoint.*. Accessed 20 November 2014 at MITRE http:/ / www. mitre. org/ work/ systems_engineering/ guide/ enterprise_engineering/ comprehensive_viewpoint/

MITRE Corporation. 2011. *Systems Engineering Guide: Systems Thinking.*. Accessed 20 November 2014 at MITRE http:/ / www. mitre. org/ work/ systems_engineering/ guide/ enterprise_engineering/ comprehensive_viewpoint/ systems_thinking.html

Senge, P. M. 1990. *The Fifth Discipline: The Art & Practice of the Learning Organization.* New York, NY: Doubleday Business.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Knowledge Area: Systems Fundamentals

Systems Fundamentals

Lead Author: *Rick Adcock*, **Contributing Authors:** *Janet Singer, Duane Hybertson*

This knowledge area (KA) provides a guide to some of the most important knowledge about a system, which forms part of systems thinking and acts as a foundation for the related worlds of integrative systems science and systems approaches to practice.

This is part of the wider systems knowledge, which can help to provide a common language and intellectual foundation and make practical systems concepts, principles, patterns and tools accessible to systems engineering (SE) as discussed in Part 2: Foundations of Systems Engineering.

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:

- Introduction to System Fundamentals
- Types of Systems
- Complexity
- Emergence
- Fundamentals for Future Systems Engineering

Introduction

The word *system* is used in many areas of human activity and at many levels. But what do systems researchers and practitioners mean when they use the word *system*? Is there some part of that meaning common to all applications? The following diagram summarizes the ways in which this question is explored in this KA.

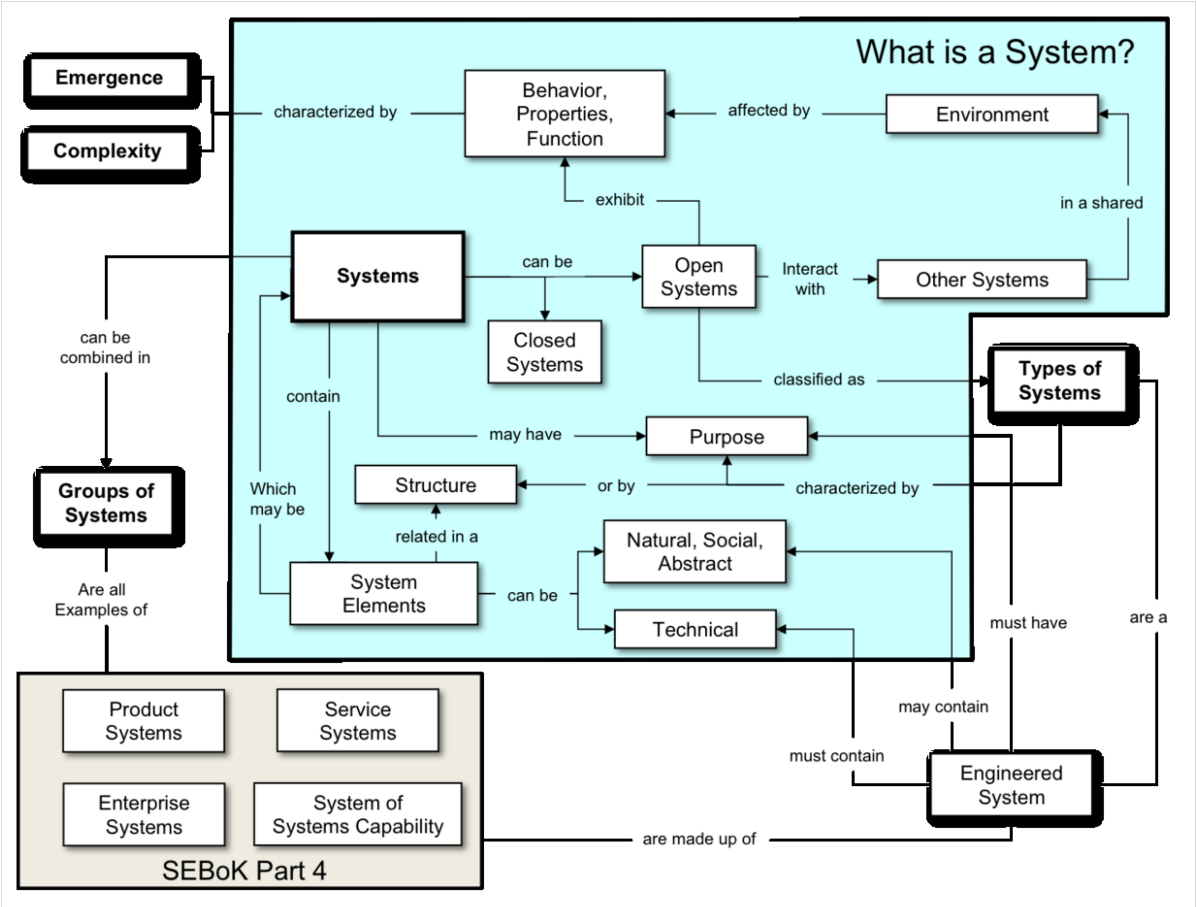


Figure 1. System Fundamentals and Engineered Systems. (SEBoK Original)

The concepts of open system and closed system are explored. Open systems, described by a set of elements and relationships, are used to describe many real world phenomena. Closed systems have no interactions with their environment. Two particular aspects of systems, complexity and emergence, are described in this KA. Between them, these two concepts represent many of the challenges which drive the need for systems thinking and an appreciation of systems science in SE.

Some systems classifications, characterized by type of element or by purpose, are presented.

Within the SEBoK an engineered system is defined as encompassing combinations of technology and people in the context of natural, social, business, public or political environments, created, used and sustained for an identified purpose. The application of the Systems Approach Applied to Engineered Systems requires the ability to position problems or opportunities in the wider system containing them, to create or change a specific engineered system-of-interest, and to understand and deal with the consequences of these changes in appropriate wider systems. The concept of a system context allows all of the system elements and relationships needed to support this to be identified.

The discussions of engineered system contexts includes the general idea of groups of systems to help deal with situations in which the elements of an engineered system are themselves independent engineered systems. To help provide a focus for the discussions of how SE is applied to real world problems, four engineered system contexts are introduced in the KA:

1. product system context
2. service system context
3. enterprise system context
4. system of systems (sos) context

The details of how SE is applied to each of these contexts are described in Part 4: Applications of Systems Engineering.

References

Works Cited

None.

Primary References

Bertalanffy, L., von. 1968. *General System Theory: Foundations, Development, Applications*, rev. ed. New York, NY, USA: Braziller.

Magee, C. L., O.L. de Weck. 2004. "Complex System Classification." Proceedings of the 14th Annual International Council on Systems Engineering International Symposium, 20-24 June 2004, Toulouse, France.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice*. Boca Raton, FL, USA: CRC Press.

Sheard, S.A. and A. Mostashari. 2009. "Principles of Complex Systems for Systems Engineering". *Systems Engineering*, 12(4): 295-311.

Tien, J.M. and D. Berg. 2003. "A Case for Service Systems Engineering". *Journal of Systems Science and Systems Engineering*, 12(1): 13-38.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Introduction to System Fundamentals

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Brian Wells, Scott Jackson, Janet Singer, Duane Hybertson*

This article forms part of the Systems Fundamentals knowledge area (KA). It provides various perspectives on systems, including definitions, scope, and context.

This article provides a guide to some of the basic concepts of systems developed by systems science and discusses how these relate to the definitions to be found in systems engineering (SE) literature. The concept of an engineered system is introduced as the system context of critical relevance to SE.

Overview

In the System Fundamentals KA we will define some terms and ideas which are foundational to the understanding and practice of Systems Engineering (SE). In particular, a number of views of system are explored; these are summarized below and described in more detail with links to relevant references in the rest of this article.

- A simple definition of **System is any set of related parts for which there is sufficient coherence between the parts to make viewing them as a whole useful**. If we consider more complex situations in which the parts of a system can also be viewed as systems we can identify useful common systems concepts to aid our understanding. This allows the creation of systems theories, models and approaches useful to anyone trying to understand, create or use collections of related things, independent of what the system is made of or the application domain considering it.
- Many of these common systems ideas relate to complex networks or hierarchies of related system elements. A **System Context is a set of system interrelationships associated with a particular system of interest (SoI) within a real world environment**. One or more views of a context allow us to focus on the SoI but not lose sight of its broader, holistic relationships and influences. Context can be used for many kinds of system but is particularly useful for scoping problems and enabling the creation of solutions which combine people and technology and operate in the natural world. These are referred to as socio-technical system contexts.
- Systems Engineering is one of the disciplines interested in socio-technical systems across their whole life. This includes where problems come from and how they are defined, how we identify and select candidate solutions, how to balance technology and human elements in the wider solution context, how to manage the complex organizational systems needed to develop new solutions, and how developed solutions are used, sustained and disposed of. To support this we define an **Engineered System as a socio-technical system which is the focus of a Systems Engineering life cycle.**
- While SE is focused on the delivery of an engineered system of interest a **SE should consider the full Engineered System Context so that the necessary understanding can be reached and the right systems engineering decisions can be made across each Life Cycle.**

A General View of Systems

The idea of a system whole can be found in both Western and Eastern philosophy. Many philosophers have considered notions of holism; that ideas, people or things must be considered in relation to the things around them to be fully understood (M'Pherson 1974).

One influential systems science definition of a system comes from general system theory (GST):

> "A System is a set of elements in interaction." (Bertalanffy 1968)

The parts of a system may be conceptual organizations of ideas in symbolic form or real objects. GST considers **abstract systems** to contain only conceptual elements and **concrete systems** to contain at least two elements that are real objects, e.g. people, information, software and physical artifacts, etc.

Similar ideas of wholeness can be found in systems engineering literature. For example:

> *We believe that the essence of a system is 'togetherness', the drawing together of various parts and the relationships they form in order to produce a new whole…* (Boardman and Sauser 2008).

The cohesive interactions between a set of parts suggest a system boundary and defines what membership of the system means. For **closed systems** all aspects of the system exist within this boundary. This idea is useful for abstract systems and for some theoretical system descriptions.

The boundary of an **open systems** (glossary) defines elements and relationships which can be considered part of the system and describe how these elements interact across the boundary with related elements in the environment (glossary). The relationships among the elements of an open system can be understood as a combination of the systems structure and behavior. The structure of a system describes a set of system elements and the allowable relationships between them. System behavior refers to the effects or outcomes produced when an instance of the system interacts with its environment. An allowable configuration of the relationships among elements is referred to as a system state. A stable system is one which returns to its original, or another stable, state following a disturbance in the environment. *System wholes entities often exhibit emergence, behavior which is meaningful only when attributed to the whole, not to its parts* (Checkland 1999).

The identification of a system and its boundary is ultimately the choice of the observer. This may be through observation and classification of sets of elements as systems, through an abstract conceptualisation of one or more possible boundaries and relationships in a given situation, or a mixture of this concrete and conceptual thinking. This underlines the fact that any particular identification of a system is a human construct used to help make better sense of a set of things and to share that understanding with others if needed.

Many natural, social and man made things can be better understood by viewing them as open systems. One of the reasons we find the idea of systems useful is that it is possible to identify shared concepts which apply to many system views. These recurring concepts or isomorphies can give useful insights into many situations, independently of the kinds of elements a particular system is made up of. The ideas of structure, behavior, emergence and state are examples of such concepts.The identification of these shared system ideas is the basis for Systems Thinking and their use in developing theories and approaches in a wide range of fields of study the system sciences.

Systems Engineering (SE), and a number of other Related Disciplines use systems concepts, patterns and models in the creation of useful outcomes or things. The concept of a network of open systems created, sustained and used to achieve a purpose within one or more environments is a powerful model that can be used to understand many complex real world situations and provide a basis for effective problem solving within them.

System Context

Bertalanffy (1968) divided open systems into nine real world types ranging from static structures and control mechanisms to socio-cultural systems. Other similar classification systems are discussed in the article Types of Systems.

The following is a simple classification of system elements which we find at the heart of many of these classifications:

- Natural system elements, objects or concepts which exist outside of any practical human control. Examples: the real number system, the solar system, planetary atmosphere circulation systems.
- Social system elements, either abstract human types or social constructs, or concrete individuals or social groups.
- Technological System elements, man-made artifacts or constructs; including physical hardware, software and information.

While the above distinctions can be made as a general abstract classification, in reality there are no hard and fast boundaries between these types of systems: e.g., natural systems are operated by, developed by, and often contain social systems and social systems depend on technical systems to fully realize their purpose. Systems which contain technical and either human or natural elements, are often called socio-technical systems. The behavior of such systems is determined both by the nature of the technical elements and by their ability to integrate with or deal with the variability of the natural and social systems around them.

Many of the original ideas upon which GST, and other branches of system study, are based come from the study of systems in the natural and social sciences. Many natural and social systems are initially formed as simple structures through the inherent cohesion among a set of elements. Once formed, they will tend to stay in this structure, as well as combine and evolve further into more complex stable states to exploit this cohesion in order to sustain themselves in the face of threats or environmental pressures. Such complex systems may exhibit specialization of elements, with elements taking on roles which contribute to the system purpose, but loosing some or all of their separate identify outside of the system. Such roles might include management of resources, defense, self-regulation or problem solving and control. Natural and social systems can be understood through an understanding of this wholeness, cohesion and specialization. They can also be guided towards the development of behaviors which not only enhance their basic survival, but also fulfill other goals of benefit to them or the systems around them. In *The Architecture of Complexity* (Simon 1962) has shown that natural or social systems which evolve via a series of stable "hierarchical intermediate forms" will be more successful and resilient to environmental change.

Thus, it is often true that the environment in which a particular system sits and the elements of that system can themselves be considered as open systems. It can be useful to consider collections of related elements as both a system and a part of one or more other systems. For example, a "holon" or system element was defined by Koestler as something which exists simultaneously a whole and as a part (Koestler 1967). At some point, the nature of the relationships between elements within and across boundaries in a hierarchy of systems may lead to complex structures and emergent behaviors which are difficult to understand or predict. Such complexity can often best be dealt with not only by looking for more detail, but also by considering the wider open system relationships.

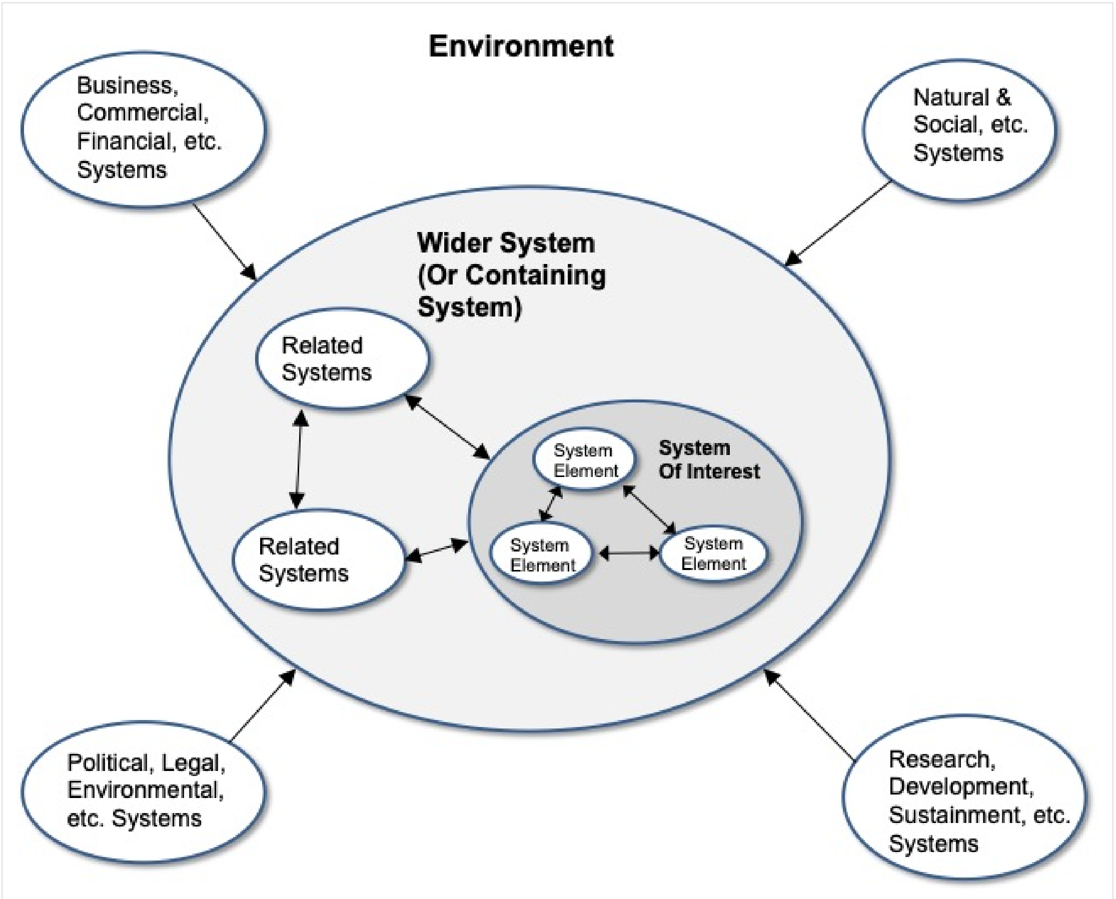

Environment
Business, Commercial, Financial, etc. Systems
Natural & Social, etc. Systems
Wider System (Or Containing System)
Related Systems
Related Systems
System Of Interest
System Element
System Element
System Element
Political, Legal, Environmental, etc. Systems
Research, Development, Sustainment, etc. Systems

Figure 1: General description of System Context (SEBoK Original)

A system context describes all of the external elements which interact across the boundary of a particular system of interest (SoI) and a sufficient view of the elements within its boundary, to allow the SoI to be better understood as part of a wider systems whole. To fully understand the context we also need to identify the environment in which the SoI and wider system sit and the systems in the environment which influence them.

Many man-made systems are designed as networks and hierarchies of related system elements to achieve desirable behaviors and the kinds of the resilience seen in natural systems. While such systems can be deliberately created to take advantage of system properties such as holism and stability, they must also consider system challenges such as complexity and emergence. Considering different views of a SoI and its context over its life can help enable this understanding. Considering systems in context allows us to focus on a SoI while maintaining the necessary wider, holistic systems perspective. This is one of the foundations of the Systems Approach described in SEBoK part 2, and forms a foundation of systems engineering.

Systems and Systems Engineering

Some of the systems ideas discussed above form part of the systems engineering body of knowledge. Systems engineering literature, standards and guides often refer to "the system" to characterize a socio-technical system with a defined purpose as the focus of SE, e.g.

- "A system is a value-delivering object" (Dori 2002).
- "A system is an array of components designed to accomplish a particular objective according to plan" (Johnson, Kast, and Rosenzweig 1963).
- "A system is defined as a set of concepts and/or elements used to satisfy a need or requirement" (Miles 1973).

The International Council on Systems Engineering Handbook (INCOSE 2015) generalizes this idea, defining system as "an interacting combination of elements to accomplish a defined objective. These include hardware, software, firmware, people, information, techniques, facilities, services, and other support elements." While these definition covers the socio-technical systems created by SE it is also necessary to consider the natural or social problem situation in which these system sits, the social systems which developed, sustained and used them, and the commercial or public enterprises in which these all sit as systems (Martin 2004).

Hence, while many SE authors talk about systems and systems ideas they are often based on a particular world view which related to engineered artifacts. It would also be useful to take a broader view of the context in which these artifacts sit, and to consider through life relationships as part of that context. To help promote this the SEBoK will try to be more precise with its use of the word system, and distinguish between general systems principles and the specific socio-technical systems created by SE.

The term socio-technical system is used by many in the systems community and may have meanings outside of that relevant to SE. Hence, we will define an engineered system as a socio-technical system forms the primary focus or system of interest (SoI) for an application of SE. A SE life cycle will consider an engineered system context, from initial problem formulation through to final safe removal from use (INCOSE 2015). A more detailed discussion of engineered system context and how it relates to the foundations of systems engineering practice can be found below.

Introduction to Engineered Systems

An engineered system defines a context containing both technology and social or natural elements, developed for a defined purpose by an engineering life cycle.

Engineered System contexts:

- are created, used and sustained to achieve a purpose, goal or mission that is of interest to an enterprise, team, or an individual.
- require a commitment of resources for development and support.
- are driven by stakeholders (glossary) with multiple views on the use or creation of the system, or with some other stake in the system, its properties or existence.
- contain engineered hardware, software, people, services, or a combination of these.
- exist within an environment that impacts the characteristics, use, sustainment and creation of the system.

Engineered systems typically

- are defined by their purpose, goal or mission.
- have a life cycle (glossary) and evolution dynamics.
- may include human operators (interacting with the systems via processes) as well as other social and natural components that must be considered in the design and development of the system.
- are part of a system-of-interest hierarchy.

Open systems are a useful way to understand many complex situations. Traditional engineering disciplines have become very good at building up detailed models and design practices to deal with the complexity of tightly integrated collections of elements within a technology domain and it is possible to model the seemingly random integration of lots of similar elements using statistical approaches. Systems Engineering makes use of both these aspects of system complexity, as discussed in the Complexity article.

SE also considers the complexity of relatively small numbers of elements taken from a range of design disciplines together with people who may not always be experienced or have detailed training in their use. Such engineered systems may be deployed in uncertain or changing environments and be used to help people achieve a number of loosely defined outcomes. For these systems relatively small changes in the internal working of their elements, or in how those elements are combined, may lead to the emergence of complex or un-expected outcomes. It can be difficult to predict and design for all such outcomes during an engineered systems creation, or to respond to them during its use. Iterative life cycle approaches which explore the complexity and emergence over a number of cycles of development and use are needed to deal with this aspect of complexity. The ways that system engineering deals with these aspects of complexity in the definition of life cycle and life cycle processes applied to an engineered system context is fully explored in Part 3

Life Cycle Definitions

As well as being a kind of system an engineered system is also the focus of a life cycle and hence part of a commercial transaction. Historically,

> *Economists divide all economic activity into two broad categories, goods and services. Goods-producing industries are agriculture, mining, manufacturing, and construction; each of them creates some kind of tangible object. Service industries include everything else: banking, communications, wholesale and retail trade, all professional services such as engineering, computer software development, and medicine, nonprofit economic activity, all consumer services, and all government services, including defense and administration of justice....* (Encyclopedia Britannica 2011).

The following diagram defines some terms related to an engineered system life cycle and the development of goods (products) and services.

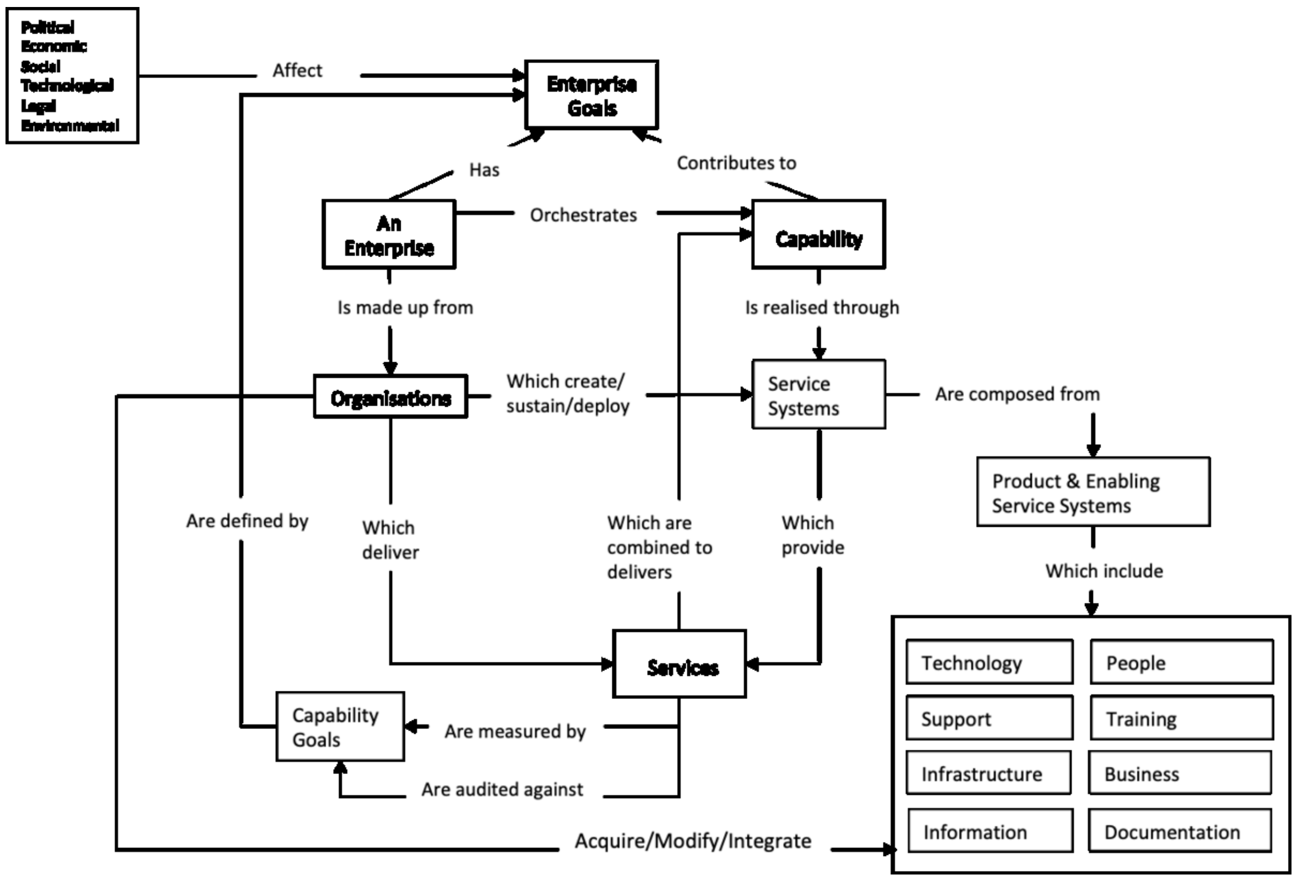


Figure 2: Life Cycle Terminology (Modified from Capability Engineering – an Analysis of Perspectives (modified from (Henshaw et al, 2011), used with permission))

In the above figure the capability needed to enable an enterprise to achieve its goals is delivered by the synchronized use of services. Those services are provided by service system which are created, sustained and deployed by one or more organisations. A service system is composed from people, technology, information, and access to related services and other necessary resources. Some of these resources are provided by enabling services and the technological elements may be developed and supplied as product systems. An enterprise system describes a collection of related capabilities and associated services which together enable the achievement of the overall purpose of an enterprise as a government, business or societal entity. Measurement and review of enterprise goals may define needs for change which require an organisation to acquire or modify, and integrate the elements needed to evolve its service systems. The general terminology above is described briefly in the associated glossary definitions and expanded in related articles in Part 4: Applications of Systems Engineering.

Engineered System Context

Engineered systems are developed as combinations of products and services within a life cycle. The figure below gives a general view of the full context for any potential application of a SE life cycle.

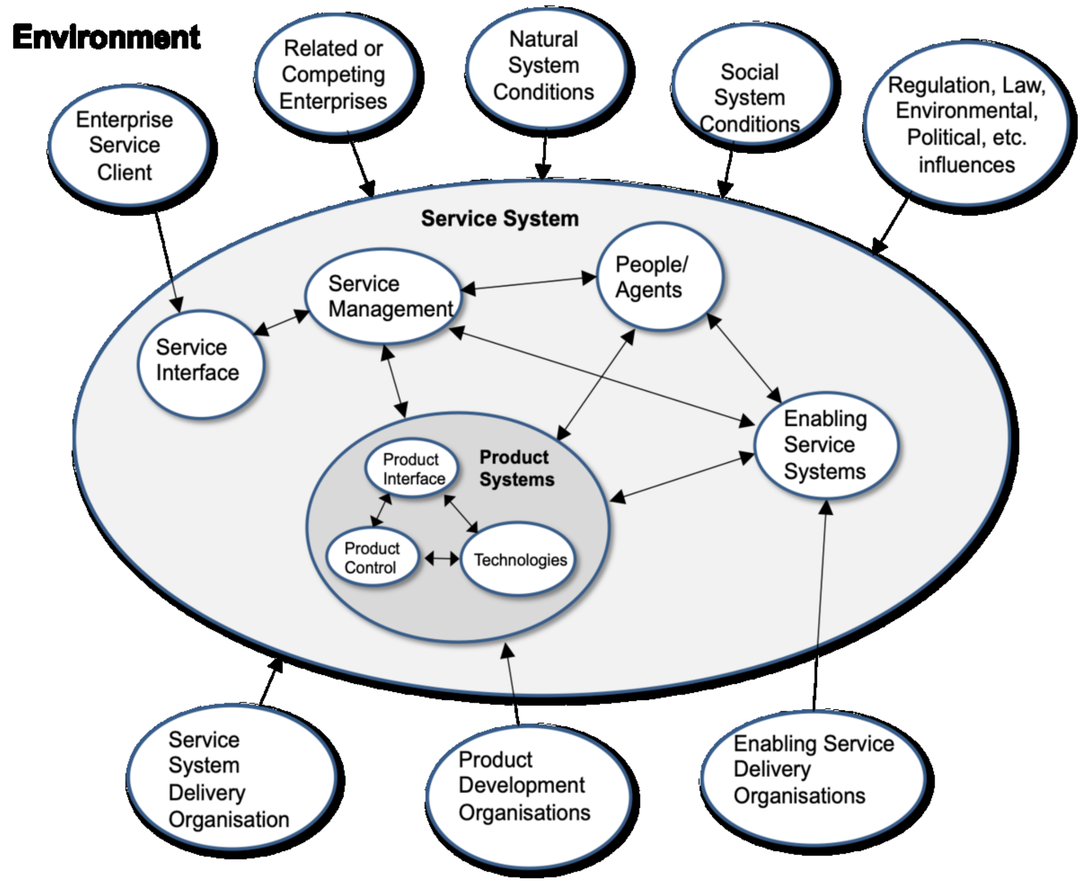

Environment
Enterprise Service Client
Related or Competing Enterprises
Natural System Conditions
Social System Conditions
Regulation, Law, Environmental, Political, etc. influences
Service System
Service Management
People/ Agents
Service Interface
Enabling Service Systems
Product Systems
Product Interface
Product Control
Technologies
Service System Delivery Organisation
Product Development Organisations
Enabling Service Delivery Organisations

Figure 3: General Engineered System Context (SEBoK original)

In this view a service system related directly to a capability need sets the overall boundary. This need establishes the problem situation or opportunity which encapsulates the starting point of any life cycle. Within this service system are the related services, products and people (or intelligent software agents) needed to fully deliver a solution to that need. The environment includes any people, organisations, rules or conditions which influence or constrain the service system or the things within it. The SoI for a particular SE life cycle may be defined at any level of this general context. While the focus of the context will vary for each life cycle it is important that some version of this general context is considered for all SE life cycles, to help maintain a holistic view of problem and solution. This is discussed in Types of Systems.

An engineered system context describes the context for a SoI so that the necessary understanding can be reached and the right systems engineering decisions can be made across the life of that SoI. This will require a number of different views of the context across a SE life cycle, both to identify all external influence on the SoI and to guide and constraint the systems engineering of the elements of the SoI. A full engineered systems context will include the problem situation from which a need for a SoI is identified, one or more socio technical solutions, the organizations needed to create and sustain new solutions and the operational environment within which those solutions must be integrated, used and eventually disposed of. The kinds of views which can be used to represent a SoI context over its life and how those views can be combined into models is discussed in the Representing Systems with Models KA in Part 2. The activities which use those models are described conceptually in the Systems Approach Applied to Engineered Systems KA in part 2 and related to more formal SE life cycle processes in Part 3.

by Janet Singer, Duane Hybertson, and Rick Adcock

References

Works Cited

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications*, rev. ed. New York: Braziller.

Boardman, J. and B. Sauser. 2008. *Systems Thinking: Coping with 21st Century Problems.* Boca Raton, FL, USA: Taylor & Francis.

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: Wiley and Sons, Inc.

Dori, D. 2002. *Object-Process Methodology – A Holistic Systems Paradigm*. Verlag, Berlin, Heidelberg, New York: Springer.

Henshaw, M., D. Kemp, P. Lister, A. Daw, A. Harding, A. Farncombe, and M. Touchin. 2011. "Capability Engineering - An Analysis of Perspectives." Presented at International Council on Systems Engineering (INCOSE) 21st International Symposium, June 20-23, 2011, Denver, CO, USA.

Hitchins, D. 2009. “What Are the General Principles Applicable to Systems?” INCOSE *Insight*, 12(4): 59-63.

INCOSE. 2015. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 4.0. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Johnson, R.A., F.W. Kast, and J.E. Rosenzweig. 1963. *The Theory and Management of Systems.* New York, NY, USA: McGraw-Hill Book Company.

Koestler, A. 1990. *The Ghost in the Machine,* 1990 reprint ed. Penguin Group.

Martin, J, 2004. "The Seven Samurai of Systems Engineering: Dealing with the Complexity of 7 Interrelated Systems". Proceedings of the 14th Annual International Council on Systems Engineering International Symposium, 20-24 June, 2004, Toulouse, France.

Miles, R.F. (ed). 1973. *System Concepts.* New York, NY, USA: Wiley and Sons, Inc.

M’Pherson, P.K. 1974. "A perspective on systems science and systems philosophy". *Futures*. 6(3):219-39.

Simon, H.A. 1962. "The Architecture of Complexity." *Proceedings of the American Philosophical Society.* 106(6) (Dec. 12, 1962): 467-482.

Primary References

Bertalanffy, L., von. 1968. *General System Theory: Foundations, Development, Applications*, rev. ed. New York, NY, USA: Braziller.

INCOSE. 2015. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 4.0. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

Additional References

Hybertson, Duane. 2009. *Model-oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boca Raton, FL, USA: CRC Press.

Hubka, Vladimir, and W. E. Eder. 1988. *Theory of Technical Systems: A Total Concept Theory for Engineering Design*. Berlin: Springer-Verlag.

Laszlo, E., ed. 1972. *The Relevance of General Systems Theory: Papers Presented to Ludwig von Bertalanffy on His Seventieth Birthday.* New York, NY, USA: George Brazillier.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Types of Systems

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Brian Wells, Scott Jackson*

This article forms part of the Systems Fundamentals knowledge area (KA). It provides various perspectives on system classifications and types of systems, expanded from the definitions presented in What is a System?.

The modern world has numerous kinds of systems that influence daily life. Some examples include transport systems; solar systems; telephone systems; the Dewey Decimal System; weapons systems; ecological systems; space systems; etc. Indeed, it seems there is almost no end to the use of the word “system” in today’s society.

This article considers the different classification systems which some systems science authors have proposed in an attempt to extract some general principles from these multiple occurrences. These classification schemes look at either the kinds of elements from which the system is composed or its reason for existing.

The idea of an engineered system (glossary) is expanded. Four specific types of engineered system context are generally recognized in systems engineering: product system, service system, enterprise system and system of systems.

System Classification

A taxonomy is "a classification into ordered categories" (Dictionary.com 2011). Taxonomies are useful ways of organizing large numbers of individual items so their similarities and differences are apparent. No single standard systems taxonomy exists, although several attempts have been made to produce a useful classification taxonomy, e.g. (Bertalanffy 1968) and (Miller 1986).

Kenneth Boulding (Boulding 1956), one of the founding fathers of general system theory, developed a systems classification which has been the starting point for much of the subsequent work. He classifies systems into nine types:

1. Structures (Bridges)
2. Clock works (Solar system)
3. Controls (Thermostat)
4. Open (Biological cells)
5. Lower organisms (Plants)
6. Animals (Birds)
7. Man (Humans)
8. Social (Families)
9. Transcendental (God)

These approaches also highlight some of the subsequent issues with these kinds of classification. Boulding implies that physical structures are closed and natural while social ones are open. However, a bridge can only be understood by considering how it reacts to traffic crossing it, and it must be sustained or repaired over time (Hitchins 2007). Boulding also separates humans from animals, which would not fit into more modern thinking.

Peter Checkland (Checkland 1999, 111) divides systems into five classes: natural systems, designed physical systems, designed abstract systems, human activity systems and transcendental systems. The first two classes are self-explanatory.

- **Designed abstract systems** – These systems do not contain any physical artifacts but are designed by humans to serve some explanatory purpose.
- **Human activity systems** – These systems are observable in the world of innumerable sets of human activities that are more or less consciously ordered in wholes as a result of some underlying purpose or mission. At one extreme is a system consisting of a human wielding a hammer. At the other extreme lies international political systems.
- **Transcendental systems** – These are systems that go beyond the aforementioned four systems classes, and are considered to be systems beyond knowledge.

Checkland refers to these five systems as comprising a "systems map of the universe". Other, similar categorizations of system types can be found in (Aslaksen 1996), (Blanchard 2005) and (Giachetti 2009).

Magee and de Weck (Magee and de Weck 2004) provide a comprehensive overview of sources on system classification such as (Maier and Rechtin 2009), (Paul 1998) and (Wasson 2006). They cover some methods for classifying natural systems, but their primary emphasis and value to the practice of systems engineer is in their classification method for human-designed, or man-made, systems. They examine many possible methods that include: degree of complexity, branch of the economy that produced the system, realm of existence (physical or in thought), boundary, origin, time dependence, system states, human involvement / system control, human wants, ownership and functional type. They conclude by proposing a functional classification method that sorts systems by their process (transform, transport, store, exchange, or control), and by the entity that they operate on matter, energy, information and value.

Types of Engineered System

The figure below is a general view of the context for any potential application of an engineered system life cycle.

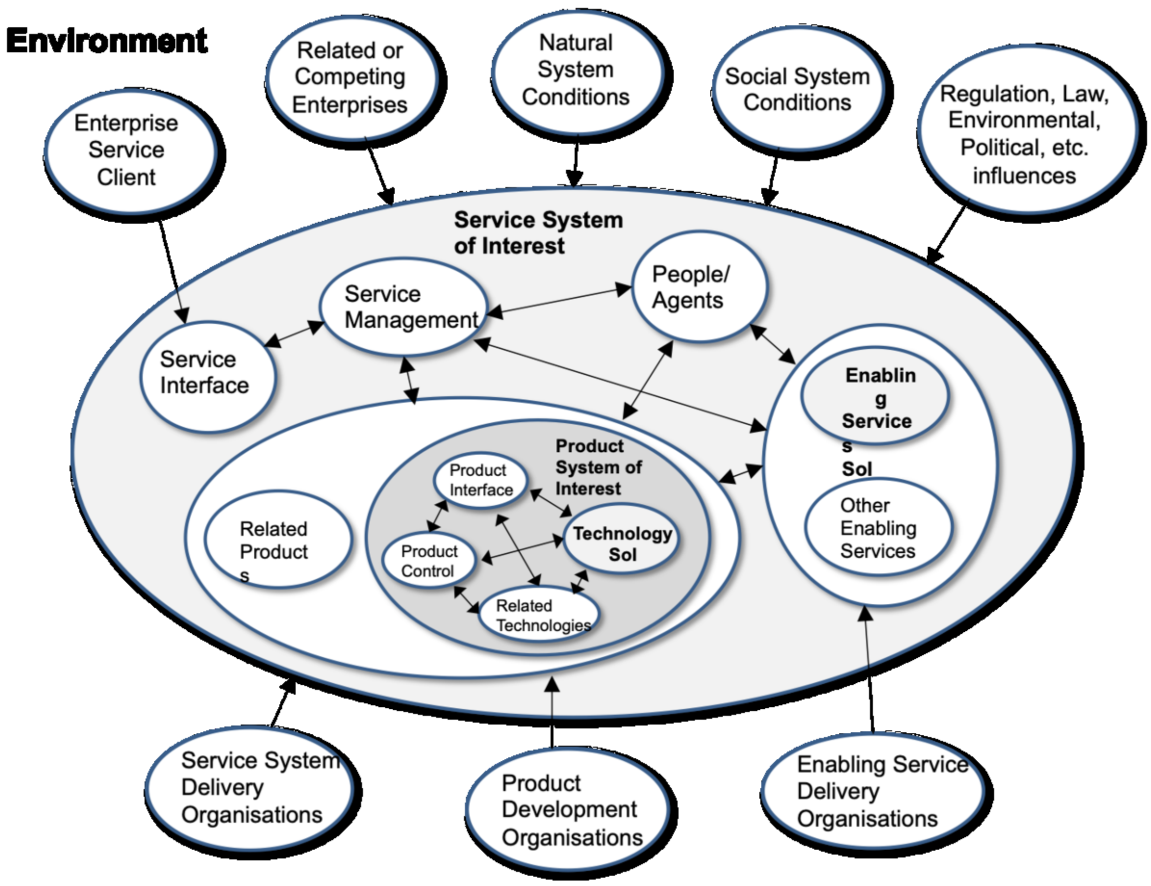

Environment
Enterprise Service Client
Related or Competing Enterprises
Natural System Conditions
Social System Conditions
Regulation, Law, Environmental, Political, etc. influences
Service System of Interest
Service Interface
Service Management
People/ Agents
Enablin g Service s SoI
Other Enabling Services
Related Product s
Product System of Interest
Product Interface
Product Control
Technology SoI
Related Technologies
Service System Delivery Organisations
Product Development Organisations
Enabling Service Delivery Organisations

Figure 1: General types of Engineered System of Interest (SoI) (SEBoK original)

Figure 1 shows four general cases of system of interest (SoI) which might be the focus of a life cycle.

- A technology focused product system SoI embedded within one or more integrated products,
- An integrated multi-technology product system SoI used directly to help provide a service,
- An enabling service system SoI supporting multiple service systems
- A service system SoI created and sustained to directly deliver capability.

Products and Product Systems

The word product is defined as "a thing produced by labor or effort; or anything produced" (Oxford English Dictionary). In a commercial sense a product is anything which is acquired, owned and sustained by an organization and used by an enterprise (hardware, software, information, personnel, etc.).

A product systems is an engineered systems in which the focus of the life cycle is to developed and delivered products to an acquirer for internal or external use to directly support the delivery of services needed by that acquirer.

A product systems life cycle context will describe a technology focused SoI plus the related products, people and services with which the SoI is required to interact. Note, the people associated with a product system over its life (e,g, operators, maintainers, producers, etc.) sit outside of the product SoI, since they are not delivered as part of the product. However, to develop a successful product it is essential to fully understand its human interfaces and influences as part of its context. The product context will also define the service systems within which it will be deployed to help provide the necessary capability (glossary) to the acquiring enterprise.

In a product life cycle this wider context defines the fixed and agreed relationships within which the SoI must operate, and the environmental influences within which the life cycle must be delivered. This gives the product developer the freedom to make solution choices within that context and to ensure these choices fit into and do not disrupt the wider context.

A product life cycle may need to recommend changes to enabling services such as recruitment and training of people, or other infrastructure upgrades. Appropriate mechanisms for the implementation of these changes must be part of the agreement between acquirer and supplier and be integrated into the product life cycle. A product life cycle may also suggest changes in the wider context which would enhance the products ownership or use, but those changes need to be negotiated and agreed with the relevant owners of the systems they relate to before they can be added to the life cycle outputs.

A more detailed discussion of the system theory associated with product systems can be found in History of Systems Science and an expansion of the application of systems engineering to service systems in the Product Systems Engineering KA in Part 4.

Services and Service Systems

A service (glossary) can be simply defined as an act of help or assistance, or as any outcome required by one or more users which can be defined in terms of outcomes and quality of service without detail to how it is provided (e.g., transport, communications, protection, data processing, etc.). Services are processes, performances, or experiences that one person or organization does for the benefit of another, such as custom tailoring a suit; cooking a dinner to order; driving a limousine; mounting a legal defense; setting a broken bone; teaching a class; or running a business’s information technology infrastructure and applications. In all cases, service involves deployment of knowledge and skills (competencies) that one person or organization has for the benefit of another (Lusch and Vargo 2006), often done as a single, customized job. To be successful, service requires substantial input from the client and related stakeholder, often referred to as the co-creation of value (Sampson 2001). For example, how can a steak be customized unless the customer tells the waiter how the customer wants the steak prepared?

A service system (glossary) is an engineered system created and sustained by an organization that provides outcomes for clients within an enterprise. A service system context contains the same kinds of system elements as a product system context, but allows greater freedom for what can be created or changed to deliver the required service.

A service system life cycle may deliver changes to how existing products and other services are deployed and used. It may also identify the need to modify existing products or create new products, in which case it may initiate a related product life cycle. In most cases the service developer will not have full freedom to change all aspects of the service system context without some negotiation with related system element owners. In particular people and infrastructure are part of the service context and changes to how system elements are used to provide desired outcomes are part of the service life cycle scope.

The description of product system context above might be viewed as a special case of a service system context in which a specific product is created and integrated into a fixed service system by an organization and used by an enterprise directly related to the organisation to provide a capability.

In a general service system context it is not necessary to deliver all hardware or software products to the service provider. In some cases some of the hardware, software or human elements may be owned by a third party who is not responsible for the service directly, but provides enabling outputs to a number of such services. In other cases the whole service may be provided by an organization that is completely separate to the enterprise which needs the service. Nor is it necessary for the exact versions of products or enabling services to be defined and integrated prior the service delivery. Some service system elements can be selected and integrated closer to the point of use. To allow for this late configuration of a service system it will contain some method of discovery, by which appropriate available elements can be found, and an overall service management element to implement and direct each instance of the service system. The use of a service system approach gives greater freedom for acquirers in how they obtain and support all of the capital equipment, software, people, etc. in order to obtain the capabilities needed to satisfy users.

Services have been part of the language of systems engineering (SE) for many years. Either as a way to describe the context of a product focused life cycle or to describe commercial arrangements for the 'out sourcing' of product ownership and operation to others. The use of the term service system in more recent times is often associated with software configurable and information intensive systems, i.e.,

> *...unique features that characterize services – namely, services, especially emerging services, are information-driven, customer-centric, e-oriented, and productivity-focused.* (Tien and Berg 2003, 13)

A more detailed discussion of the system theory associated with service systems can be found in History of Systems Science and an expansion of the application of systems engineering to service systems in the Service Systems Engineering KA in Part 4.

Enterprises and Enterprise Systems

An enterprise (glossary) is one or more organizations or individuals sharing a definite mission, goals, and objectives to offer an output such as a product or service.

An enterprise system (glossary) consists of a purposeful combination (network) of interdependent resources (e.g., people; processes; organizations; supporting technologies; and funding) that interact with each other (e.g., to coordinate functions; share information; allocate funding; create workflows; and make decisions), and their environment(s), to achieve business and operational goals through a complex web of interactions distributed across geography and time (Rebovich and White 2011, 4, 10, 34-35).

Both product and service systems require an enterprise system to create them and an enterprise to use the product system to deliver services either internally to the enterprise or externally to a broader community.

Enterprise systems are unique, compared to product and service systems, in that they are constantly evolving; they rarely have detailed configuration controlled requirements; they typically have the goal of providing shareholder value and customer satisfaction, which are constantly changing and are difficult to verify; and they exist in a context (or environment) that is ill-defined and constantly changing.

While an enterprise system cannot be described using the general system context above, an enterprise may wish to create a model of the capabilities and services it needs to achieve its strategy and goals. Such a model can be extended to describe a baseline of service system and product system contexts related to its current capabilities, and to proposed future capabilities. These are referred to as enterprise architectures or enterprise reference architectures.

A more detailed discussion of the system theory associated with service systems can be found in History of Systems Science and an expansion of the application of systems engineering to service systems in the Enterprise Systems Engineering KA in Part 4. The notion of enterprises and enterprise systems also permeates Part 5 Enabling Systems Engineering.

Systems of Systems

A product, service or enterprise context can be defined as a hierarchy of system elements, with the additional definition of which elements are part of a SoI solution, which form the related problem context and which influence any life cycle associated with that context.

The additional concepts of a Systems of Systems (SoS) or Federations of Systems (FoS) is used for some contexts. In terms of the general description in figure 1 above this would apply to any life cycle context in which elements within the SoI have independent life cycle relationships. This concept could apply to any of the life cycle contexts above, although it is of particular relevance to the service and enterprise contexts.

It is important to understand that the term SoS is an addition to the general concept of system hierarchy that applies to all system. Maier examined the meaning of System of Systems in detail and used a characterization approach which emphasizes the independent nature of the system elements, (Maier 1998, 268). Maier describes both independence in how a system element operates (e.g. an element in the SoI also has its own separate mission or is part of another SoI) and in how an element is developed or sustained (e.g. an element is made available, modified or configured by a different organization to the one responsible for the rest of SoI).

There are advantages to being able to have elements shared across a number of engineered systems and to being able to quickly create solutions to problems by combining existing engineered systems. As the technology to enable integration of independent systems becomes more common this SoS approach is becoming a common aspect of many SE life cycle.

Wherever system elements in an engineered system context have any degree of independence from the SoI life cycle this adds a further complexity; specifically, by constraining how the resulting engineered system can be changed or controlled. This dimension of complexity affects the management and control aspects of the systems approach.

A more detailed discussion of the different system grouping taxonomies developed by systems science can be found in Part 4 Applications of Systems Engineering and an expansion of the ways we deal with SoS complexity can be found n the Systems of Systems KA in part 4.

Applying Engineered System Contexts

From the discussions of product and service contexts above it should be clear that they require similar systems understanding to be successful and that the difference between them is more about the scope of life cycle choices and the authority to make changes than about what kinds of systems they are.

They are presented here as generalizations of the system engineering approach. All real projects may have both product and service system dimensions to them. In this general view of engineered systems there is always an enterprise system directly interested in the service system context and either directly owning and operating any product systems and enabling services or gaining access to them as needed. This enterprise system may be explicitly involved in initiating and managing an engineering system life cycle, or may be implicit in the shared ownership of a problem situation. Any engineered system context may have aspects of the SoS independence discussed above. This may be part of the context in the wider system or environment or it may be related to the choice of elements within the SoI.

A real SE life cycle typically combines different aspects of these general contexts into a unique problem and solution context and associated acquirer and supplier commercial relationships. These must be identified by that life cycle as part of its SE activities. More details of these different life cycle contexts are given in part 2 and how these apply to SE practice is expanded in Part 4.

A good example of a general description of the above is given by Ring (Ring, 1998), who defines the overall context as the Problem Suppression System, and describes a cycle by which an enterprise will explore its current needs, use these to identify one or more life cycle interventions, relevant organisations then conduct and deliver those life cycles, and integrate their outputs into the PSS, the enterprise can then review the results in the environment and begin the cycle again.

This general systems approach is described in part 2 and used as a focus to identify areas of foundational knowledge. The current practices of SE described in the rest of the SEBoK make reference to these foundations as appropriate.

References

Works Cited

Aslaksen, E.W. 1996. *The Changing Nature of Engineering*. New York, NY, USA: McGraw-Hill.

Bertalanffy, L. von. 1968. *General System Theory*. New York, NY, USA: Brazillier.

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.

Boulding, K. 1956 "General Systems Theory: Management Science, 2, 3 (Apr. 1956) pp.197-208; reprinted in General Systems, Yearbook of the Society for General Systems Research, vol. 1, 1956.

Checkland, P.B. 1999. *Systems Thinking, Systems Practice*. Chichester, UK: John Wiley & Sons Ltd.

Dictionary.com, s.v. "Taxonomy," Accessed December 3 2014 at Dictionary.com http://dictionary.reference.com/browse/taxonomy.

Encyclopedia Britannica, s.v. "Service Industry," Accessed December 3 2014 at Dictionary.com http://www.britannica.com/EBchecked/topic/535980/service-industry.

DeRosa, J. K. 2005. "Enterprise Systems Engineering." Air Force Association, Industry Day, Day 1, 4 August 2005, Danvers, MA.

Giachetti, R.E. 2009. *Design of Enterprise Systems: Theory, Architectures, and Methods.* Boca Raton, FL, USA: CRC Press.

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

Lusch, R.F. and S. L. Vargo (Eds). 2006. *The service-dominant logic of marketing: Dialog, debate, and directions*. Armonk, NY: ME Sharpe Inc.

Magee, C.L. and O.L. de Weck. 2004. "Complex System Classification". Proceedings of the 14th Annual International Symposium of the International Council on Systems Engineering, 20-24 June, 2004, Toulouse, France.

Maier, M. W. 1998. "Architecting Principles for Systems-of-Systems". *Systems Engineering*, 1(4): 267-84.

Maier, M., and E. Rechtin. 2009. *The Art of Systems Architecting*, 3rd Ed.. Boca Raton, FL, USA: CRC Press.

Miller J. G. 1986. "Can Systems Theory Generate Testable Hypothesis?: From Talcott Parsons to Living Systems Theory?" *Systems Research.* 3:73-84.

Paul, A.S. 1998. "Classifying Systems." Proceedings of The 8th Annual International Council on Systems Engineering International Symposium, 26-30 July, 1998, Vancouver, BC, Canada.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice*. Boca Raton, FL, USA: CRC Press.

Ring, J., 1998. "A Value Seeking Approach to the Engineering of Systems." Proceedings of the IEEE Conference on Systems, Man, and Cybernetics. p. 2704-2708.

Sampson, S.E. 2001. *Understanding Service Businesses*. New York, NY, USA: John Wiley.

Tien, J.M. and D. Berg. 2003. "A Case for Service Systems Engineering." *Journal of Systems Science and Systems Engineering*. 12(1): 13-38.

Wasson, C.S. 2006. *System Analysis, Design and Development.* Hoboken, NJ, USA: John Wiley and Sons.

Primary References

Checkland, P. B. 1999. *Systems Thinking, Systems Practice.* Chichester, UK: John Wiley & Sons.

Magee, C. L., O.L. de Weck. 2004. "Complex System Classification." Proceedings of the 14th Annual International Council on Systems Engineering International Symposium, 20-24 June 2004, Toulouse, France.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice*. Boca Raton, FL, USA: CRC Press.

Tien, J.M. and D. Berg. 2003. "A Case for Service Systems Engineering". *Journal of Systems Science and Systems Engineering.* 12(1): 13-38.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Complexity

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Hillary Sillitto, Sarah Sheard*

This article is part of the Systems Fundamentals knowledge area (KA). It gives the background and an indication of current thinking on complexity and how it influences systems engineering (SE) practice.

Complexity is one of the most important and difficult to define system concepts. Is a system's complexity in the eye of the beholder, or is there inherent complexity in how systems are organized? Is there a single definitive definition of complexity and, if so, how can it be assessed and measured? This topic will discuss how these ideas relate to the general definitions of a system given in What is a System?, and in particular to the different engineered system contexts. This article is closely related to the emergence topic that follows it.

Defining System Complexity

Complexity has been considered by a number of authors, from various perspectives, some of the discussions of complexity relevant to systems are described in the final section of this article. Sheard and Mostashari (Sheard and Mostashari 2011) synthesize many of these ideas to categorize complexity as follows:

1. **Structural Complexity** looks at the system elements and relationships. In particular, structural complexity looks at how many different ways system elements can be combined. Thus, it is related to the potential for the system to adapt to external needs.
2. **Dynamic Complexity** considers the complexity which can be observed when systems are used to perform particular tasks in an environment. There is a time element to dynamic complexity. The ways in which systems interact in the short term is directly related to system behavior; the longer term effects of using systems in an environment is related to system evolution.
3. **Socio-political Complexity** considers the effect of individuals or groups of people on complexity. People-related complexity has two aspects. One is related to the perception of a situation as complex or not, due to multiple stakeholder viewpoints within a system context and social or cultural biases which add to the wider influences on a system context. The other involves either the “irrational” behavior of an individual or the swarm behavior of many people behaving individually in ways that make sense; however, the emergent behavior is unpredicted and perhaps counterproductive. This latter type is based on the interactions of the people according to their various

interrelationships and is often graphed using systems dynamics formalisms.

Thus, complexity is a measure of how difficult it is to understand how a system will behave or to predict the consequences of changing it. It occurs when there is no simple relationship between what an individual element does and what the system as a whole will do, and when the system includes some element of adaptation or problem solving to achieve its goals in different situations. It can be affected by objective attributes of a system such as by the number, types of and diversity of system elements and relationships, or by the subjective perceptions of system observers due to their experience, knowledge, training, or other sociopolitical considerations.

This view of complex systems provides insight into the kind of system for which systems thinking and a systems approach is essential.

Complexity and Engineered Systems

The different perspectives on complexity are not independent when considered across a systems context. The Structural complexity of a system-of-interest (SoI) may be related to dynamic complexity when the SoI also functions as part of a wider system in different problem scenarios. People are involved in most system contexts, as part of the problem situation, as system elements and part of the operating environment. The human activity systems which we create to identify, design, build and support an engineered system and the wider social and business systems in which they sit are also likely to be complex and affect the complexity of the systems they produce and use.

Sheard and Mostashari (Sheard and Mostashari 2011) show the ways different views of complexity map onto product system, service system and enterprise system contexts, as well as to associated development and sustainment systems and project organizations. Ordered systems occur as system components and are the subject of traditional engineering. It is important to understand the behaviors of such systems when using them in a complex system. One might also need to consider both truly random or chaotic natural or social systems as part of the context of an engineered system. The main focus for systems approaches is **organized complexity** (see below). This kind of complexity cannot be dealt with by traditional analysis techniques, nor can it be totally removed by the way we design or use solutions. A systems approach must be able to recognise and deal with such complexity across the life of the systems it interacts with..

Sillitto (Sillitto 2014) considers the link between the types of system complexity and system architecture. The ability to understand, manage and respond to both objective and subjective complexity in the problem situation, the systems we develop or the systems we use to develop and sustain them is a key component of the Systems Approach Applied to Engineered Systems and hence to the practice of systems engineering.

Origins and Characteristics of Complexity

This section describes some of the prevailing ideas on complexity. Various authors have used different language to express these ideas. While a number of common threads can be seen, some of the ideas take different viewpoints and may be contradictory in nature.

One of the most widely used definitions of complexity is the degree of difficulty in predicting the properties of a system if the properties of the system's parts are given (generally attributed to Weaver). This, in turn, is related to the number of elements and connections between them. Weaver (Weaver 1948) relates complexity to types of elements and how they interact. He describes simplicity as problems with a finite number of variables and interaction, and identifies two kinds of complexity:

1. **Disorganized Complexity** is found in a system with many loosely coupled, disorganized and equal elements, which possesses certain average properties such as temperature or pressure. Such a system can be described by “19th Century” statistical analysis techniques.

2. **Organized Complexity** can be found in a system with many strongly coupled, organized and different elements which possess certain emergent properties and phenomena such as those exhibited by economic, political or social systems. Such a system can not be described well by traditional analysis techniques.

Weaver's ideas about this new kind of complex problem are one of the foundational ideas of systems thinking. (See also Systems Thinking.)

Later authors, such as Flood and Carson (Flood and Carson 1993) and Lawson (Lawson 2010), expand organized complexity to systems which have been organized into a structure intended to be understood and thus amenable to engineering and life cycle management (Braha et al. 2006). They also suggest that disorganized complexity could result from a heterogeneous complex system evolving without explicit architectural control during its life (complexity creep). This is a different use of the terms organized and disorganized to that used by Weaver. Care should be taken in mixing these ideas

Complexity should not be confused with "complicated". Many authors make a distinction between ordered and disordered collections of elements.

Ordered systems have fixed relationships between elements and are not adaptable. Page (Page 2009) cites a watch as an example of something which can be considered an ordered system. Such a system is complicated, with many elements working together. Its components are based on similar technologies, with clear mapping between form and function. If the operating environment changes beyond prescribed limits, or one key component is removed, the watch will cease to perform its function.

In common usage, chaos is a state of disorder or unpredictability characterized by elements which are not interconnected and behave randomly with no adaptation or control. Chaos Theory (Kellert 1993) is applied to certain dynamic systems (e.g., the weather) which, although they have structure and relationships, exhibit unpredictable behavior. These systems may include aspects of randomness but can be described using deterministic models from which their behavior can be described given a set of initial conditions. However, their structure is such that (un-measurably) small perturbations in inputs or environmental conditions may result in unpredictable changes in behavior. Such systems are referred to as deterministically chaotic or, simply, chaotic systems. Simulations of chaotic systems can be created and, with increases in computing power, reasonable predictions of behavior are possible at least some of the time.

On a spectrum of order to complete disorder complexity is somewhere in the middle, with more flexibility and change than complete order and more stability than complete disorder (Sheard and Mostashari 2009).

Complex systems may evolve “to the edge of chaos”, resulting in systems which can appear deterministic but which exhibit counter intuitive behavior compared to that of more ordered systems. The statistics of chance events in a complex system are often characterized by a power-law distribution, the “signature of complexity” (Sheard 2005). The power-law distribution is found in a very wide variety of natural and man-made phenomena, and it means that the probability of a low probability—large impact event is much higher than a Gaussian distribution would suggest. Such a system may react in a non-linear way to exhibit abrupt phase changes. These phase changes can be either reversible or irreversible. This has a major impact on engineered systems in terms of the occurrence, impact and public acceptance of risk and failure.

Objective complexity is an attribute of complex systems and is a measure of where a system sits on this spectrum. It is defined as the extent to which future states of the system cannot be predicted with certainty and precision, regardless of our knowledge of current state and history. Subjective complexity is a measure of how easy it is for an observer to understand a system or predict what it will do next. As such, it is a function of the perspective and comprehension of each individual. It is important to be prepared to mitigate subjective complexity with consistent, clear communication and strong stakeholder engagement (Sillitto 2009).

The literature has evolved to a fairly consistent definition of the characteristics of system elements and relationships for objective systems complexity. The following summary is given by Page (Page 2009):

1. **Independence**: Autonomous system elements which are able to make their own decisions; influenced by information from other elements and the adaptability algorithms it carries with it (Sheard and Mostashari 2009).
2. **Interconnectedness**: System elements connect via a physical connection, shared data or simply a visual awareness of where the other elements are and what they are doing, as in the case of the flock of geese or the squadron of aircraft.
3. **Diversity**: System elements which are either technologically or functionally different in some way. For example, elements may be carrying different adaptability algorithms.
4. **Adaptability**: Self-organizing system element which can do what it wants to do to support itself or the entire system in response to their environment (Sheard and Mostashari 2009). Adaptability is often achieved by human elements but can be achieved with software. Pollock and Hodgson (2004) describe how this can be done in a variety of complex system types, including power grids and enterprise systems.

Due to the variability of human behavior as part of a system and the perceptions of people outside the system, the inclusion of people in a system is often a factor in their complexity. People may be viewed as observing systems or as system elements which contribute to the other types of complexity (Axelrod and Cohen 1999). The rational or irrational behavior of individuals in particular situations is a vital factor in respect to complexity (Kline 1995). Some of this complexity can be reduced through education, training and familiarity with a system. Some is irreducible, and must be managed as part of a problem or solution. Checkland (Checkland 1999) argues that a group of stakeholders will have its own world views which lead them to form different, but equally valid, understandings of a system context. These differences cannot be explained away or analyzed out, and must be understood and considered in the formulation of problems and the creation of potential solutions.

Warfield (Warfield 2006) developed a powerful methodology for addressing complex issues, particularly in the socio-economic field, based on a relevant group of people developing an understanding of the issue in the form of a set of interacting problems - what he called the “problematique”. The complexity is then characterized via several measures, such as the number of significant problems, their interactions and the degree of consensus about the nature of the problems. What becomes clear is that how, why, where and by whom a system is used may all contribute to its perceived complexity.

Sheard and Mostashari (Sheard and Mostashari 2011) sort the attributes of complexity into causes and effects. Attributes that cause complexity include being non-linear; emergent; chaotic; adaptive; tightly coupled; self-organized; decentralized; open; political (as opposed to scientific); and multi-scale; as well as having many pieces. The effects of those attributes which make a system be perceived as complex include being uncertain; difficult to understand; unpredictable; uncontrollable; unstable; unrepairable; unmaintainable and costly; having unclear cause and effect; and taking too long to build.

References

Works Cited

Axelrod, R. and M. Cohen. 1999. *Harnessing Complexity: Organizational Implications of a Scientific Frontier*. New York, NY, USA: Simon and Schuster.

Braha, D., A. Minai, and Y. Bar-Yam (eds.). 2006. *Complex Engineered Systems: Science Meets Technology*. New York, NY, USA: Springer.

Checkland, P. 1999. *Systems Thinking, Systems Practice*. New York, NY, USA: John Wiley & Sons.

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.

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

Kellert, S. 1993. *In the Wake of Chaos: Unpredictable Order in Dynamical Systems*, Chicago, IL, USA: University of Chicago Press. p. 32.

Kline, S. 1995. *Foundations of Multidisciplinary Thinking*. Stanford, CA, USA: Stanford University Press.

Page, Scott E. 2009. *Understanding Complexity*. Chantilly, VA, USA: The Teaching Company.

Pollock, J.T. and R. Hodgson. 2004. *Adaptive Information*. Hoboken, NJ, USA: John Wiley & Sons.

Senge, P.M. 1990. *The Fifth Discipline: The Art & Practice of The Learning Organization*. New York, NY, USA: Doubleday/Currency.

Sheard, S.A. 2005. "Practical Applications of Complexity Theory for Systems Engineers". *Proceedings of the Fifteenth Annual International Council on Systems Engineering*. Volume 15 Issue 1.

Sheard, S.A. and A. Mostashari. 2009. "Principles of Complex Systems for Systems Engineering." *Systems Engineering*, 12(4): 295-311.

Sheard, SA. and A. Mostashari. 2011. "Complexity Types: From Science to Systems Engineering." Proceedings of the 21st Annual of the International Council on Systems Engineering (INCOSE) International Symposium, 20-23 June 2011, Denver, Colorado, USA.

Sillitto H 2014, "Architecting Systems - Concepts, Principles and Practice", College Publications 2014

Warfield, J.N. 2006. *An Introduction to Systems Science*. London, UK: World Scientific Publishing.

Weaver, W. 1948. "Science and Complexity." *American Science.* 36: 536-544.

Primary References

Flood, R. L., & 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.

Page, Scott E. 2009. *Understanding Complexity*. Chantilly, VA, USA: The Teaching Company.

Sheard, S.A. and A. Mostashari. 2009. "Principles of Complex Systems for Systems Engineering". *Systems Engineering*, 12(4): 295-311.

Additional References

Ashby, W.R. 1956. *An Introduction to Cybernetics*. London, UK: Chapman and Hall.

Aslaksen, E.W. 2004. "System Thermodynamics: A Model Illustrating Complexity Emerging from Simplicity". *Systems Engineering*, 7(3). Hoboken, NJ, USA: Wiley.

Aslaksen, E.W. 2009. *Engineering Complex Systems: Foundations of Design in the Functional Domain*. Boca Raton, FL, USA: CRC Press.

Aslaksen, E.W. 2011. "Elements of a Systems Engineering Ontology". Proceedings of SETE 2011, Canberra, Australia.

Eisner, H. 2005. *Managing Complex Systems: Thinking Outside the Box*. Hoboken, NJ, USA: John Wiley & Sons.

Jackson, S., D. Hitchins, and H. Eisner. 2010. What is the Systems Approach? INCOSE *Insight* 13(1) (April 2010): 41-43.

MITRE. 2011. "Systems Engineering Strategies for Uncertainty and Complexity." *Systems Engineering Guide.* Accessed 9 March 2011 at http:/ / www. mitre. org/ work/ systems_engineering/ guide/ enterprise_engineering/ comprehensive_viewpoint/sys_engineering_strategies_uncertainty_complexity.html.

Ryan, A. 2007. "Emergence Is Coupled to Scope, Not Level, Complexity". A condensed version appeared in INCOSE *Insight*, 11(1) (January 2008): 23-24.

Sillitto H.G. 2009. "On Systems Architects and Systems Architecting: Some Thoughts on Explaining The Art and Science of System Architecting." Proceedings of the 19th Annual International Council on Systems Engineering

(INCOSE) International Symposium, 20-23 July 2009, Singapore.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.0, released 2 June 2019

Emergence

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Scott Jackson, Dick Fairley, Janet Singer, Duane Hybertson*

This topic forms part of the Systems Fundamentals knowledge area (KA). It gives the background to some of the ways in which emergence has been described, as well as an indication of current thinking on what it is and how it influences systems engineering (SE) practice. It will discuss how these ideas relate to the general definitions of systems given in What is a System?; in particular, how they relate to different engineered system contexts. This topic is closely related to the complexity topic that precedes it.

Emergence is a consequence of the fundamental system concepts of holism and interaction (Hitchins 2007, 27). System wholes have behaviors and properties arising from the organization of their elements and their relationships, which only become apparent when the system is placed in different environments.

Questions that arise from this definition include: What kinds of systems exhibit different kinds of emergence and under what conditions? Can emergence be predicted, and is it beneficial or detrimental to a system? How do we deal with emergence in the development and use of engineered systems? Can it be planned for? How?

There are many varied and occasionally conflicting views on emergence. This topic presents the prevailing views and provides references for others.

Overview of Emergence

As defined by Checkland, emergence is “the principle that entities exhibit properties which are meaningful only when attributed to the whole, not to its parts.” (Checkland 1999, 314). Emergent system behavior can be viewed as a consequence of the interactions and relationships between system elements rather than the behavior of individual elements. It emerges from a combination of the behavior and properties of the system elements and the systems structure or allowable interactions between the elements, and may be triggered or influenced by a stimulus from the systems environment.

Emergence is common in nature. The pungent gas ammonia results from the chemical combination of two odorless gases, hydrogen and nitrogen. As individual parts, feathers, beaks, wings, and gullets do not have the ability to overcome gravity. Properly connected in a bird, however, they create the emergent behavior of flight. What we refer to as “self-awareness” results from the combined effect of the interconnected and interacting neurons that make up the brain (Hitchins 2007, 7).

Hitchins also notes that technological systems exhibit emergence. We can observe a number of levels of outcome which arise from interaction between elements in an engineered system context. At a simple level, some system outcomes or attributes have a fairly simple and well defined mapping to their elements; for example, center of gravity or top speed of a vehicle result from a combination of element properties and how they are combined. Other behaviors can be associated with these simple outcomes, but their value emerges in complex and less predictable ways across a system. The single lap performance of a vehicle around a track is related to center of gravity and speed; however, it is also affected by driver skill, external conditions, component ware, etc. Getting the 'best' performance from a vehicle can only be achieved by a combination of good design and feedback from real laps under race conditions.

There are also outcomes which are less tangible and which come as a surprise to both system developers and users. How does lap time translate into a winning motor racing team? Why is a sports car more desirable to many than other vehicles with performances that are as good or better?

Emergence can always be observed at the highest level of system. However, Hitchins (2007, 7) also points out that to the extent that the systems elements themselves can be considered as systems, they also exhibit emergence. Page (Page 2009) refers to emergence as a “macro-level property.” Ryan (Ryan 2007) contends that emergence is coupled to scope rather than system hierarchical levels. In Ryan’s terms, scope has to do with spatial dimensions (how system elements are related to each other) rather than hierarchical levels.

Abbott (Abbott 2006) does not disagree with the general definition of emergence as discussed above. However, he takes issue with the notion that emergence operates outside the bounds of classical physics. He says that “such higher-level entities…can always be reduced to primitive physical forces.”

Bedau and Humphreys (2008) and Francois (2004) provide comprehensive descriptions of the philosophical and scientific background of emergence.

Types of Emergence

A variety of definitions of types of emergence exists. See Emmeche et al. (Emmeche et al. 1997), Chroust (Chroust 2003) and O’Connor and Wong (O’Connor and Wong 2006) for specific details of some of the variants. Page (Page 2009) describes three types of emergence: "simple", "weak", and "strong".

According to Page, **simple emergence** is generated by the combination of element properties and relationships and occurs in non-complex or “ordered” systems (see Complexity) (2009). To achieve the emergent property of “controlled flight” we cannot consider only the wings, or the control system, or the propulsion system. All three must be considered, as well as the way these three are interconnected-with each other, as well as with all the other parts of the aircraft. Page suggests that simple emergence is the only type of emergence that can be predicted. This view of emergence is also referred to as synergy (Hitchins 2009).

Page describes **weak emergence** as expected emergence which is desired (or at least allowed for) in the system structure (2009). However, since weak emergence is a product of a complex system, the actual level of emergence cannot be predicted just from knowledge of the characteristics of the individual system components.

The term **strong emergence** is used to describe unexpected emergence; that is, emergence not observed until the system is simulated or tested or, more alarmingly, until the system encounters in operation a situation that was not anticipated during design and development.

Strong emergence may be evident in failures or shutdowns. For example, the US-Canada Blackout of 2003 as described by the US-Canada Power System Outage Task Force (US-Canada Power Task Force 2004) was a case of cascading shutdown that resulted from the design of the system. Even though there was no equipment failure, the shutdown was systemic. As Hitchins points out, this example shows that emergent properties are not always beneficial (Hitchins 2007, 15).

Other authors make a different distinction between the ideas of strong, or unexpected, emergence and unpredictable emergence:

- Firstly, there are the unexpected properties that could have been predicted but were not considered in a systems development: "Properties which are unexpected by the observer because of his incomplete data set, with regard to the phenomenon at hand" (Francois, C. 2004, 737). According to Jackson et al. (Jackson et al. 2010), a desired level of emergence is usually achieved by iteration. This may occur as a result of evolutionary processes, in which element properties and combinations are "selected for", depending on how well they contribute to a systems effectiveness against environmental pressures or by iteration of design parameters through simulation or build/test cycles. Taking this view, the specific values of weak emergence can be refined and examples of strong emergence can be considered in subsequent iterations so long as they are amenable to analysis.

- Secondly, there are unexpected properties which cannot be predicted from the properties of the system's components: "Properties which are, in and of themselves, not derivable a priori from the behavior of the parts of the system" (Francois, C. 2004, 737). This view of emergence is a familiar one in social or natural sciences, but more controversial in engineering. We should distinguish between a theoretical and a practical unpredictability (Chroust 2002). The weather forecast is theoretically predictable, but beyond certain limited accuracy practically impossible due to its chaotic nature. The emergence of consciousness in human beings cannot be deduced from the physiological properties of the brain. For many, this genuinely unpredictable type of complexity has limited value for engineering. (See **Practical Considerations** below.)

A type of system particularly subject to strong emergence is the system of systems (sos). The reason for this is that the SoS, by definition, is composed of different systems that were designed to operate independently. When these systems are operated together, the interaction among the parts of the system is likely to result in unexpected emergence. Chaotic or truly unpredictable emergence is likely for this class of systems.

Emergent Properties

Emergent properties can be defined as follows: "A property of a complex system is said to be 'emergent' [in the case when], although it arises out of the properties and relations characterizing its simpler constituents, it is neither predictable from, nor reducible to, these lower-level characteristics" (Honderich 1995, 224).

All systems can have emergent properties which may or may not be predictable or amenable to modeling, as discussed above. Much of the literature on complexity includes emergence as a defining characteristic of complex systems. For example, Boccara (Boccara 2004) states that "The appearance of emergent properties is the single most distinguishing feature of complex systems". In general, the more ordered a systems is, the easier its emergent properties are to predict. The more complex a system is, the more difficult predicting its emergent properties becomes.

Some practitioners use the term "emergence" only when referring to "strong emergence". These practitioners refer to the other two forms of emergent behavior as synergy or "system level behavior" (Chroust 2002). Taking this view, we would reserve the term "Emergent Property" for unexpected properties, which can be modeled or refined through iterations of the systems development.

Unforeseen emergence causes nasty shocks. Many believe that the main job of the systems approach is to prevent undesired emergence in order to minimize the risk of unexpected and potentially undesirable outcomes. This review of emergent properties is often specifically associated with identifying and avoiding system failures (Hitchins 2007).

Good SE isn't just focused on avoiding system failure, however. It also involves maximizing opportunity by understanding and exploiting emergence in engineered systems to create the required system level characteristics from synergistic interactions between the components, not just from the components themselves (Sillitto 2010).

One important group of emergent properties include properties such as agility and resilience. These are critical system properties that are not meaningful except at the whole system level.

Practical Considerations

As mentioned above, one way to manage emergent properties is through iteration. The requirements to iterate the design of an engineered system to achieve desired emergence results in a design process are more lengthy than those needed to design an ordered system. Creating an engineered system capable of such iteration may also require a more configurable or modular solution. The result is that complex systems may be more costly and time-consuming to develop than ordered ones, and the cost and time to develop is inherently less predictable.

Sillitto (2010) observes that "engineering design domains that exploit emergence have good mathematical models of the domain, and rigorously control variability of components and subsystems, and of process, in both design and operation". The iterations discussed above can be accelerated by using simulation and modeling, so that not all the iterations need to involve building real systems and operating them in the real environment.

The idea of domain models is explored further by Hybertson in the context of general models or patterns learned over time and captured in a model space (Hybertson 2009). Hybertson states that knowing what emergence will appear from a given design, including side effects, requires hindsight. For a new type of problem that has not been solved, or a new type of system that has not been built, it is virtually impossible to predict emergent behavior of the solution or system. Some hindsight, or at least some insight, can be obtained by modeling and iterating a specific system design; however, iterating the design within the development of one system yields only limited hindsight and often does not give a full sense of emergence and side effects.

True hindsight and understanding comes from building multiple systems of the same type and deploying them, then observing their emergent behavior in operation and the side effects of placing them in their environments. If those observations are done systematically, and the emergence and side effects are distilled and captured in relation to the design of the systems — including the variations in those designs — and made available to the community, then we are in a position to predict and exploit the emergence.

Two factors are discovered in this type of testing environment: what works (that is, what emergent behavior and side effects are desirable); and what does not work (that is, what emergent behavior and side effects are undesirable). What works affirms the design. What does not work calls for corrections in the design. This is why multiple systems, especially complex systems, must be built and deployed over time and in different environments; to learn and understand the relations among the design, emergent behavior, side effects, and environment.

These two types of captured learning correspond respectively to patterns and “antipatterns”, or patterns of failure, both of which are discussed in a broader context in the Principles of Systems Thinking and Patterns of Systems Thinking topics.

The use of iterations to refine the values of emergent properties, either across the life of a single system or through the development of patterns encapsulating knowledge gained from multiple developments, applies most easily to the discussion of strong emergence above. In this sense, those properties which can be observed but cannot be related to design choices are not relevant to a systems approach. However, they can have value when dealing with a combination of engineering and managed problems which occur for system of systems contexts (Sillitto 2010). (See Systems Approach Applied to Engineered Systems.)

References

Works Cited

Abbott, R. 2006. "Emergence Explained: Getting Epiphenomena to Do Real Work". *Complexity.* 12(1) (September-October): 13-26.

Bedau, M.A. and P. Humphreys, P. (eds.). 2008. "Emergence" In Contemporary Readings in Philosophy and Science. Cambridge, MA, USA: The MIT Press.

Boccara, N. 2004. *Modeling Complex Systems.* New York: Springer-Verlag.

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: John Wiley & Sons.

Chroust. G. 2002. "Emergent Properties in Software Systems." 10th Interdisciplinary Information Management Talks; Hofer, C. and Chroust, G. (eds.). Verlag Trauner Linz, pages 277-289.

Chroust, G., C. Hofer, C. Hoyer (eds.). 2005. *The Concept of Emergence in Systems Engineering." The 12th Fuschl Conversation, April 18-23, 2004, Institute for Systems Engineering and Automation, Johannes Kepler University Linz. pp. 49-60.*

Emmeche, C., S. Koppe, and F. Stjernfelt. 1997. "Explaining Emergence: Towards an Ontology of Levels." *Journal for General Philosophy of Science.* 28: 83-119 (1997). Accessed December 3 2014 at Claus Emmeche http://www. nbi.dk/~emmeche/coPubl/97e.EKS/emerg.html.

Francois, C. 2004. *International Encyclopedia of Systems and Cybernetics*, 2nd edition, 2 volumes. K.G.Saur, Munchen.

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

Honderich. T. 1995. *The Oxford Companion to Philosophy*. New York: Oxford University Press.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Auerbach/CRC Press, Boca Raton, FL.

Jackson, S., D. Hitchins, and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight.* 13(1) (April 2010): 41-43.

O'Connor, T. and H. Wong. 2006. "Emergent Properties". *Stanford Encyclopedia of Philosophy*. Accessed December 3 2014 at Stanford Encyclopedia of Philosophy http://plato.stanford.edu/entries/properties-emergent/.

Page, S.E. 2009. *Understanding Complexity.* The Great Courses. Chantilly, VA, USA: The Teaching Company.

Ryan, A. 2007. "Emergence is Coupled to Scope, Not Level." *Complexity.* 13(2) (November-December).

Sillitto, H.G. 2010. "Design Principles for Ultra-Large-Scale Systems". Proceedings of the 20th Annual International Council on Systems Engineering (INCOSE) International Symposium, July 2010, Chicago, IL, USA, reprinted in "The Singapore Engineer", April 2011.

US-Canada Power System Outage Task Force. 2004. *Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations*. April, 2004. Washington-Ottawa. Accessed December 3 2014 at US Department of Energy http://energy.gov/oe/downloads/blackout-2003-final-report-august-14-2003-blackout-united-states-and-canada-causes-and

Primary References

Emmeche, C., S. Koppe, and F. Stjernfelt. 1997. "Explaining Emergence: Towards an Ontology of Levels." *Journal for General Philosophy of Science*, 28: 83-119 (1997). http://www.nbi.dk/~emmeche/coPubl/97e.EKS/emerg.html.

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

Page, S. E. 2009. *Understanding Complexity*. The Great Courses. Chantilly, VA, USA: The Teaching Company.

Additional References

Sheard, S.A. and A. Mostashari. 2008. "Principles of Complex Systems for Systems Engineering." *Systems Engineering*. 12: 295-311.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Fundamentals for Future Systems Engineering

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Duane Hybertson*

This article forms part of the Systems Fundamentals knowledge area (KA). It considers future trends in SE and how these might influence the evolution of future fundamentals.

The SEBoK contains a guide to generalized knowledge about the practice of SE. It does not pass judgement on that knowledge. However, it can be useful in some cases to indicate which parts of the knowledge are rooted in existing practice and which points towards the future evolution of SE.

This article provides a sketch of how SE is changing and suggests how these changes may affect the future of systems engineering, the SEBoK, and the foundations in Part 2.

INCOSE Vision

The INCOSE Vision 2025 statement (INCOSE 2014) depicts some future directions in:

Broadening SE Application Domains

- SE relevance and influence will go beyond traditional aerospace and defense systems and extend into the broader realm of engineered, natural and social systems.
- SE will be applied more widely to assessments of socio-physical systems in support of policy decisions and other forms of remediation

More intelligent and autonomous systems

- Systems of the future need to become smarter, self-organized, sustainable, resource-efficient, robust and safe
- More autonomous vehicles and transportation systems
- Systems become more “intelligent” and dominate human-safety critical applications

Theoretical Foundations

- SE will be supported by a more encompassing foundation of theory and sophisticated model-based methods and tools allowing a better understanding of increasingly complex systems and decisions in the face of uncertainty
- Challenge: A core body of systems engineering foundations is defined and taught consistently across academia and forms the basis for systems engineering practice

In this article we will consider how the fundamentals of SE might need to evolve to support this vision.

How will SE Change?

In Types of Systems we describe three general contexts in which a SE life cycle can be applied. In a product system context the outputs of SE focus on the delivery of technologically systems. While such systems are designed to be used by people and fit into a wider problem solving context, this context has been seen as largely a fixed and external to SE. The service system context allows SE to consider all aspects of the solution system as part of its responsibility. This is currently seen as a special case of SE application largely focused on software intensive solutions. The enterprise system context offers the potential for an application of SE directly to tackle complex socio-technical problems, by supporting the planning, development and use of combinations of service systems. While this is done it can be difficult to connect up to the product focused life cycles of many SE projects.

The role of the systems engineer has already begun to change somewhat due to the first two of the future trends above. Changes to the scope of SE application and the increased use of software intensive, reconfigurable and autonomous solutions will make the service system context the primary focus of most SE life cycles. To enable this most product systems will need to become more general and configurable, allowing them to be used in a range of service systems as needed. Increasingly these life cycles are initiated and managed as part of an enterprise portfolio of related life cycles.

In this evolution of SE the systems engineer cannot consider as many aspects of the context to be fixed, making the problem and possible solution options more complex and harder to anticipate. This also means the systems engineer has greater freedom to consider solutions which combine existing and new technologies and in which the role of people and autonomous software can be changed to help deliver desired outcomes. For such systems to be successful they will need to include the ability to change, adapt and grow both in operation and over several iterations of their life cycle. This change moves SE to be directly involved in enterprise strategy and planning, as part of an ongoing and iterative approach to tackling the kinds of societal problems identified in the INCOSE vision.

This evolution of both the role and scope of SE will also see the system of systems aspects of all system contexts increase. We can expect system of systems engineering to become part of the systems engineering of many, if not most, SE life cycles.

Evolution of Fundamentals

These ongoing change to SE places more emphasis on the role of autonomus agents in systems engineering, and agency will be an area of increased emphasis in the systems engineering and SEBoK of the future. Hybertson (Hybertson, 2019) spells out in more detail the increased role of agents and agency in future SE. Moving from a total control model to a shared responsibility model changes the nature of engineering to something more like collective actualization, as proposed by Hybertson (2009 and 2019). Systems will represent a combination and interplay of technology and social factors, and they can range from technical product to service provider to social entity. In many cases they will be a socio-technical combination or hybrid.

The above trends have an impact on SE foundations, including technical aspects, social aspects, and ethical aspects. Inclusion of people in systems implies significant expansion of foundation sciences, to provide principles, theories, models, and patterns of the human, biological, social, and agent realm as well as the technical and physical realm. Emphasis on agents implies a revised conceptualization of system change, from the traditional model of mechanistic and controlled fixes and upgrades to a more organic change model that involves growth, self-learning, self-organizing, and self-adapting. Ethical considerations will include how to allocate responsibility for a system in a shared responsibility model. Further discussion of the expanded foundation and a list of foundation disciplines for future SE are presented in (Hybertson 2009 and 2019).

References

Works Cited

Hybertson, D. (2009). Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems, Auerbach/CRC Press, Boca Raton, FL.

Hybertson, D. (2019 forthcoming). Systems Engineering Science. Chapter in G. S. Metcalf, H. Deguchi, and K. Kijima (editors in chief). Handbook of Systems Science. Tokyo: Springer.

INCOSE (2014). A world in motion: systems engineering vision 2025. International Council on Systems Engineering.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Knowledge Area: Systems Approach Applied to Engineered Systems

Systems Approach Applied to Engineered Systems

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Janet Singer, Duane Hybertson*

This knowledge area (KA) provides a guide for applying the systems approach as a means of identifying and understanding complex problems and opportunities, synthesizing possible alternatives, analyzing and selecting the best alternative, implementing and approving a solution, as well as deploying, using and sustaining engineered system solutions. The active participation of stakeholders during all the activities of the systems approach is the key to the success of the systems approach.

In an engineered system context, a systems approach is a holistic approach that spans the entire life of the system; however, it is usually applied in the development and operational/support life cycle stages. This knowledge area defines a systems approach using a common language and intellectual foundation to ensure that practical systems concepts, principles, patterns and tools are accessible to perform systems engineering (SE), as is discussed in the introduction to Part 2: Foundations of Systems Engineering.

Topics

Each part of the Guide to the SE Body of Knowledge (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:

- Overview of the Systems Approach
- Engineered System Context
- Identifying and Understanding Problems and Opportunities
- Synthesizing Possible Solutions
- Analysis and Selection between Alternative Solutions
- Implementing and Proving a Solution
- Deploying, Using, and Sustaining Systems to Solve Problems
- Applying the Systems Approach

Systems Approach

This KA describes a high-level framework of activities and principles synthesized from the elements of the systems approach, as described earlier in Part 2 of the SEBoK, and is mapped to the articles Concepts of Systems Thinking, Principles of Systems Thinking, and Patterns of Systems Thinking. The concept map in Figure 1 describes how the knowledge is arranged in this KA and the linkage to the KA in Part 3.

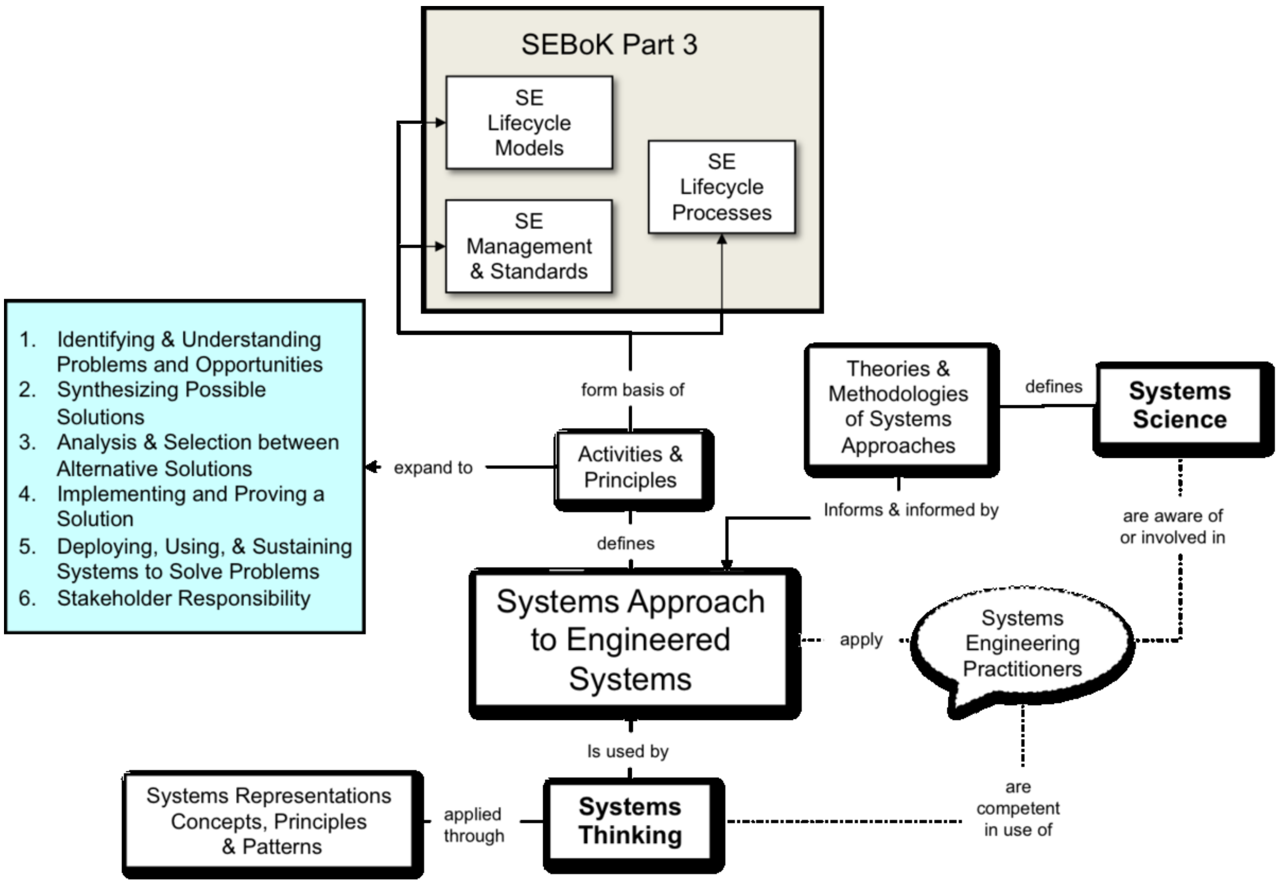

SEBoK Part 3
SE Lifecycle Models
SE Lifecycle Processes
SE Management & Standards
form basis of
Theories & Methodologies of Systems Approaches
defines
Systems Science
1. Identifying & Understanding Problems and Opportunities
2. Synthesizing Possible Solutions
3. Analysis & Selection between Alternative Solutions
4. Implementing and Proving a Solution
5. Deploying, Using, & Sustaining Systems to Solve Problems
6. Stakeholder Responsibility
expand to
Activities & Principles
Informs & informed by
are aware of or involved in
defines
Systems Approach to Engineered Systems
apply
Systems Engineering Practitioners
Is used by
Systems Representations Concepts, Principles & Patterns
applied through
Systems Thinking
are competent in use of

Figure 1. Systems Engineering and the Systems Approach. (SEBoK Original)

According to Jackson et al. (Jackson et al. 2010, 41-43), the systems approach to engineered systems is a problem-solving paradigm. It is a comprehensive problem identification and resolution approach based upon the principles, concepts, and tools of systems thinking and systems science, along with the concepts inherent in engineering problem-solving. It incorporates a holistic systems view that covers the larger context of the system, including engineering and operational environments, stakeholders, and the entire life cycle.

Successful systems practice should not only apply systems thinking to the system being created, but should also utilize systems thinking in consideration of the way in which work is planned and conducted. See Part 5: Enabling Systems Engineering for further discussions on how individuals, teams, businesses and enterprises may be enabled to perform systems engineering.

References

Works Cited

Jackson, S., D. Hitchins, and H. Eisner. 2010. "What is the Systems Approach?". INCOSE *Insight.* 13(1): 41-43.

Primary References

Checkland, P. 1999. *Systems Thinking, Systems Practice*. New York, NY, USA: John Wiley & Sons.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4).

Jackson, S., D. Hitchins, and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight.* 13(1): 41-43.

Additional References

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

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

Senge, P. M. 1990. *The Fifth Discipline: The Art and Practice of the Learning Organization*. New York, Doubleday/Currency.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Overview of the Systems Approach

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Janet Singer, Duane Hybertson, Hillary Sillitto, Bud Lawson, Brian Wells, James Martin*

This knowledge area (KA) considers how a systems approach relates to engineered systems and to systems engineering (SE). The article Applying the Systems Approach considers the dynamic aspects of how the approach is used and how this relates to elements of SE.

Systems Approach and Systems Engineering

The term systems approach is used by systems science authors to describe a systems *"thinking"* approach, as it pertains to issues outside of the boundary of the immediate system-of-interest (Churchman 1979). This systems approach is essential when reductionist assumptions (the notion that the whole system has properties derived directly from the properties of their components) no longer apply to the system-of-interest (SoI) and when emergence and complexity at multiple levels of a system context necessitate a holistic approach.

The systems approach for engineered systems is designed to examine the *"whole system, whole lifecycle, and whole stakeholder community"* as well as to ensure that the purpose of the system (or systemic intervention) is achieved sustainably without causing any negative unintended consequences. This prevents the engineer from *"transferring the burden"* (in systems thinking terms) to some other part of the environment that unable to sustain that burden (Senge 2006). This also deters issues involving *sub-optimization* that could occur when whole systems are not kept in mind in achieving the purpose of the system (Sillitto 2012).

The systems approach (derived from systems thinking) and systems engineering (SE) have developed and matured, for the most part, independently; therefore, the systems science and the systems engineering communities differ in their views as to what extent SE is based on a systems approach and how well SE uses the concepts, principles, patterns and representations of systems thinking. These two views are discussed in the following sections.

Systems Science View

As discussed in the Systems Science article, some parts of the systems movement have been developed as a reaction to the perceived limitations of systems engineering (Checkland 1999).

According to Ryan (2008):

> ***Systems engineering has a history quite separate to the systems movement.*** *Its closest historical link comes from the application of systems analysis techniques to parts of the systems engineering process . . . The recent popularity of the SoS buzzword in the systems engineering literature has prompted* ***the expansion of systems engineering techniques to include methods that can cope with evolving networks of semi-autonomous systems. This has led many systems engineers to read more widely across the systems literature, and is providing a re-conceptualization of systems engineering as part of the systems movement, despite its historical independence. This is reflected in the latest INCOSE hand-book [INCOSE 2011, page 52], which states "the systems engineering perspective is based on systems thinking", which "recognizes circular causation, where a variable is both the cause and the effect of another and recognizes the primacy of interrelationships and non-linear and organic thinking—a way of thinking where the primacy of the whole is acknowledged.*** **(emphases added)**

Thus, for many in the systems science community, systems thinking is *not* naturally embedded in either SE definitions or practice.

Systems Engineering View

Many SE authors see a clear link between SE and systems thinking. For example, Hitchins (Hitchins 2007) describes generic models for the application of systems thinking to engineered system contexts. He suggests that these could form the foundation for descriptions and standards for the practices of SE. Hitchins also proposes a set of *guiding principles which have been the foundations of SE, apparently since its inception* (Hitchins 2009):

- **SE Principle A: The Systems Approach** - *"SE is applied to a system-of-interest (SoI) in a wider systems context"*
- **SE Principle B: Synthesis** - *"SE must bring together a collection of parts to create whole system solutions"*
- **SE Principle C: Holism** - *"Always consider the consequences on the wider system when making decisions about the system elements"*
- **SE Principle D: Organismic Analogy** - *"Always consider systems as having dynamic "living" behavior in their environment"*
- **SE Principle E: Adaptive Optimizing** - *"Solve problems progressively over time"*
- **SE Principle F: Progressive Entropy Reduction** - *"Continue to make systems work over time, through maintenance, sustainment, upgrade activities."*
- **SE Principle G: Adaptive Satisfying** - *"A system will succeed only if it makes winners of its success-critical stakeholders, so the lifecycle of a system must be driven by how well its outputs contribute to stakeholder purpose"*

Hitchins considers principles A-D as pillars of SE that identify key aspects of systems thinking which should underpin the practice of SE. Principles E-G consider the dynamics of SE life cycle thinking, the *why*, *when* and *how often* of SE.

The following sections consider the systems approach to engineered systems against four themes.

1. Whole System

The system coupling diagram (Figure 1), describes the scope of a systems approach to engineered systems (Lawson 2010).

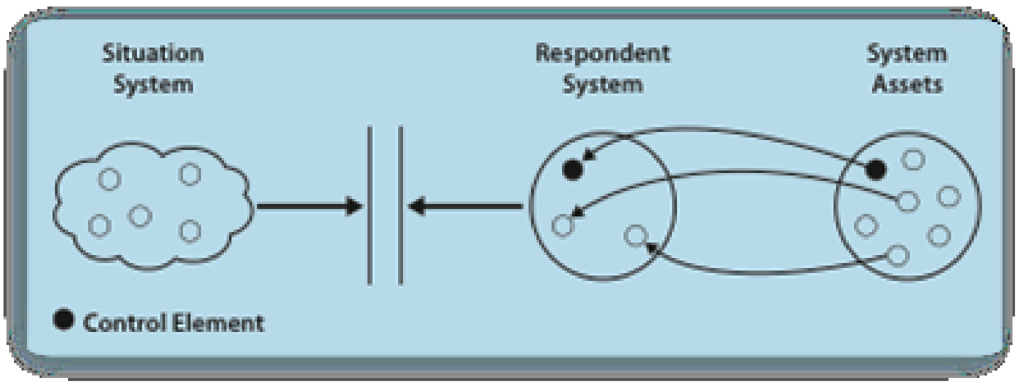


Figure 1. System Coupling Diagram (Lawson 2010). Reprinted with permission of Harold "Bud" Lawson. All other rights are reserved by the copyright owner.

- The **situation system** is the problem or opportunity either unplanned or planned. The situation may be natural, man-made, a combination of both, or a postulated situation used as a basis for deeper understanding and training (e.g. business games or military exercises).
- The **respondent system** is the system created to respond to the situation. The parallel bars indicate that this system interacts with the situation and transforms it to a new situation. Respondent systems have several names, project, program, mission, task force, or in a scientific context, experiment.
- **System assets** are the sustained assets of one or more enterprises to be used in response to situations. System assets must be adequately managed throughout the life of a system to ensure that they perform their function when instantiated in a respondent system. Examples include: value-added products or services, facilities, instruments and tools, and abstract systems, such as theories, knowledge, processes and methods.

Martin (Martin 2004) describes seven types of system, or *"the seven samurai of systems engineering"*, all of which, system developers need to understand to develop successful systems:

- the context system
- the intervention system
- the realization system
- the deployed system
- collaborating systems
- the sustainment system
- competing systems

Martin contends that all seven systems must be explicitly acknowledged and understood when engineering a solution for a complex adaptive situation.

These views, and others, describe one aspect of the systems approach when applied to engineered systems; in addition, it is applicable to understanding a problem, it organizes the resolution of that problem, and creates and integrates any relevant assets and capabilities to enable that solution.

2. Whole Lifecycle

Ring (Ring 1998) provides a powerful framework for the continuing management and periodic upgrade of long-life and *"immortal"* systems. It also accurately represents the *"continuous"* or very rapid product launch and refreshment cycle driven by market feedback and constant innovation that is seen in most product and service system consumer markets.

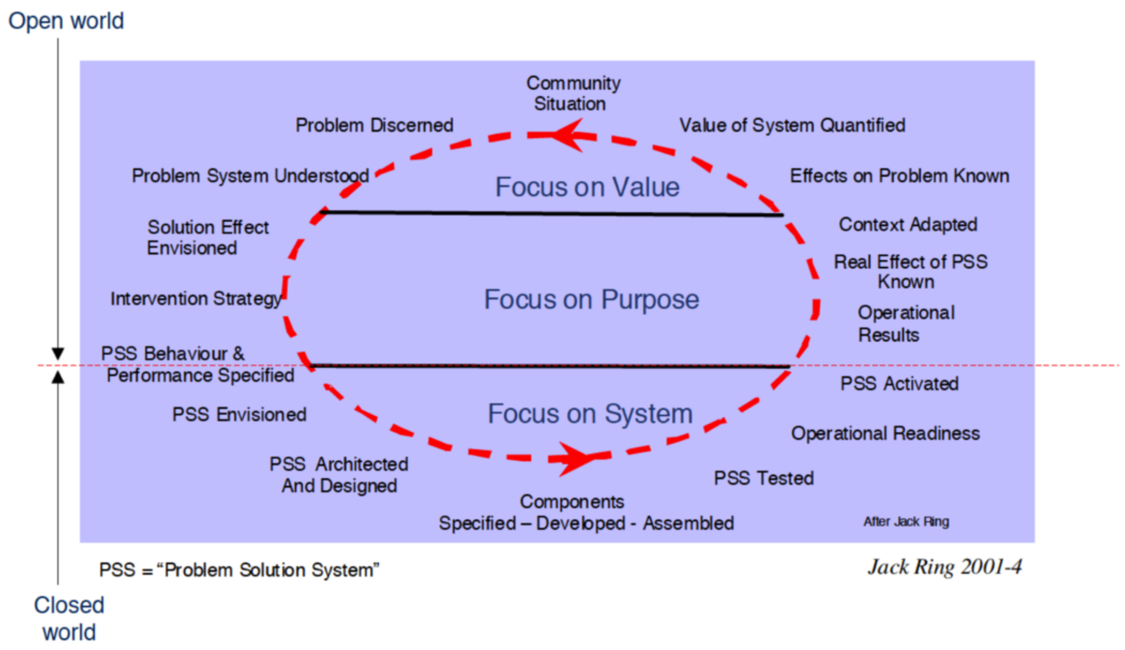


Figure 2. Ellipse Graphic (Ring 1998). © 1998 IEEE. Reprinted, with permission, from Jack Ring, Engineering Value-Seeking Systems, IEEE-SMC. Conference Proceedings. All other rights are reserved by the copyright owner.

Enterprise systems engineering may be considered in multiple concurrent instances of this model for different sub-sets of enterprise assets and services, in order to maintain a capability to pursue enterprise goals in a complex and dynamic external environment.

The dynamic nature of this cycle and its relationship to Life Cycle thinking is discussed in the article Applying the Systems Approach.

3. Whole Problem

The article Identifying and Understanding Problems and Opportunities considers the nature of problem situations. It discusses the relationship between hard system and soft system views of problems and how they relate to engineered systems. Engineered systems are designed to operate with and add value to a containing social and/or ecological system. The scope of problems is captured by frameworks, such as Political, Economic, Social, Technological, Legal and Environmental (PESTLE) (Gillespie 2007) or Social, Technical, Economic, Environmental, Political, Legal, Ethical and Demographic (STEEPLED).

The idea of a wicked problem (Rittel and Webber 1973) is also discussed. These problems cannot be quantified and solved in a traditional engineering sense.

Sillitto (Sillitto 2010) describes a lifecycle model in which the decision as to what parts of problems can be *"solved"* and what parts must be *"managed"* is the first key decision and emphasizes the need for a solution approach that provides flexibility in the solution to match the level of uncertainty and change in the problem and stakeholder expectations. It is now normal to view a problem as one that *"changes over time"* and to promote the belief that value is determined by the perceptions of key stakeholders.

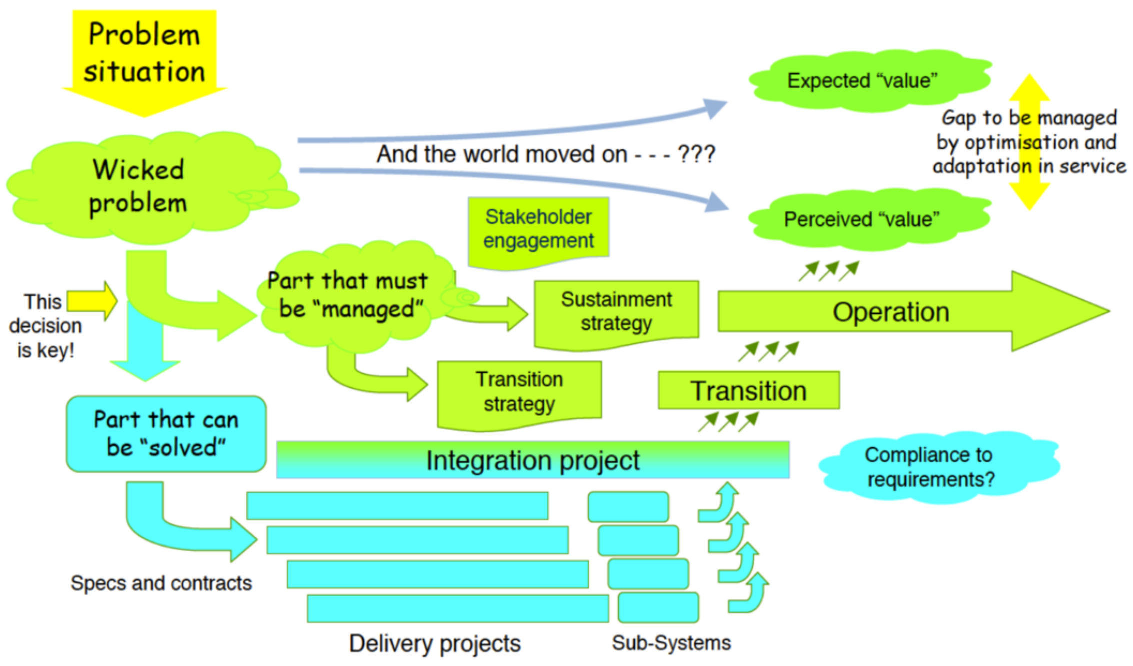


Figure 3. Engineered vs Managed Problems (Sillitto 2010). Reproduced with permission of Hillary Sillitto. All other rights are reserved by the copyright owner.

Thus, a systems approach can be useful when addressing all levels of a problematic situation, from individual technologies to the complex socio-technical issues that come about in the area of engineered systems development.

4. Multi-Disciplinary

As discussed by Sillitto (Sillitto 2012), the methods and thinking applied by many practicing systems engineers have become optimized to the domains of practice. While systems thinking concepts, patterns and methods are used widely, they are not endemic in SE practice. As a result, SE practitioners find it difficult to share systems ideas with others involved in a systems approach. Part 4: Applications of Systems Engineering describes traditional (product based) SE (Lawson 2010) and examines this against the SE approaches that are applicable to service, enterprise, and system of systems capability. These approaches require more use of problem exploration, a broader solution context, and a purpose driven life cycle thinking.

SE and Systems Approach

From the above discussions, there are three ways in which SE could make use of a systems approach:

- in its overall problem solving approach
- in the scope of problem and solution system contexts considered
- in the embedding of systems thinking and systems thinking tools and in all aspects of the conduct of that approach

The current SE standards and guides, as described in Part 3: Systems Engineering and Management, encapsulate many of the elements of a systems approach. However, they tend to focus primarily on the development of system solutions while the wider purpose-driven thinking of a full systems approach (Ring 1998) and the wider consideration of all relevant systems (Martin 2004) are embedded in the acquisition and operational practices of their application domains.

The inclusion of systems thinking in SE competency frameworks (INCOSE 2010) represents a general move toward a desire for more use of systems thinking in SE practice. There is a wide stakeholder desire to acquire the benefits of a systems approach through the application of SE, particularly in areas where current SE approaches are inadequate or irrelevant. Hence, there is a need for a better articulation of the *systems approach* and how to apply it to non-traditional problems.

Synthesis for SEBOK

The systems approach presented in the SEBoK uses the following activities:

- identify and understand the relationships between the potential problems and opportunities in a real world situation
- gain a thorough understanding of the problem and describe a selected problem or opportunity in the context of its wider system and its environment
- synthesize viable system solutions to a selected problem or opportunity situation
- analyze and choose between alternative solutions for a given time/cost/quality version of the problem.
- provide evidence that a solution has been correctly implemented and integrated
- deploy, sustain, and apply a solution to help solve the problem (or exploit the opportunity)

All of the above are considered within a life cycle (glossary) framework which may need concurrent, recursive (glossary) and iterative applications of some or all of the systems approach.

When the systems approach is executed in the real world of an engineered system (glossary), a number of engineering and management disciplines emerge, including SE. Part 3: Systems Engineering and Management and Part 4: Applications of Systems Engineering contain a detailed guide to SE with references to the principles of the systems approach, where they are relevant. Part 5: Enabling Systems Engineering provides a guide to the relationships between SE and the organizations and Part 6: Related Disciplines also offers a guide to the relationship between SE and other disciplines.

More detailed discussion of how the systems approach relates to these engineering and management disciplines is included in the article Applying the Systems Approach within this KA.

References

Works Cited

Checkland, P. 1999. *Systems Thinking, Systems Practice*. New York, NY, USA: John Wiley & Sons.

Churchman, C.W. 1979. *The Systems Approach and Its Enemies.* New York, NY: Basic Books.

Gillespie. 2007. *Foundations of Economics - Additional chapter on Business Strategy.* Oxford University Press.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?". INCOSE *Insight.* 12(4).

Hitchins, D. 2007. *Systems Engineering, a 21st Century Systems Methodology.* Hoboken, NJ: Wiley.

INCOSE. 2010. *Systems Engineering Competencies Framework 2010-0205.* San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2010-003.

INCOSE. 2012. *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.

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

Martin, J, 2004. "The Seven Samurai of Systems Engineering: Dealing with the Complexity of 7 Interrelated Systems." Proceedings of the 14th Annual INCOSE International Symposium, 20-24 June 2004, Toulouse, France.

Ring, J., 1998. "A Value Seeking Approach to the Engineering of Systems." Proceedings of the IEEE Conference on Systems, Man, and Cybernetics. p. 2704-2708.

Rittel, H. and M. Webber. 1973. "Dilemmas in a General Theory of Planning." *Policy Sciences*. 4:155–169.

Ryan, A. 2008. “What is a Systems Approach?” *Journal of Non-linear Science*.

Senge, P.M. 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization*, 2nd ed. New York, NY, USA: Doubleday Currency.

Sillitto, H. 2010. “Design principles for ultra-large scale systems.” Proceedings of the INCOSE International Symposium, Chicago, July 2010, re-printed in "The Singapore Engineer" July 2011.

Sillitto, H.G. 2012: "Integrating Systems Science, Systems Thinking, and Systems Engineering: understanding the differences and exploiting the synergies", Proceedings of the INCOSE International Symposium, Rome July 2012.

Primary References

Checkland, P. 1999. *Systems Thinking, Systems Practice*. New York, NY, USA: John Wiley & Sons.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight*. 12(4).

Senge, P.M. 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization*, 2nd ed. New York, NY, USA: Doubleday Currency.

Additional References

Biggs, J.B. 1993. "From Theory to Practice: A Cognitive Systems Approach". *Journal of Higher Education & Development.* Accessed December 4 2014 Taylor and Francis from http:/ / www. informaworld. com/ smpp/ content~db=all~content=a758503083.

Boardman, J. and B. Sauser 2008. *Systems Thinking: Coping with 21st Century Problems*. Boca Raton, FL, USA: CRC Press.

Edson, R. 2008. *Systems Thinking. Applied. A Primer.* Arlington, VA, USA: Applied Systems Thinking (ASysT) Institute, Analytic Services Inc.

Ring J. 2004. "Seeing an Enterprise as a System." INCOSE *Insight.* 6(2): 7-8.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Engineered System Context

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Brian Wells, Scott Jackson, James Martin*

This article is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It describes knowledge related to the further expansion of the ideas of an engineered system and engineered system context that were introduced in the Systems Fundamentals KA.

The single most important principle of the systems approach is that it is applied to an engineered system context and not just to a single system (INCOSE 2012). The systems approach includes models and activities useful for the understanding, creation, use, and sustainment of engineered systems to enable the realization of stakeholder needs. Disciplines that use a systems approach (like systems engineering (SE)) consider an engineered system context that defines stakeholder needs, and look for the best ways to provide value by applying managed technical activities to one or more selected engineered systems of interest (SoI).

Generally, four specific types of engineered system contexts are recognized in SE:

- product system (glossary)
- service system (glossary)
- enterprise system (glossary)
- system of systems (SoS) capability

One of the key distinctions between these system contexts pertains to the establishment of how and when the SoI boundary is drawn.

Engineered System-of-Interest

We use the idea of an engineered system context to define an engineered SoI and to capture and agree on the important relationships between it, the systems it works directly with, and any other systems with which it work. All applications of a systems approach (and hence of SE) are applied in a system context rather than only to an individual system.

A system context can be constructed around the following set of open system relationships (Flood and Carson 1993):

- The **Narrower System-of-Interest** (NSoI) is the system of direct concern to the observer. The focus of this system is driven by the scope of authority or control with implicit recognition that this scope may not capture all related elements.
- The **Wider System-of-Interest** (WSoI) describes a logical system boundary containing all of the elements needed to fully understand system behavior. The observer may not have authority over all of the elements in the WSoI, but will be able to establish the relationships between WSoI elements and NSoI elements.
- The WSoI exists in an **environment**. The immediate environment contains engineered, natural, and/or social systems, with which the WSoI (and thus some elements of the NSoI) directly interact with for the purpose of exchanging material, information, and/or energy to achieve its goals or objective.
- A **Wider Environment** completes the context and contains systems that have no direct interaction with the SoI, but which might influence decisions related to it during its life cycle.
- "Some Theoretical Considerations of Mathematical Modeling" (Flood 1987) extends this context to include a **meta-system** (MS) that exists outside of the WSoI and exercises direct control over it.

The choice of the SoI boundary for particular activities depends upon what can be changed and what must remain fixed. The SoI will always include one or more NSoI, but may also include WSoI and a MS if appropriate, such as when considering a service or an enterprise system.

Applying the System Context

For lower-level and less-complex systems, the WSoI can represent levels of a product system hierarchy. An example of this would be an engine management unit as part of an engine, or an engine as part of a car. The WSoI in a system context may encapsulate some aspects of SoS ideas for sufficiently complex systems. In these cases, the WSoI represents a collection of systems with their own objectives and ownership with which the NSoI must cooperate with in working towards a shared goal. An example of this would be a car and a driver contributing to a transportation service.

This view of a SoS context being used as a means to support the engineering of a NSoI product system is one way in which a systems approach can be applied. It can also be applied directly to the SoS. Examples of this include a flexible multi-vehicle transportation service or transportation as part of a commercial enterprise. In this case, the NSoI aspect of the context no longer applies. The WSoI will consist of a set of cooperating systems, each of which might be changed or replaced to aid in the synthesis of a solution. The context may also need to represent **loose coupling**, with some systems moving in or out of the context depending on the need, or **late binding** with systems joining the context only at, or close to the delivery of the service.

Thus, a context allows a reductionist view of the SoI that is of direct concern to an observer, as it provides for the system relationships and influences that are needed to maintain a holistic (glossary) view of the consequence of any actions taken.

Product System Context

The distinction between a product (glossary) and a product system (glossary) is discussed in the article Types of Systems.

A product system context would be one in which the SoI is the product itself. The wider system context for a product system can be a higher level of product hierarchy, a service, or an enterprise system that uses the product directly to help provide value to the user. A significant aspect of a product systems context is the clear statement of how the product is intended to be used and ensures that this information is given to the acquirer upon delivery. The customer will be required to accept the system, typically through a formal process, agreeing not to go against the terms of use.

If a systems approach is applied to a product context, it is done with the purpose of engineering a narrow system product to be integrated and used in a wider system product hierarchy or to enable the delivery of a wider system service directly to a user by an enterprise.

This view of the relationship between product and service is specific to product systems engineering. While some engineering of the acquirer's static service system may occur, it is done with a product focus. The definition of service system in a service systems engineering context describes a more dynamic view of service systems.

Service System Context

Services are activities that cause a transformation of the state of an entity (people, product, business, and region or nation) by mutually agreed terms between the service provider and the customer (Spohrer 2008). The distinction between service and a service system is discussed in the article Types of Systems.

A service system context is one in which the SoI is the service system. This SoI contains all of the technology, infrastructure, people, resources, etc. that are needed to enable the service. The WSoI describes the enterprise providing the service as well as its relationship with other services that are impact the success of the enterprise.

If a systems approach is applied to a service system, it is done with the purpose of engineering a service system to enable the outcomes required by an enterprise to satisfy its clients. When operating in the service system context, all options to provide the service must be considered, providing that they fit within the constraints of the enterprise. This will include interfaces to other services, people, and resources in the enterprise. If an option for providing the service makes use of existing products or resources within or outside of the enterprise, it must be ensured that they are available for this use and that this does not adversely affect other services. Part of getting the right service may require the negotiation of changes to the wider enterprise context, but this must be by agreement with the relevant authority.

For a service system, and also when considering the service system context, the value is realized only through service transactions. The end-user co-creates value at the time of the request to use the service. For example, to make a flight reservation using a smart phone, the service system is composed of many service system entities (the caller, the person called, the smart phone, the access network, the core Internet Protocol (IP) network, the Internet Service provider (ISP), the World Wide Web (WWW), data centers, etc. All these are necessary to enable the service. When a caller makes a reservation and then books the flight, the value has been created.

This definition of a service system, as associated with dynamic Information Technology (IT) services, is discussed further in the article Service Systems Engineering.

Enterprise System Context

The distinction between an enterprise and an enterprise system is discussed in the article Types of Systems.

An enterprise system context is one in which the SoI is the enterprise system. This system contains all of the technology, infrastructure, people, resources, etc. needed to enable the service. The WSoI describes the business environment within which the enterprise sits.

It is to be noted that an enterprise context is not equivalent to an **organization** according to this definition. An enterprise includes not only the organizations that participate in it, but also the people, knowledge, and other assets, such as processes, principles, policies, practices, doctrines, theories, beliefs, facilities, land, and intellectual property that compose the enterprise.

An enterprise may contain or employ service systems along with product systems. An enterprise might even contain sub-enterprises. Enterprise systems are unique when compared to product and service systems in that:

- they are constantly evolving
- they rarely have detailed configuration controlled requirements
- they typically have (constantly changing) goals of providing shareholder value and customer satisfaction
- they exist in a context (or environment) that is ill-defined and constantly changing

The enterprise systems engineer must consider and account for these factors in their processes and methods.

Both product and service systems require an enterprise system context to create them and an enterprise to use the product system and deliver services, either internally to the enterprise or externally to a broader community. Thus, the three types of engineered system contexts are linked in all instances, regardless of which type of system the developers consider as the object of the development effort that is delivered to the customer.

References

Works Cited

Flood, R.L. 1987. "Some Theoretical Considerations of Mathematical Modeling." In Problems of Constancy and Change (proceedings of 31st Conference of the International Society for General Systems Research, Budapest), Volume 1, p. 354 - 360.

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.

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.

Spohrer, J. 2008. "Service Science, Management, Engineering, and Design (SSMED): An Emerging Discipline-Outline & References." *International Journal of Information Systems in the Service Sector.* 1(3) (May).

Primary References

Chang, C.M., 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence.* Hoboken, NJ, USA: John Wiley and 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.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice.* Boca Raton, FL, USA: CRC Press.

Rouse, W.B. 2005. "Enterprise as Systems: Essential Challenges and Enterprise Transformation". *Systems Engineering*. 8(2): 138-50.

Tien, J.M. and D. Berg. 2003. "A Case for Service Systems Engineering". *Journal of Systems Science and Systems Engineering.* 12(1): 13-38.

Additional 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.

Bernus, P., L. Nemes, and G. Schmidt (eds.). 2003. *Handbook on Enterprise Architecture.* Heidelberg, Germany: Springer.

Chang, C.M., 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence.* Hoboken, NJ, USA: John Wiley and Sons.

DeRosa, J. K. 2006. “An Enterprise Systems Engineering Model.” Proceedings of the 16th Annual International Council on Systems Engineering, 9-13 July 2006, Orlando, FL, USA.

Giachetti, R.E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods.* Boca Raton, FL, USA: CRC Press.

Joannou, P. 2007. "Enterprise, Systems, and Software—The Need for Integration." IEEE *Computer*. 40(5) (May): 103-105.

Katzan, H. 2008. *Service Science.* Bloomington, IN, USA: iUniverse Books.

Maglio P., S. Srinivasan, J.T. Kreulen, and J. Spohrer. 2006. “Service Systems, Service Scientists, SSME, and Innovation." *Communications of the ACM*. 49(7) (July).

Martin J.N. 1997. *Systems Engineering Guidebook.* Boca Raton, FL, USA: CRC Press.

Rebovich, G., and B.E. White (eds.). 2011. *Enterprise Systems Engineering: Advances in the Theory and Practice.* Boca Raton, FL, USA: CRC Press.

Rouse, W.B. 2009. "Engineering the Enterprise as a System". In *Handbook of Systems Engineering and Management*, 2nd ed. A.P. Sage and W.B. Rouse (eds.). New York, NY, USA: Wiley and Sons.

Tien, J.M. and D. Berg. 2003. "A Case for Service Systems Engineering". *Journal of Systems Science and Systems Engineering.* 12(1): 13-38.

Valerdi, R. and D.J. Nightingale. 2011. "An Introduction to the Journal of Enterprise Transformation." *Journal of Enterprise Transformation* 1(1):1-6.

SEBoK v. 2.1, released 31 October 2019

Identifying and Understanding Problems and Opportunities

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Brian Wells, Scott Jackson, Janet Singer, Duane Hybertson, Bud Lawson*

This topic is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It describes knowledge related to the identification and exploration of problems or opportunities in detail. The problem situations described by the activities in this topic may form a starting point for Synthesizing Possible Solutions. Any of the activities described below may also need to be considered concurrently with other activities in the systems approach at a particular point in the life of a system-of-interest (SoI).

The activities described below should be considered in the context of the Overview of the Systems Approach topic at the start of this KA. The final topic in this knowledge area, Applying the Systems Approach, considers the dynamic aspects of how these activities are used as part of the systems approach and how this relates in detail to elements of systems engineering (SE).

The phrase "problem or opportunity" used herein recognizes that the "problem" is not always a negative situation and can also be a positive opportunity to improve a situation.

Introduction

According to Jenkins (1969), the first step in the systems approach is “the recognition and formulation of the problem.” The systems approach described in the Guide to the SE Body of Knowledge (SEBoK) is predominantly a hard system approach. The analysis, synthesis, and proving parts of the approach assume a problem or opportunity has been identified and agreed upon and that a "new" engineered system (glossary) solution is needed.

However, the systems approach does not have to apply to the development and use of a newly designed and built technical solution. Abstract or experimental solutions to potential problems might be explored to help achieve agreement on a problem context. Solutions may involve reorganizing existing system of systems (SoS) contexts or the modification or re-use of existing products and services. The problem and opportunity parts of the approach overlap with soft system approaches. This is discussed in more detail below.

One thing that must be considered in relation to system complexity is that the opportunity situation may be difficult to fully understand; therefore, system solutions may not solve the problem the first time, but is still useful in increasing the understanding of both problem issues and what to try next to work toward a solution.

Hence, problem exploration and identification is often not a one-time process that specifies the problem, but is used in combination with solution synthesis and analysis to progress toward a more complete understanding of problems and solutions over time (see Applying the Systems Approach for a more complete discussion of the dynamics of this aspect of the approach).

Problem Exploration

Soft system thinking does not look for "the problem", but considers a problematic situation. Forming systems views of this situation can help stakeholders better understand each other's viewpoints and provide a starting point for directed intervention in the current system context. If a full soft systems intervention is undertaken, such as a soft systems methodology (SSM) (Checkland 1999), it will not include formal analysis, synthesis, and proving. However, the SSM method was originally based on hard methodologies, in particular one presented by Jenkins (1969). It follows the basic principles of a systems approach: "analyzing" conceptual models of shared understanding, "synthesizing" intervention strategies, and "proving" improvements in the problematic situation.

Often, the distinction between hard and soft methods is not as clear cut as the theory might suggest. Checkland himself has been involved in applications of SSM as part of the development of information system design (Checkland and Holwell 1998). It is now agreed upon by many that while there is a role for a "pure soft system" approach, the service and enterprise problems now being tackled can only be dealt with successfully by a combination of soft problematic models and hard system solutions. Mingers and White (Mingers and White 2009) give a number of relevant examples of this. In particular they reference "Process and Content: Two Ways of Using SSM" (Checkland and Winters 2006). It is likely in the future that engineered system problems will be stated, solved, and used as part of a predominately soft intervention, which will place pressure on the speed of development needed in the solution space. This is discussed more fully in the topic Life Cycle Models.

The critical systems thinking and multi-methodology approaches (Jackson 1985) take this further by advocating a "pick and mix" approach, in which the most appropriate models and techniques are chosen to fit the problem rather than following a single methodology (Mingers and Gill 1997). Thus, even if the hard problem identification approach described below is used, some use of the soft system techniques (such as rich pictures, root definitions, or conceptual models) should be considered within it.

Problem Identification

Hard system thinking is based on the premise that a problem exists and can be stated by one or more stakeholders in an objective way. This does not mean that hard systems approaches start with a defined problem. Exploring the potential problem with key stakeholders is still an important part of the approach.

According to Blanchard and Fabrycky (Blanchard and Fabrycky 2006, 55-56), defining a problem is sometimes the most important and difficult step. In short, a system cannot be defined unless it is possible to clearly describe what it is supposed to accomplish.

According to Edson (Edson 2008, 26-29), there are three kinds of questions that need to be asked to ensure we fully understand a problem situation. First, how difficult or well understood is the problem? The answer to this question will help define the tractability of the problem. Problems can be “tame,” “regular,” or “wicked”:

- For tame problems, the solution may be well-defined and obvious.
- Regular problems are those that are encountered on a regular basis. Their solutions may not be obvious, thus serious attention should be given to every aspect of them.
- Wicked problems (Rittel and Webber 1973) cannot be fully solved, or perhaps even fully defined. Additionally, with wicked problems, it is not possible to understand the full effect of applying systems to the problem.

Next, who or what is impacted? There may be elements of the situation that are causing the problem, elements that are impacted by the problem, and elements that are just in the loop. Beyond these factors, what is the environment and what are the external factors that affect the problem? In examining these aspects, the tools and methods of systems thinking can be productively applied.

Finally, what are the various viewpoints of the problem? Does everyone think it is a problem? Perhaps there are conflicting viewpoints. All these viewpoints need to be defined. Persons affected by the system, who stand to benefit from the system, or can be harmed by the system, are called stakeholders. Wasson (Wasson 2006, 42-45) provides a comprehensive list of stakeholder types. The use of soft systems models, as discussed above, can play an important part in this. Describing a problem using situation views can be useful when considering these issues, even if a single problem perspective is selected for further consideration.

Operations research is a hard systems method which concentrates on solving problem situations by deploying known solutions. The problem analysis step of a typical approach asks questions about the limitation and cost of the current system to identify efficiency improvements that need to be made (Flood and Carson 1993).

Traditional SE methods tend to focus more on describing an abstract model of the problem, which is then used to develop a solution that will produce the benefits stakeholders expect to see (Jenkins 1969). The expectation is often that a new solution must be created, although this need not be the case. Jenkins suggests that SE is just as applicable to a redesign of existing systems. A clear understanding of stakeholder expectations in this regard should produce a better understanding of part of the problem. Do stakeholders expect a new solution or modifications to their existing solutions, or are they genuinely open to solution alternatives which consider the pros and cons of either. Such expectations will influence suggestions of solution alternatives, as discussed in the Synthesizing Possible Solutions article.

An important factor in defining the desired stakeholder outcomes, benefits, and constraints is the operational environment, or scenario, in which the problem or opportunity exists. Armstrong (Armstrong 2009, 1030) suggests two scenarios: the first is the descriptive scenario, or the situation as it exists now, and the second is the normative scenario, or the situation as it may exist sometime in the future.

All of these aspects of problem understanding can be related to the concept of a system context.

Problem Context

The Engineered System Context topic identifies a way by which a complex system situation can be resolved around a system-of-interest (glossary) (SoI). The initial identification of a "problem context" can be considered as the outcome of this part of the systems approach.

The systems approach should not consider only soft or hard situations. More appropriately, a problem or opportunity should be explored using aspects of both. In general, the application of the systems approach with a focus on engineered system contexts will lead to hard system contexts in which an identified SoI and required outcome can be defined.

An initial description of the wider SoI and environment serves as the problem or opportunity problem scope. Desired stakeholder benefits are expressed as outcomes in the wider system and some initial expression of what the SoI is intended for may be identified. Jenkins (1969) defines a problem formulation approach where one:

- states the aim of the SoI
- defines the wider SoI
- defines the objectives of the wider SoI
- defines the objectives of the system
- defines economic, informational, and other conditions.

In a hard system problem context, a description of a logical or ideal system solution may be included. This ideal system cannot be implemented directly, but describes the properties required of any realizable system solution.

To support this problem or opportunity description, a soft context view of the SoI will help ensure wider stakeholder concerns are considered. If a soft system context has been defined, it may include a conceptual model (Checkland 1999) which describes the logical elements of a system that resolve the problem situation and how they are perceived by different stakeholders. Unlike the hard system view, this does not describe the ideal solution, but provides an alternative view on how aspects of any solution would be viewed by potential stakeholders.

In problem contexts with a strong coercive dimension, the problem context should include an identification of the relative power and the importance of stakeholders.

The problem context should include some boundaries on the cost, time to deployment, time in use, and operational effectiveness needed by stakeholders. In general, both the full problem context and an agreed version of the problem to be tackled next are described. (see Applying the Systems Approach).

References

Works Cited

Armstrong, Jr., J.E., 2009. "Issue Formulation." In A.P. Sage and W.B. Rouse (eds.). *Handbook of Systems Engineering and Management,* 2nd edition. Hoboken, NJ, USA: Wiley.

Blanchard, B. and W.J. Fabrycky. 2006. *Systems Engineering and Analysis*. Upper Saddle River, NJ, USA: Prentice Hall.

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: Wiley.

Checkland, P. and S. Holwell. 1998. *Information, Systems and Information Systems: Making Sense of the Field.* Hoboken, NJ, USA: Wiley.

Checkland, P. and M. Winter. 2006. "Process and Content: Two Ways of Using SSM". *Journal of Operational Research Society.* 57(12): 1435-1441.

Edson, R. 2008. *Systems Thinking. Applied. A Primer.* Arlington, VA, USA: Applied Systems Thinking (ASysT) Institute, Analytic Services Inc.

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.

Jackson, M. 1985. "Social Systems Theory and Practice: The Need for A Critical Approach". *International Journal of General Systems*. 10: 135-151.

Jenkins, G.M. 1969. "The Systems Approach." *The Journal of Systems Engineering.* 1(1).

Mingers, J. and A. Gill. 1997. *Multimethodology: Theory and Practice of Combining Management Science Methodologies*. Chichester, UK: Wiley.

Mingers, J. and L. White. 2009. *A Review of Recent Contributions of Systems Thinking to Operational Research and Management Science*, Working Paper 197. Canterbury, UK: Kent Business School.

Rittel, H. and M. Webber. 1973. "Dilemmas in a General Theory of Planning." In *Policy Sciences*. 4:155–169.

Wasson, C.S. 2006. *System Analysis, Design, and Development.* Hoboken, NJ, USA: Wiley.

Primary References

Blanchard, B. & W.J. Fabrycky. 2006. *Systems Engineering and Analysis*. Upper Saddle River, NJ, USA: Prentice Hall.

Edson, R. 2008. *Systems Thinking. Applied. A Primer*. Arlington, VA, USA: Applied Systems Thinking (ASysT) Institute, Analytic Services Inc.

Additional References

None

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Synthesizing Possible Solutions

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Scott Jackson, Janet Singer, Duane Hybertson*

This topic is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It describes knowledge related to the synthesis of possible solution options in response to the problem situations described by activities from Identifying and Understanding Problems and Opportunities topic. The solution options proposed by the synthesis activities will form the starting point for the Analysis and Selection between Alternative Solutions. Any of the activities described below may also need to be considered concurrently with other activities in the systems approach at a particular point in the life of a system-of-interest (SoI).

The activities described below should be considered in the context of the Overview of the Systems Approach topic at the start of this KA. The final topic in this KA, Applying the Systems Approach, considers the dynamic aspects of how these activities are used as part of the systems approach and how this relates in detail to elements of systems engineering (SE).

Synthesis Overview

System synthesis is an activity within the systems approach that is used to describe one or more system solutions based upon a problem context for the life cycle of the system to:

- Define options for a SoI with the required properties and behavior for an identified problem or opportunity context.
- Provide solution options relevant to the SoI in its intended environment, such that the options can be assessed to be potentially realizable within a prescribed time limit, cost, and risk described in the problem context.
- Assess the properties and behavior of each candidate solution in its wider system context.

The iterative activity of system synthesis develops possible solutions and may make some overall judgment regarding the feasibility of said solutions. The detailed judgment on whether a solution is suitable for a given iteration of the systems approach is made using the Analysis and Selection between Alternative Solutions activities.

Essential to synthesis is the concept of holism, according to Hitchins (Hitchins 2009), and states that a system must be considered as a whole and not simply as a collection of its elements. The holism of any potential solution system requires that the behavior of the whole be determined by addressing a system within an intended environment and not simply the accumulation of the properties of the elements. The latter process is known as reductionism and is the opposite of holism, which Hitchins (Hitchins 2009, 60) describes as the notion that “the properties, capabilities, and behavior of a system derive from its parts, from interactions between those parts, and from interactions with other systems.”

When the system is considered as a whole, properties called emergent properties often appear (see Emergence). These properties are often difficult to predict from the properties of the elements alone. They must be evaluated within the systems approach to determine the complete set of performance levels of the system. According to Jackson (Jackson 2010), these properties can be considered in the design of a system, but to do so, an iterative approach is required.

In complex systems, individual elements will adapt to the behavior of the other elements and to the system as a whole. The entire collection of elements will behave as an organic whole. Therefore, the entire synthesis activity, particularly in complex systems, must itself be adaptive.

Hence, synthesis is often not a one-time process of solution design, but is used in combination with problem understanding and solution analysis to progress towards a more complete understanding of problems and solutions over time (see Applying the Systems Approach topic for a more complete discussion of the dynamics of this aspect of the approach).

Problem or Opportunity Context

System synthesis needs the problem or opportunity that the system is intended to address to have already been identified and described and for non-trivial systems, the problem or opportunity needs to be identified and understood concurrently with solution synthesis activities.

As discussed in Identifying and Understanding Problems and Opportunities, the systems approach should not consider strictly soft or hard situations. In general, the application of the systems approach, with a focus on engineered system contexts, will lead to hard system contexts in which an identified SoI and required outcome be defined. Even in these cases, a soft context view of the SoI context will help ensure wider stakeholder concerns are considered.

The problem context should include some boundaries on the cost, time to deployment, time in use, and operational effectiveness needed by stakeholders. In general, the goal is not to synthesize the perfect solution to a problem, but rather to find the best available solution for the agreed version of the problem.

Synthesis Activities

The following activities provide an outline for defining the SoI: grouping of elements, identification of the interactions among the elements, identification of interfaces between elements, identification of external interfaces to the SoI boundary and common sub-elements within the SoI boundary.

The activities of systems synthesis are built on the idea of a balanced reduction vs. holism approach as discussed in What is Systems Thinking? topic. It is necessary to divide system elements and functions to create a description of the SoI which is realizable, either through combinations of available elements or through the design and construction of new elements. However, if the system is simply decomposed into smaller and smaller elements, the holistic nature of systems will make it more and more difficult to predict the function and behavior of the whole. Thus, synthesis progresses through activities that divide, group, and allocate elements, and then assesses the complete system’s properties in context relevant to the user need the SoI will fulfill. Hence, synthesis occurs over the entire life cycle of the system as the system and its environment change.

Identification of the Boundary of a System

Establishing the boundary of a system is essential to synthesis, the determination of the system's interaction with its environment and with other systems, and the extent of the SoI. Buede (Buede 2009, 1102) provides a comprehensive discussion of the importance of, and methods of, defining the boundary of a system in a SE context.

Identification of the Functions of the System

The function of a system at a given level of abstraction is critical to synthesis since the primary goal of the synthesis activity is to propose realizable system descriptions which can provide a given function. The function of a system is distinct from its behavior as it describes what the system can be used for or is asked to do in a larger system context.

Buede (Buede 2009, 1091-1126) provides a comprehensive description of functional analysis in a SE context.

Identification of the Elements of a System

System synthesis calls for the identification of the elements of a system. Typical elements of an Engineered System Context may be physical, conceptual, or processes. Physical elements may be hardware, software, or humans. Conceptual elements may be ideas, plans, concepts, or hypotheses. Processes may be mental, mental-motor (writing, drawing, etc.), mechanical, or electronic (Blanchard and Fabrycky 2006, 7).

In addition to the elements of the system under consideration (i.e., a SoI), ISO 15288 (ISO/IEC/IEEE 15288 2015) also calls for the identification of the enabling systems. These are systems (or services) utilized at various stages in the life cycle, e.g., development, utilization or support stages, to facilitate the SoI in achieving its objectives.

Today's systems often include existing elements. It is rare to find a true "greenfield" system, in which the developers can specify and implement all new elements from scratch. "Brownfield" systems, wherein legacy elements constrain the system structure, capabilities, technology choices, and other aspects of implementation, are much more typical (Boehm 2009).

Division of System Elements

System synthesis may require elements to be divided into smaller elements. The division of elements into smaller elements allows the systems to be grouped and leads to the SE concept of physical architecture, as described by Levin (Levin 2009, 493-495). Each layer of division leads to another layer of the hierarchical view of a system. As Levin points out, there are many ways to depict the physical architecture, including the use of wiring diagrams, block diagrams, etc. All of these views depend on arranging the elements and dividing them into smaller elements. According to the principle of recursion, these decomposed elements are either terminal elements, or are able to be further decomposed. The hierarchical view does not imply a top-down analytical approach to defining a system. It is simply a view. In the systems approach, levels of the hierarchy are defined and considered recursively with one level forming the context for the next.

Grouping of System Elements

System synthesis may require that elements be grouped. This leads to the identification of the sub-systems that are essential to the definition of a system. Synthesis determines how a system may be partitioned and how each sub-system fits and functions within the whole system. The largest group is the SoI, also called the relevant system by Checkland (Checkland 1999, 166). According to Hitchins, some of the properties of a SoI are as follows: the SoI is open and dynamic, the SoI interacts with other systems, and the SoI contains sub-systems (Hitchins 2009, 61). The SoI is brought together through the concept of synthesis.

Identification of the Interactions among System Elements

System synthesis may require the identification of the interactions among system elements. These interactions lead to the SE process of interface analysis. Integral to this aspect is the principle of interactions. Interactions occur both with other system elements as well as with external elements and the environment. In a systems approach, interfaces have both a technical and managerial importance. Managerial aspects include the contracts between interfacing organizations. Technical aspects include the properties of the physical and functional interfaces. Browning provides a list of desirable characteristics of both technical and managerial interface characteristics (Browning 2009, 1418-1419) .

System synthesis will include activities to understand the properties of system elements, the structure of proposed system solutions, and the resultant behavior of the composed system. A number of system concepts for describing system behavior are discussed in Concepts of Systems Thinking topic. It should be noted that in order to fully understand a system’s behavior, we must consider the full range of environments in which it might be placed and its allowable state in each. According to Page, in complex systems, the individual elements of the system are characterized by properties which enhance the systems as a whole, such as their adaptability (Page 2009).

Defining the System-of-Interest

Flood and Carson provide two ways to identify system boundaries: a bottom-up, or **structural approach**, which starts with significant system elements and builds out, and a top down, or **behavioral approach**, in which major systems needed to fulfill a goal are identified and then the work flows downward (Flood and Carson 1993). They identify a number of rules proposed by Beishon (Beishon 1980) and Jones (Jones 1982) to help in the selection of the best approach.

In either case, the ways in which system elements are refined, grouped, and allocated must be driven towards the synthesis of a realizable system solution description. A realizable solution must consider elements that are either already available, can be created from existing system elements, or are themselves described as system contexts which will need to be synthesized at a future point. In the third case, it is one of the outcomes of the Analysis and Selection between Alternative Solutions activities that is used to assess the risk that a given element may not be able to be synthesized in the required time limit or cost budget.

A top down approach might start with a system boundary and an overall description of system functions. Through the repeated application of element identification, division, grouping, and allocation of functions, a complete description of the elements needed for the SoI can be defined. In this case, the choice of system elements and allocation of functions may be guided by pre-defined ways of solving a given problem or by identified system patterns; both can support as well as insert bias into the synthesis. For example, one might start with the need to provide energy to a new housing project and propose solution options based around connections to an existing power grid, local power generators, renewable energy sources, increased energy efficiency, etc.

The iterative nature of analysis also reflects the need to change the solution as the life cycle progresses and changes the system's environment; thereby, possibly changing what a "best" solution is.

A bottom up approach starts with major elements and interactions. Again, division, grouping, and identification allows for the construction of a full system description that is capable of providing all the necessary functions, at which point the final SoI boundary can be set. In this case, the choice of system elements and groupings will be driven by the goal of ensuring that the major system elements can be formed together into a viable system whole. For example, there may be a need to replace an existing delivery vehicle and produce solution options that consider vehicle ownership/leasing, driver training, petrol, diesel or electric fuel, etc.

The systems approach aspect of synthesis leads to SE terms such as “design” and “development.” Wasson describes synthesis from a SE point of view (Wasson 2006, 390-690). White provides a comprehensive discussion of methods of achieving design synthesis (White 2009, 512-515). The systems approach treats synthesis at the abstract level while the SE process definitions provide the concrete steps.

The SoI brings together elements, sub-systems and systems through the concept of synthesis to identify a solution option.

Synthesis of possible solutions may result in the development of artifacts documenting the synthesis itself and provide the basis for analysis and selection between alternative solutions. These artifacts are dynamic and will change as the SoI changes its environment throughout the system life cycle.

References

Works Cited

Beishon, J. 1980. *Systems Organisations: the Management of Complexity*. Milton Keynes; Open University press.

Blanchard, B. and W.J. Fabrcky. 2006. *Systems Engineering and Analysis*. Upper Saddle River, NJ: Prentice Hall.

Boehm, B. 2009. "Applying the Incremental Commitment Model to Brownfield System Development". Proceedings of the 7th Annual Conference on Systems Engineering Research (CSER), Loughborough, UK.

Browning, T.R. 2009. "Using the Design Structure Matrix to Design Program Organizations". In Sage, A.P. and W.B. Rouse (eds.). *Handbook of Systems Engineering and Management,* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

Buede, D.M. 2009. "Functional Analysis". In Sage, A.P. and W.B. Rouse (eds.). *Handbook of Systems Engineering and Management,* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

Checkand, P. 1999. *Systems Thinking, Systems Practice*. New York, NY, USA: John Wiley & Sons.

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.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight*. 12(4) (December 2009): 59-63.

INCOSE. 1998. "INCOSE SE Terms Glossary." INCOSE Concepts and Terms WG (eds.). Seattle, WA, USA: International Council on Systems Engineering.

Jackson, S., D. Hitchins, and H. Eisner. 2010. "What is the Systems Approach?". INCOSE *Insight*. 13(1) (April 2010): 41-43.

Jones, L. 1982. "Defining System Boundaries in Practice: Some Proposals and Guidelines." *Journal of Applied Systems Analysis*, 9: 41-55.

Levin, A.H. 2009. "System Architectures". In Sage, A.P. and W.B. Rouse (eds.). *Handbook of Systems Engineering and Management*, 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

Page, S.E. 2009. "Understanding Complexity". The Great Courses. Chantilly, VA, USA: The Teaching Company.

Wasson, C.S. 2006. *System Analysis, Design, and Development*. Hoboken, NJ, USA: John Wiley & Sons.

White, Jr., K.P. 2009. "Systems Design." In Sage, A.P. and W.B. Rouse (eds.). *Handbook of Systems Engineering and Management*, 2nd ed. Hoboken, NJ, USA: John Wiley & Sons.

Primary References

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4) (December 2009): 59-63.

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 Engineer. ISO/IEC/IEEE 15288:2015.

Jackson, S., D. Hitchins and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight*. 13(1) (April 2010): 41-43.

Additional References

None

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Analysis and Selection between Alternative Solutions

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Brian Wells, Scott Jackson, Janet Singer, Duane Hybertson*

This topic is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It describes knowledge related to the analysis and selection of a preferred solution from the possible options, which may have been proposed by Synthesizing Possible Solutions. Selected solution options may form the starting point for Implementing and Proving a Solution. Any of the activities described below may also need to be considered concurrently with other activities in the systems approach at a particular point in the life of a system-of-interest (SoI).

The activities described below should be considered in the context of the Overview of the Systems Approach topic at the start of this KA. The final topic in this KA, Applying the Systems Approach, considers the dynamic aspects of how these activities are used as part of the systems approach and how this relates in detail to elements of systems engineering (SE).

System Analysis

System analysis is an activity in the systems approach that evaluates one or more system artifacts created during the activities involved in Synthesizing Possible Solutions, such as:

- Defining assessment criteria based on the required properties and behavior of an identified problem or opportunity system situation.
- Accessing the properties and behavior of each candidate solution in comparison to the criteria.
- Comparing the assessments of the candidate solutions and identification of any that could resolve the problem or exploit the opportunities, along with the selection of candidates that should be further explored.

As discussed in Synthesizing Possible Solutions topic, the problem context for an engineered system will include a logical or ideal system solution description. It is assumed that the solution that “best” matches the ideal one will be the most acceptable solution to the stakeholders. Note, as discussed below, the “best” solution should include an understanding of cost and risk, as well as effectiveness. The problem context may include a soft system conceptual model describing the logical elements of a system to resolve the problem situation and how these are perceived by different stakeholders (Checkland 1999). This soft context view will provide additional criteria for the analysis process, which may become the critical issue in selecting between two equally effective solution alternatives.

Hence, analysis is often not a one-time process of solution selection; rather, it is used in combination with problem understanding and solution synthesis to progress towards a more complete understanding of problems and solutions over time (see Applying the Systems Approach topic for a more complete discussion of the dynamics of this aspect of the approach).

Effectiveness Analysis

Effectiveness studies use the problem or opportunity system context as a starting point.

The effectiveness of a synthesized system solution will include performance criteria associated with both the system's primary and enabling functions. These are derived from the system's purpose, in order to enable the realization of stakeholder needs in one or more, wider system contexts.

For a product system there are a set of generic non-functional qualities that are associated with different types of solution patterns or technology, e.g., safety, security, reliability, maintainability, usability, etc. These criteria are often explicitly stated as parts of the domain knowledge of related technical disciplines in technology domains.

For a service system or enterprise system the criteria will be more directly linked to the identified user needs or enterprise goals. Typical qualities for such systems include agility, resilience, flexibility, upgradeability, etc.

In addition to assessments of the absolute effectiveness of a given solution system, systems engineers must also be able to combine effectiveness with the limitations of cost and timescales included in the problem context. In general, the role of system analysis is to identify the proposed solutions which can provide some effectiveness within the cost and time allocated to any given iteration of the systems approach (see Applying the Systems Approach for details). If none of the solutions can deliver an effectiveness level that justifies the proposed investment, then it is necessary to return to the original framing of the problem. If at least one solution is assessed as sufficiently effective, then a choice between solutions can be proposed.

Trade-Off Studies

In the context of the definition of a system, a trade-off study consists of comparing the characteristics of each candidate system element to those of each candidate system architecture, in order to determine the solution that globally balances the assessment criteria in the best way. The various characteristics analyzed are gathered in cost analysis, technical risks analysis, and effectiveness analysis (NASA 2007). To accomplish a trade off study there are a variety of methods, often supported by tooling. Each class of analysis is the subject of the following topics:

- Assessment criteria are used to classify the various candidate solutions. They are either absolute or relative. For example, the maximum cost per unit produced is c$, cost reduction shall be x%, effectiveness improvement is y%, and risk mitigation is z%.
- **Boundaries** identify and limit the characteristics or criteria to be taken into account at the time of analysis (e.g., the kind of costs to be taken into account, acceptable technical risks, and the type and level of effectiveness).
- **Scales** are used to quantify the characteristics, properties, and/or criteria and to make comparisons. Their definition requires knowledge of the highest and lowest limits, as well as the type of evolution of the characteristic (linear, logarithmic, etc.).
- An assessment score is assigned to a characteristic or criterion for each candidate solution. The goal of the trade-off study is to succeed in quantifying the three variables (and their decomposition in sub-variables) of cost, risk, and effectiveness for each candidate solution. This operation is generally complex and requires the use of models.
- The **optimization** of the characteristics or properties improves the scoring of interesting solutions.

A decision-making process is not an accurate science; ergo, trade-off studies have limits. The following concerns should be taken into account:

- Subjective Criteria – personal bias of the analyst; for example, if the component has to be beautiful, what constitutes a "beautiful" component?
- Uncertain Data – for example, inflation has to be taken into account to estimate the cost of maintenance during the complete life cycle of a system, how can a systems engineer predict the evolution of inflation over the next five years?

- Sensitivity Analysis – A global assessment score that is designated to every candidate solution is not absolute; thus, it is recommended that a robust selection is gathered by performing a sensitivity analysis that considers small variations of assessment criteria values (weights). The selection is robust if the variations do not change the order of scores.

A thorough trade-off study specifies the assumptions, variables, and confidence intervals of the results.

Systems Principles of System Analysis

From the discussions above, the following general principles of systems analysis can be defined:

- Systems analysis is an iterative activity consisting of trade studies made between various solution options from the systems synthesis activity.
- Systems analysis uses assessment criteria based upon a problem or opportunity system description.
  - These criteria will be based around an ideal system description that assumes a hard system problem context can be defined.
  - The criteria must consider required system behavior and properties of the complete solution in all of the possible wider system contexts and environments.
  - Trade studies require equal consideration to the primary system and the enabling system working as a single sytem to address the User need. These trades need to consider system requirements for Key Performance Parameters (KPPs), systems safety, security, and affordability across the entire life cycle
  - This ideal system description may be supported by soft system descriptions from which additional “soft” criteria may be defined (e.g., a stakeholder preference for or against certain kinds of solutions and relevant social, political, or cultural conventions to be considered in the likely solution environment, etc.).
- At a minimum, the assessment criteria should include 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 “system of assessment criteria”, designating special attention to 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.

References

Works Cited

Checkland, P.B. 1999. *Systems Thinking, Systems Practice*. Chichester, UK: John Wiley & Sons Ltd.

NASA. 2007. *Systems Engineering Handbook*, Revision 1. Washington, DC, USA: National Aeronautics and Space Administration (NASA). NASA/SP-2007-6105.

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 Engineer. ISO/IEC/IEEE 15288:2015.

Jackson, S., D. Hitchins and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight.* 13(1) (April 2010): 41-43.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Implementing and Proving a Solution

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Brian Wells, Scott Jackson, Janet Singer, Duane Hybertson*

This topic is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It describes knowledge related to the implementation and proving of a preferred solution that may have been selected by activities described in the Analysis and Selection between Alternative Solutions topic. The activities that apply to an implemented solution during its operational life are described in Deploying, Using, and Sustaining Systems to Solve Problems topic and how systems fit into commercial and acquisition relationships is discussed in the Introduction to System Fundamentals topic. Any of the activities described below may also need to be considered concurrently with other activities in the systems approach at a particular point in the life of a system-of-interest (SoI).

The activities described below should be considered in the context of the Overview of the Systems Approach topic at the start of this KA. The final topic in this KA, Applying the Systems Approach, considers the dynamic aspects of how these activities are used as part of the systems approach and how this relates in detail to elements of systems engineering (SE).

Proving the System Overview

This topic covers both the sub-topics of verification and validation.

Verification

Verification is the determination that each element of the system meets the requirements of a documented specification (see principle of elements). Verification is performed at each level of the system hierarchy. In the systems approach this topic pertains to the more abstract level of providing evidence that the system will accomplish what it was meant to do. In SE this topic pertains to providing quantitative evidence from tests and other methods for verifying the performance of the system.

Validation

Validation is the determination that the entire system meets the needs of the stakeholders. Validation only occurs at the top level of the system hierarchy. In the systems approach this topic pertains to the more abstract level of making sure the system meets the needs of the stakeholders. In SE, this topic pertains to the detailed demonstrations and other methods that are used to promote stakeholder satisfaction.

In a SE context, Wasson provides a comprehensive guide to the methods of both system verification and system validation (Wasson 2006, 691-709).

References

Works Cited

Wasson, C. S. 2006. *System Analysis, Design, and Development.* Hoboken, NJ, USA: John Wiley & Sons.

Primary References

Jackson, S., D. Hitchins and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight*. 13(1) (April 2010): 41-43.

Additional References

MITRE. 2012. "Verification and Validation." *Systems Engineering Guide.* http:/ / mitre. org/ work/ systems_engineering/ guide/ se_lifecycle_building_blocks/ test_evaluation/ verification_validation. html (accessed September 11, 2012)

Wasson, C. S. 2006. *System Analysis, Design, and Development.* Hoboken, NJ, USA: John Wiley & Sons.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Deploying, Using, and Sustaining Systems to Solve Problems

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Brian Wells, Scott Jackson, Janet Singer, Duane Hybertson*

This topic is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It describes knowledge related to the deployment, sustainment, and use of a solution, that may have been developed through the activities described in the Implementing and Proving a Solution topic. Discussion of how a deployed systems fit into commercial and acquisition relationships in Introduction to System Fundamentals. Any of the activities described below may also need to be considered concurrently with other activities in the systems approach at a particular point in the life of a system-of-interest (SoI).

The activities described below should be considered in the context of the Overview of the Systems Approach topic at the start of this KA. The final topic in this KA, Applying the Systems Approach, considers the dynamic aspects of how these activities are used as part of the systems approach and how this relates in detail to elements of systems engineering (SE).

Introduction

Part 3, Systems Engineering and Management, of the Guide to the SE Body of Knowledge (SEBoK) provides two additional KAs that address the engineering aspects of these steps of the systems approach. In KA Product and Service Life Management and System Deployment and Use, in Part 3 explain the SE aspects of deployment, operation, maintenance, logistics, service life extension, updates, upgrades, disposal and the retirement of systems.

A systems approach considers the total system and the total life cycle of the system. This includes all aspects of the system and the system throughout its life until the day users depose of the system and the external enterprises complete the handling of the disposed system products. Creation of the system is rarely the step that solves the stakeholders' problems. It is the use of the system solution that solves the problem. So from this perspective the deployment, use and sustainment of the system are important concepts that must be a part of the systems approach.

Engineered systems are eventually owned by an individual, team, or enterprise. Those who own the system during development may not be the ones who own the system when it is in operation. Moreover, the owners may not be the users; e.g., service systems may be used by the general public but owned by a specific business that is offering the service. The transition of a system from development to operations is often itself a complex task, involving such activities as training those who will operate the system, taking legal actions to complete the transfer, and establishing logistical arrangements so that the operators can keep the system running once the transition is completed.

A complete systems approach must also consider the many enterprises involved in the system from initial conception through the completion of the disposal process. These enterprises are all stakeholders with requirements , which all have interfaces that must be considered as part of a total systems approach.

There is very little in the literature pertaining to the application of the systems approach to these phases of the life cycle. However, a basic premise of this KA is that the systems approach pertains to all phases of a system's life cycle. Hence, to properly build systems to solve problems or for other uses, it can be inferred that the systems approach pertains to the deployment, use, and the sustainment of the systems. Many of the available references in this topic area are from SE literature rather than from literature associated with the systems approach so the reader should also see Part 3, Systems Engineering and Management, of the SEBoK.

Deployment: The Transition from Development to Operation

Transferring custody of the SoI and responsibility for its support from one organization to another occurs during deployment and is often called transition (INCOSE 2011). Transition of a product system includes the integration of the system into the acquiring organization's infrastructure. Deployment and transition involves the activity of moving the system from the development to the operational locations, along with the support systems necessary to accomplish the relocation.

Transition includes the initial installation of a system and the determination that it is compatible with the wider system and does not cause any significant wider system issues. This process of acceptance and release for use varies between domains and across businesses and enterprises and can be thought of as an initial assessment of the system's effectiveness (Hitchins 2007). Generally, transition may be considered as having two parts: 1.) ensuring that the new system interoperability with the systems around it and 2.) ensuring the resulting system is safe and possesses other critical operational properties.

It is particularly important to considered emergent properties when a new system is added to the existing organization's system of systems (SoS) network, as well as the complexity of the organization into which the new system is transitioned (see also Complexity). The more complex the receiving organization is the more challenging the transition will be, and the greater the likelihood of unintended interactions and consequences from the new system's insertion. Dealing with the consequences of this complexity starts in transition and continues into operation, maintenance, and disposal.

Transition of a service system is often performed in two stages. First, the service system infrastructure is accepted and released. Second, each realization of the service is accepted and released. There can be significant problems during the second stage if the required responsiveness of the service does not leave sufficient time to ensure that the service meets necessary functional and quality attributes, including interoperability, safety, and security. (See Service Systems Engineering).

Transition and deployment of a system may introduce unique requirements that are not necessary for operation or use. These requirements can influence the design of the system; therefore, must be considered during the initial requirements and design stages. The most common examples are related to the need to transport the system or system elements, which often limits the size and weight of the system elements.

Transition can also require its own enabling systems, each of which can be realized using a systems approach.

Use: Operation

Use of the system to help enable delivery of user services is often called “operations” (INCOSE 2011). A system’s effectiveness is normally considered throughout the operational life of a system. For a complex system, emergent behavior should be considered in three ways:

- to identify and plan for emergent properties within the system realization process (See System Realization KA in Part 3, Systems Engineering and Management)
- to incorporate mechanisms for identifying and handling unexpected emergent properties within the system during its use
- to provide necessary procedures for dealing with wider system consequences of unexpected emergent properties in the enterprise (e.g., emergency responses or medical first aid)

Operations require their own enabling systems, each of which can be realized using a systems approach.

System Sustainment and Maintenance

System sustainment requires maintenance of the system throughout its useful life (INCOSE 2011). In system terms, maintenance implements systems that handle entropy and maintaining the SoI in a viable state. Since an open system maintains its existence by continual exchange of energy, information, and materiel with it’s environment, one aspect of its maintenance must be the management of resources in the environment.

Hitchins (Hitchins 2007) describes generic approaches to resource management and viability management based on systems concepts. Resource management identifies the need to consider the acquisition, storage, distribution, conversion, and disposal of resources. Viability management should consider systems to maintain homeostasis and a means for ensuring resilience to environmental disturbance and adaptability to environmental change.

Maintenance will require its own enabling systems, each of which can be realized using a systems approach. Maintenance success is more likely if it is considered as part of the system concept and design well before the system enters service.

Disposal

A total life cycle systems approach cannot be considered complete without consideration of how disposal of the system will be accomplished. The purpose of disposal is to remove a system element from the operational environment with the intent of permanently terminating its use and remove any hazardous or toxic materials or waste products (INCOSE 2011).

During disposal the entirety of the open system crosses the boundary from the system side to the environment. A complete systems approach must consider how it crosses the boundary and what remains that must be managed by enterprises other than the ones that developed, used or sustained the system. Including disposal in the system approach expands the stakeholders, the enterprises and the external systems that must be considered.

Disposal requires its own enabling systems, each of which can be realized using a systems approach. Some of these may be contained within the system boundaries and others may be external to the system. For the external disposal systems, the interface where the handover occurs must be considered. As with maintenance, a large part of successful disposal requires related issues to have been considered early on in the system’s life cycle.

The topic Disposal and Retirement in Part 3, Systems Engineering and Management, of the SEBoK provides information on the engineering aspects of system disposal.

References

Works Cited

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

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

Primary References

INCOSE. 2011. *Systems Engineering Handbook*, version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering. INCOSE-TP-2003-002-03.2.1.

Additional References

MITRE. 2011. "Transformation Planning and Organizational Change." . *Systems Engineering Guide.* Accessed December 4 2014 at MITRE http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/transformation_planning_org_change/.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Applying the Systems Approach

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Brian Wells, Scott Jackson, Janet Singer, Duane Hybertson, Hillary Sillitto, Bud Lawson, James Martin*

The systems approach relates to both the dynamics of problem resolution and stakeholder value over time, as well as to the levels of system relationship, detailed management, and the engineering activities this implies.

This article builds on the concepts introduced in Overview of the Systems Approach topic. It is part of the Systems Approach Applied to Engineered Systems knowledge area (KA), which describes, primarily through five groups of activities, the application of an approach based around systems thinking to engineered system contexts throughout their lives.

Life Cycle

Engineered Systems provide outcomes which deliver benefits to stakeholders by helping them achieve something of value in one or more problem situations. Ultimately, a system is successful only if it enables successful outcomes for its stakeholders (Boehm and Jain 2006). In complex real world situations value can best be provided through a continuing process of adapting the system needs and developing associated solutions in response to changing circumstances, according to the principle of **Progressive Satisfying** (Hitchins 2009).

A value cycle associating the systems approach to the deliver real world stakeholder benefits is discussed in the Overview of the Systems Approach topic. A greater understanding of the value of an engineered system within its context enables agreement on the problem situation and appropriate system interventions to be created, deployed, and used overall, in turn enabling a more effective application of the systems approach. Value is fully realized only when considered within the context of time, cost, funding and other resource issues appropriate to key stakeholders (Ring 1998).

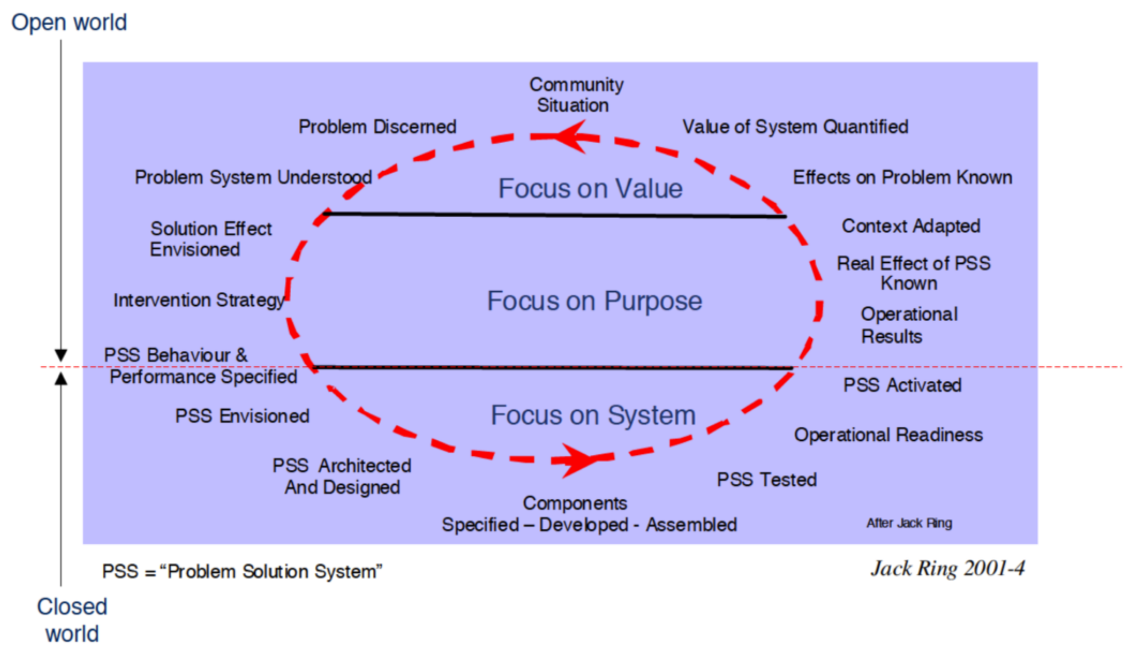


Figure 1. Ellipse Graphic (Ring 1998). © 1998 IEEE. Reprinted, with permission, from Jack Ring, Engineering Value-Seeking Systems, IEEE-SMC. Conference Proceedings. All other rights are reserved by the copyright owner.

The views in Figure 1 apply the idea of **Systemic Intervention** to the resolution of problem situations in which one or more engineered system solution might be required. For each turn of the cycle an agreement is made between stakeholders and developers that an Engineered System to solve problem X with effectiveness Y in agreed conditions Z has a chance of delivering value A for which we are willing to invest cost B and other resources C.

It is in the nature of wicked problems that this proposition cannot be a certainty. life cycle approaches to understand and manage the shared risk of tackling such problems are discussed in Life Cycle Models. The idea of Systemic Intervention comes from soft systems thinking (see Systems Approaches).

For each of the engineered system problems, the solutions agreed above must be developed such that they are effective in terms of cost, performance and other properties relevant to the problem domain. A developer must consider not only what to do, but when and how much to do to provide real value (Senge 1990). In systems engineering (SE) and management practices this leads to the two key concepts (INCOSE 2011):

- **Life Cycles:** Stakeholder value and problem resolution described as a set of life cycle stages over which problems can be explored and resolved, and resources can be managed.
- **Life Cycle Processes:** Systems of activities focused on creation and sharing of knowledge associated with the systems approach, that can be employed to promote a holistic approach over a life cycle.

Life cycle management provides the framework in which to take a systems approach to all aspects of an engineered system context, which includes not only the system product or service but also the systems to create, deploy and support it (Martin 2004). The following sections consider how the systems approach should be applied to an identified problem statement, within the context of the overall value cycle discussed above.

Application Principles

Concurrency

Within any application of the systems approach the activities of problem identification, solution synthesis and selection; solution implementation and proving and deployment, sustainment and use should be applied concurrently, reflecting their interrelationships and dependencies.

The system value cycle (Ring 1998) can be taken as a generic model of the life of an engineered system within a problem resolution cycle driven by stakeholder value. For practical reasons it is necessary to break this life down into a set of finite stages, to allow activities to be organized. We can express the value cycle as six groups of questions to cycle around value, problem, and solution questions that are related to the systems approach:

1. What values do Stakeholders want/need?
2. What system outcomes could improve this value?
3. What system can provide these outcomes?
4. How do we create such a system?
5. How do we deploy and use the system to achieve the outcomes?
6. Do these outcomes provide the expected improvement in value?

The above questions focus on each iteration of the systems approach to deliver stakeholder goals within an enterprise context. Activities 1 and 6 are part of the business cycles of providing stakeholder value within an enterprise, whereas activities 2 through 5 can be mapped directly to product, service, and enterprise engineering life cycles. A distinction is made here between the normal business of an enterprise and the longer-term strategic activities of Enterprise Systems Engineering.

The following diagram illustrates the concurrent nature of the activities of a systems approach over time.

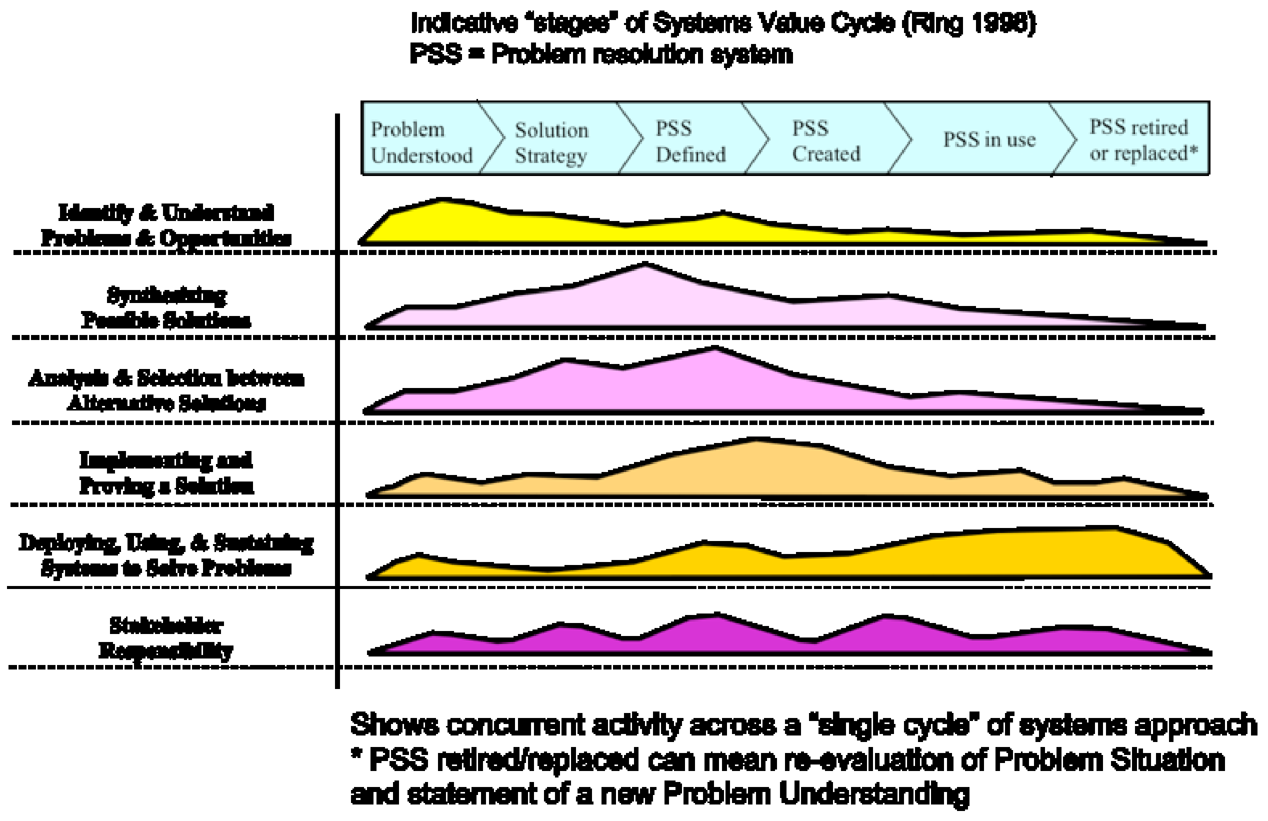

Indicative "stages" of Systems Value Cycle (Ring 1998)
PSS = Problem resolution system
Problem Understood
Solution Strategy
PSS Defined
PSS Created
PSS in use
PSS retired or replaced*
Identify & Understand Problems & Opportunities
Synthesizing Possible Solutions
Analysis & Selection between Alternative Solutions
Implementing and Proving a Solution
Deploying, Using, & Sustaining Systems to Solve Problems
Stakeholder Responsibility
Shows concurrent activity across a "single cycle" of systems approach
* PSS retired/replaced can mean re-evaluation of Problem Situation and statement of a new Problem Understanding

Figure 2. Activities of the Systems Approach Applied within a System Life Cycle. (SEBoK Original)

The lines on Figure 2 represent activity in each of the activity areas over a simple (not to scale) life cycle based on the questions above. Activities may have a primary focus in certain stages, but need to span the whole of life to ensure a holistic approach. For example, problem identification has a large input during the Problem Understanding stage, but problems are refined, reviewed, and reassessed over the rest of the life cycle. Similarly, Implement and Proving activities are conducted during the transition from Create to Use. This is only possible if proving issues, strategies, and risks are considered in earlier stages. This diagram is a schematic representation of these activity mappings, sometimes called a hump diagram (Kruchten 2003).

For the generic systems approach, the following fundamental life cycle principles apply:

- A life cycle has groups of stages which cover understanding stakeholder value; exploration of a problem situation (see System Definition); creation of a system solution (see System Realization); and System Deployment and Use.
- Life cycle processes define a system of engineering and management activities based on the detailed information needed to ensure a systems approach across a life cycle (e.g., requirements, architecture, verification, and validation).
- Activities in any of the processes may be employed in all of the stages to allow for appropriate concurrency.
- The sequence and control of the life cycle stages and concurrent process activities must be tailored to the problem situation and commercial environment (Lawson 2010), thus leading to the selection of an appropriate life cycle model.
- Appropriate management activities must be included in the life cycle to ensure consideration of time, cost, and resource drivers.
- In focusing on the creation of a specific system-of-interest (SoI) to provide solutions within the cycle, it is important to recognize the need to employ the right balance between reductionism and holism by considering the appropriate system context.

The ways in which this idea of concurrent process activity across a life cycle has been implemented in SE are discussed in Systems Engineering and Management.

Iteration

The systems approach can be applied in an iterative way to move towards an acceptable solution to a problem situation within a larger cycle of stakeholder value.

The systems approach can be applied to multiple systems within an engineered system context, as discussed below. At each level, the approach may be applied iteratively to cycle between what is needed and versions of the solutions within a life cycle model.

Hitchins (Hitchins 2009) defines two principle related to iterations:

- **Adaptive Optimizing —** Continual redesign addresses the problem space, detecting and addressing changes in situation, operational environment, other interacting systems, and other factors; it continually conceives, designs, and implements or reconfigures the whole solution system to perform with optimal effectiveness in the contemporary operational environment.
- **Progressive Entropy Reduction —** Continual performance and capability improvement of systems in operation may be undertaken by customer or user organizations with or without support from industry, as they seek to “get the best” out of their systems in demanding situations. In terms of knowledge or information, this process involves progressively reducing entropy, going from a condition of high entropy (that is, disorder) at the outset to low entropy (order) at the finish.

In general, these two cycles of iterations can be realized from combinations of three life cycle types (Adcock 2005):

- Sequential: With iteration between the stages to solve detailed issues as they arise, a single application of the systems approach is sufficient.

- Incremental: Successive versions of the sequential approach are necessary for a solution concept. Each increment adds functionality or effectiveness to the growing solution over time.
- Evolutionary: A series of applications of the sequential approach for alternative solutions intended to both provide stakeholder value and increase problem understanding. Each evolutionary cycle provides an opportunity to examine how the solution is used so these lessons learned can be incorporated in the next iteration.

These aspects of the systems approach form the basis for life cycle models in Life Cycle Models.

Recursion

The stakeholder value, problem resolution, and system creation aspects of the system value cycle may each require the use of a focused systems approach. These might be soft systems to prove a better understanding of a situation, product systems and/or service systems solutions to operational needs, enabling systems to support an aspect of the product or service life cycle, or enabling systems used directly by the enterprise system.

Each of these systems may be identified as a system-of-interest (SoI)and require the application of the systems approach. This application may be sequential (the start of one system approach dependent on the completion of another) or parallel (independent approaches which may or may not overlap in time), but will often be recursive in nature.

Recursion is a technique borrowed from computer science. In computer science recursion occurs when a function calls itself repeatedly to logically simplify an algorithm. In a recursive application applied to systems, the systems approach for one system-of-interest is nested inside another. Examples include cases where

- trades made at one level of the system require trades to be made for system elements;
- the analysis of a system requires analysis of a system element;
- the synthesis of a solution system requires one or more sub-system elements; and
- the verification of a product system requires verification of system elements.

In each case, the “outer” system approach may continue in parallel with the “inner” to some extent, but depends on key outcomes for its own progress.

As with all recursive processes, at some stage the application of the approach must reach a level at which it can be completed successfully. This then "rolls up" to allow higher levels to move forward and eventually complete all nested applications successfully.

The INCOSE *Systems Engineering Handbook* (INCOSE 2011) describes a recursive application of SE to levels of system element with each application representing a system project. Martin (1997) describes the recursive application of SE within a product system hierarchy until a component level is reached, at which point procurement of design and build processes can be used to create solution elements.

The principle of recursive application and how it relates to life cycle models is described in Life Cycle Models. This topic is part of the Systems Approach Applied to Engineered Systems knowledge area (KA). It summarizes various aspects of stakeholder responsibility for acquisition and ownership during the system life cycle processes covered by such sources as the International Council on Systems Engineering Handbook (INCOSE 2012). Any of the activities described below may also need to be considered concurrently with other activities in the systems approach at a particular point in the life of a system-of-interest (SoI).

The activities described below should be considered in the context of the Overview of the Systems Approach topic at the start of this KA. The final topic in this KA, Applying the Systems Approach, considers the dynamic aspects of how these activities are used as part of the systems approach and how this relates in detail to elements of systems engineering (SE).

Stakeholder Responsibilities

The general principles of life cycle application discussed above apply as necessary to each application of SE. The following sections clarify the different kinds of stakeholder and the roles they take for the different system contexts discussed in the SEBoK.

Products, Services, and Enterprises

Most often, the terms "product" and "service" describe the effects that are exchanged in a customer and supplier agreement. This may be a commercial agreement, one funded publicly by a charity, or provided by a government agency. The difference between a product and a service is that a product is an artifact acquired to achieve an outcome while a service is an outcome supplied directly to a user.

The terms “customer” and “user” are often used interchangeably in engineering and management disciplines. The INCOSE *Systems Engineering Handbook* (INCOSE 2012) makes the following specific distinctions among the stakeholders associated with a system:

- The acquirer is the stakeholder that acquires or procures a product or service from a supplier.
- The supplier is an organization or individual that enters into an agreement with the acquirer to supply a product or service.
- The operator is an individual or organization that that uses knowledge, skills and procedures to perform the functions of the system to provide the product or service.
- The user or customer is the individual or group that benefit from the operation of the system.

These terms define the roles stakeholders take; however, they may not always lie within these distinct entities (e.g. sometimes the acquirer may also be the user). This also applies to service systems, as some of the entities may also overlap in roles. Parnell et al., (Parnell et al. 2011) offer an alternative list of stakeholders that include decision authority, client, owner, user, consumer, and interconnected.

Product systems consist of hardware, software, and humans, and they have traditionally been the focus of SE efforts. These systems are delivered to the acquirer and operated to accomplish the goals that led to the requirements for the system. These requirements were derived from the need to provide products and services to one or more users as part of an enterprise.

The delivery (supplying) of a service is indicative of the direct delivery of an outcome, which is often related to the delivery of products (e.g., a maintenance, training, or cleaning service). This is not the same as the delivery of a service system (see the discussion below).

In traditional SE, the term “service” or “service system” refers to the wider system context that describes the acquirer's need to deliver user value. In this case, the service system is a fixed system definition that dictates the manner in which the acquiring enterprise will utilize the products to enable the delivery of services to users. Product systems are designed to be integrated and operated as appropriate to enable this service to be maintained or improved as required. In this view, a service system is static and contains dedicated products, people, and resources; that is, hierarchies of products are engineered to provide acquirers with the ability to offer predefined services to users or customers.

More recently, the term "service systems" has been used to describe a system that is engineered in a manner that allows enterprises to offer services directly to users, bypassing the need to hold all of the necessary products and services within the enterprise itself. This requires the expansion of the definition of a “supplier” as follows:

- A **product supplier** is an organization or individual that enters into an agreement with an acquirer to supply a product or related product support services.
- A **service system supplier** is an organization or individual that enters into an agreement with an acquirer to supply a service system.
- A **service supplier** is an organization or individual that enters into an agreement with a user to supply a service.

These service systems tend to be configured dynamically to deal with problems that traditional static service find challenging to address. This view of a service system employs "late binding" with product systems that are not owned by the enterprise, but are used to enable the service to be offered as closely to given time demands as possible. This is the definition of a service system used in the Service Systems Engineering topic in Part 4, Applications of Systems Engineering.

Stakeholder Needs

One of the most critical stakeholder responsibilities is to identify the needs and requirements for the system that provides the products or services (INCOSE 2012).These needs and requirements are expressed in agreements between acquirers and suppliers.

There are other stakeholders who shape system requirements based on their needs, but who are not necessarily acquirers or suppliers. The stakeholders and the requirements engineers share the responsibility to identify their needs during the requirements process.

Acquirer/Supplier Agreements

Lawson (Lawson 2010) provides a perspective on what it means to own systems, trade in system products and services, and the implications of supply chains in respect to the value added and ownership of the systems, its products and services. INCOSE (INCOSE 2012) defines two life cycle processes related to acquisition and supply. The acquisition process includes activities to identify, select, and reach commercial agreements with a product or service supplier.

In many larger organizations, there is a tradition of system ownership vested in individuals or, in some cases, enterprise entities (groups or teams). Ownership implies the authority and responsibility to create, manage, and dispose of a system-of-interest (SoI), as well as sometimes to operate the SoI.

Product Acquire/Supply

In some industries, a supplier works directly with an acquirer to help understand the acquirer’s needs and then engineer one or more products to satisfy these needs. In certain cases, a single supplier will provide the complete worthy product system. In other cases, a supply chain will be formed to deliver product systems with a system integrator to ensure they fit together and integrate into the wider context. This is a theoretical view of product systems engineering in which the context is fixed and the product is designed to fit into it. A good systems engineer may suggest changes to the enterprise as a better way to solve the problem and then modify the product system’s requirements accordingly. However, at some point, an agreed context will be set and a product system developed to work within it.

For many commercial products, such as mobile phones, a supplier creates a representative user profile to generate the requirement and then markets the product to real users once it is realized. In these cases, the other elements of the systems approach are performed by the acquirer/user and may not follow formal SE processes. It is important that a product supplier takes this into account when considering the best manner in which to engineer a system, as additional help or support services may need to be offered with the purchased product. The idea of a supplier offering support services for users with a type of product purchased elsewhere (e.g., an auto-mechanic servicing different makes of cars) begins to overlap with the service systems context, as discussed in the next topic.

For an institutionalized infrastructure in which SoIs are entirely owned by an enterprise or parties thereof, the entire responsibility of life cycle management, including operation, is often vested with the system owners. These systems belong to the system asset portfolio of an enterprise or multiple enterprises and provide the system resources, including the planned systems that are developed during life cycle management.

Service Acquire/Supply

Organizations providing service systems need not own the individual products and services that they deliver to their users and customers. With this viewpoint, the supplied service system includes the means to identify and gain access to appropriate products or services when needed. The service systems then would then be the bundle of products and services assembled for the user; for example, assembling software applications and service agreements for a mobile phone already owned by a user. The enterprises providing service systems may, in turn, offer infrastructure services to a wide range of different technologies or application domains. This can mean that the transition, operation, maintenance and disposal activities associated with system ownership may not be embedded in the acquiring service system enterprise, and will therefore need to be treated as separate system services. More detail can be found in Product Systems Engineering, Service Systems Engineering, and Enterprise Systems Engineering, in Part 4, Applications of Systems Engineering in the *Guide to the Systems Engineering Body of Knowledge* (SEBoK).

The service systems engineer helps the service supplier create and sustain the service system that can be used to discover, integrate, and use specific versions of generic products or services when needed. The realization of service systems requires the ability to make use of product systems; however, these product systems are developed and owned outside of the service system. The service system must be able to gain access to a product or service when needed, as well as to interface with it effectively. The use of open interface standards, such as standard power supplies, interface connections (e.g., Universal Serial Bus (USB)), or file formats (e.g., Portable Document Format (PDF)) can help make this easier.

Enterprise Evolution

A useful distinction between product system design and enterprise system design is that "enterprise design does not occur at a single point in time like the design of most systems. Instead, enterprises evolve over time and are constantly changing, or are constantly being designed" (Giachetti 2010, xiii).

The enterprise developer may also aim to optimize back stage processes (the internal operations) of an organization or an institution by exploiting advances in technology, particularly information technology (IT) and associated processes. In these cases, the engineered systems are considered to be enterprise systems.

Enterprise systems may offer products (goods) and/or services. From an enterprise engineering viewpoint, an enterprise concurrent with its product SE must not only look at the development and delivery of the products but also look at the alignment and optimization of the product delivery within the enterprise objectives. Similarly, in service SE, the main focus is on an intangible value delivery to the end-customer (externally focused: front stage), in which internal and external processes must be synchronized. However, with the rapid advances in information and communications technologies (ICT), in many cases the boundaries between internal and external processes are quite blurred. Current SE research is extending product methods, processes, and tools into the enterprise transformation and service innovation fields to exploit advances in business process methodologies and technologies.

Enterprise SE must not only do the engineering of the enterprise itself, but may also be involved in the engineering of the service systems and products systems that are necessary for the enterprise to achieve its goals.

Activity Mapping

This topic belongs to the Systems Approach Applied to Engineered Systems KA from Part 2 Foundations of Systems Engineering. Other topics about activities, from the same KA, relate to high level technical processes defined in KAs in Part 3, Systems Engineering and Management, in the following way:

- Identifying and Understanding Problems and Opportunities topic relates to the Concept Definition KA.
- Synthesizing Possible Solutions and Analysis and Selection between Alternative Solutions topics relate to the System Definition KA.
- Implementing and Proving a Solution topic relates to the System Realization KA.
- Deploying, Using, and Sustaining Systems to Solve Problems topic relates to the Product and Service Life Management KA.

Part 3 discusses the principles defined in each of the systems approach activities, and how they help shape the technical processes to which they are mapped.

References

Works Cited

Adcock, R.D. 2005. "Tailoring Systems Engineering Lifecycle Processes to meet the challenges of Project and Programme". *INCOSE International Symposium 2005*. Volume 15. Issue 1.

Boehm, B. and A. Jain. 2006. "A value-based theory of Systems Engineering." Presented at the 16th Annual INCOSE Systems Engineering Conference (Orlando FL).

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4).

INCOSE. 2011. *INCOSE Systems Engineering Handbook*, version 3.2.1. San Diego, CA, USA: International Council on Systems Engineering. INCOSE-TP-2003-002-03.2.1.

Giachetti, R.E. 2010. *Design of Enterprise Systems: Theory, Architecture, and Methods.* Boca Raton, FL, USA: CRC Press.

Kruchten, P. 2003. *The Rational Unified Process: An Introduction,* 3rd edition. Boston, MA: Addison Wesley.

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

Martin J.N. 1997. *Systems Engineering Guidebook*. Boca Raton, FL, USA: CRC Press.

Martin, J, 2004. "The Seven Samurai of Systems Engineering: Dealing with the Complexity of 7 Interrelated Systems." INCOSE 2004 - 14th Annual International Symposium Proceedings.

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.

Ring, J. 1998. "A Value Seeking Approach to the Engineering of Systems." Proceedings of the IEEE Conference on Systems, Man, and Cybernetics. p. 2704-2708.

Senge, P. 1990. *The Fifth Discipline: The Art and Practice of the Learning Organization*. New York, NY, USA: Doubleday/Currency.

Primary References

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4).

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

Additional References

Blanchard, B. and W.J. Fabrycky 2006. *Systems Engineering and Analysis.* Upper Saddle River, NJ: Prentice Hall.

Carlock, P.G. and R.E. Fenton. 2001. "System of Systems (SoS) enterprise systems engineering for information-intensive organizations." *Systems Engineering.* 4(4): 242–261.

Checkland, P. 1999. *Systems Thinking, Systems Practice*. New York, NY, USA: John Wiley & Sons.

Rouse, W.B. 2005. "Enterprises as Systems: Essential Challenges and Approaches to Transformation." *Systems Engineering.* 8(2): 138-150.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Knowledge Area: Systems Science

Systems Science

Lead Author: *Rick Adcock*

This knowledge area (KA) provides a guide to some of the major developments in systems science which is an interdisciplinary field of science that studies the nature of complex systems in nature, society, and engineering.

This is part of the wider systems knowledge which can help to provide a common language and intellectual foundation; and make practical systems concepts, principles, patterns and tools accessible to systems engineering (SE) as discussed in the Introduction to Part 2.

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:

- History of Systems Science
- Systems Approaches

Introduction

The following diagram summarizes the relationships between systems science and other sections of the SEBoK

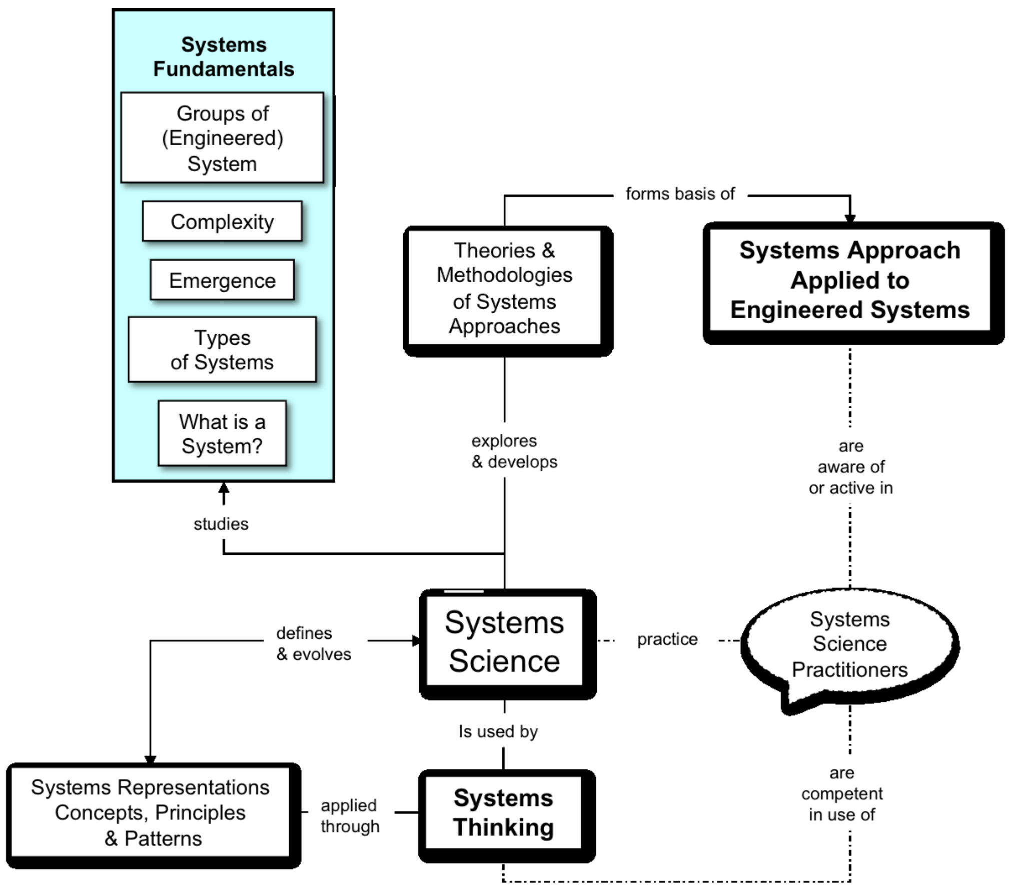

Systems Fundamentals
Groups of (Engineered) System
Complexity
Emergence
Types of Systems
What is a System?
studies
forms basis of
Theories & Methodologies of Systems Approaches
Systems Approach Applied to Engineered Systems
explores & develops
are aware of or active in
defines & evolves
Systems Science
practice
Systems Science Practitioners
Is used by
are competent in use of
Systems Representations Concepts, Principles & Patterns
applied through
Systems Thinking

Figure 1. Systems Science in the SEBoK. (SEBoK Original)

Systems science brings together research into all aspects of systems with the goal of identifying, exploring, and understanding patterns of complexity which cross disciplinary fields and areas of application. It seeks to develop interdisciplinary foundations which can form the basis of theories applicable to all types of systems, independent of element type or application; additionally, it could form the foundations of a meta-discipline unifying traditional scientific specialisms.

The History of Systems Science article describes some of the important multidisciplinary fields of research of which systems science is composed.

A second article presents and contrasts the underlying theories behind some of the system approaches taken in applying systems science to real problems.

People who think and act in a systems way are essential to the success of both research and practice. Successful systems research will not only apply systems thinking to the topic being researched but should also consider a systems thinking approach to the way the research is planned and conducted. It would also be of benefit to have people involved in research who have, at a minimum, an awareness of system practice and ideally are involved in practical applications of the theories they develop.

References

Works Cited

None.

Primary References

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: John Wiley & Sons.

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY, USA: Braziller.

Flood, R.L. 1999. *Rethinking the Fifth Discipline: Learning within the Unknowable.* London, UK: Routledge.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

History of Systems Science

Lead Author: *Rick Adcock*, ***Contributing Authors:*** *Scott Jackson, Janet Singer, Duane Hybertson*

This article is part of the Systems Science knowledge area (KA). It describes some of the important multidisciplinary fields of research comprising systems science in historical context.

Systems science, is an integrative discipline which brings together ideas from a wide range of sources which share a common systems theme. Some fundamental concepts now used in systems science have been present in other disciplines for many centuries, while equally fundamental concepts have independently emerged as recently as 40 years ago (Flood and Carson 1993).

The “Systems Problem”

Questions about the nature of systems, organization, and complexity are not specific to the modern age. As International Council on Systems Engineering (INCOSE) pioneer and former International Society for System Sciences (ISSS) President John Warfield put it, “Virtually every important concept that backs up the key ideas emergent in systems literature is found in ancient literature and in the centuries that follow.” (Warfield 2006) It was not until around the middle of the 20th Century, however, that there was a growing sense of a need for, and possibility of a scientific approach to problems of organization and complexity in a “science of systems” per se.

The explosion of knowledge in the natural and physical sciences during the 18th and 19th centuries had made the creation of specialist disciplines inevitable: in order for science to advance, there was a need for scientists to become expert in a narrow field of study. The creation of educational structures to pass on this knowledge to the next generation of specialists perpetuated the fragmentation of knowledge (M’Pherson 1973).

This increasing specialization of knowledge and education proved to be a strength rather than a weakness for problems which were suited to the prevailing scientific methods of experimental isolation and analytic reduction. However there were areas of both basic and applied science that were not adequately served by those methods alone. The systems movement has its roots in two such areas of science: the biological-social sciences, and a mathematical-managerial base stemming first from cybernetics and operations research, and later from organizational theory.

Biologist Ludwig von Bertalanffy was one of the first to argue for and develop a broadly applicable scientific research approach based on **Open System Theory** (Bertalanffy 1950). He explained the scientific need for systems research in terms of the limitations of analytical procedures in science.

These limitations, often expressed as emergent evolution or "the whole is more than a sum of its parts", are based on the idea that an entity can be resolved into and reconstituted from its parts, either material or conceptual:

> *This is the basic principle of "classical" science, which can be circumscribed in different ways: resolution into isolable causal trains or seeking for "atomic" units in the various fields of science, etc.*

He stated that while the progress of "classical" science has shown that these principles, first enunciated by Galileo and Descartes, are highly successful in a wide realm of phenomena, but two conditions are required for these principles to apply:

> *The first is that interactions between "parts" be non-existent or weak enough to be neglected for certain research purposes. Only under this condition, can the parts be "worked out," actually, logically, and mathematically, and then be "put together." The second condition is that the relations describing the behavior of parts be linear; only then is the condition of summativity given, i.e., an equation describing the behavior of the total is of the same form as the equations describing the behavior of the parts.*
>
> *These conditions are not fulfilled in the entities called systems, i.e. consisting of parts "in interaction" and description by nonlinear mathematics. These system entities describe many real world situations: populations, eco systems, organizations and complex man made technologies. The methodological problem of systems theory is to provide for problems beyond the analytical-summative ones of classical science.* (Bertalanffy 1968, 18-19)

Bertalanffy also cited a similar argument by mathematician and co-founder of information theory Warren Weaver in a 1948 American Scientist article on "Science and Complexity". Weaver had served as Chief of the Applied Mathematics Panel at the U.S. Office of Scientific Research and Development during WWII. Based on those experiences, he proposed an agenda for what he termed a new "science of problems of organized complexity".

Weaver explained how the mathematical methods which had led to great successes of science to date were limited to problems where appropriate simplifying assumptions could be made. What he termed "problems of simplicity" could be adequately addressed by the mathematics of mechanics, while "problems of disorganized complexity" could be successfully addressed by the mathematics of statistical mechanics. But with other problems, making the simplifying assumptions in order to use the methods would not lead to helpful solutions. Weaver placed in this category problems such as, how the genetic constitution of an organism expresses itself in the characteristics of the adult, and to what extent it is safe to rely on the free interplay of market forces if one wants to avoid wide swings from prosperity to depression. He noted that these were complex problems which involved "analyzing systems which are organic wholes, with their parts in close interrelation."

> *These problems-and a wide range of similar problems in the biological, medical, psychological, economic, and political sciences-are just too complicated to yield to the old nineteenth century techniques which were so dramatically successful on two-, three-, or four-variable problems of simplicity. These new problems, moreover, cannot be handled with the statistical techniques so effective in describing average behavior in problems of disorganized complexity [problems with elements exhibiting random or unpredictable behaviour].*

These new critical global problems require science to make a third great advance,

> *An advance that must be even greater than the nineteenth-century conquest of problems of simplicity or the twentieth-century victory over problems of disorganized complexity. Science must, over the next 50 years, learn to deal with these problems of organized complexity [problems for which complexity "emerges" from the coordinated interaction between its parts.* (Weaver 1948)

Weaver identified two grounds for optimism in taking on this great challenge: 1.) developments in mathematical modeling and digital simulation, and 2.) the success during WWII of the “mixed team” approach of operations analysis, where individuals from across disciplines brought their skills and insights together to solve critical, complex problems.

The importance of modeling and simulation and the importance of working across disciplinary boundaries have been the key recurring themes in development of this “third way” science for systems problems of organized complexity.

The Development of Systems Research

The following overview of the evolution of systems science is broadly chronological, but also follows the evolution of different paradigms in system theory.

Open Systems and General Systems Theory

General system theory (GST) attempts to formulate principles relevant to all open systems (Bertalanffy 1968). GST is based on the idea that correspondence relationships (homologies) exist between systems from different disciplines. Thus, knowledge about one system should allow us to reason about other systems. Many of the generic system concepts come from the investigation of GST.

In 1954, Bertalanffy co-founded, along with Kenneth Boulding (economist), Ralph Gerard (physiologist) and 'Anatol Rapoport (mathematician), the Society for General System Theory (renamed in 1956 the Society for General Systems Research, and in 1988 the ISSS.

The initial purpose of the society was "to *encourage the development of theoretical systems which are applicable to more than one of the traditional departments of knowledge ... and promote the unity of science through improving the communication among specialists*." (Bertalanffy 1968)

This group is considered by many to be the founders of **System Age Thinking** (Flood 1999).

Cybernetics

Cybernetics was defined by Wiener, Ashby and others as the study and modeling of communication, regulation, and control in systems (Ashby 1956; Wiener 1948). Cybernetics studies the flow of information through a system and how information is used by the system to control itself through feedback mechanisms. Early work in cybernetics in the 1940s was applied to electronic and mechanical networks, and was one of the disciplines used in the formation of early systems theory. It has since been used as a set of founding principles for all of the significant system disciplines.

Some of the key concepts of feedback and control from Cybernetics are expanded in the Concepts of Systems Thinking article.

Operations Research

Operations Research (OR) considers the use of technology by an organization. It is based on the use of mathematical modeling and statistical analysis to optimize decisions on the deployment of the resources under an organization's control. An interdisciplinary approach based on scientific methods, OR arose from military planning techniques developed during World War II.

Operations Research and Management Science (ORMS) was formalized in 1950 by Ackoff and Churchman applying the ideas and techniques of OR to organizations and organizational decisions (Churchman et. al. 1950).

Systems Analysis

Systems analysis was developed by RAND Corporation in 1948. It borrowed from and extended OR, including using black boxes and feedback loops from cybernetics to construct block diagrams and flow graphs. In 1961, the Kennedy Administration decreed that systems analysis techniques should be used throughout the government to provide a quantitative basis for broad decision-making problems, combining OR with cost analysis. (Ryan 2008)

Systems Dynamics

Systems dynamics (SD) uses some of the ideas of cybernetics to consider the behavior of systems as a whole in their environment. SD was developed by Jay Forrester in the 1960's (Forrester 1961). He was interested in modeling the dynamic behavior of systems such as populations in cities, industrial supply chains. See Systems Approaches for more details.

SD is also used by Senge (Senge 1990) in his influential book *The Fifth Discipline*. This book advocates a system thinking approach to organization and also makes extensive use of SD notions of feedback and control.

Organizational Cybernetics

Stafford Beer was one of the first to take a cybernetics approach to organizations (Beer 1959). For Beer the techniques of ORMS are best applied in the context of an understanding of the whole system. Beer also developed a **Viable Systems Model** (Beer 1979), which encapsulates the effective organization needed for a system to be viable (to survive and adapt in its environment).

Work in cybernetics and ORMS consider the mechanism for communication and control in complex systems, and particularly in organizations and management sciences. They provide useful approaches for dealing with operational and tactical problems within a system, but do not allow consideration of more strategic organizational problems (Flood 1999).

Hard and Soft Systems Thinking

Action research is an approach, first described by Kurt Lewin, as a reflective process of progressive problem solving in which reflection on action leads to a deeper understanding of what is going on and to further investigation (Lewin 1958).

Peter Checkland's action research program in the 1980's led to an Interpretative-based Systemic Theory which seeks to understand organizations by not only observing the actions of people, but also by building understandings of the cultural context, intentions and perceptions of the individuals involved. Checkland, himself starting from a systems engineering (SE) perspective, successively observed the problems in applying a SE approach to the more fuzzy, ill-defined problems found in the social and political arenas (Checkland 1978). Thus he introduced a distinction between hard systems and soft systems - see also Systems Approaches.

Hard systems (glossary) views of the world are characterized by the ability to define purpose, goals, and missions that can be addressed via engineering methodologies in an attempt to, in some sense, "optimize" a solution.

In hard system approaches the problems may be complex and difficult, but they are known and can be fully expressed by the investigator. Such problems can be solved by selecting from the best available solutions (possibly with some modification or integration to create an optimum solution). In this context, the term "systems" is used to describe real world things; a solution system is selected, created and then deployed to solve the problem.

Soft systems (glossary) views of the world are characterized by complex, problematical, and often mysterious phenomena for which concrete goals cannot be established and which require learning in order to make improvement. Such systems are not limited to the social and political arenas and also exist within and amongst enterprises where complex, often ill-defined patterns of behavior are observed that are limiting the enterprise's ability to improve.

Soft system approaches reject the idea of a single problem and consider **problematic** situations in which different people will perceive different issues depending upon their own viewpoint and experience. These problematic situations are not solved, but managed through interventions which seek to reduce "discomfort" among the participants. The term system is used to describe systems of ideas, conceptual systems which guide our understanding of the situation, or help in the selection of intervention strategies.

These three ideas of “problem vs. problematic situation”,“solution vs. discomfort reduction”, and “the system vs. systems understanding” encapsulate the differences between hard and soft approaches (Flood and Carson 1993).

Critical Systems Thinking

The development of a range of hard and soft methods naturally leads to the question of which method to apply in what circumstances (Jackson 1989). Critical systems thinking (CST), or **critical management science** (Jackson 1985), attempts to deal with this question.

The word **critical** is used in two ways. Firstly, critical thinking considers the limits of knowledge and investigates the limits and assumptions of hard and soft systems, as discussed in the above sections. The second aspect of critical thinking considers the ethical, political and coercive dimension and the role of system thinking in society, see also Systems Approaches.

Service Science and Service Systems Engineering

The world economies have transitioned over the past few decades from manufacturing economies that provide goods - to service based economies. Harry Katzan defined the newly emerging field of service science: "Service science is defined as the application of scientific, engineering, and management competencies that a service-provider organization performs that creates value for the benefit of the client of customer" (Katzan 2008, vii).

The disciplines of service science and service engineering have developed to support this expansion and are built on principles of systems thinking but applied to the development and delivery of service systems.

Service Systems Engineering is described more fully in the Service Systems Engineering KA in Part 4 of the SEBoK.

References

Works Cited

Ackoff, R.L. 1971. "Towards a System of Systems Concepts". *Management Science.* 17(11).

Ashby, W. R. 1956. *Introduction to Cybernetics.* London, UK: Methuen.

Beer, S. 1959. *Cybernetics and Management.* London, UK: English Universities; New York: Wiley and Sons.

Beer, S. 1979. *The Heart of the Enterprise*. Chichester, UK: Wiley.

Bertalanffy, L. von. 1950. "The Theory of Open Systems in Physics and Biology". *Science*, New Series, 111(2872) (Jan 13): 23-29

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY, USA: Braziller.

Checkland, P. 1978. "The Origins and Nature of “Hard” Systems Thinking." *Journal of Applied Systems Analysis*, 5(2): 99-110.

Checkland, P. 1999. *Systems Thinking, Systems Practice*, New York, NY, USA: John Wiley & Sons.

Churchman, C.W. 1968. *The Systems Approach.* New York, NY, USA: Dell Publishing.

Churchman, C.W., R.L. Ackoff. and E.L. Arnoff. 1950. *Introduction to Operations Research.* New York, NY, USA: Wiley and Sons.

Flood, R.L. 1999. *Rethinking the Fifth Discipline: Learning within the Unknowable.* London, UK: Routledge.

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.

Forrester, J. 1961. *Industrial Dynamics*. Cambridge, MA, USA: MIT Press.

Jackson, M. 1985. "Social Systems Theory and Practice: the Need for a Critical Approach." *International Journal of General Systems.* 10: 135-151.

Jackson, M. 1989. "Which Systems Methodology When? Initial Results from a Research Program." In: R Flood, M Jackson and P Keys (eds). *Systems Prospects: the Next Ten Years of Systems Research.* New York, NY, USA: Plenum.

Jackson, M. 2003. *Systems Thinking: Creating Holisms for Managers*. Chichester, UK: Wiley.

Katzan, H. 2008. *Service Science.* Bloomington, IN, USA: iUniverse Books.

Lewin, K. 1958. *Group Decision and Social Change.* New York, NY, USA: Holt, Rinehart and Winston. p. 201.

Magee, C. L., O.L. de Weck. 2004. "Complex System Classification." Proceedings of the 14th Annual International Council on Systems Engineering International Symposium, 20-24 June 2004, Toulouse, France.

M'Pherson, P, K. 1974. "A Perspective on Systems Science and Systems Philosophy." *Futures* 6(3) (June 1974): 219-239.

Miller, J.G. 1986. "Can Systems Theory Generate Testable Hypothesis?: From Talcott Parsons to Living Systems Theory." *Systems Research.* 3: 73-84.

Ryan, A. 2008. “What is a Systems Approach?” *Journal of Nonlinear Science.*

Senge, P.M. 1990. *The fifth discipline: The Art & Practice of the Learning Organization.* New York, NY, USA: Doubleday Business.

Weaver, W. (1948). “Science and complexity.” *American Scientist*. 36: 536-544.

Wiener, N. 1948. *Cybernetics or Control and Communication in the Animal and the Machine.* Paris, France: Hermann & Cie Editeurs; Cambridge, MA, USA: The Technology Press; New York, NY, USA: John Wiley & Sons Inc.

Primary References

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY, USA: Braziller.

Chang, C.M., 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence.* Hoboken, NJ, USA: John Wiley and Sons.

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: John Wiley & Sons.

Flood, R. L. 1999. *Rethinking the Fifth Discipline: Learning within the Unknowable.* London, UK: Routledge.

Jackson, M. 1985. "Social Systems Theory and Practice: the Need for a Critical Approach." *International Journal of General Systems.* 10: 135-151.

Additional References

Ackoff, R.L. 1981. *Creating the Corporate Future.* New York, NY, USA: Wiley and Sons.

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.

Bowler, D.T. 1981. *General Systems Thinking: Its Scope and Applicability.* Amsterdam: The Netherlands: Elsevier.

Boulding, K.E. 1996. *The World as a Total System.* Beverly Hills, CA, USA: Sage Publications.

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

Laszlo, E. (ed). 1972. *The Relevance of General Systems Theory.* New York, NY, USA: George Brazillier.

Skyttner, L. 1996. *General Systems Theory - An Introduction.* Basingstoke, UK: Macmillan Press.

Warfield, J.N. 2006. *An Introduction to Systems Science.* Singapore: World Scientific Publishing Co. Pte Ltd. Chang, C.M., 2010. *Service Systems Management and Engineering: Creating Strategic Differentiation and Operational Excellence.* Hoboken, NJ, USA: John Wiley and Sons.

Lusch, R.F. and S. L. Vargo (Eds). 2006. *The service-dominant logic of marketing: Dialog, debate, and directions.* Armonk, NY: ME Sharpe Inc.

Maglio P., S. Srinivasan, J.T. Kreulen, and J. Spohrer. 2006. “Service Systems, Service Scientists, SSME, and Innovation." *Communications of the ACM*. 49(7) (July).

Popper, K. R. 1979. *Objective Knowledge*, 2nd edition. Oxford, UK: Oxford University Press.

Salvendy, G. and W. Karwowski (eds.). 2010. *Introduction to Service Engineering*. Hoboken, NJ, USA: John Wiley and Sons.

Sampson, S.E. 2001. *Understanding Service Businesses*. New York, NY, USA: John Wiley.

Spohrer, J. and P. P. Maglio. 2008. "The emergence of service science: Toward systematic service innovations to accelerate co-creation of value." *Production and Operations Management* 17 (3): 238-246, cited by Spohrer, J. and P. Maglio. 2010. "Service Science: Toward a Smarter Planet." In *Introduction to Service Engineering*. Ed. G Salvendy and W Karwowski. 3-30. Hoboken, NJ, USA: John Wiley & Sons, Inc.

Tien, J.M. and D. Berg. 2003. "A Case for Service Systems Engineering." *Journal of Systems Science and Systems Engineering.* 12(1): 13-38.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Systems Approaches

Lead Author: *Rick Adcock*, **Contributing Authors:** *Scott Jackson, Janet Singer, Duane Hybertson*

This article is part of the Systems Science knowledge area (KA). It presents issues in the comparison and analysis of systems approaches by the systems science community. Some of these ideas contribute to basic theory and methods that are used in systems thinking discussed in the Systems Thinking KA.

What is a Systems Approach?

In Bertalanffy's introduction to his 1968 book *General System Theory* (GST), he characterizes a systems approach as:

> *A certain objective is given; to find ways and means for its realization requires the system specialist (or team of specialists) to consider alternative solutions and to choose those promising optimization at maximum efficiency and minimum cost in a tremendously complex network of interactions.* (Bertalanffy 1968, 4)

He goes on to list as possible elements of a systems approach: “classical” systems theory (differential equations), computerization and simulation, compartment theory, set theory, graph theory, net theory, cybernetics, information theory, theory of automata, game theory, decision theory, queuing theory, and models in ordinary language.

This description is similar to what Warren Weaver identified as the methods used successfully by “mixed teams” during World War II (WWII) on “problems of organized complexity”. However, some conditions that had contributed to success during wartime did not hold after the war, such as a clear focus on well-defined common goals that motivated participants to work across disciplinary boundaries.

By the early 1970’s, there was growing disillusionment with the promise that a systems approach would provide easy solutions for all complex problems. There was particular criticism from some, including pioneers of Operations Research and Management Science (ORMS) like Ackoff and Churchman, that reliance on rote mathematical methods to identify optimal solutions among fixed alternatives had become just as inflexible and unimaginative an approach to complex problems as whatever it had replaced. Interest grew in examining and comparing methods and methodologies to better understand what could help ensure the best thinking and learning in terms of systems in systems approaches to practice.

Issues in Systems Approaches

A systems approach is strongly associated with systems thinking and how it helps to guides systems practice. In What is Systems Thinking? the key ideas of considering a system holistically, setting a boundary for a problem/solution of interest, and considering the resulting system-of-interest from outside its boundary are identified (Churchman 1979; Senge 2006).

A systems approach can view a system as a “holon” – an entity that is itself a “whole system” that interacts with a mosaic of other holons in its wider environment (Hybertson 2009), while also being made up of interacting parts. We can use this model recursively – each part of the system may be a system in its own right, and can itself be viewed both as an entity as seen from outside, and as a set of interacting parts. This model also applies in upwards recursion, so the original “system-of-interest” is an interacting part of one or more wider systems.

This means that an important skill in a systems approach is to identify the “natural holons” in the problem situation and solution systems and to make the partitioning of responsibilities match the “natural holons”, so as to minimize the coupling between parallel activities when applying a solution. This is the “cohesive/loose coupling” heuristic that has been around for a long time in many design disciplines.

Another consequence of the holistic nature of a systems approach is that it considers not only a problem situation and a solution system but also the system created and deployed to apply one to the other. A systems approach must consider both the boundary of the system of concern as well as the boundary of the system inquiry (or model). Real systems are always open, i.e., they interact with their environment or supersystem(s). On the other hand, real models are always "closed" due to resource constraints — a fixed boundary of consideration must be set. So there is an ongoing negotiation to relate the two in systems practice and the judgment to do so is greatly helped by an appreciation of the difference between them.

Thus, a systems approach can be characterized by how it considers problems, solutions and the problem resolution process itself:

- Consider problems holistically, setting problem boundaries though understanding of natural system relationships and trying to avoid unwanted consequences.
- Create solutions based on sound system principles, in particular creating system structures which reduce organized complexity and unwanted emergent properties.
- Use understanding, judgment and models in both problem understanding and solution creation, while understanding the limitations of such views and models.

Systems Methodologies

One topic that has received significant attention in the systems science community is the analysis and comparison of methodologies which implement a systems approach. A methodology is a body of tools, procedures, and methods applied to a problem situation, ideally derived from a theoretical framework. These describe structured approaches to problem understanding and/or resolution making use of some of the concepts of systems thinking. These methodologies are generally associated with a particular system paradigm or way of thinking, which has a strong influence on the three aspects of a systems approach described above.

The most widely used groups of methodologies are as follows, see also History of Systems Science:

- hard system methodologies (Checkland 1978) set out to select an efficient means to achieve a predefined and agreed end.
- soft system methodologies (Checkland 1999) are interactive and participatory approaches to assist groups of diverse participants to alleviate a complex, problematic situation of common interest.
- critical systems thinking methodologies (Jackson 1985) attempts to provide a framework in which appropriate hard and soft methods can be applied as appropriate to the situation under investigation.

Systems Dynamics

Systems dynamics (SD) uses some of the ideas of cybernetics to consider the behavior of systems as a whole in their environment. SD was developed by Jay Forrester in the 1960's. He was interested in modeling the dynamic behavior of systems such as populations in cities, or industrial supply chains.

System dynamics, (Forrester 1961), is an approach to understanding the behavior of complex systems over time. It deals with internal feedback loops and time delays that affect the behavior of the entire system. The main elements of SD are:

- The understanding of the dynamic interactions in a problem or solution as a system of feedback loops, modeled using a Causal Loop Diagram.
- Quantitative modeling of system performance as an accumulation of stocks (any entity or property which varies over time) and flows (representations of the rate of change of a stock).
- The creation of dynamic simulations, exploring how the value of key parameters change over time. A wide range of software tools are available to support this.

These elements help describe how even seemingly simple systems display baffling non-linearity.

Hard Systems Methodologies

Checkland (Checkland 1975) classifies hard system (glossary) methodologies, which set out to select an efficient means to achieve a predefined end, under the following headings:

- system analysis - the systematic appraisal of the costs and other implications of meeting a defined requirement in various ways.
- systems engineering (SE) - the set of activities that together lead to the creation of a complex man-made entity and/or the procedures and information flows associated with its operation.

Operational Research is also considered a hard system approach, closely related to the systems analysis approach developed by the Rand Corporation, in which solutions are known but the best combinations of these solutions must be found. There is some debate as to whether system dynamics is a hard approach, which is used to assess the objective behavior of real situations. Many application of SD have focused on the system, however it can and has also be used as part of a soft approach including the modeling of subjective perceptions (Lane 2000).

SE allows for the creation of new solution systems, based upon available technologies. This hard view of SE as a solution focused approach applied to large, complex and technology focused solutions, is exemplified by (Jenkins 1969; Hall 1962) and early defense and aerospace standards.

It should be noted that historically the SE discipline was primarily aimed at developing, modifying or supporting hard systems. More recent developments in SE have incorporated problem focused thinking and agile solution approaches. It is this view of SE that is described in the SEBoK.

All of these hard approaches can use systems thinking to ensure complete and viable solutions are created and/or as part of the solution optimization process. These approaches are appropriate to unitary problems, but not when the problem situation or solution technologies are unclear.

Soft Systems and Problem Structured Methods

Problem Structuring Methods (PSM) are interactive and participatory approaches to assist groups of diverse participants to alleviate a complex, problematic situation of common interest. Typically the hardest element of the situation is framing the issues which constitute the problem (Minger and Resenhead 2004).

PSM use systems and systems thinking as an abstract framework for investigation, rather than a structure for creating solutions. Systems descriptions are used to understand the current situation and describe an idealized future. Interventions directly in the current organization to move towards the idea recognize that the assumptions and mental models of the participants are an important obstruction to change and that these differing views cannot be dismissed, but instead must form part of the intervention approach.

Peter Checkland's action research program, see Systems Science, in the 1980's forms the basis of work by Checkland, Wilson and others in the development of soft systems methodology (SSM) (Checkland 1999; Wilson 2001). SSM formalizes the idea of a soft approach using systemic thinking to expose the issues in a problem situation and guide interventions to reduce them. SSM provides a framework of ideas and models to help guide participants through this systemic thinking.

Other PSM approaches include interactive planning approach (Ackoff 1981), social systems design (Churchman 1968), and strategic assumptions surfacing and testing (Mason and Mitroff 1981).

SSM and other soft approaches use systems thinking to ensure problem situations are fully explored and resolved. These approaches are appropriate to pluralist (glossary) problems. Critics of SSM suggest that it does not consider the process of intervention, and in particular how differences in power between individuals and social groups impact the effectiveness of interventions.

Critical Systems Thinking and Multi-Methodology

The development of a range of hard and soft methods naturally leads to the question of which method to apply in what set of circumstances (Jackson 1989). Critical systems thinking (CST) or **Critical Management Science** Jackson (Jackson 1985) attempts to deal with this question.

The word critical is used in two ways. Firstly, critical thinking considers the limits of knowledge and investigates the limits and assumptions of hard and soft systems, as discussed in the above sections. From this comes frameworks and meta-methodology that establish when to apply different methods such as **total systems intervention** (TSI) (Flood and Jackson 1991). Critical or "pluralist" or "pragmatic" , **multi-methodology** approaches take this aspect of critical thinking one stage further to recognize the value of combining techniques from several hard, soft , or custom methods as needed (Mingers and Gill 1997). Many in the systems science community believe that the multi-methodology approach has been accepted as the de facto systems approach and that the challenges now are in refining tools and methods to support it.

Churchman (Churchman, 1979) and others have also considered broader ethics political and social questions related to management science, with regards to the relative power and responsibility of the participants in system interventions. The second aspect of critical thinking considers the ethical, political, and coercive dimension in Jackson's System of Systems Methodologies (SOSM) framework (Jackson 2003) and the role of system thinking in society.

Selecting Systems Methodologies

Jackson proposes a frame for considering which approach should be applied, please see Jackson's Framework [1]. In Jackson's framework the following definitions apply to the participants involved in solving the problem:

- unitary - A problem situation in which participants "have similar values, beliefs and interests. They share common purposes and are all involved, in one way or another, in decision-making about how to realize their agreed objectives." (Jackson 2003, 19)
- pluralist - A problem situation involving participants in which "although their basic interests are compatible, they do not share the same values and beliefs. Space needs to be made available within which debate, disagreement, even conflict, can take place. If this is done, and all feel they have been involved in decision-making, then accommodations and compromises can be found. Participants will come to agree, at least temporarily, on productive ways forward and will act accordingly." (Jackson 2003, 19)
- coercive - A problem situation in which the participants "have few interests in common and, if free to express them, would hold conflicting values and beliefs. Compromise is not possible and so no agreed objectives direct action. Decisions are taken on the basis of who has most power and various forms of coercion employed to ensure adherence to commands." (Jackson 2003, 19)

Jackson's framework suggests that for simple and complex systems with unitary participants, hard and dynamic systems thinking applies, respectively. For simple and complex systems with pluralist participants, soft systems thinking applies. For simple and complex systems with coercive participants, **emancipatory** and postmodernist system thinking applies, respectively. These thinking approaches consider all attempts to look for system solutions to be temporary and ineffective in situations where the power of individuals and groups of people dominate any system structures we create. They advocate an approach which encourages diversity, free-thinking and creativity of individuals and in the organization's structures. Thus, modern system thinking has the breadth needed to deal with a broad range of complex problems and solutions.

These ideas sit at the extreme of system thinking as a tool for challenging assumptions in and stimulating innovative solutions in problem solving. Jackson (Jackson 2003) identifies the work of some authors who have included these ideas into their systems approach.

References

Works Cited

Bertalanffy, L. 1968. *General System Theory: Foundations, Development, Applications*. New York, NY, USA: George Braziller, Inc.

Ackoff, R.L. 1981. *Creating the Corporate Future*. New York, NY, USA: Wiley and Sons.

Checkland, P. 1978. "The Origins and Nature of “Hard” Systems Thinking." *Journal of Applied Systems Analysis.* 5(2): 99-110.

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: John Wiley & Sons.

Churchman, C.W. 1968. *The Systems Approach*. New York, NY, USA: Dell Publishing.

Churchman, C. West. 1979. *The Systems Approach and Its Enemies*. New York: Basic Books.

Flood, R. and M. Jackson. 1991. *Creative Problem Solving: Total Systems Intervention*. London, UK: Wiley.

Forrester, J. 1961. *Industrial Dynamics*. Cambridge, MA, USA: MIT Press.

Hall, A.D. 1962. *A Methodology for Systems Engineering*. New York, NY, USA: Van Nostrand Reinhold.

Hybertson, D, 2009. *Model-oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems.* Series in Complex and Enterprise Systems Engineering. Boston, MA, USA: Auerbach Publications.

Jackson, M. 1985. "Social Systems Theory and Practice: the Need for a Critical Approach." *International Journal of General Systems.* 10: 135-151.

Jackson, M. 1989. "Which Systems Methodology When? Initial Results from a Research Program." In: R Flood, M Jackson and P Keys (eds). *Systems Prospects: the Next Ten Years of Systems Research*. New York, NY, USA: Plenum.

Jackson, M. 2003. *Systems Thinking: Creating Holisms for Managers*. Chichester, UK: Wiley.

Jenkins, G.M. 1969. "The Systems Approach." In Beishon, J. and G. Peters (eds.), *Systems Behavior*, 2nd ed. New York, NY, USA: Harper and Row.

Lane, D. 2000. “Should System Dynamics be Described as a `Hard' or `Deterministic' Systems Approach?” *Systems Research and Behavioral Science.* 17: 3–22 (2000).

Mason, R.O. and I.I. Mitroff. 1981. *Challenging Strategic Planning Assumptions: Theory, Case and Techniques*. New York, NY, USA: Wiley and Sons.

Mingers, J. and A. Gill. 1997. *Multimethodology: Theory and Practice of Combining Management Science Methodologies*. Chichester, UK: Wiley.

Senge, P.M. 1990, 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization*. New York, Doubleday/Currency.

Wilson, B. 2001. *Soft Systems Methodology—Conceptual Model Building and Its Contribution*. New York, NY, USA: J.H.Wiley.

Primary References

Checkland, P. 1999. *Systems Thinking, Systems Practice.* New York, NY, USA: John Wiley & Sons.

Forrester, J. 1961. *Industrial Dynamics*. Cambridge, MA, USA: MIT Press.

Jackson, M. 1985. "Social Systems Theory and Practice: the Need for a Critical Approach." *International Journal of General Systems.* 10: 135-151.

Additional References

Jackson, M.C. and P. Keys. 1984. "Towards a System of Systems Methodologies." *The Journal of the Operational Research Society.* 35(6) (Jun. 1984): 473-486.

Mingers, J. and J. Rosenhead. 2004. "Problem Structuring Methods in Action." *European Journal of Operations Research.* 152(3) (Feb. 2004): 530-554. Sterman, J.D. 2001. "System dynamics modeling: Tools for learning in a complex world." *California Management Review.* 43(4): 8–25.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

References

[1] http://www.systemswiki.org/index.php?title=System_of_Systems_Methodologies_(SOSM)

Knowledge Area: Systems Thinking

Systems Thinking

Lead Author: *Rick Adcock*

This knowledge area (KA) provides a guide to knowledge about systems thinking which is the integrating paradigm for systems science and systems approaches to practice.

This is part of the wider systems knowledge which can help to provide a common language and intellectual foundation; and make practical systems concepts, principles, patterns and tools accessible to systems engineering (SE), as discussed in the Introduction to Part 2.

Topics

Each part of the Guide to the SE Body of Knowledge (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:

- What is Systems Thinking?
- Concepts of Systems Thinking
- Principles of Systems Thinking
- Patterns of Systems Thinking

Introduction

Systems thinking is concerned with understanding or intervening in problem situations, based on the principles and concepts of the systems paradigm. This KA offers some basic definitions of systems thinking. The following diagram summarizes how the knowledge is presented.

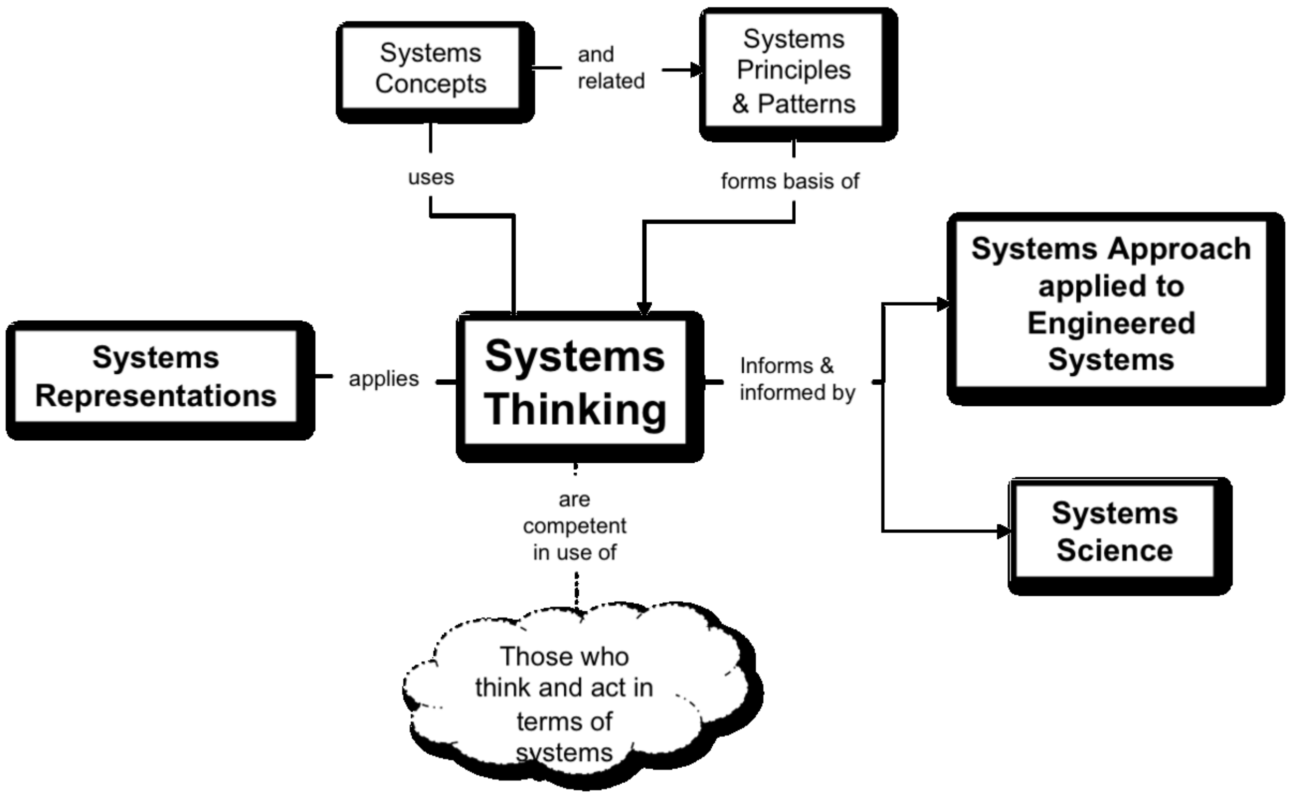


Figure 1. Systems Thinking in the SEBoK. (SEBoK Original)

Systems thinking considers the similarities between systems from different domains in terms of a set of common systems concepts, principles and patterns:

- A principle is a rule of conduct or behavior. To take this further, a principle is a “basic generalization that is accepted as true and that can be used as a basis for reasoning or conduct” (WordWeb.com).
- A concept is an abstraction, or a general idea inferred or derived from specific instances.

Principles depend on concepts in order to state a “truth.” Hence, principles and concepts go hand in hand; principles cannot exist without concepts and concepts are not very useful without principles to help guide the proper way to act (Lawson and Martin 2008).

Many sources combine both concepts and the principles based on them. The Concepts of Systems Thinking article presents concepts extracted from a variety of theory and practice sources. The Principles of Systems Thinking article, in turn, presents a summary of important principles referring back to the concepts upon which they are based is also provided.

A pattern is an expression of observable similarities found in systems from different domains. Patterns exist in both natural and man-made systems and are used in systems science and systems engineering. A summary of the different classes of patterns and the use of patterns to support a systems approach is discussed in the final Patterns of Systems Thinking article.

The practical application of systems thinking often employs the use of abstract system representations or models. Some mention of models is made in this KA; additionally, a more complete guide provided in Representing Systems with Models.

References

Works Cited

Lawson, H., and J.N. Martin. 2008. "On the Use of Concepts and Principles for Improving Systems Engineering Practice". Proceedings of the 18th Annual International Council on Systems Engineering (INCOSE) International Symposium, 5-19 June 2008, Utrecht, The Netherlands.

WordWeb Online. n.d. "Definition:Principle." Accessed Dec 3 2014 At WordWeb Online http:/ / www. wordwebonline.com/en/PRINCIPLE

Primary References

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY: Braziller.

Checkland, P. 1999. *Systems Thinking, Systems Practice*, New York, NY, USA: John Wiley & Sons.

Churchman, C. W. 1968. *The Systems Approach and its Enemies*. New York, NY, USA: Dell Publishing.

Flood, R. L. 1999. *Rethinking the Fifth Discipline: Learning Within The Unknowable.* London UK: Routledge.

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-2003-002-03.2.1.

Additional References

Ackoff, R.L. 1971. "Towards a System of Systems Concepts". *Management Science.* 17(11).

Hitchins, D. 2009. "What Are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4).

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

Ramage, Magnus, and Karen Shipp. 2009. *Systems Thinkers*. Dordrecht: Springer.

Weinberg, Gerald M. 1975. *An Introduction to General Systems Thinking*. New York: Wiley.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

What is Systems Thinking?

Lead Author: *Rick Adcock*, **Contributing Authors:** *Brian Wells, Bud Lawson*

This topic is part of the Systems Thinking knowledge area (KA). The scope of systems thinking is a starting point for dealing with real world situations using a set of related systems concept discussed in Concepts of Systems Thinking topic, systems principles discussed in Principles of Systems Thinking topic, and system patterns discussed in Patterns of Systems Thinking topic.

Introduction

The concepts, principles, and patterns of systems thinking have arisen both from the work of systems scientists and from the work of practitioners applying the insights of systems science to real-world problems.

Holism has been a dominant theme in systems thinking for nearly a century, in recognition of the need to consider a system as a whole because of observed phenomena such as emergence. Proponents have included Wertheimer, Smuts, Bertalanffy, Weiss, (Ackoff 1979), (Klir 2001), and (Koestler 1967) among many others.

A more detailed discussion of the most important movements in systems theory can be found in History of Systems Science.

Identifying Systems of Interest

When humans observe or interact with a system, they allocate boundaries and names to parts of the system. This naming may follow the natural hierarchy of the system, but will also reflect the needs and experience of the observer to associate elements with common attributes of purposes relevant to their own. Thus, a number of systems of interest (SoIs) (Flood and Carson 1993) must be identified and they must be both relevant and include a set of elements which represent a system whole. This way of observing systems wherein the complex system relationships are focused around a particular system boundary is called **systemic resolution**.

Systems thinking requires an ongoing process of attention and adaptation to ensure that one has appropriately identified boundaries, dependencies, and relationships. Churchman (Churchman 1968) and others have also considered broader ethical, political, and social questions related to management science with regards to the relative power and responsibility of the participants in system interventions. These are seen by critical systems thinkers as key factors to be considered in defining problem system boundaries.

A system context can be used to define a SoI and to capture and agree on the important relationships between it, such as the systems it works directly with and the systems which influence it in some way. When this approach is used to focus on part of a larger system, a balance of reductionism and holism is applied. This balance sits at the heart of a systems approach. A systems context provides the tool for applying this balance, and is thus an essential part of any systems approach and hence, of systems engineering (SE) as well. Approaches for describing the context of the different types of engineered systems are discussed in Engineered System Context topic within the Systems Approach Applied to Engineered Systems KA.

Thoughts on Systems Thinking

Senge (1990) discusses systems thinking in a number of ways as

> *a discipline for seeing wholes ... a framework for seeing interrelationships rather than things ... a process of discovery and diagnosis ... and as a sensibility for the subtle interconnectedness that gives living systems their unique character.*(Senge 2006, 68-69)

Churchman came to define a systems approach as requiring consideration of a system from the viewpoint of those outside its boundary (Churchman 1979). There are many demonstrations that choosing too narrow a boundary, either in terms of scope or timeline, results in the problem of the moment being solved only at the expense of a similar or bigger problem being created somewhere else in space, community, or time (Senge 2006) and (Meadows 1977). This is the “shifting the burden” archetype described in Patterns of Systems Thinking topic.

Churchman believes that an important component of system knowledge comes from "others" or "enemies" outside the system; the systems approach begins when first you see the world through the eyes of another (Churchman 1968). In this famous phrase, Churchman suggests that people can step outside a system they are in and mentally try to consider it through the lenses of other people's values. Churchman (1979) identified four main enemies of the systems approach namely: politics, morality, religion and aesthetics.

To Churchman, the "enemies" of the systems approach provide a powerful way of learning about the systems approach, precisely because they enable the rational thinker to step outside the boundary of a system and to look at it. It means that systems thinkers are not necessarily just involved within a system but are essentially involved in reasoning and decisions "outside" of systems rationality.

Some additional perspectives on systems thinking definitions are as follows:

- “Systems thinking requires the consciousness of the fact that we deal with models of our reality and not with the reality itself.” (Ossimitz 1997, 1)
- “…what is often called “systemic thinking” …is …a bundle of capabilities, and at the heart of it is the ability to apply our normal thought processes, our common sense, to the circumstances of a given situation.” (Dörner 1996, 199)
- “Systems thinking provides a powerful way of taking account of causal connections that are distant in time and space.” (Stacey 2000, 9)

Chaos and complexity theories have also impacted the development of systems thinking, including the treatment of such concepts as emergence. According to Gharajedaghi:

> *Systems thinking is the art of simplifying complexity. It is about seeing through chaos, managing interdependency, and understanding choice. We see the world as increasingly more complex and chaotic because we use inadequate concepts to explain it. When we understand something, we no longer see it as chaotic or complex.* (Gharajedaghi 1999, 283)

Kasser considers systems thinking to be one element in a wider system of holistic thinking. Kasser defines holistic thinking as follows: "...the combination of analysis [in the form of elaboration], systems thinking and critical thinking." (Kasser 2010)

Systems Thinking and the Guide to the Systems Engineering Body of Knowledge

From these discussions one can see systems thinking as both a set of founding ideas for the development of systems theories and practices and also as a pervasive way of thinking need by those developing and applying them.

The SEBoK is particularly focused on how systems thinking can support a systems approach to engineered systems.

In order to examine a SoI in more detail, to understand, use, or change them in some way, practitioners are faced with an apparent “systems thinking paradox”. One can only truly understand a system by considering all of its possible relationships and interactions, inside and outside of its boundary and in all possible future situations (of both system creation and life), but this makes it apparently impossible for people to understand a system or to predict all of the consequences of changes to it.

If this means that all possible system relationships and environmental conditions must be considered to fully understand the consequences of creating or changing a system, what useful work can be done?

In many ways this is the essence of all human endeavors, whether they are technical, managerial, social or political, the so called known knowns and unknown unknowns. The systems approach is a way of tackling real world problems and making use of the concepts, principle and patterns of systems thinking to enable systems to be engineered and used.

The systems principles of encapsulation and separation of concerns in Principles of Systems Thinking relate to this issue. Some of the detail of complex situations must be hidden to allow focus on changes to a system element. The impact must be considered of any changes that might be made across sufficient related system components to fit within the acceptable commercial and social risks that must be considered. Engineering and management disciplines deal with this by gathering as much knowledge as necessary to proceed at a risk level acceptable to the required need. The assessment of what is enough and how much risk to take can, to some extent, be codified with rules and regulations, and managed through processes and procedures; however, it is ultimately a combination of the skill and judgment of the individuals performing the work.

References

Works Cited

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY: Braziller.

Churchman, C.W. 1968. *The Systems Approach*. Delacorte Press.

Churchman, C.W. 1979. "The Systems Approach and Its Enemies". New York: Basic Books.

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

Dorner, H., and A. Karpati. 2008. "Mentored innovation in teacher training using two virtual collaborative learning environments." In *Beyond knowledge: The legacy of competence--meaningful computer-based learning environments*., eds. J. Zumbach, N. Schwartz, T. Seufert and L. Kester. Vol. VIII. New York, NY: Springer.

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.

Gharajedaghi, J. 1999. *Systems Thinking: Managing Chaos and Complexity: A platform for designing.* 1st ed. Woburn, MA: Butterworth-Heinemann.

Jackson, M. 1989. "Which Systems Methodology When? Initial Results from a Research Program." In: R Flood, M Jackson and P Keys (eds). *Systems Prospects: the Next Ten Years of Systems Research.* New York, NY, USA: Plenum.

Kasser, J. 2010. "Holistic thinking and how it can produce innovative solutions to difficult problems." Paper presented at 7th Bi-annual European Systems Engineering Conference (EuSEC), 24-27 May 2010, Stockholm, Sweden.

Meadows, Donella H. et al. 1977. "Limits to Growth: A Report for the Club of Rome's Project on the Predicament of Mankind." New American Library, paperback, ISBN 0-451-13695-0; Universe Books, hardcover, 1972, ISBN 0-87663-222-3 (scarce).

Ossimitz, G. "The development of systems thinking skills using system dynamics modeling tools." in Universitat Klagenfurt [database online]. Klagenfurt, Austria, 1997 [cited November 12 2007]. Available from http://wwwu.uni-klu.ac.at/gossimit/sdyn/gdm_eng.htm.

Senge, P.M. 1990, 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization.* New York, NY, USA: Doubleday Currency.

Stacey, R.D., D. Griffin, and P. Shaw. 2000. *Complexity and management: Fad or radical challenge to systems thinking?* London, U.K.: Routledge.

Primary References

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY: Braziller.

Churchman, C.W.. 1979. "The Systems Approach and its Enemies". New York: Basic Books.

Gharajedaghi, J. 1999. *Systems Thinking: Managing Chaos and Complexity: A platform for designing.* 1st ed. Woburn, MA: Butterworth-Heinemann.

Senge, P.M. 1990, 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization.* New York, NY, USA: Doubleday Currency.

Additional References

Jackson, M. 2003. *Systems Thinking: Creating Holisms for Managers.* Wiley; Chichester

Edson, R. 2008. *Systems Thinking. Applied. A Primer*. In: ASYST Institute (ed.). Arlington, VA: Analytic Services.

Klir, G. 2001. *Facets of Systems Science, 2nd ed*. New York: Kluwer Academic/Plenum Publishers.

Koestler, A. 1967. *The Ghost in the Machine. New York*, NY: Macmillan. Lawson, H. 2010. *A Journey Through the Systems Landscape.* London, Kings College, UK.

MITRE. 2012. “Systems Engineering Guide.” Accessed September 11, 2012. Available at http://www.mitre.org/work/systems_engineering/guide.

Rebovich, G., Jr. 2005. "Systems Thinking for the Enterprise (New and Emerging Perspectives)," In *Volume 2 of Enterprise Systems Engineering Theory and Practice*. The MITRE Corporation.

Senge, P.M., A. Klieiner, C. Roberts, R.B. Ross, and B.J. Smith. 1994. *The Fifth Discipline Fieldbook: Strategies and tools for building a learning organization*. New York, NY: Crown Business.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Concepts of Systems Thinking

Lead Author: *Rick Adcock*, **Contributing Authors:** *Scott Jackson, Janet Singer, Duane Hybertson*

This article forms part of the Systems Thinking knowledge area (KA). It describes systems concepts, knowledge that can be used to understand problems and solutions to support systems thinking.

The concepts below have been synthesized from a number of sources, which are themselves summaries of concepts from other authors. Ackoff (1971) proposed a system of system concepts as part of general system theory (GST); Skyttner (2001) describes the main GST concepts from a number of systems science authors; Flood and Carlson (1993) give a description of concepts as an overview of systems thinking; Hitchins (2007) relates the concepts to systems engineering practice; and Lawson (2010) describes a system of system concepts where systems are categorized according to fundamental concepts, types, topologies, focus, complexity, and roles.

Wholeness and Interaction

A system is defined by a set of elements which exhibit sufficient cohesion, or "togetherness", to form a bounded whole (Hitchins 2007; Boardman and Sauser 2008).

According to Hitchins, interaction between elements is the "key" system concept (Hitchins 2009, 60). The focus on interactions and holism is a push-back against the perceived reductionist focus on parts and provides recognition that in complex systems, the interactions among parts is at least as important as the parts themselves.

An open system is defined by the interactions between system elements within a system boundary and by the interaction between system elements and other systems within an environment (see What is a System?). The remaining concepts below apply to open systems.

Regularity

Regularity is a uniformity or similarity that exists in multiple entities or at multiple times (Bertalanffy 1968). Regularities make science possible and engineering efficient and effective. Without regularities, we would be forced to consider every natural and artificial system problem and solution as unique. We would have no scientific laws, no categories or taxonomies, and each engineering effort would start from a clean slate.

Similarities and differences exist in any set or population. Every system problem or solution can be regarded as unique, but no problem/solution is in fact entirely unique. The nomothetic approach assumes regularities among entities and investigates what the regularities are. The idiographic approach assumes each entity is unique and investigates the unique qualities of entities, (Bertalanffy 1975).

A very large amount of regularity exists in both natural systems and engineered systems. Patterns of systems thinking capture and exploit that regularity.

State and Behavior

Any quality or property of a system element is called an attribute. The state of a system is a set of system attributes at a given time. A **system event** describes any change to theenvironment of a system, and hence its state:

- **Static** - A single state exists with no events.
- **Dynamic** - Multiple possible stable states exist.
- **Homeostatic** - System is static but its elements are dynamic. The system maintains its state by internal adjustments.

A stable state is one in which a system will remain until another event occurs.

State can be monitored using state variables, values of attributes which indicate the system state. The set of possible values of state variables over time is called the "'state space'". State variables are generally continuous, but can be modeled using a finite state model (or, "state machine").

Ackoff (Ackoff 1971) considers "change" to be how a system is affected by events, and system behavior as the effect a system has upon its environment. A system can

- **react** to a request by turning on a light,
- **respond** to darkness by deciding to turn on the light
- **act** to turn on the lights at a fixed time, randomly or with discernible reasoning.

A stable system is one which has one or more stable states within an environment for a range of possible events:

- **Deterministic** systems have a one-to-one mapping of state variables to state space, allowing future states to be predicted from past states.
- **Non-Deterministic** systems have a many-to-many mapping of state variables; future state cannot be reliably predicted.

The relationship between determinism and system complexity, including the idea of chaotic systems, is further discussed in the Complexity article.

Survival Behavior

Systems often behave in a manner that allows them to sustain themselves in one or more alternative viable states. Many natural or social systems have this goal, either consciously or as a "self organizing" system, arising from the interaction between elements.

Entropy is the tendency of systems to move towards disorder or disorganization. In physics, entropy is used to describe how organized heat energy is “lost” into the random background energy of the surrounding environment (the 2nd Law of Thermodynamics). A similar effect can be seen in engineered systems. What happens to a building or garden left unused for any time? Entropy can be used as a metaphor for aging, skill fade, obsolescence, misuse, boredom, etc.

"Negentropy" describes the forces working in a system to hold off entropy. Homeostasis is the biological equivalent of this, describing behavior which maintains a "steady state" or "dynamic equilibrium". Examples in nature include human cells, which maintain the same function while replacing their physical content at regular intervals. Again, this can be used as a metaphor for the fight against entropy, e.g. training, discipline, maintenance, etc.

Hitchins (Hitchins 2007) describes the relationship between the viability of a system and the number of connections between its elements. Hitchins's concept of connected variety states that stability of a system increases with its connectivity (both internally and with its environment). (See variety.)

Goal Seeking Behavior

Some systems have reasons for existence beyond simple survival. Goal seeking is one of the defining characteristics of engineered systems:

- A **goal** is a specific outcome which a system can achieve in a specified time
- An **objective** is a longer term outcome which can be achieved through a series of goals.
- An **ideal** is an objective which cannot be achieved with any certainty, but for which progress towards the objective has value.

Systems may be single goal seeking (perform set tasks), multi-goal seeking (perform related tasks), or reflective (set goals to tackle objectives or ideas). There are two types of goal seeking systems:

- purposive systems have multiple goals with some shared outcome. Such a system can be used to provide pre-determined outcomes within an agreed time period. This system may have some freedom to choose how to achieve the goal. If it has memory it may develop processes describing the behaviors needed for defined goals. Most machines or software systems are purposive.
- purposeful systems are free to determine the goals needed to achieve an outcome. Such a system can be tasked to pursue objectives or ideals over a longer time through a series of goals. Humans and sufficiently complex machines are purposeful.

Control Behavior

Cybernetics, the science of control, defines two basic control mechanisms:

- **Negative feedback**, maintaining system state against a set objectives or levels.
- **Positive feedback**, forced growth or contraction to new levels.

One of the main concerns of cybernetics is the balance between stability and speed of response. A black-box system (glossary) view looks at the whole system. Control can only be achieved by carefully balancing inputs with outputs, which reduces speed of response. A white-box system (glossary) view considers the system elements and their relationships; control mechanisms can be imbedded into this structure to provide more responsive control and associated risks to stability.

Another useful control concept is that of a "meta-system", which sits over the system and is responsible for controlling its functions, either as a black-box or white-box. In this case, behavior arises from the combination of system and meta-system.

Control behavior is a trade between

- **Specialization**, the focus of system behavior to exploit particular features of its environment, and
- flexibility, the ability of a system to adapt quickly to environmental change.

While some system elements may be optimized for either specialization, a temperature sensitive switch, flexibility, or an autonomous human controller, complex systems must strike a balance between the two for best results. This is an example of the concept of dualism, discussed in more detail in Principles of Systems Thinking.

Variety describes the number of different ways elements can be controlled, and is dependent on the different ways in which they can then be combined. The Law of Requisite Variety states that a control system must have at least as much variety as the system it is controlling (Ashby 1956).

Function

Ackoff defines function as outcomes which contribute to goals or objectives. To have a function, a system must be able to provide the outcome in two or more different ways. (This is called **equifinality**).

This view of function and behavior is common in systems science. In this paradigm, all system elements have behavior of some kind; however, to be capable of functioning in certain ways requires a certain richness of behaviors.

In most hard systems approaches, a set of functions are described from the problem statement, and then associated with one or more alternative element structures (Flood and Carson 1993). This process may be repeated until a system component (implementable combinations of function and structure) has been defined (Martin 1997). Here, function is defined as either a task or activity that must be performed to achieve a desired outcome or as a transformation of inputs to outputs. This transformation may be:

- **Synchronous**, a regular interaction with a closely related system, or
- **Asynchronous**, an irregular response to a demand from another system that often triggers a set response.

The behavior of the resulting system is then assessed as a combination of function and effectiveness. In this case behavior is seen as an external property of the system as a whole and is often described as analogous to human or organic behavior (Hitchins 2009).

Hierarchy, Emergence and Complexity

System behavior is related to combinations of element behaviors. Most systems exhibit **increasing variety**; i.e., they have behavior resulting from the combination of element behaviors. The term "synergy", or weak emergence, is used to describe the idea that the whole is greater than the sum of the parts. This is generally true; however, it is also possible to get **reducing variety**, in which the whole function is less than the sum of the parts, (Hitchins 2007).

Complexity frequently takes the form of hierarchies (glossary). Hierarchic systems have some common properties independent of their specific content, and they will evolve far more quickly than non-hierarchic systems of comparable size (Simon 1996). A natural system hierarchy is a consequence of wholeness, with strongly cohesive elements grouping together forming structures which reduce complexity and increase robustness (Simons 1962).

Encapsulation is the enclosing of one thing within another. It may also be described as the degree to which it is enclosed. System encapsulation encloses system elements and their interactions from the external environment, and usually involves a system boundary that hides the internal from the external; for example, the internal organs of the human body can be optimized to work effectively within tightly defined conditions because they are protected from extremes of environmental change.

Socio-technical systems form what are known as control hierarchies, with systems at a higher level having some ownership of control over those at lower levels. Hitchins (2009) describes how systems form "preferred patterns" which can be used to the enhanced stability of interacting systems hierarchies.

Looking across a hierarchy of systems generally reveals increasing complexity at the higher level, relating to both the structure of the system and how it is used. The term emergence describes behaviors emerging across a complex system hierarchy.

Effectiveness, Adaptation and Learning

Systems effectiveness is a measure of the system's ability to perform the functions necessary to achieve goals or objectives. Ackoff (Ackoff 1971) defines this as the product of the number of combinations of behavior to reach a function and the efficiency of each combination.

Hitchins (2007) describes effectiveness as a combination of **performance** (how well a function is done in ideal conditions), **availability** (how often the function is there when needed), and **survivability** (how likely is it that the system will be able to use the function fully).

System elements and their environment change in a positive, neutral or negative way in individual situations. An adaptive system is one that is able to change itself or its environment if its effectiveness is insufficient to achieve its current or future objectives. Ackoff (Ackoff 1971) defines four types of adaptation, changing the environment or the system in response to internal or external factors.

A system may also **learn**, improving its effectiveness over time, without any change in state or goal.

References

Works Cited

Ackoff, R.L. 1971. "Towards a System of Systems Concepts". *Management Science.* 17(11).

Ackoff, R. 1979. "The Future of Operational Research is Past." *Journal of the Operational Research Society*. 30(2): 93–104, Pergamon Press.

Ashby, W R. 1956. "Chapter 11". *Introduction to Cybernetics*. London, UK: Wiley.

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications,* Revised ed. New York, NY, USA: Braziller.

Bertalanffy, L. von. 1975. 'Perspectives on General System Theory. *E. Taschdjian, ed. New York: George Braziller.*

Boardman, J. and B. Sauser. 2008. *Systems Thinking: Coping with 21st Century Problems.* Boca Raton, FL, USA: Taylor & Francis.

Flood, R.L. and E.R. Carson. 1993. *Dealing With Complexity: An Introduction to the Theory and Application of Systems Science*. New York, NY, USA: Plenum Press.

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

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4): 59-63.

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

Martin J. N. 1997. *Systems Engineering Guidebook.* Boca Raton, FL, USA: CRC Press.

Skyttner, L. 2001. *General Systems Theory: Ideas and Applications.* Singapore: World Scientific Publishing Co. p. 53-69.

Simon, H.A. 1962. "The Architecture of Complexity." *Proceedings of the American Philosophical Society.* 106(6) (Dec. 12, 1962): 467-482.

Simon, H. 1996. *The Sciences of the Artificial*, 3rd ed. Cambridge, MA: MIT Press.

Primary References

Ackoff, R.L. 1971. "Towards a System of Systems Concept." *Management Science.* 17(11).

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight.* 12(4): 59-63.

Additional References

Edson, Robert. 2008. *Systems Thinking. Applied. A Primer.* Arlington, VA, USA: Applied Systems Thinking Institute (ASysT), Analytic Services Inc.

Jackson, S., D. Hitchins, and H. Eisner. 2010. "What is the Systems Approach?" INCOSE *Insight.* 13(1) (April 2010): 41-43.

Waring, A. 1996. "Chapter 1." *Practical Systems Thinking*. London, UK: International Thomson Business Press.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Principles of Systems Thinking

Lead Author: *Rick Adcock*, **Contributing Authors:** *Scott Jackson, Janet Singer, Duane Hybertson*

This topic forms part of the Systems Thinking knowledge area (KA). It identifies systems principles as part of the basic ideas of systems thinking.

Some additional concepts more directly associated with engineered systems are described, and a summary of system principles associated with the concepts already defined is provided. A number of additional “laws” and heuristics are also discussed.

Systems Principles, Laws, and Heuristics

A principle is a general rule of conduct or behavior (Lawson and Martin 2008). It can also be defined as a basic generalization that is accepted as true and that can be used as a basis for reasoning or conduct (WordWeb 2012c). Thus, systems principles can be used as a basis for reasoning about systems thinking or associated conduct (systems approaches).

Separation of Concerns

A systems approach is focused on a systems-of-interest (SoI) of an open system. This SoI consists of open, interacting subsystems that as a whole interact with and adapt to other systems in an environment. The systems approach also considers the SoI in its environment to be part of a larger, wider, or containing system (Hitchins 2009).

In the What is Systems Thinking? topic, a “systems thinking paradox” is discussed. How is it possible to take a holistic system view while still being able to focus on changing or creating systems?

Separation of concerns describes a balance between considering parts of a system problem or solution while not losing sight of the whole (Greer 2008). Abstraction is the process of taking away characteristics from something in order to reduce it to a set of base characteristics (SearchCIO 2012). In attempting to understand complex situations it is easier to focus on bounded problems, whose solutions still remain agnostic to the greater problem (Erl 2012). This process sounds reductionist, but it can be applied effectively to systems. The key to the success of this approach is ensuring that one of the selected problems is the concerns of the system as a whole. Finding balance between using abstraction to focus on specific concerns while ensuring we continue to consider the whole is at the center of systems approaches. A view is a subset of information observed of one or more entities, such as systems. The physical or conceptual point from which a view is observed is the viewpoint, which can be motivated by one or more observer concerns. Different views of the same target must be both separate, to reflect separation of concerns, and integrated such that all views of a given target are consistent and form a coherent whole (Hybertson 2009). Some sample views of a system are internal (What does it consist of?); external (What are its properties and behavior as a whole?); static (What are its parts or structures?); and dynamic (interactions).

encapsulation, which encloses system elements and their interactions from the external environment, is discussed in Concepts of Systems Thinking. Encapsulation is associated with modularity, the degree to which a system's components may be separated and recombined (Griswold 1995). Modularity applies to systems in natural, social, and engineered domains. In engineering, encapsulation is the isolation of a system function within a module and provides precise specifications for the module (IEEE Std. 610.12-1990).

Dualism is a characteristic of systems in which they exhibit seemingly contradictory characteristics that are important for the system (Hybertson 2009). The yin yang concept in Chinese philosophy emphasizes the interaction between dual elements and their harmonization, ensuring a constant dynamic balance through a cyclic dominance of one element and then the other, such as day and night (IEP 2006).

From a systems perspective the interaction, harmonization, and balance between system properties is important. Hybertson (Hybertson 2009) defines **leverage** as the duality between

- **Power**, the extent to which a system solves a specific problem, and
- **Generality**, the extent to which a system solves a whole class of problems.

While some systems or elements may be optimized for one extreme of such dualities, a dynamic balance is needed to be effective in solving complex problems.

Summary of Systems Principles

A set of systems principles is given in Table 1 below. The "Names" segment points to concepts underlying the principle. (See Concepts of Systems Thinking). Following the table, two additional sets of items related to systems principles are noted and briefly discussed: prerequisite laws for design science, and heuristics and pragmatic principles.

Table 1. A Set of Systems Principles. (SEBoK Original)

Name	Statement of Principle
Abstraction	A focus on essential characteristics is important in problem solving because it allows problem solvers to ignore the nonessential, thus simplifying the problem. (Sci-Tech Encyclopedia 2009; SearchCIO 2012; Pearce 2012)
Boundary	A boundary or membrane separates the system from the external world. It serves to concentrate interactions inside the system while allowing exchange with external systems. (Hoagland, Dodson, and Mauck 2001)
Change	Change is necessary for growth and adaptation, and should be accepted and planned for as part of the natural order of things rather than something to be ignored, avoided, or prohibited (Bertalanffy 1968; Hybertson 2009).
Dualism	Recognize dualities and consider how they are, or can be, harmonized in the context of a larger whole (Hybertson 2009)
Encapsulation	Hide internal parts and their interactions from the external environment. (Klerer 1993; IEEE 1990)
Equifinality	In open systems, the same final state may be reached from different initial conditions and in different ways (Bertalanffy 1968). This principle can be exploited, especially in systems of purposeful agents.
Holism	A system should be considered as a single entity, a whole, not just as a set of parts. (Ackoff 1979; Klir 2001)
Interaction	The properties, capabilities, and behavior of a system are derived from its parts, from interactions between those parts, and from interactions with other systems. (Hitchins 2009 p. 60)
Layer Hierarchy	The evolution of complex systems is facilitated by their hierarchical structure (including stable intermediate forms) and the understanding of complex systems is facilitated by their hierarchical description. (Pattee 1973; Bertalanffy 1968; Simon 1996)

Leverage	Achieve maximum leverage (Hybertson 2009). Because of the power versus generality tradeoff, leverage can be achieved by a complete solution (power) for a narrow class of problems, or by a partial solution for a broad class of problems (generality).
Modularity	Unrelated parts of the system should be separated, and related parts of the system should be grouped together. (Griswold 1995; Wikipedia 2012a)
Network	The network is a fundamental topology for systems that forms the basis of togetherness, connection, and dynamic interaction of parts that yield the behavior of complex systems (Lawson 2010; Martin et al. 2004; Sillitto 2010)
Parsimony	One should choose the simplest explanation of a phenomenon, the one that requires the fewest assumptions (Cybernetics 2012). This applies not only to choosing a design, but also to operations and requirements.
Regularity	Systems science should find and capture regularities in systems, because those regularities promote systems understanding and facilitate systems practice. (Bertalanffy 1968)
Relations	A system is characterized by its relations: the interconnections between the elements. Feedback is a type of relation. The set of relations defines the network of the system. (Odum 1994)
Separation of Concerns	A larger problem is more effectively solved when decomposed into a set of smaller problems or concerns. (Erl 2012; Greer 2008)
Similarity/ Difference	Both the similarities and differences in systems should be recognized and accepted for what they are. (Bertalanffy 1975 p. 75; Hybertson 2009). Avoid forcing one size fits all, and avoid treating everything as entirely unique.
Stability/ Change	Things change at different rates, and entities or concepts at the stable end of the spectrum can and should be used to provide a guiding context for rapidly changing entities at the volatile end of the spectrum (Hybertson 2009). The study of complex adaptive systems can give guidance to system behavior and design in changing environments (Holland 1992).
Synthesis	Systems can be created by choosing (conceiving, designing, selecting) the right parts, bringing them together to interact in the right way, and in orchestrating those interactions to create requisite properties of the whole, such that it performs with optimum effectiveness in its operational environment, so solving the problem that prompted its creation" (Hitchins 2008: 120).
View	Multiple views, each based on a system aspect or concern, are essential to understand a complex system or problem situation. One critical view is how concern relates to properties of the whole. (Edson 2008; Hybertson 2009)

The principles are not independent. They have synergies and tradeoffs. Lipson (Lipson 2007), for example, argued that “scalability of open-ended evolutionary processes depends on their ability to exploit functional modularity, structural regularity and hierarchy.” He proposed a formal model for examining the properties, dependencies, and tradeoffs among these principles. Edson (Edson 2008) related many of the above principles in a structure called the conceptagon, which he modified from the work of Boardman and Sauser (Boardman and Sauser 2008). Edson also provided guidance on how to apply these principles. Not all principles apply to every system or engineering decision. Judgment, experience, and heuristics (see below) provide understanding into which principles apply in a given situation.

Several principles illustrate the relation of view with the dualism and yin yang principle; for example, holism and separation of concerns. These principles appear to be contradictory but are in fact dual ways of dealing with complexity. Holism deals with complexity by focusing on the whole system, while separation of concerns divides a problem or system into smaller, more manageable elements that focus on particular concerns. They are reconciled by the fact that both views are needed to understand systems and to engineer systems; focusing on only one or the other does not give sufficient understanding or a good overall solution. This dualism is closely related to the systems thinking paradox described in What is Systems Thinking?.

Rosen (Rosen 1979) discussed “false dualisms” of systems paradigms that are considered incompatible but are in fact different aspects or views of reality. In the present context, they are thus reconcilable through yin yang harmonization. Edson (Edson 2008) emphasized viewpoints as an essential principle of systems thinking; specifically, as a way to understand opposing concepts.

Derick Hitchins (Hitchins 2003) produced a systems life cycle theory described by a set of seven principles forming an integrated set. This theory describes the creation, manipulation and demise of engineered systems. These principles consider the factors which contribute to the stability and survival of man made systems in an environment.

Stability is associated with the principle of **connected variety**, in which stability is increased by variety plus the **cohesion** and **adaptability** of that variety. Stability is limited by allowable relations, resistance to change, and patterns of interaction. Hitchins describes how interconnected systems tend toward a **cyclic progression**, in which variety is generated, dominance emerges to suppress variety, dominant modes decay and collapse and survivors emerge to generate new variety.

Guidance on how to apply many of these principles to engineered systems is given in the topic Synthesizing Possible Solutions, as well as in System Definition and other knowledge areas in Part 3 of the SEBoK.

Prerequisite Laws of Design Science

John Warfield (Warfield 1994) identified a set of laws of generic design science that are related to systems principles. Three of these laws are stated here:

1. ''Law of Requisite Variety'': A design situation embodies a variety that must be matched by the specifications. The variety includes the diversity of stakeholders. This law is an application of the design science of the Ashby (1956) Law of Requisite Variety, which was defined in the context of cybernetics and states that to successfully regulate a system, the variety of the regulator must be at least as large as the variety of the regulated system.
2. ''Law of Requisite Parsimony'': Information must be organized and presented in a way that prevents human information overload. This law derives from Miller's findings on the limits of human information processing capacity (Miller 1956). Warfield's structured dialog method is one possible way to help achieve the requisite parsimony.
3. ''Law of Gradation'': Any conceptual body of knowledge can be graded in stages or varying degrees of complexity and scale, ranging from simplest to most comprehensive, and the degree of knowledge applied to any design situation should match the complexity and scale of the situation. A corollary, called the Law of Diminishing Returns, states that a body of knowledge should be applied to a design situation to the stage at which the point of diminishing returns is reached.

Heuristics and Pragmatic Principles

A heuristic is a common sense rule intended to increase the probability of solving some problem (WordWeb 2012b). In the present context it may be regarded as an informal or pragmatic principle. Maier and Rechtin (Maier and Rechtin 2000) identified an extensive set of heuristics that are related to systems principles. A few of these heuristics are stated here.

- Relationships among the elements are what give systems their added value. This is related to the ''Interaction'' principle.
- Efficiency is inversely proportional to universality. This is related to the ''Leverage'' principle.
- The first line of defense against complexity is simplicity of design. This is related to the ''Parsimony'' principle.
- In order to understand anything, you must not try to understand everything (attributed to Aristotle). This is related to the ''Abstraction'' principle.

An International Council on Systems Engineering (INCOSE) working group (INCOSE 1993) defined a set of “pragmatic principles” for systems engineering (SE). They are essentially best practice heuristics for engineering a system. For example:

- Know the problem, the customer, and the consumer
- Identify and assess alternatives so as to converge on a solution
- Maintain the integrity of the system

Hitchins defines a set of SE principles which include principles of holism and synthesis as discussed above, as well as principles describing how systems problems should be resolved that are of particular relevance to a Systems Approach Applied to Engineered Systems (Hitchins 2009).

References

Works Cited

Ackoff, R. 1979. "The Future of Operational Research is Past." *Journal of the Operational Research Society.* 30(2): 93–104, Pergamon Press.

Ashby, W.R. 1956. "Requisite variety and its implications for the control of complex systems." *Cybernetica.* 1(2):1–17.

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications*. Revised ed. New York, NY: Braziller.

Bertalanffy, L. von. 1975. *Perspectives on General System Theory*. E. Taschdjian, ed. New York: George Braziller.

Boardman, J. and B. Sauser. 2008. *Systems Thinking: Coping with 21st Century Problems*. Boca Raton, FL: Taylor & Francis.

Cybernetics (Web Dictionary of Cybernetics and Systems). 2012. "Principle of Parsimony or Principle of Simplicity." Accessed December 3 2014 at Web Dictionary of Cybernetics and Systems http://pespmc1.vub.ac.be/ASC/PRINCI_SIMPL.html

Edson, R. 2008. *Systems Thinking. Applied. A Primer*. Arlington, VA, USA: Applied Systems Thinking (ASysT) Institute, Analytic Services Inc.

Erl, T. 2012. "SOA Principles: An Introduction to the Service Orientation Paradigm." Accessed December 3 2014 at Arcitura http://www.soaprinciples.com/p3.php

Greer, D. 2008. "The Art of Separation of Concerns." Accessed December 3 2014 at Aspiring Craftsman http://aspiringcraftsman.com/tag/separation-of-concerns/

Griswold, W. 1995. "Modularity Principle." Accessed December 3 2014 at William Griswold http://cseweb.ucsd.edu/users/wgg/CSE131B/Design/node1.html

Hitchins D. K. 2003. *Advanced systems thinking engineering and management.* Boston, MA, USA: Artech House.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight*. 12(4): 59-63.

Hoagland, M., B. Dodson, and J. Mauck. 2001. *Exploring the Way Life Works*. Burlington, MA, USA: Jones and Bartlett Publishers, Inc.

Holland, J. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA: MIT Press.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boca Raton, FL, USA: Auerbach/CRC Press.

IEEE. 1990. *IEEE Standard Glossary of Software Engineering Terminology*. Geneva, Switzerland: Institute of Electrical and Electronics Engineers. IEEE Std 610.12-1990.

IEP (Internet Encyclopedia of Philosophy). 2006. "Yinyang (Yin-yang)." Accessed December 3 2014 at Internet Encyclopedia of Philosophy http://www.iep.utm.edu/yinyang/

INCOSE 1993. *An Identification of Pragmatic Principles - Final Report*. SE Principles Working Group, January 21, 1993.

Klerer, S. “System Management Information Modeling.” *IEEE Communications*. 31(5)May 1993: 38-44.

Klir, G. 2001. *Facets of Systems Science*, 2nd ed. New York, NY: Kluwer Academic/Plenum Publishers.

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

Lawson, H. and J. Martin. 2008. "On the Use of Concepts and Principles for Improving Systems Engineering Practice." INCOSE International Symposium 2008, 15-19 June 2008, The Netherlands.

Lipson, H. 2007. "Principles of modularity, regularity, and hierarchy for scalable systems." *Journal of Biological Physics and Chemistry.* 7: 125–128.

Maier, M. and E. Rechtin. 2000. *The Art of Systems Architecting, 2nd ed*. Boca Raton, FL: CRC Press.

Miller, G. 1956. "The magical number seven, plus or minus two: some limits on our capacity for processing information." *The Psychological Review.* 63: 81–97.

Odum, H. 1994. *Ecological and General Systems: An Introduction to Systems Ecology (Revised Edition).* Boulder, CO: University Press of Colorado.

Pattee, H. (ed.) 1973. *Hierarchy Theory: The Challenge of Complex Systems*. New York, NY: George Braziller.

Pearce, J. 2012. "The Abstraction Principle." Posting date unknown. Accessed December 3 2014 at Jon Pearce, San Jose State University http://www.cs.sjsu.edu/~pearce/modules/lectures/ood/principles/Abstraction.htm

Rosen, R. 1979. "Old trends and new trends in general systems research." *International Journal of General Systems.* 5(3): 173-184.

Sci-Tech Encyclopedia. 2009. "Abstract Data Type." *McGraw-Hill Concise Encyclopedia of Science and Technology, Sixth Edition*, The McGraw-Hill Companies, Inc.

SearchCIO. 2012. "Abstraction." Accessed December 3 2014 at SearchCIO http://searchcio-midmarket.techtarget.com/definition/abstraction

Sillitto, H. 2010. "Design principles for Ultra-Large-Scale (ULS) Systems." 'Proceedings of INCOSE International Symposium 2010', 12-15 July 2010, Chicago, IL.

Simon, H. 1996. *The Sciences of the Artificial, 3rd ed*. Cambridge, MA: MIT Press.

Warfield, J.N. 1994. *A Science of Generic Design*. Ames, IA: Iowa State University Press.

Wikipedia. 2012a. "Modularity." Accessed December 3 2014 at Wikipedia http://en.wikipedia.org/wiki/Modularity

WordWeb. 2012b. "Dualism." Accessed December 3 2014 at WordWeb http://www.wordwebonline.com/en/DUALISM.

WordWeb. 2012c. "Heuristic." Accessed December 3 2014 at WordWeb http://www.wordwebonline.com/en/HEURISTIC.

WordWeb. 2012d. "Principle." Accessed December 3 2014 at WordWeb http://www.wordwebonline.com/en/PRINCIPLE.

Primary References

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications*. Revised ed. New York, NY: Braziller.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Auerbach/CRC Press, Boca Raton, FL.

Klir, G. 2001. *Facets of Systems Science, 2nd ed.* New York: Kluwer Academic/Plenum Publishers.

Additional References

Francois, F. (ed.). 2004. *International Encyclopedia of Systems and Cybernetics,* 2nd ed. Munich, Germany: K. G. Saur Verlag.

Meyers, R. (ed.). 2009. *Encyclopedia of Complexity and Systems Science.* New York, NY: Springer.

Midgley, G. (ed.). 2003. *Systems Thinking.* Thousand Oaks, CA: Sage Publications Ltd.

Volk, T., and J.W. Bloom. (2007). "The use of metapatterns for research into complex systems of teaching, learning, and schooling. Part I: Metapatterns in nature and culture." *Complicity: An International Journal of Complexity and Education.* 4(1): 25—43.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Patterns of Systems Thinking

Lead Author: *Rick Adcock*, **Contributing Authors:** *Scott Jackson, Janet Singer, Duane Hybertson*

This topic forms part of the Systems Thinking knowledge area (KA). It identifies systems patterns as part of the basic ideas of systems thinking. The general idea of patterns and a number of examples are described. A brief conclusion discusses the maturity of systems science from the perspective of principles and patterns.

Systems Patterns

This section first discusses definitions, types, and pervasiveness of patterns. Next, samples of basic patterns in the form of hierarchy and network patterns, metapatterns, and systems engineering (SE) patterns are discussed. Then samples of patterns of failure (or “antipatterns”) are presented in the form of system archetypes, along with antipatterns in software engineering and other fields. Finally, a brief discussion of patterns as maturity indicators is given.

Pattern Definitions and Types

The most general definition of pattern is that it is an expression of an observed regularity. Patterns exist in both natural and artificial systems and are used in both systems science and systems engineering (SE). Theories in science are patterns. Building architecture styles are patterns. Engineering uses patterns extensively.

Patterns are a representation of similarities in a set or class of problems, solutions, or systems. In addition, some patterns can also represent uniqueness or differences, e.g., uniqueness pattern or unique identifier, such as automobile vehicle identification number (VIN), serial number on a consumer product, human fingerprints, DNA. The pattern is that a unique identifier, common to all instances in a class (such as fingerprint), distinguishes between all instances in that class.

The term pattern has been used primarily in building architecture and urban planning by Alexander (Alexander et al. 1977, Alexander 1979) and in software engineering (e.g., Gamma et al. 1995; Buschmann et al. 1996). Their definitions portray a pattern as capturing design ideas as an archetypal and reusable description. A design pattern provides a generalized solution in the form of templates to a commonly occurring real-world problem within a given context. A design pattern is not a finished design that can be transformed directly into a specific solution. It is a description or template for how to solve a problem that can be used in many different specific situations (Gamma et al. 1995; Wikipedia 2012b). Alexander placed significant emphasis on the pattern role of reconciling and resolving competing forces, which is an important application of the yin yang principle.

Other examples of general patterns in both natural and engineered systems include: conventional designs in engineering handbooks, complex system models such as evolution and predator-prey models that apply to multiple application domains, domain taxonomies, architecture frameworks, standards, templates, architecture styles, reference architectures, product lines, abstract data types, and classes in class hierarchies (Hybertson 2009). Shaw and Garlan (Garlan 1996) used the terms pattern and style interchangeably in discussing software architecture. Lehmann and Belady (Lehmann 1985) examined a set of engineered software systems and tracked their change over time and observed regularities that they captured as evolution laws or patterns.

Patterns have been combined with model-based systems engineering (MBSE) to lead to pattern-based systems engineering (PBSE) (Schindel and Smith 2002, Schindel 2005).

Patterns also exist in systems practice, both science and engineering. At the highest level, Gregory (1966) defined science and design as behavior patterns:

> *The scientific method is a pattern of problem-solving behavior employed in finding out the nature of what exists, whereas the design method is a pattern of behavior employed in inventing things of value which do not yet exist.*

Regularities exist not only as positive solutions to recurring problems, but also as patterns of failure, i.e., as commonly attempted solutions that consistently fail to solve recurring problems. In software engineering these are called antipatterns, originally coined and defined by Koenig (Koenig 1995): An antipattern is just like a pattern, except that instead of a solution it gives something that looks superficially like a solution but isn't one. Koenig's rationale was that if one does not know how to solve a problem, it may nevertheless be useful to know about likely blind alleys. Antipatterns may include patterns of pathologies (i.e., common diseases), common impairment of normal functioning, and basic recurring problematic situations. These antipatterns can be used to help identify the root cause of a problem and eventually lead to solution patterns. The concept was expanded beyond software to include project management, organization, and other antipatterns (Brown et al. 1998; AntiPatterns Catalog 2012).

Patterns are grouped in the remainder of this section into basic foundational patterns and antipatterns (or patterns of failure).

Basic Foundational Patterns

The basic patterns in this section consist of a set of hierarchy and network patterns, followed by a set of metapatterns and SE patterns.

Hierarchy and Network Patterns

The first group of patterns are representative types of hierarchy patterns distinguished by the one-to-many relation type (extended from Hybertson 2009, 90), as shown in the table below. These are presented first because hierarchy patterns infuse many of the other patterns discussed in this section.

Table 1. Hierarchy Patterns. (SEBoK Original)

Relation	Hierarchy Type or Pattern
Basic: Repeating One-to-Many Relation	General: Tree structure
Part of a Whole	Composition (or Aggregation) hierarchy
Part of + Dualism: Each element in the hierarchy is a holon, i.e., is both a whole that has parts and a part of a larger whole	Holarchy (composition hierarchy of holons) (Koestler 1967) - helps recognize similarities across levels in multi-level systems
Part of + Interchangeability: The parts are clonons, i.e., interchangeable	Composition Hierarchy of Clonons (Bloom 2005). Note: This pattern reflects horizontal similarity.
Part of + Self-Similarity: At each level, the shape or structure of the whole is repeated in the parts, i.e., the hierarchy is self-similar at all scales.	Fractal. Note: This pattern reflects vertical similarity.
Part of + Connections or Interactions among Parts	System composition hierarchy
Control of Many by One	Control hierarchy—e.g., a command structure
Subtype or Sub-Class	Type or specialization hierarchy; a type of generalization
Instance of Category	Categorization (object-class; model-metamodel…) hierarchy; a type of generalization

Network patterns are of two flavors. First, traditional patterns are network topology types, such as bus (common backbone), ring, star (central hub), tree, and mesh (multiple routes) (ATIS 2008). Second, the relatively young science of networks has been investigating social and other complex patterns, such as percolation, cascades, power law, scale-free, small worlds, semantic networks, and neural networks (Boccara 2004; Neumann et al. 2006).

Metapatterns

The metapatterns identified and defined in the table below are from (Bloom 2005), (Volk and Bloom 2007), and (Kappraff 1991). They describe a metapattern as convergences exhibited in the similar structures of evolved systems across widely separated scales (Volk and Bloom 2007).

Table 2. Metapatterns. (SEBoK Original)

Name	Brief Definition	Examples
Spheres	Shape of maximum volume, minimum surface, containment	Cell, planet, dome, ecosystem, community
Centers	Key components of system stability	Prototypes, purpose, causation; Deoxyribonucleic acid (DNA), social insect centers, political constitutions and government, attractors
Tubes	Surface transfer, connection, support	Networks, lattices, conduits, relations; leaf veins, highways, chains of command
Binaries Plus	Minimal and thus efficient system	Contrast, duality, reflections, tensions, complementary/symmetrical/reciprocal relationships; two sexes, two-party politics, bifurcating decision process
Clusters, Clustering	Subset of webs, distributed systems of parts with mutual attractions	Bird flocks, ungulate herds, children playing, egalitarian social groups
Webs or Networks	Parts in relationships within systems (can be centered or clustered, using clonons or holons)	Subsystems of cells, organisms, ecosystems, machines, society
Sheets	Transfer surface for matter, energy, or information	Films; fish gills, solar collectors
Borders and Pores	Protection, openings for controlled exchange	Boundaries, containers, partitions, cell membranes, national borders
Layers	Combination of other patterns that builds up order, structure, and stabilization	Levels of scale, parts and wholes, packing, proportions, tiling

Similarity	Figures of the same shape but different sizes	Similar triangles, infant-adult
Emergence	General phenomenon when a new type of functionality derives from binaries or webs.	Creation (birth), life from molecules, cognition from neurons
Holarchies	Levels of webs, in which successive systems are parts of larger systems	Biological nesting from biomolecules to ecosystems, human social nesting, engineering designs, computer software
Holons	Parts of systems as functionally unique	Heart-lungs-liver (holons) of body
Clonons	Parts of systems as interchangeable	Skin cells (clonons) of the skin; bricks in constructing a house
Arrows	Stability or gradient-like change over time	Stages, sequence, orientation, stress, growth, meanders, biological homeostasis, growth, self-maintaining social structures
Cycles	Recurrent patterns in systems over time	Alternating repetition, vortex, spiral, turbulence, helices, rotations; protein degradation and synthesis, life cycles, power cycles of electricity generating plants, feedback cycles
Breaks	Relatively sudden changes in system behavior	Transformation, change, branching, explosion, cracking, translations; cell division, insect metamorphosis, coming-of-age ceremonies, political elections, bifurcation points
Triggers	Initiating agents of breaks, both internal and external	Sperm entering egg or precipitating events of war
Gradients	Continuum of variation between binary poles	Chemical waves in cell development, human quantitative and qualitative values

Systems Engineering Patterns

Some work has been done on various aspects of explicitly applying patterns to SE. A review article of much of this work was written by Bagnulo and Addison (Bagnulo and Addison 2010), covering patterns in general, capability engineering, pattern languages, pattern modeling, and other SE-related pattern topics. Cloutier (Cloutier 2005) discussed applying patterns to SE, based on architecture and software design patterns. Haskins (Haskins 2005), and Simpson and Simpson (Simpson and Simpson 2006) discussed the use of SE pattern languages to enhance the adoption and use of SE patterns. Simpsons identified three high-level, global patterns that can be used as a means of organizing systems patterns:

- Anything can be described as a system.
- The problem system is always separate from the solution system.
- Three systems, at a minimum, are always involved in any system activity: the environmental system, the product system, and the process system.

Haskins (Haskins 2008) also proposed the use of patterns as a way to facilitate the extension of SE from traditional technological systems to address social and socio-technical systems. Some patterns have been applied and identified in this extended arena, described as patterns of success by Rebovich and DeRosa (Rebovich and DeRosa2012). Stevens (Stevens 2010) also discussed patterns in the engineering of large-scale, complex “mega-systems.”

A common SE activity in which patterns are applied is in system design, especially in defining one or more solution options for a system-of-interest. See Synthesizing Possible Solutions for a discussion. The more specific topic of using patterns (and antipatterns, as described below) to understand and exploit emergence is discussed in the Emergence topic.

Patterns of Failure: Antipatterns

System Archetypes

The system dynamics community has developed a collection of what are called system archetypes. The concept was originated by Forrester (Forrester 1969), while Senge (Senge 1990) appears to have introduced the system archetype term. According to Braun (2002), the archetypes describe common patterns of behavior that help answer the question, “Why do we keep seeing the same problems recur over time?” They focus on behavior in organizations and other complex social systems that are repeatedly but unsuccessfully used to solve recurring problems. This is why they are grouped here under antipatterns, even though the system dynamics community does not refer to the archetypes as antipatterns. The table below summarizes the archetypes. There is not a fixed set, or even fixed names for a given archetype. The table shows alternative names for some archetypes.

Table 3. System Archetypes. (SEBoK Original)

Name (Alternates)	Description	Reference**
Counterintuitive Behavior	Forrester identified three “especially dangerous” counter-intuitive behaviors of social systems, which correspond respectively to three of the archetypes discussed below: (1) Low-Leverage Policies: Ineffective Actions; (2) High Leverage Policies: Often Wrongly Applied; and (3) Long-Term vs. Short-Term Trade-offs	F1, F2
Low-Leverage Policies: Ineffective Actions (Policy Resistance)	Most intuitive policy changes in a complex system have very little leverage to create change; this is because the change causes reactions in other parts of the system that counteract the new policy.	F1, F3, M
High Leverage Policies: Often Wrongly Applied (High Leverage, Wrong Direction)	A system problem is often correctable with a small change, but this high-leverage solution is typically counter-intuitive in two ways: (1) the leverage point is difficult to find because it is usually far removed in time and place from where the problem appears, and (2) if the leverage point is identified, the change is typically made in the wrong direction, thereby intensifying the problem.	F1, F3, M
Long-Term vs. Short-Term Trade-offs (Fixes that Fail, Shifting the Burden, Addiction)	Short-term solutions are intuitive, but in complex systems there is nearly always a conflict or tradeoff between short-term and long-term goals. Thus, a quick fix produces immediate positive results, but its unforeseen and unintended long-term consequences worsen the problem. Furthermore, a repeated quick fix approach makes it harder to change to a more fundamental solution approach later.	F1, F3, M, S, B
Drift to Low Performance (Eroding Goals, Collapse of Goals)	There is a strong tendency for complex system goals to drift downward. A gap between current state and goal state creates pressure to lower the goal rather than taking difficult corrective action to reach the goal. Over time the continually lowered goals lead to crisis and possible collapse of the system.	F1, F3, M, B
Official Addiction – Shifting the Burden to the Intervener	The ability of a system to maintain itself deteriorates when an intervener provides help and the system then becomes dependent on the intervener	M, S
Limits to Growth (a.k.a. Limits to Success)	A reinforcing process of accelerating growth (or expansion) will encounter a balancing process as the limit of that system is approached and continuing efforts will produce diminishing returns as one approaches the limits.	S, B
Balancing Process with Delay	Delay in the response of a system to corrective action causes the correcting agent to either over-correct or to give up due to no visible progress.	S
Escalation	Two systems compete for superiority, with each escalating its competitive actions to get ahead, to the point that both systems are harmed.	B
Success to the Successful	Growth leads to decline elsewhere. When two equally capable systems compete for a limited resource, if one system receives more resources, it is more likely to be successful, which results in it’s receiving even more resources, in a reinforcing loop.	S, B
Tragedy of the Commons	A shared resource is depleted as each system abuses it for individual gain, ultimately hurting all who share it.	H, S, B

Growth and Underinvestment	In a situation where capacity investments can overcome limits, if such investments are not made, then growth stalls, which then rationalizes further underinvestment.	S, B
Accidental Adversaries	Two systems destroy their relationship through escalating retaliations for perceived injuries.	B
Attractiveness Principle	In situations where a system faces multiple limiting or impeding factors, the tendency is to consider each factor separately to select which one to address first, rather than a strategy based on the interdependencies among the factors.	B

**** B**—(Braun 2002); **F1**—(Forrester 1969); **F2**—(Forrester 1995); **F3**—(Forrester 2009); **H**—(Hardin 1968); **M**—(Meadows 1982); **S**—(Senge 1990).

Relations among system archetypes were defined by Goodman and Kleiner (Goodman and Kleiner 1993/1994) and republished in Senge et al. (Senge et al. 1994).

Software and Other Antipatterns

Antipatterns have been identified and collected in the software community in areas that include: Architecture, development, project management, user interface, organization, analysis, software design, programming, methodology, and configuration management (AntiPatterns Catalog 2012, Wikibooks 2012). A brief statement of three of them follows; the first two are organization and the third is software design.

- Escalation of commitment - Failing to revoke a decision when it proves wrong.
- Moral hazard - Insulating a decision-maker from the consequences of his or her decision.
- Big ball of mud - A system with no recognizable structure,

A link between the software community and the system archetypes is represented in a project at the Software Engineering Institute (SEI) (2012), which is exploring the system archetypes in the context of identifying recurring software acquisition problems as “acquisition archetypes.” They refer to both types of archetypes as patterns of failure.

Another set of antipatterns in the general systems arena has been compiled by Troncale (Troncale 2010; Troncale 2011) in his systems pathologies project. Sample pathology types or patterns include:

- Cyberpathologies - Systems-level malfunctions in feedback architectures.
- Nexopathologies - Systems-level malfunctions in network architectures or dynamics.
- Heteropathologies - systems-level malfunctions in hierarchical, modular structure & dynamics.

Some treatments of antipatterns, including Senge (Senge 1990) and SEI (SEI 2012), also provide some advice on dealing with or preventing the antipattern.

Patterns and Maturity

Patterns may be used as an indicator of the maturity of a domain of inquiry, such as systems science or systems engineering. In a mature and relatively stable domain, the problems and solutions are generally understood and their similarities are captured in a variety of what are here called patterns. A couple of observations can be made in this regard on the maturity of systems science in support of systems engineering.

In the arenas of physical systems and technical systems, systems science is relatively mature; many system patterns of both natural physical systems and engineered technical systems are reasonably well defined and understood.

In the arena of more complex systems, including social systems, systems science is somewhat less mature. Solution patterns in that arena are more challenging. A pessimistic view of the possibility of science developing solutions to social problems was expressed by Rittel and Webber (Rittel and Webber 1973) in their classic paper on wicked problems: “The search for scientific bases for confronting problems of social policy is bound to fail, because . . . they are ‘wicked’ problems, whereas science has developed to deal with ‘tame’ problems.” A more optimistic stance toward social problems has characterized the system dynamics community. They have been pointing out for over 40 years the problems with conventional solutions to social problems, in the form of the system archetypes and associated feedback loop models. That was an important first step. Nevertheless, they have had difficulty achieving the second step; producing social patterns that can be applied to solve those problems. The antipatterns characterize problems, but the patterns for solving those problems are elusive.

Despite the difficulties, however, social systems do exhibit regularities, and social problems are often solved to some degree. The social sciences and complex systems community have limited sets of patterns, such as common types of organization structures, common macro-economic models, and even patterns of insurgency and counter-insurgency. The challenge for systems science is to capture those regularities and the salient features of those solutions more broadly, and make them explicit and available in the form of mature patterns. Then perhaps social problems can be solved on a more regular basis. As systems engineering expands its scope from the traditional emphasis on technical aspects of systems to the interplay of the social and technical aspects of socio-technical systems, such progress in systems science is becoming even more important to the practice of systems engineering.

References

Works Cited

Alexander, C. 1979. *The Timeless Way of Building*. New York: Oxford University Press.

Alexander, C., S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel. 1977. *A Pattern Language: Towns – Buildings – Construction*. New York: Oxford University Press.

ATIS. 2008. *ATIS Telecom Glossary 2007*. Washington, D.C.: Alliance for Telecommunications Industry Solutions. Accessed December 3 2014 at ATIS http://www.atis.org/glossary/definition.aspx?id=3516.

Bagnulo, A. and T. Addison. 2010. *State of the Art Report on Patterns in Systems Engineering and Capability Engineering.* Contract Report 2010-012 by CGI Group for Defence R&D Canada – Valcartier. March 2010.

Bloom, J. 2005. "The application of chaos, complexity, and emergent (meta)patterns to research in teacher education." *Proceedings of the 2004 Complexity Science and Educational Research Conference* (pp. 155-191), Sep 30–Oct 3 • Chaffey's Locks, Canada. http://www.complexityandeducation.ca.

Boccara, N. 2004. *Modeling Complex Systems*. New York, NY: Springer-Verlag.

Braun, T. 2002. "The System Archetypes." Accessed December 3 at http:/ / www. albany. edu/ faculty/ gpr/ PAD724/724WebArticles/sys_archetypes.pdf

Brown, W., R. Malveau, H. McCormick, and T. Mowbray. 1998. *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. John Wiley & Sons.

Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. 1996. *Pattern-Oriented Software Architecture: A System of Patterns*. Chichester, U.K.: John Wiley.

Cloutier, R. 2005. "Toward the Application of Patterns to Systems Engineering." 'Proceedings of the Conference on Systems Engineering Research (CSER) 2005, March 23-25, Hoboken, NJ, USA.

Forrester, J. 1969. *Urban Dynamics*. Waltham, MA: Pegasus Communications.

Forrester, J. 1995. "Counterintuitive Behavior of Social Systems." *Technology Review.* 73(3), Jan. 1971: 52-68.

Forrester, J. 2009. Learning through System Dynamics as Preparation for the 21st Century.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley.

Goodman, G. and A. Kleiner. 1993/1994. "Using the Archetype Family Tree as a Diagnostic Tool." *The Systems Thinker.* December 1993/January 1994.

Gregory, S. 1966. "Design and the design method," in S. Gregory (ed.). *The Design Method*. London, Engladh: Butterworth.

Hardin, G. 1968. "The Tragedy of the Commons." *Science.* 162(13 December 1968):1243-1248. DOI: 10.1126/science.162.3859.1243.

Haskins, C. 2005. "Application of Patterns and Pattern Languages to Systems Engineering." *Proceedings of the 15th Annual INCOSE International Symposium*. Rochester, NY, July 10-13, 2005.

Haskins, C. 2008. "Using patterns to transition systems engineering from a technological to social context." *Systems Engineering*, 11(2), May 2008: 147-155.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boca Raton, FL: Auerbach/CRC Press.

Kappraff, J. (1991). *Connections: The geometric bridge between art and science*. New York, NY: McGraw-Hill.

Koenig, A. 1995. "Patterns and Antipatterns". *Journal of Object-Oriented Programming.* 8(1)March/April 1995: 46–48.

Koestler, A. 1967. *The Ghost in the Machine*. New York, NY: Macmillan.

Lehmann, M. and L. Belady. 1985. *Program Evolution*. London, England: Academic Press.

Meadows, D. 1982. Whole Earth Models and Systems. *The Co-Evolution Quarterly.* Summer 1982: 98-108.

Odum, H.1994. *Ecological and General Systems: An Introduction to Systems Ecology (Revised Edition).* Boulder, CO: University Press of Colorado.

Rebovich, G. and J. DeRosa 2012. "Patterns of Success in Systems Engineering of IT-Intensive Government Systems." *Procedia Computer Science.* 8(2012): 303 – 308.

Rittel, H. and M. Webber. 1973. "Dilemmas in a general theory of planning." *Policy Sciences.* 4:155–169.

Schindel, W. 2005. "Pattern-based systems engineering: An extension of model-based systems engineering." INCOSE TIES tutorial presented at 2005 INCOSE Symposium, 10-15 July 2005, Rochester, NY.

Schindel, W. and V. Smith. 2002. *Results of applying a families-of-systems approach to systems engineering of product line families.* Technical Report 2002-01-3086. SAE International.

SEI 2012. Patterns of Failure: System Archetypes. Accessed December 3 2014 at SEI http://www.sei.cmu.edu/acquisition/research/pofsa.cfm

Senge, P. 1990. *The Fifth Discipline: Discipline: The Art and Practice of the Learning Organization*. New York, NY: Currency Doubleday.

Senge, P., A. Kleiner, C. Roberts and R. Ross. 1994. *The Fifth Discipline Fieldbook: Strategies and Tools for Building a Learning Organization*. New York, NY: Currency Doubleday.

Shaw, M. and D. Garlan. 1996. *Software Architecture: Perspectives on an Emerging Discipline*. Upper Saddle River, NJ: Prentice Hall.

Simpson, J. and M. Simpson. 2006. "Foundational Systems Engineering Patterns for a SE Pattern Language." *Proceedings of the 16th Annual INCOSE Symposium, July, 2006, Orlando, FL.*

Stevens, R. 2011. *Engineering Mega-Systems: The Challenge of Systems Engineering in the Information Age*. Boca Raton, FL: Auerbach/Taylor & Francis.

Troncale, L. 2010. "Would a Rigorous Knowledge Base in “Systems Pathology” Add to the S.E. Portfolio?" Presented at 2010 LA Mini-Conference, 16 October 2010, Loyola Marymount University, Los Angeles, CA.

Troncale, L. 2011. “Would A Rigorous Knowledge Base in Systems Pathology Add Significantly to the SE Portfolio?" Proceedings of the Conference on Systems Engineering Research (CSER), April 14-16, Redondo Beach, CA.

Volk, T., and J.W. Bloom. 2007. "The use of metapatterns for research into complex systems of teaching, learning, and schooling. Part I: Metapatterns in nature and culture." *Complicity: An International Journal of Complexity and Education*, 4(1): 25—43.

Wikibooks. 2012a. "AntiPatterns." http:/ / en. wikibooks. org/ wiki/ Introduction_to_Software_Engineering/ Architecture/Anti-Patterns.

Wikipedia. 2012b. "Software design pattern." http://en.wikipedia.org/wiki/Software_design_pattern

Primary References

Alexander, C. 1979. *The Timeless Way of Building*. New York, NY: Oxford University Press.

Bertalanffy, L. von. 1968. *General System Theory: Foundations, Development, Applications*. Revised ed. New York, NY: Braziller.

Bloom, J. 2005. "The application of chaos, complexity, and emergent (meta)patterns to research in teacher education." *Proceedings of the 2004 Complexity Science and Educational Research Conference* (pp. 155-191), Sep 30–Oct 3, Chaffey's Locks, Canada. http://www.complexityandeducation.ca.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Auerbach/CRC Press, Boca Raton, FL.

Additional References

Principia Cybernetica. 1996. "Cybernetics and Systems Theory." Accessed 21 April 2013. Available at: http:// pespmc1.vub.ac.be/CYBSYSTH.html

Erl, T. 2009. *SOA: Design Patterns*. Upper Saddle River, NJ: Prentice Hall.

Erl, T. 2008. *SOA: Principles of Service Design*. Upper Saddle River, NJ: Prentice Hall.

Francois, F. (ed.). 2004. *International Encyclopedia of Systems and Cybernetics*, 2nd ed.. Munich, Germany: K. G. Saur Verlag.

Meyers, R. (ed.). 2009. *Encyclopedia of Complexity and Systems Science.* New York, NY: Springer.

Midgley, G. (ed.). 2003. *Systems Thinking.* Thousand Oaks, CA: Sage Publications Ltd.

Principia Cybernetica Web. 2013. "Web Dictionary of Cybernetics and Systems." Accessed 21 April 2013. Available at: http://pespmc1.vub.ac.be/ASC/indexASC.html

Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Knowledge Area: Representing Systems with Models

Representing Systems with Models

Lead Author: *Sanford Friedenthal,* ***Contributing Authors:*** *Dov Dori, Yaniv Mordecai*

A model is a simplified representation of a system at some particular point in time or space intended to promote understanding of the real system. As an abstraction of a system, it offers insight about one or more of the system's aspects, such as its function, structure, properties, performance, behavior, or cost.

Overview

The modeling of systems as holistic, value-providing entities has been gaining recognition as a central process of systems engineering. The use of modeling and simulation during the early stages of the system design of complex systems and architectures can:

- document system functions and requirements,
- assess the mission performance,
- estimate costs,
- evaluate tradeoffs, and
- provide insights to improve performance, reduce risk, and manage costs.

Modeling and analysis can complement testing and evaluation which occur later in the life cycle. In some systems, modeling and simulation may be the only way to fully evaluate performance (e.g., ballistic missile defense) or to evaluate system performance in severe scenarios (e.g., response to weapons of mass destruction attacks on the homeland). Furthermore, advanced simulations, e.g. flight simulators and command and control center simulations, can be a cost-effective technique for personnel training in accompaniment with operational system training (INCOSE 2012).

Modeling serves to make concepts concrete and formal, enhance quality, productivity, documentation, and innovation, as well as to reduce the cost and risk of systems development.

Modeling occurs at many levels: component, subsystem, system, and systems-of-systems; and throughout the life cycle of a system. Different types of models may be needed to represent systems in support of the analysis, specification, design, and verification of systems. This knowledge area provides an overview of models used to represent different aspects of systems.

Modeling is a common practice that is shared by most engineering disciplines, including:

- electrical engineering, which uses electrical circuit design models
- mechanical engineering, which uses three-dimensional computer-aided design models
- software engineering, which uses software design and architecture models.

Each of these disciplines has its own language with its syntax and semantics, serving as a means of communication among professionals in that discipline. Analytic models are used to support power, thermal, structural, and embedded real-time analysis.

Modeling Standards play an important role in defining system modeling concepts that can be represented for a particular domain of interest and enable the integration of different types of models across domains of interest.

Topics

Each part of the Guide to the Systems Engineering Body of Knowledge (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:

- What is a Model?
- Why Model?
- Types of Models
- System Modeling Concepts
- Modeling Standards
- Integrating Supporting Aspects into System Models

References

Works Cited

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.

Primary References

Dori, D. 2002. *Object-Process Methodology – A Holistic Systems Paradigm*. Berlin, Germany: Springer Verlag.

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

Friedenthal, S., A. Moore, R. Steiner, and M. Kaufman. 2012. *A Practical Guide to SysML: The Systems Modeling Language,* 2nd Edition. Needham, MA, USA: OMG Press.

Guizzardi, G. 2007. *On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models*. Proceedings of the Databases and Information Systems IV Conference, Amsterdam, Netherlands. Accessed December 4 2014 at ACM http://portal.acm.org/citation.cfm?id=1565425.

INCOSE. 2007. *Systems Engineering Vision 2020*. Seattle, WA, USA: International Council on Systems Engineering. September 2007. INCOSE-TP-2004-004-02.

Wymore, A.W. 1993. *Model-Based Systems Engineering*. Boca Raton, FL, USA: CRC Press, Inc.

Additional References

Holt, J. and S. Perry. 2008. *SysML for systems engineering*. Stevenage: Institution of Engineering and Technology. Accessed December 4 2014 at Ebrary http://site.ebrary.com/id/10263845.

Grobshtein, Y. and D. Dori. 2011. "Generating SysML Views from an OPM Model: Design and Evaluation." *Systems Engineering.* 14(3), Sept.

West, P., J. Kobza, and S. Goerger. 2011. "Chapter 4, Systems Modeling and Analysis," in Parnell, G.S., P.J. Driscoll, and D.L Henderson (eds). *Decision Making for Systems Engineering and Management*, 2nd ed. Wiley Series in Systems Engineering. Hoboken, NJ: Wiley & Sons Inc.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

What is a Model?

Lead Author: *Sanford Friedenthal*, ***Contributing Authors:*** *Dov Dori, Yaniv Mordecai*

This topic provides foundational concepts, such as definitions of a model and a modeling language, and expresses their relationships to modeling tools and model-based systems engineering (MBSE).

Definition of a Model

There are many definitions of the word *model*. The following definitions refer to a model as a representation of selected aspects of a domain of interest to the modeler:

- a physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process (DoD 1998);
- a representation of one or more concepts that may be realized in the physical world (Friedenthal, Moore, and Steiner 2009);
- a simplified representation of a system at some particular point in time or space intended to promote understanding of the real system (Bellinger 2004);
- an abstraction of a system, aimed at understanding, communicating, explaining, or designing aspects of interest of that system (Dori 2002); and
- a selective representation of some system whose form and content are chosen based on a specific set of concerns; the model is related to the system by an explicit or implicit mapping (Object Management Group 2010).

In the context of systems engineering, a model that represents a system and its environment is of particular importance to the system engineer who must specify, design, analyze, and verify systems, as well as share information with other stakeholders. A variety of system models are used to represent different types of systems for different modeling purposes. Some of the purposes for modeling systems are summarized in the topic Why Model?, and a simple taxonomy of the different types of models are described in the topic Types of Models. The modeling standards topic refers to some of the standard system modeling languages and other modeling standards that support MBSE.

A model can have different forms as indicated in the first definition above, including a physical, mathematical, or logical representation. A physical model can be a mockup that represents an actual system, such as a model airplane. A mathematical model may represent possible flight trajectories in terms of acceleration, speed, position, and orientation. A logical model may represent logical relationships that describe potential causes of airplane failure, such as how an engine failure can result in a loss of power and cause the airplane to lose altitude, or how the parts of the system are interconnected. It is apparent that many different models may be required to represent a system-of-interest (SoI).

Modeling Language

A physical model is a concrete representation of an actual system that can be felt and touched. Other models may be more abstract representations of a system or entity. These models rely on a modeling language to express their meaning as explained in "On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models" (Guizzardi 2007).

Just as engineering drawings express the 3D structure of mechanical and architectural designs, conceptual models are the means by which systems are conceived, architected, designed, and built. The resulting models are the counterparts of the mechanical design blueprint. The difference, however, is that while blueprints are exact representations of physical artifacts with a precise, agreed-upon syntax and long tradition of serving as a means of communication among professionals, conceptual models are just beginning to make headway toward being a complete and unambiguous representation of a system under development. The articles in the special section of Communications of the Association for Computing Machinery (ACM) (Dori 2003) present the abstract world of systems analysis and architecting by means of conceptual modeling, and, how to evaluate, select, and construct models.

Modeling languages are generally intended to be both human-interpretable and computer-interpretable, and are specified in terms of both syntax and semantics.

The abstract syntax specifies the model constructs and the rules for constructing the model. In the case of a natural language like English, the constructs may include types of words such as verbs, nouns, adjectives, and prepositions, and the rules specify how these words can be used together to form proper sentences. The abstract syntax for a mathematical model may specify constructs to define mathematical functions, variables, and their relationship. The abstract syntax for a logical model may also specify constructs to define logical entities and their relationships. A well-formed model abides by the rules of construction, just as a well-formed sentence must conform to the grammatical rules of the natural language.

The concrete syntax specifies the symbols used to express the model constructs. The natural language English can be expressed in text or Morse code. A modeling language may be expressed using graphical symbols and/or text statements. For example, a functional flow model may be expressed using graphical symbols consisting of a combination of graphical nodes and arcs annotated with text; while a simulation modeling language may be expressed using a programming language text syntax such as the C programming language.

The semantics of a language define the meaning of the constructs. For example, an English word does not have explicit meaning until the word is defined. Similarly, a construct that is expressed as a symbol, such as a box or arrow on a flow chart, does not have meaning until it is defined. The language must give meaning to the concept of a verb or noun, and must give specific meaning to a specific word that is a verb or noun. The definition can be established by providing a natural language definition, or by mapping the construct to a formalism whose meaning is defined. As an example, a graphical symbol that expresses $sin(x)$ and $cos(x)$ is defined using a well-defined mathematical formalism for the sine and cosine function. If the position of a pendulum is defined in terms of $sin(\theta)$ and $cos(\theta)$, the meaning of the pendulum position is understood in terms of these formalisms.

Modeling Tools

Models are created by a modeler using modeling tools. For physical models, the modeling tools may include drills, lathes, and hammers. For more abstract models, the modeling tools are typically software programs running on a computer. These programs provide the ability to express modeling constructs using a particular modeling language. A word processor can be viewed as a tool used to build text descriptions using natural language. In a similar way, modeling tools are used to build models using modeling languages. The tool often provides a tool palette to select symbols and a content area to construct the model from the graphical symbols or other concrete syntax. A modeling tool typically checks the model to evaluate whether it conforms to the rules of the language, and enforces such rules to help the modeler create a well-formed model. This is similar to the way a word processor checks the text to see that it conforms to the grammar rules for the natural language.

Some modeling tools are commercially available products, while others may be created or customized to provide unique modeling solutions. Modeling tools are often used as part of a broader set of engineering tools which constitute the systems development environment. There is increased emphasis on tool support for standard modeling languages that enable models and modeling information to be interchanged among different tools.

Relationship of Model to Model-Based Systems Engineering

The International Council of Systems Engineering (INCOSE) INCOSE Systems Engineering Vision 2020 (2007) defines MBSE as “the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.” In MBSE, the models of the system are primary artifacts of the systems engineering process, and are managed, controlled, and integrated with other parts of the system technical baseline. This contrasts with the traditional document-centric approach to systems engineering, where text-based documentation and specifications are managed and controlled. Leveraging a model-based approach to systems engineering is intended to result in significant improvements in system specification and design quality, lower risk and cost of system development by surfacing issues early in the design process, enhanced productivity through reuse of system artifacts, and improved communications among the system development and implementation teams.

In addition to creating models, the MBSE approach typically includes methods for model management which aim to ensure that models are properly controlled and methods for model validation which aim to ensure that models accurately represent the systems being modeled.

The jointly sponsored INCOSE/Object Management Group (OMG) MBSE Wiki [1] provides additional information on the INCOSE MBSE Initiative including some applications of MBSE and some key topics related to MBSE such as sections on Methodology and Metrics, and Model Management.

The Final Report of the Model Based Engineering (MBE) Subcommittee, which was generated by the the National Defense Industrial Association (NDIA) Modeling and Simulation Committee of the Systems Engineering Division, highlights many of the benefits, risks, and challenges of a model-based approach, and includes many references to case studies of MBE (NDIA 2011).

Brief History of System Modeling Languages and Methods

Many system modeling methods and associated modeling languages have been developed and deployed to support various aspects of system analysis, design, and implementation. Functional modeling languages include the data flow diagram (DFD) (Yourdon and Constantine 1979), Integration Definition for Functional Modeling (IDEF0) (Menzel and Maier 1998), and enhanced functional flow block diagram (eFFBD). Other behavioral modeling techniques include the classical state transition diagram, statecharts (Harel 1987), and process flow diagrams. Structural modeling techniques include data structure diagrams (Jackson 1975), entity relationship diagrams (Chen 1976), and object modeling techniques (Rumbaugh et al. 1991), which combine object diagrams, DFDs, and statecharts.

In 2008, Estefan conducted an extensive survey of system modeling methods, processes, and tools and documented the results in A Survey of Model-Based Systems Engineering (MBSE) Methodologies (Estefan 2008). This survey identifies several candidate MBSE methodologies and modeling languages that can be applied to support an MBSE approach. Additional modeling methods are available from the MBSE Wiki [1] under the section on Methodology and Metrics [2] . The modeling standards section refers to some of the standard system modeling languages and other modeling standards that support MBSE. Since Estefan's report, a number of surveys have been conducted to understand the acceptance and barriers to model-based systems engineering (Bone, 2010, Cloutier 2014, 2015).

References

Works Cited

Bellinger, G. 2004. "Modeling & Simulation: An Introduction," in *Mental Model Musings*. Available at http://www.systems-thinking.org/modsim/modsim.htm.

Bone, M. & Cloutier, R. 2010. The Current State of Model Based Systems Engineering: Results from the OMG™ SysML Request for Information 2009. 8 th Conference on Systems Engineering Research March 17-19, 2010, Hoboken, NJ Paper #1569270919. Available at http:/ / www. omgsysml. org/ SysML_2009_RFI_Response_Summary-bone-cloutier.pdf.Last accessed 5/26/2019

Cloutier, R. & Bone, M. 2014. MBSE Survey Presented January 2015 INCOSE IW Los Angeles, CA. Available at http:/ / www. omgwiki. org/ MBSE/ lib/ exe/ fetch. php?media=mbse:incose_mbse_survey_results_initial_report_2015_01_24.pdf.Last accessed 5/26/2019

Cloutier, R. 2015. Current Modeling Trends in Systems Engineering. INCOSE Insight, 18(2)

Chen, P. 1976. "The Entity Relationship Model – Toward a Unifying View of Data." ACM *Transactions on Data Base Systems* 1(1): 9-36.

DoD. 1998. "'DoD Modeling and Simulation (M&S) Glossary" in *DoD Manual 5000.59-M*. Arlington, VA, USA: US Department of Defense. January. P2.13.22. Available at http:/ / www. dtic. mil/ whs/ directives/ corres/ pdf/ 500059m.pdf

Dori, D. 2002. *Object-Process Methodology: A Holistic System Paradigm*. New York, NY, USA: Springer.

Dori, D. 2003. "Conceptual Modeling and System Architecting." *Communications of the ACM*, 46(10), pp. 62-65. [3]

Estefan, J. 2008. "A Survey of Model-Based Systems Engineering (MBSE) Methodologies," rev. B, Seattle, WA, USA: International Council on Systems Engineering. INCOSE-TD-2007-003-02.

Friedenthal, S., A. Moore, R. Steiner, and M. Kaufman. 2012. *A Practical Guide to SysML: The Systems Modeling Language*, 2nd Edition. Needham, MA, USA: OMG Press.

Guizzardi, G. 2007. "On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models" Proceedings of Seventh International Baltic Conference. Amsterdam, The Netherlands. Available at http:/ / portal. acm.org/citation.cfm?id=1565425.

Harel, D. 1987. "Statecharts: A Visual Formalism for Complex Systems." *Science of Computer Programming.* 8(3): 231–74.

Jackson, M.A. 1975. *Principles of Program Design.* New York, NY, USA: Academic Press.

Menzel, C. and R.J. Mayer. 1998. "The IDEF Family of Languages." in P. Bernus, K. Mertins, and G. Schmidt (eds.). *Handbook on Architectures for Information Systems.* Berlin, Germany: Springer-Verlag. p. 209-241.

OMG. 2010. *MDA Foundation Model*. Needham, MA, USA: Object Management Group. ORMSC/2010-09-06.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorenson. 1990. *Object-Oriented Modeling and Design.* Upper Saddle River, NJ: Prentice Hall.

Press, Y. and L.L. Constantine. 1976. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design.* Upper Saddle River, NJ: Prentice Hall.

Yourdon E. and Constantine L.L. 1973. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA. 1st Edition.

Primary References

Estefan, J. 2008. *A Survey of Model-Based Systems Engineering (MBSE) Methodologies,* Rev. B. San Diego, CA, USA: International Council on Systems Engineering. INCOSE-TD-2007-003-02. Available at http://www.incose.org/ProductsPubs/pdf/techdata/MTTC/MBSE_Methodology_Survey_2008-0610_RevB-JAE2.pdf.

Guizzardi, G. 2007. "On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models." Proceedings of Seventh International Baltic Conference. Amsterdam, The Netherlands. Available at http://portal.acm.org/citation.cfm?id=1565425.

INCOSE. 2007. *Systems Engineering Vision 2020.* Seattle, WA, USA: International Council on Systems Engineering. September 2007. INCOSE-TP-2004-004-02.

NDIA. 2011. *Final Report of the Model Based Engineering (MBE) Subcommittee*. Arlington, VA, USA: National Defense Industrial Association. Available at: http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/Committees/M_S%20Committee/Reports/MBE_Final_Report_Document_(2011-04-22)_Marked_Final_Draft.pdf

Additional References

Downs, E., P. Clare, and I. Coe. 1992. *Structured Systems Analysis and Design Method: Application and Context*. Hertfordshire, UK: Prentice-Hall International.

Eisner, H. 1988. *Computer-Aided Systems Engineering*. Englewood Cliffs, NJ, USA: Prentice Hall.

Harel, D. 1987. "Statecharts: A Visual Formalism for Complex Systems." *Science of Computer Programming.* 8(3): 231–74.

Kossiakoff, A. and W. Sweet. 2003. "Chapter 14" in *Systems Engineering Principles and Practice*. New York, NY, USA: Wiley and Sons.

OMG. "MBSE Wiki." Object Management Group (OMG). Accessed 11 September 2011. Available at: http://www.omgwiki.org/MBSE/doku.php.

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

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

References

[1] http://www.omgwiki.org/MBSE/doku.php

[2] http://www.omgwiki.org/MBSE/doku.php?id=mbse:methodology|

[3] http://esml.iem.technion.ac.il/site/wp-content/uploads/2011/02/article169.pdf

Why Model?

Lead Author: *Sanford Friedenthal*, ***Contributing Authors:*** *Dov Dori, Yaniv Mordecai*

System models can be used for many purposes. This topic highlights some of those purposes, and provides indicators of an effective model, in the context of model-based systems engineering (MBSE).

Purpose of a Model

Models are representations that can aid in defining, analyzing, and communicating a set of concepts. System models are specifically developed to support analysis, specification, design, verification, and validation of a system, as well as to communicate certain information. One of the first principles of modeling is to clearly define the purpose of the model. Some of the purposes that models can serve throughout the system life cycle are

- **Characterizing an existing system:** Many existing systems are poorly documented, and modeling the system can provide a concise way to capture the existing system design. This information can then be used to facilitate maintaining the system or to assess the system with the goal of improving it. This is analogous to creating an architectural model of an old building with overlays for electrical, plumbing, and structure before proceeding to upgrade it to new standards to withstand earthquakes.
- **Mission and system concept formulation and evaluation:** Models can be applied early in the system life cycle to synthesize and evaluate alternative mission and system concepts. This includes clearly and unambiguously defining the system's mission and the value it is expected to deliver to its beneficiaries. Models can be used to explore a trade-space by modeling alternative system designs and assessing the impact of critical system parameters such as weight, speed, accuracy, reliability, and cost on the overall measures of merit. In addition to bounding the system design parameters, models can also be used to validate that the system requirements meet stakeholder needs before proceeding with later life cycle activities such as synthesizing the detailed system design.
- **System design synthesis and requirements flowdown:** Models can be used to support architecting system solutions, as well as flow mission and system requirements down to system components. Different models may be required to address different aspects of the system design and respond to the broad range of system requirements. This may include models that specify functional, interface, performance, and physical requirements, as well as other non-functional requirements such as reliability, maintainability, safety, and security.
- **Support for system integration and verification:** Models can be used to support integration of the hardware and software components into a system, as well as to support verification that the system satisfies its requirements. This often involves integrating lower level hardware and software design models with system-level design models which verify that system requirements are satisfied. System integration and verification may also include replacing selected hardware and design models with actual hardware and software products in order to incrementally verify that system requirements are satisfied. This is referred to as hardware-in-the-loop testing and software-in-the-loop testing. Models can also be used to define the test cases (glossary) and other aspects of the test program to assist in test planning and execution.
- **Support for training:** Models can be used to simulate various aspects of the system to help train users to interact with the system. Users may be operators, maintainers, or other stakeholders. Models may be a basis for developing a simulator (glossary) of the system with varying degrees of fidelity to represent user interaction in different usage scenarios.
- **Knowledge capture and system design evolution:** Models can provide an effective means for capturing knowledge about the system and retaining it as part of organizational knowledge. This knowledge, which can be reused and evolved, provides a basis for supporting the evolution of the system, such as changing system requirements in the face of emerging, relevant technologies, new applications, and new customers. Models can

also enable the capture of families of products.

Indicators of an Effective Model

When modeling is done well, a model's purposes are clear and well-defined. The value of a model can be assessed in terms of how effectively it supports those purposes. The remainder of this section and the topics Types of Models, System Modeling Concepts, and Modeling Standards describe indicators of an effective model (Friedenthal, Moore, and Steiner 2012).

Model Scope

The model must be scoped to address its intended purpose. In particular, the types of models and associated modeling languages selected must support the specific needs to be met. For example, suppose models are constructed to support an aircraft's development. A system architecture model may describe the interconnection among the airplane parts, a trajectory analysis model may analyze the airplane trajectory, and a fault tree analysis model may assess potential causes of airplane failure.

For each type of model, the appropriate breadth, depth, and fidelity should be determined to address the model's intended purpose. The model breadth reflects the system requirements coverage in terms of the degree to which the model must address the functional, interface, performance, and physical requirements, as well as other non-functional requirements, such as reliability, maintainability, and safety. For an airplane functional model, the model breadth may be required to address some or all of the functional requirements to power up, takeoff, fly, land, power down, and maintain the aircraft's environment.

The model's depth indicates the coverage of system decomposition from the system context down to the system components. For the airplane example, a model's scope may require it to define the system context, ranging from the aircraft, the control tower, and the physical environment, down to the navigation subsystem and its components, such as the inertial measurement unit; and, perhaps down to lower-level parts of the inertial measurement unit.

The model's fidelity indicates the level of detail the model must represent for any given part of the model. For example, a model that specifies the system interfaces may be fairly abstract and represent only the logical information content, such as aircraft status data; or, it may be much more detailed to support higher fidelity information, such as the encoding of a message in terms of bits, bytes, and signal characteristics. Fidelity can also refer to the precision of a computational model, such as the time step required for a simulation.

Indicators of Model Quality

The quality of a model should not be confused with the quality of the design that the model represents. For example, one may have a high-quality, computer-aided design model of a chair that accurately represents the design of the chair, yet the design itself may be flawed such that when one sits in the chair, it falls apart. A high quality model should provide a representation sufficient to assist the design team in assessing the quality of the design and uncovering design issues.

Model quality is often assessed in terms of the adherence of the model to modeling guidelines and the degree to which the model addresses its intended purpose. Typical examples of modeling guidelines include naming conventions, application of appropriate model annotations, proper use of modeling constructs, and applying model reuse considerations. Specific guidelines are different for different types of models. For example, the guidelines for developing a geometric model using a computer-aided design tool may include conventions for defining coordinate systems, dimensioning, and tolerances.

Model-based Metrics

Models can provide a wealth of information that can be used for both technical and management metrics to assess the modeling effort, and, in some cases, the overall systems engineering (SE) effort. Different types of models provide different types of information. In general, models provide information that enables one to

- assess progress;
- estimate effort and cost;
- assess technical quality and risk; and
- assess model quality.

Models can capture metrics similar to those captured in a traditional document-based approach to systems engineering, but potentially with more precision given the more accurate nature of models compared to documents. Traditional systems engineering metrics are described in *Metrics Guidebook for Integrated Systems and Product Development* (Wilbur 2005).

A model's progress can be assessed in terms of the completeness of the modeling effort relative to the defined scope of the model. Models may also be used to assess progress in terms of the extent to which the requirements have been satisfied by the design or verified through testing. When augmented with productivity metrics, the model can be used to estimate the cost of performing the required systems engineering effort to deliver the system.

Models can be used to identify critical system parameters and assess technical risks in terms of any uncertainty that lies in those parameters. The models can also be used to provide additional metrics that are associated with its purpose. For example, when the model's purpose is to support mission and system concept formulation and evaluation, then a key metric may be the number of alternative concepts that are explored over a specified period of time.

References

Works Cited

Friedenthal, S., A. Moore, R. Steiner, and M. Kaufman. 2012. *A Practical Guide to SysML: The Systems Modeling Language*, 2nd Edition. Needham, MA, USA: OMG Press.

Wilbur, A., G. Towers, T. Sherman, D. Yasukawa, and S. Shreve. 2005. *Metrics Guidebook for Integrated Systems and Product Development*. Seattle, WA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-1995-002-01.

Primary References

Friedenthal, S., A. Moore, R. Steiner, and M. Kaufman. 2012. *A Practical Guide to SysML: The Systems Modeling Language*, 2nd Edition. Needham, MA, USA: OMG Press.

Wilbur, A., G. Towers, T. Sherman, D. Yasukawa, and S. Shreve. 2005. *Metrics Guidebook for Integrated Systems and Product Development*. Seattle, WA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-1995-002-01. Accessed April 13 at https://www.incose.org/ProductsPublications/techpublications/GuideMetrics

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Types of Models

Lead Author: *Sanford Friedenthal*, ***Contributing Authors:*** *Dov Dori, Yaniv Mordecai*

There are many different types of models (glossary) expressed in a diverse array of modeling languages and tool sets. This article offers a taxonomy of model types and highlights how different models must work together to support broader engineering efforts.

Model Classification

There are many different types of models and associated modeling languages to address different aspects of a system and different types of systems. Since different models serve different purposes, a classification of models can be useful for selecting the right type of model for the intended purpose and scope.

Formal versus Informal Models

Since a system model is a representation of a system, many different expressions that vary in degrees of formalism could be considered models. In particular, one could draw a picture of a system and consider it a model. Similarly, one could write a description of a system in text, and refer to that as a model. Both examples are representations of a system. However, unless there is some agreement on the meaning of the terms, there is a potential lack of precision and the possibility of ambiguity in the representation.

The primary focus of system modeling is to use models supported by a well-defined modeling language. While less formal representations can be useful, a model must meet certain expectations for it to be considered within the scope of model-based systems engineering (MBSE). In particular, the initial classification distinguishes between informal and formal models as supported by a modeling language with a defined syntax and the semantics for the relevant domain of interest.

Physical Models versus Abstract Models

The United States “Department of Defense Modeling and Simulation (M&S) Glossary” asserts that “a model can be [a] physical, mathematical, or otherwise logical representation of a system” (1998). This definition provides a starting point for a high level model classification. A physical model is a concrete representation that is distinguished from the mathematical and logical models, both of which are more abstract representations of the system. The abstract model can be further classified as descriptive (similar to logical) or analytical (similar to mathematical). Some example models are shown in Figure 1.

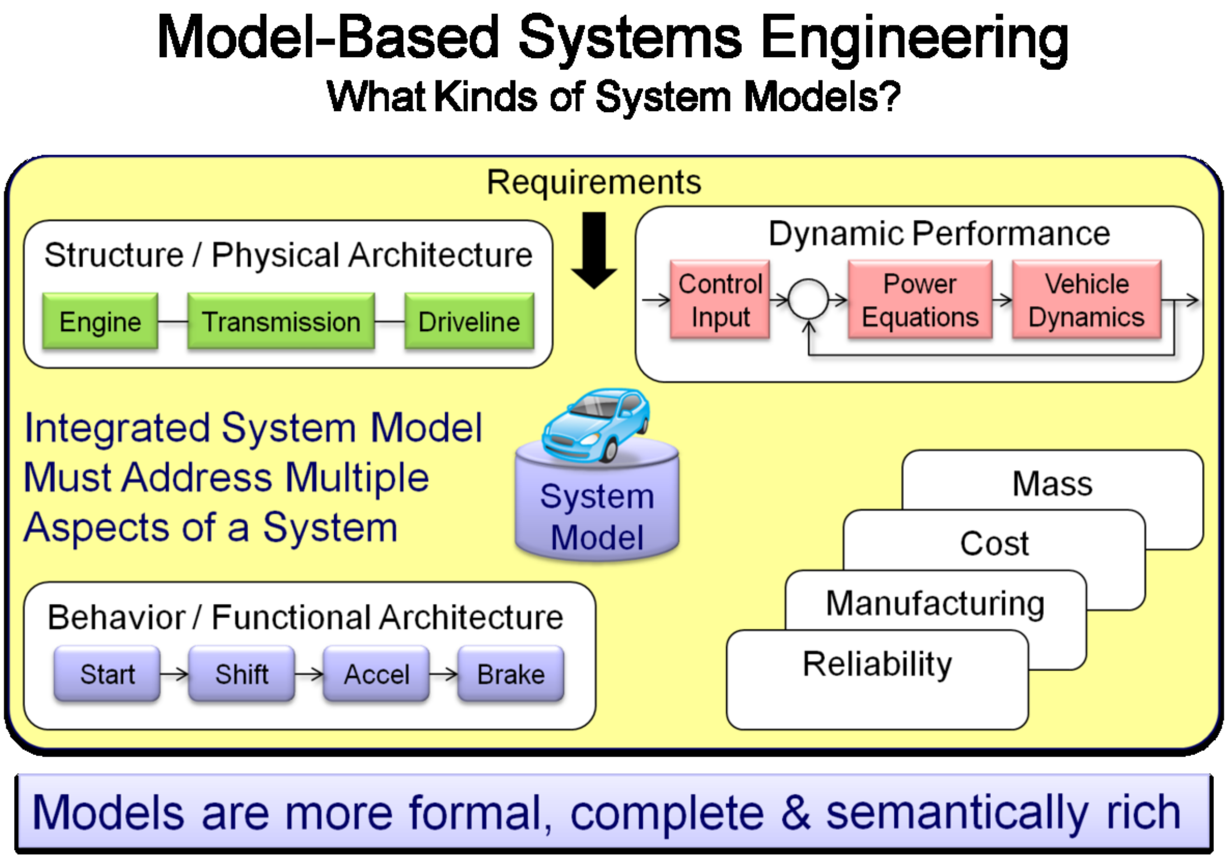

Model-Based Systems Engineering
What Kinds of System Models?
Requirements
Structure / Physical Architecture
Engine
Transmission
Driveline
Dynamic Performance
Control Input
Power Equations
Vehicle Dynamics
Integrated System Model Must Address Multiple Aspects of a System
System Model
Mass
Cost
Manufacturing
Reliability
Behavior / Functional Architecture
Start
Shift
Accel
Brake
Models are more formal, complete & semantically rich

Figure 1. Model-Based Systems Engineering (Paredis 2011). Reprinted with permission of Chris Paredis from Georgia Tech. All other rights are reserved by the copyright owner.

Descriptive Models

A descriptive model describes logical relationships, such as the system's whole-part relationship that defines its parts tree, the interconnection between its parts, the functions that its components perform, or the test cases that are used to verify the system requirements. Typical descriptive models may include those that describe the functional or physical architecture of a system, or the three dimensional geometric representation of a system.

Analytical Models

An analytical model (glossary) describes mathematical relationships, such as differential equations that support quantifiable analysis about the system parameters. Analytical models can be further classified into dynamic and static models. Dynamic models describe the time-varying state of a system, whereas static models perform computations that do not represent the time-varying state of a system. A dynamic model may represent the performance of a system, such as the aircraft position, velocity, acceleration, and fuel consumption over time. A static model may represent the mass properties estimate or reliability prediction of a system or component.

Hybrid Descriptive and Analytical Models

A particular model may include descriptive and analytical aspects as described above, but models may favor one aspect or the other. The logical relationships of a descriptive model can also be analyzed, and inferences can be made to reason about the system. Nevertheless, logical analysis provides different insights than a quantitative analysis of system parameters.

Domain-specific Models

Both descriptive and analytical models can be further classified according to the domain that they represent. The following classifications are partially derived from the presentation on *OWL, Ontologies and SysML Profiles: Knowledge Representation and Modeling* (Web Ontology Language (OWL) & Systems Modeling Language (SysML)) (Jenkins 2010):

- properties of the system, such as performance, reliability, mass properties, power, structural, or thermal models;
- design and technology implementations, such as electrical, mechanical, and software design models;
- subsystems and products, such as communications, fault management, or power distribution models; and
- system applications, such as information systems, automotive systems, aerospace systems, or medical device models.

The model classification, terminology and approach is often adapted to a particular application domain. For example, when modeling organization or business, the behavioral model may be referred to as workflow or process model, and the performance modeling may refer to the cost and schedule performance associated with the organization or business process.

A single model may include multiple domain categories from the above list. For example, a reliability, thermal, and/or power model may be defined for an electrical design of a communications subsystem for an aerospace system, such as an aircraft or satellite.

System Models

System models can be hybrid models that are both descriptive and analytical. They often span several modeling domains that must be integrated to ensure a consistent and cohesive system representation. As such, the system model must provide both general-purpose system constructs and domain-specific constructs that are shared across modeling domains. A system model may comprise multiple views to support planning, requirements, design, analysis, and verification.

Wayne Wymore is credited with one of the early efforts to formally define a system model using a mathematical framework in *A Mathematical Theory of Systems Engineering: The Elements* (Wymore 1967). Wymore established a rigorous mathematical framework for designing systems in a model-based context. A summary of his work can be found in A Survey of Model-Based Systems Engineering (MBSE) Methodologies.

Simulation versus Model

The term simulation, or more specifically computer simulation, refers to a method for implementing a model over time (DoD 1998). The computer simulation includes the analytical model which is represented in executable code, the input conditions and other input data, and the computing infrastructure. The computing infrastructure includes the computational engine needed to execute the model, as well as input and output devices. The great variety of approaches to computer simulation is apparent from the choices that the designer of computer simulation must make, which include

- stochastic or deterministic;
- steady-state or dynamic;
- continuous or discrete; and

- local or distributed.

Other classifications of a simulation may depend on the type of model that is being simulated. One example is an agent-based simulation that simulates the interaction among autonomous agents to predict complex emergent behavior (Barry 2009). They are many other types of models that could be used to further classify simulations. In general, simulations provide a means for analyzing complex dynamic behavior of systems, software, hardware, people, and physical phenomena.

Simulations are often integrated with the actual hardware, software, and operators of the system to evaluate how actual components and users of the system perform in a simulated environment. Within the United States defense community, it is common to refer to simulations as live, virtual, or constructive, where live simulation refers to live operators operating real systems, virtual simulation refers to live operators operating simulated systems, and constructive simulations refers to simulated operators operating with simulated systems. The virtual and constructive simulations may also include actual system hardware and software in the loop as well as stimulus from a real systems environment.

In addition to representing the system and its environment, the simulation must provide efficient computational methods for solving the equations. Simulations may be required to operate in real time, particularly if there is an operator in the loop. Other simulations may be required to operate much faster than real time and perform thousands of simulation runs to provide statistically valid simulation results. Several computational and other simulation methods are described in Simulation Modeling and Analysis (Law 2007).

Visualization

Computer simulation results and other analytical results often need to be processed so they can be presented to the users in a meaningful way. Visualization techniques and tools are used to display the results in various visual forms, such as a simple plot of the state of the system versus time to display a parametric relationship. Another example of this occurs when the input and output values from several simulation executions are displayed on a response surface showing the sensitivity of the output to the input. Additional statistical analysis of the results may be performed to provide probability distributions for selected parameter values. Animation is often used to provide a virtual representation of the system and its dynamic behavior. For example, animation can display an aircraft's three-dimensional position and orientation as a function of time, as well as project the aircraft's path on the surface of the Earth as represented by detailed terrain maps.

Integration of Models

Many different types of models may be developed as artifacts of a MBSE effort. Many other domain-specific models are created for component design and analysis. The different descriptive and analytical models must be integrated in order to fully realize the benefits of a model-based approach. The role of MBSE as the models integrate across multiple domains is a primary theme in the International Council on Systems Engineering (INCOSE) INCOSE Systems Engineering Vision 2020 (INCOSE 2007).

As an example, system models can be used to specify the components of the system. The descriptive model of the system architecture may be used to identify and partition the components of the system and define their interconnection or other relationships. Analytical models for performance, physical, and other quality characteristics, such as reliability, may be employed to determine the required values for specific component properties to satisfy the system requirements. An executable system model that represents the interaction of the system components may be used to validate that the component requirements can satisfy the system behavioral requirements. The descriptive, analytical, and executable system model each represent different facets of the same system.

The component designs must satisfy the component requirements that are specified by the system models. As a result, the component design and analysis models must have some level of integration to ensure that the design model is traceable to the requirements model. The different design disciplines for electrical, mechanical, and software each create their own models representing different facets of the same system. It is evident that the different models must be sufficiently integrated to ensure a cohesive system solution.

To support the integration, the models must establish semantic interoperability to ensure that a construct in one model has the same meaning as a corresponding construct in another model. This information must also be exchanged between modeling tools.

One approach to semantic interoperability is to use model transformations between different models. Transformations are defined which establish correspondence between the concepts in one model and the concepts in another. In addition to establishing correspondence, the tools must have a means to exchange the model data and share the transformation information. There are multiple means for exchanging data between tools, including file exchange, use of application program interfaces (API), and a shared repository.

The use of modeling standards for modeling languages, model transformations, and data exchange is an important enabler of integration across modeling domains.

References

Works Cited

Barry, P.S., M.T.K. Koehler, and B.F. Tivnan. 2009. *Agent-Directed Simulation for Systems Engineering.* McLean, VA: MITRE, March 2009, PR# 09-0267.

DoD. 1998. "'DoD Modeling and Simulation (M&S) Glossary" in *DoD Manual 5000.59-M*. Arlington, VA, USA: US Department of Defense. January 1998.

Wymore, A. 1967. *A Mathematical Theory of Systems Engineering: The Elements*. New York, NY, USA: John Wiley.

Wymore, A. 1993. *Model-Based Systems Engineering*. Boca Raton, FL, USA: CRC Press.

Primary References

Law, A. 2007. *Simulation Modeling and Analysis*, 4th ed. New York, NY, USA: McGraw Hill.

Wymore, A. 1993. *Model-Based Systems Engineering*. Boca Raton, FL, USA: CRC Press.

Additional References

Estefan, J. 2008. *Survey of Candidate Model-Based Systems Engineering (MBSE) Methodologies,* Revision B. Pasadena, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TD-2007-003-02.

Hybertson, D. 2009. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boca Raton, FL, USA: Auerbach/CRC Press.

INCOSE. 2007. *Systems Engineering Vision 2020.* Seattle, WA, USA: International Council on Systems Engineering. September 2007. INCOSE-TP-2004-004-02.

Rouquette, N. and S. Jenkins. 2010. *OWL Ontologies and SysML Profiles: Knowledge Representation and Modeling.* Proceedings of the NASA-ESA PDE Workshop, June 2010.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

System Modeling Concepts

Lead Author: *Sanford Friedenthal*, ***Contributing Authors:*** *Dov Dori, Yaniv Mordecai*

A system model represents aspects of a system and its environment. There are many different types of models, as there a variety of purposes for which they are built. It is useful to have a common way to talk about the concepts underlying the many different types of models (e.g., many modeling techniques enable the understanding of system behavior, while others enable the understanding of system structure). This article highlights several concepts used for modeling systems.

Abstraction

Perhaps the most fundamental concept in systems modeling is abstraction, which concerns hiding unimportant details in order to focus on essential characteristics. Systems that are worth modeling have too many details for all of them to reasonably be modeled. Apart from the sheer size and structural complexity that a system may possess, a system may be behaviorally complex as well, with emergent properties, non-deterministic behavior, and other difficult-to-characterize properties. Consequently, models must focus on a few vital characteristics in order to be computationally and intellectually tractable. Modeling techniques address this complexity through various forms of abstraction. For example, a model may assume that structural characteristics of many individual components of a particular type are all the same, ignoring the small order differences between individuals in instances that occur in real life. In that case, those differences are assumed to be unimportant to modeling the structural integrity of those components. Of course, if that assumption is wrong, then the model could lead to false confidence in that structural integrity. There are two key concepts that are applied in regard to modeling different levels of abstraction, which are: view and viewpoint and black-box and white-box modeling, which are described below. Although these two modeling methods are the most widely recognized, different modeling languages and tools employ other techniques as well.

View and Viewpoint

IEEE 1471, a standard for architecture modeling, defines "view" and "viewpoint" as follows:

- view - A representation of a whole system from the perspective of a related set of concerns.
- viewpoint - A specification of the conventions necessary for constructing and using a view; a pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis.

Even though IEEE 1471 is focused on architecture models, the concepts of view and viewpoint are general and could apply to models for other purposes as well (IEEE 2000). The viewpoint addresses the concerns of the stakeholders and provides the necessary conventions for constructing a view to address those concerns; therefore, the view represents aspects of the system that address the concerns of the stakeholder. Models can be created to represent the different views of the system. A systems model should be able to represent multiple views of the system to address a range of stakeholder concerns. Standard views may include requirements, functional, structural, and parametric views, as well as a multitude of discipline-specific views to address system reliability, safety, security, and other quality characteristics.

Black-Box and White-Box Models

A very common abstraction technique is to model the system as a black-box, which only exposes the features of the system that are visible from an external observer and hides the internal details of the design. This includes externally visible behavior and other physical characteristics, such as the system’s mass or weight. A white-box model of a system, on the other hand, shows the internal structure and displays the behavior of the system. Black-box and white-box modeling can be applied to the next level of design decomposition in order to create a black-box and white-box model of each system component.

Conceptual Model

A conceptual model is the set of concepts within a system and the relationships among those concepts (e.g., view and viewpoint). A system conceptual model describes, using one diagram type (such as in Object-Process Methodology (OPM)) or several diagram types (such as in Systems Modeling Language (SysML)) the various aspects of the system. The conceptual model might include its requirements, behavior, structure, and properties. In addition, a system conceptual model is accompanied by a set of definitions for each concept. Sometimes, system concept models are defined using an entity relationship diagram, an object-process diagram (OPD), or a Unified Modeling Language (UML) class diagram.

A preliminary conceptual (or concept) model for systems engineering (Systems Engineering Concept Model) was developed in support of the integration efforts directed toward the development of the Object Management Group (OMG) SysML and the International Organization for Standardization (ISO) AP233 *Data Exchange Standard for Systems Engineering* (ISO 2010). The concept model was originally captured in an informal manner; however, the model and associated concepts were rigorously reviewed by a broad representation of the systems engineering community, including members from the International Council on Systems Engineering (INCOSE), AP233, and SysML development teams.

A fragment from the top level systems engineering concept model is included in Figure 1. This model provides concepts for requirements, behavior, structure and properties of the system, as well as other concepts common to systems engineering and project management, such as stakeholder. The concept model is augmented by a well-defined glossary of terms called the semantic dictionary. The concept model and the semantic dictionary contributed greatly to the requirements for the OMG Systems Modeling Language written in the UML for Systems Engineering Request for Proposal.

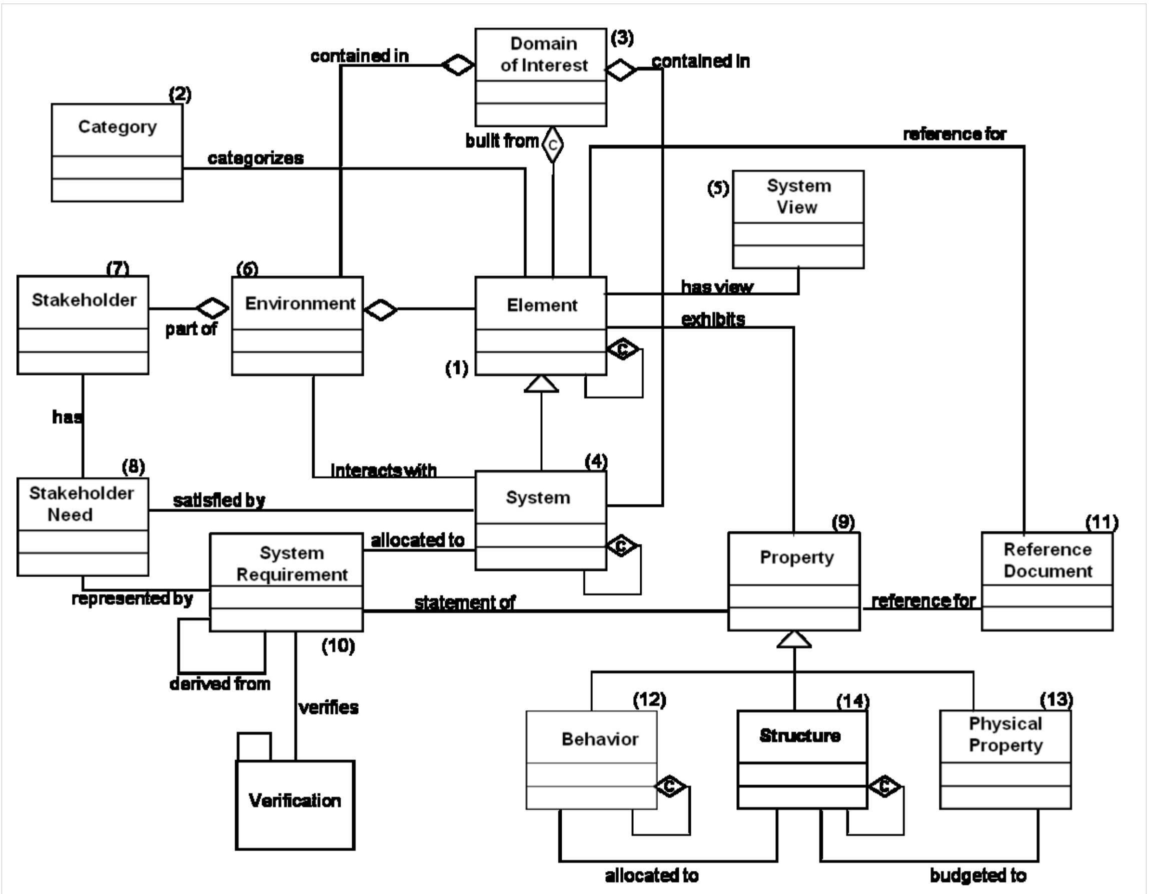


Figure 1. Fragment of the Object Management Group System Concept Model (Oliver 2003, Slide 3). Permission granted by David Oliver on behalf of INCOSE MDSD Working Group. All other rights are reserved by the copyright owner.

A concept model is sometimes referred to as a meta-model, domain meta-model, or schema, and can be used to specify the abstract syntax of a modeling language (refer to the Model Driven Architecture (MDA®) Foundation Model (OMG 2010)). Several other systems engineering concept models have been developed but not standardized. Future standardization efforts should establish a standard systems engineering concept model. The model can then evolve over time as the systems engineering community continues to formalize and advance the practice of systems engineering.

References

Works Cited

IEEE. 2000. *Recommended practice for architectural description for software-intensive systems*. New York, NY: Institute of Electrical and Electronics Engineers (IEEE), IEEE 1471-2000.

ISO. 2010. *OMG System Modeling Language (OMG SysML),* version 1.2. Needham, MA, USA: Object Management Group.

OMG. 2010. *MDA Foundation Model*. Needham, MA, USA: Object Management Group. Document number ORMSC/2010-09-06.

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.

Dori, D. 2002. *Object-Process Methodology – A Holistic Systems Paradigm*. New York, NY, USA: Springer-Verlag.

Guizzardi, G. 2007. "On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models". Proceeding of the 2007 Conference on Databases and Information Systems IV. Available at http://portal.acm.org/citation.cfm?id=1565425.

IEEE. 2000. *Recommended practice for architectural description for software-intensive systems*. New York, NY: Institute of Electrical and Electronics Engineers (IEEE), IEEE 1471-2000.

INCOSE. 2003. *Systems Engineering Concept Model. Draft 12 Baseline.* Seattle, WA: *International Council on Systems Engineering*. Available at http://syseng.omg.org/SE_Conceptual%20Model/SE_Conceptual_Model.htm.

OMG. 2003. *UML for Systems Engineering Request for Proposal*. Needham, MA: Object Management Group. OMG document number ad/2003-3-41. Available at http://www.omg.org/cgi-bin/doc?ad/2003-3-41.

Additional References

None

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Integrating Supporting Aspects into System Models

Lead Author: *Sanford Friedenthal*, ***Contributing Authors:*** *Dov Dori, Yaniv Mordecai*

This article discusses the integrated modeling of systems and supporting aspects using Model-Based Systems Engineering methodologies and frameworks. Supporting aspects of systems engineering include:

- Engineering Management
- Project Management
- Requirements Engineering and Management
- Risk Modeling, Analysis, and Management
- Quality Assurance, Testing, Verification, and Validation
- System Integration and Employment
- Analysis of "-ilities" (e.g., Reliability, Availability, Maintainability, Safety, And Security (RAMSS), Manufacturability, Extensibility, Robustness, Resilience, Flexibility, and Evolvability)

These aspects can pertain to physical facets, as well as to functional, structural, behavioral, social, and environmental facets of the core system model. The article focuses on three main aspects:

1. Project and Engineering Management
2. Risk Modeling, Analysis, and Management
3. Requirements Definition and Management

Background

The model-based approach to systems engineering considers the system model as much more than a plain description of the system; the model is the central common basis for capturing, representing, and integrating the various system aspects listed above. The model is essential to the design and understanding of the system as well as to managing its life cycle and evolution. Modeling languages constitute the basis for standardized, formal descriptions of systems, just like natural languages form the basis for human communication.

As systems progressively become more complex and multidisciplinary, the conceptual modeling of systems needs to evolve and will become more critical for understanding complex design (Dori 2002). In addition to facilitating communication among clients, designers, and developers, conceptual modeling languages also assist in clearly describing and documenting various domains, systems, and problems, and define requirements and constraints for the design and development phases (Wand and Weber 2002). The importance of model-based analysis is demonstrated by the variety of conceptual modeling methodologies and frameworks, although *de facto* standards are slow to emerge. While certain disciplines of engineering design, such as structural analysis or circuit design, have established modeling semantics and notation, the conceptual modeling of complex systems and processes has not yet converged on a unified, consolidated modeling framework (Estefan 2007). The challenge is not only to integrate multiple aspects and support the various phases of the system's life cycle, but also to capture the multidisciplinary nature of the system, which has led to the creation of various frameworks. Nevertheless, the information systems analysis paradigm is currently the most widely used, perhaps due to the need to integrate complex systems via information-intensive applications and interactions.

Integrated Modeling of Systems and Projects

This section discusses the integrated modeling of systems and projects and of the project-system relationship (often called Project-Product Integration). The fields of project management and systems engineering have been advancing hand-in-hand for the last two decades, due to the understanding that successful projects create successful systems. Many of the main systems engineering resources pay considerable attention to project management and consider it to be a critical process and enabler of systems engineering (INCOSE 2012; NASA 2007; Sage and Rouse 2011). The integration of system-related aspects and concepts into project plans is more common than the integration of project-related aspects and concepts into system models. Because the project is a means to an end, it is the process that is expected to deliver the system. Indeed, project activities are often named after or in accord with the deliverables that they are aimed at facilitating (e.g., "console design", "software development", "hardware acquisition", or "vehicle assembly"). Each is a function name, consisting of an object (noun), or the system to be attained, and a process (verb), being the project or part of the project aimed at attaining the end system.

The specific process associated with each of these examples refers to different stages or phases of the project and to different maturity levels of the system or sub-system it applies to. Moreover, the mere inclusion of system and part names in activity names does not truly associate system model artifacts with these activities. Overall, it is not truly possible to derive the set of activities associated with a particular part or functionality of the system which that will be delivered by the project. Project-Product integration is not straightforward, as project models and system models are traditionally disparate and hardly interface. A model-based approach to project-system integration follows a system-centric paradigm and focuses on incorporating project-related aspects and concepts into the core system model, as opposed to the project-centric approach described in the previous paragraph. Such aspects and concepts include schedule, budget and resources, deliverables, work-packages, constraints and previous relations. The integrated system-project model should provide useful information on the mutual effects of project activities and system components and capabilities. Some examples of integrated system-project modeling include the following:

- The set of project activities associated with a particular system component, feature, or capability.
- The set of resources required for performing a task of designing or developing a particular component of the system.

- The team or subcontractor responsible for delivering each system component.
- The preexisting dependencies between activities of system components deployment.
- The cost associated with each system component, feature, or capability.
- The parts of the system negotiated for each delivery, deployment, build, release, or version.

The Work Breakdown Structure (WBS) is designed to support the division of the project scope (work content) amongst the individuals and organizations participating in the project (Golany and Shtub 2001). The WBS is traditionally organization or activity-oriented; however, one of its main cornerstones focuses on the deliverable, which corresponds to the system, sub-system, component, or a capability or feature of one or more of these. A deliverable-oriented WBS, in which the high-level elements correspond to primary sub-systems, is advocated, as it is likely to allow the WBS to be more product-oriented (Rad 1999). An integrated approach to project planning and system modeling (Sharon and Dori 2009) merges the system model with the project's WBS using Object-Process Methodology (OPM) (Dori 2002). The unified OPM model captures both the project activities and the system components and functionalities.

The Design Structure Matrix (DSM) is a common method for enhancing and analyzing the design of products and systems. DSMs can be component-based, task-based, parameter-based, or team-based (Browning 2001). A DSM for an OPM-based project-product model derives a hybrid DSM of project activities and system building blocks from the unified OPM model, accounting for dependencies between project activities and system components, as well as replacing the two monolithic and separate component-based and task-based DSM views (Sharon, De-Weck, and Dori 2012). The underlying OPM model assures model consistency and traceability. The integrated project-product OPM model includes both a diagram and an equivalent auto-generated textual description. The DSM derived from this model visualizes a dependency loop comprising both system components and project activities.

Another model-based approach (Demoly et al. 2010) employs System Modeling Language (SysML) in order to create various views that meet the needs of various system stakeholders, such as the project/process manager. The approach includes both product-oriented and process-oriented views.

Integrated Modeling of Systems and Requirements

Requirements are statements that describe operational, functional, or design-related aspects of a system. Requirements definition and management is an important SE process, as it both initiates and facilitates the entire SE effort by defining the expected functions and performance of the engineered system. Several challenges associated with requirements include:

- Defining the requirements in a structured, controlled manner.
- Tracing these requirements to system components, aspects, and decisions.
- Testing and verifying compliance of the system with these requirements.

The extension of conceptual system models to include requirements has several significant benefits:

1. Requirements provide the rationale for the system's architecture and design by making and justifying architectural and design decisions based on specific requirements.
2. Modeling the internal logic and the hierarchy and dependency relations among requirements enables identification and elimination of redundant and contradictory requirements.
3. Responsibility for satisfying specific requirements can often be assigned to teams and persons responsible for delivering various system components. While the advantages of having good requirements engineering is clear, it is often a challenge to directly trace requirements to specific system artifacts, especially when the requirements are defined in a holistic, solution-independent manner.

There are several methods to incorporate requirements into system models, including SysML Requirements Engineering and Object-Process Methodology(OPM)-based Requirements Engineering and Authoring.

SysML-Based Requirements Engineering

The SysML requirements diagram makes it possible to capture the requirements and the relations among them in a visual manner, which is more intuitive than the textual manner in which requirements are traditionally edited and managed. The diagram was added to the basic set of UML diagrams that formed the basis for SysML (Friedenthal, Moore, and Steiner 2006), and is not a native UML diagram. Tracing requirements to the system blocks and artifacts satisfying them can be captured in the SysML Block Definition Diagram, which is primarily designated to capture the relations among types of system elements and components. The < > link between the block and the requirement captures the trace.

OPM-Based Requirements Engineering

Object-Process Methodology (OPM) is a methodology and language for conceptual modeling of complex systems and processes with a bimodal textual and graphical representation (Dori 2002). OPM’s textual representation is coordinated with the graphical representation; additionally, each visual model construct in the Object-Process Diagram (OPD) is described by a formal structured textual statement in Object-Process Language (OPL), which is a subset of natural English. OPM facilitates model-based requirements engineering, authoring, and specification, in three possible modes:

1. OPM can be used to generate conceptual models which initially focus on the requirements level—the problem domain, rather than the design level or the solution domain, which facilitates automated model-based requirements generation (Blekhman and Dori 2011). The requirements model is solution-neutral and it can be the basis for one or more architectural solutions for achieving the functions specified in the requirements.
2. Utilizing OPM in order to generate requirement-oriented OPDs in a manner similar to the SysML Requirements Diagram enables an engineer to capture the requirements specification as the skeleton for the system model. User-defined tagged structural relations, such as "is realized by" or is allocated to", provide for associating requirements with system model functions (objects and processes that transform them). This approach is similar to the SysML requirements diagram; however, instead of using a unique notation in a separate diagram type, the requirements are seamlessly incorporated into the single system model.
3. OPM can be used for the purpose of generating visual system models from formally specified requirements by tracing the textually authored requirements to system model inserts and artifacts (Dori et al. 2004).

Integrating Risk into System Models

Risk is an expression and a measure of the negative or adverse impact of uncertainty. Risk exists whenever uncertainty can lead to several results, of which some may be negative (adverse) and some positive. A system faces risks from other systems or from the environment, and it can also pose risks to other systems or to the environment. Systems are characterized by such attributes, such as: goals, objectives, inputs, outputs, variables, parameters, processes, events, states, subsystems, interfaces, mechanisms, and methods. System vulnerability is the system's total potential to be harmed or negatively affected in any one of these attributes. Analogously, system harmfulness is the system's total potential to harm others or to generate negative effects, which can be manifested in one or more of these attributes (Haimes 2009). Model-based risk analysis (MBRA) enables structured analysis and risk-related process control. Several model-based risk analysis approaches are available in the literature. MBRA is presently common in the information technology and information security domains more than the systems engineering domain; however, some of the methods are generally applicable to complex systems as well. The ISO-IEC-IEEE collaborative software development and operation lifecycle standard (ISO and IEC 2004) proposes a concurrent approach to IT Risk Management. This approach consists of six main activities:

- Plan and Implement Risk Management
- Manage the Project Risk Profile
- Perform Risk Analysis

- Perform Risk Monitoring
- Perform Risk Treatment
- Evaluate the Risk Management Process

These activities are executed concurrently, affect and provide feedback to each other, and interact with other software life cycle processes, such as the technical management and the design processes (ISO and IEC 2004).

The CORAS approach (Fredriksen et al. 2002; den Braber et al. 2006; Lund, Solhaug, and Stølen 2011) is a UML-derivative for IT security risk modeling and assessment. This framework consists mostly of the UML use case (UC) diagram, extended for misuse cases. Additional notation was added to the UC notation in order to capture risk sources, effects, and results (e.g., the "bad actor" icon, moneybag for asset-in-risk). A misuse diagram can include, for example, the risk of loss of legal protection of proprietary know-how due to information theft and distribution by an unfaithful employee. The treatment for the risk source of insufficient security policy, which contributes to the above risk, is illustrated in a separate treatment diagram.

A quantitative risk assessment method for component-based systems (Grunske and Joyce 2008) supports component vulnerability analysis and specification using modular attack trees. In addition, it provides attacker profiling, which enables supporting econometric approaches to risk response. The methodology utilizes SysML as its underpinning language, and especially the SysML block definition diagram and parametric diagram, in order to capture parametric relations and constraints as a means to defining risk profiles.

System-Theoretic Accident Model and Processes (STAMP) is a method for system and component design for safety (Leveson 2011). STAMP reformulates the safety problem as a control problem as opposed to a reliability problem. STAMP is optimized for safety-oriented systems engineering and design and for hazard avoidance and mitigation, specifically in complex socio-technical systems. A model-based adaptation of STAMP was also proposed (Leveson 2004) and was implemented in various safety-critical and mission-critical systems, including aircraft collision avoidance systems (CAS) (Leveson 2004) and ballistic missile defense systems (Pereira, Lee, and Howard 2006). Risk-Oriented Systems Engineering (ROSE) (Mordecai and Dori 2013) is a method based on Object-Process Methodology (OPM) for integrating risk into system models. Being system-centric, ROSE is responsible for capturing risk layers and aspects on top of and in sync with the core system model, while improving and immunizing it against captured risks, as well as for generating system robustness and resilience by design in response to various risk-posing scenarios. The risk handling meta-model includes risk mitigation during the design phase and risk response during the operational phase.

References

Works Cited

Blekhman, A. and D. Dori. 2011. "Model-Based Requirements Authoring - Creating Explicit Specifications with OPM." In 6th International Conference on Systems Engineering. Herzeliyya, Israel.

Browning, T.R. 2001. "Applying the Design Structure Matrix to System Decomposition and Integration Problems: a Review and New Directions." IEEE Transactions on Engineering Management 48 (3): 292–306. doi:10.1109/17.946528. Accessed December 4 2014 at IEEE http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=946528.

Demoly, F., D. Monticolo, B. Eynard, L. Rivest, and S. Gomes. 2010. "Multiple Viewpoint Modelling Framework Enabling Integrated Product–process Design." International Journal on Interactive Design and Manufacturing (IJIDeM) 4 (4) (October 12): 269–280. doi:10.1007/s12008-010-0107-3. Accessed December 4 2014 at Springer http://link.springer.com/10.1007/s12008-010-0107-3.

Den Braber, F., G. Brændeland, H.E.I. Dahl, I. Engan, I. Hogganvik, M.S. Lund, B. Solhaug, K. Stølen, and F. Vraalsen. 2006. *The CORAS Model-based Method for Security Risk Analysis*. SINTEF, Oslo. Oslo: SINTEF.

Dori, D. 2002. *Object-Process Methodology – A Holistic Systems Paradigm*. New York, NY, USA: Springer-Verlag.

Dori, D., N. Korda, A. Soffer, and S. Cohen. 2004. "SMART: System Model Acquisition from Requirements Text." Lecture Notes in *Computer Science: Business Process Management* 3080: 179–194. Accessed December 4 2014 at Springer http://link.springer.com/chapter/10.1007/978-3-540-25970-1_12.

Estefan, J. 2008. *A Survey of Model-Based Systems Engineering (MBSE) Methodologies,* Rev. B. San Diego, CA, USA: International Council on Systems Engineering. INCOSE-TD-2007-003-02.

Friedenthal, S., A. Moore, and R. Steiner. 2006. "OMG Systems Modeling Language (OMG SysML™) Tutorial" (July).

Golany, B. and A. Shtub. 2001. "Work Breakdown Structure." In Salvendy, G. (ed.) 2001. *Handbook of Industrial Engineering, Technology and Operations Management,* 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, Inc. 1263–1280.

Grunske, L. and D. Joyce. 2008. "Quantitative Risk-based Security Prediction for Component-based Systems with Explicitly Modeled Attack Profiles." Journal of Systems and Software 81 (8): 1327–1345. Haimes, YY. 2009. "On the Complex Definition of Risk: A Systems-Based Approach." *Risk Analysis.* 29 (12): 1647–1654. Accessed December 4 2014 at Wiley http://onlinelibrary.wiley.com/doi/10.1111/j.1539-6924.2009.01310.x/full.

ISO/IEC/IEEE. 2004. *Systems and software engineering - Life cycle processes - Risk management*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC/IEEE 16085:2006.

Leveson, N.G. 2004. "Model-based Analysis of Socio-technical Risk." Cambridge, MA: Massachusetts Institute of Technology (MIT) Working Paper Series. ESD-WP-2004-08.

Leveson, N.G. 2011. *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA: MIT Press.

Lund, M.S., B. Solhaug, and K. Stølen. 2011. *Model-Driven Risk Analysis: The CORAS Approach*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-12323-8. Accessed December 4 2014 at Springer http://www.springerlink.com/index/10.1007/978-3-642-12323-8.

Mordecai, Y., and D. Dori. 2013. "Model-Based Risk-Oriented Robust Systems Design with Object-Process Methodology." *International Journal of Strategic Engineering Asset Management*. TBD (CESUN 2012 Special Issue).

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

Pereira, S.J., G. Lee, and J. Howard. 2006. "A System-Theoretic Hazard Analysis Methodology for a Non-advocate Safety Assessment of the Ballistic Missile Defense System". Vol. 1606. Accessed December 4 2014 at Defense Technical Information Center http:/ / oai. dtic. mil/ oai/ oai?verb=getRecord& metadataPrefix=html& identifier=ADA466864.

Rad, P.F. 1999. "Advocating a Deliverable-oriented Work Breakdown Structure." Sage, Andrew P., and William B. Rouse. 2011. *Handbook of Systems Engineering and Management*. Edited by A.P. Sage and W.B. Rouse. 2nd ed. John Wiley & Sons.

Sharon, A., O.L. de-Weck, and D. Dori. 2012. "Improving Project-Product Lifecycle Management with Model-Based Design Structure Matrix : A Joint Project Management and Systems Engineering Approach." *Systems Engineering*: 1–14. doi:10.1002/sys.

Sharon, A., and D. Dori. 2009. "A Model-Based Approach for Planning Work Breakdown Structures of Complex Systems Projects." In Proc. 14th IFAC Symposium on Information Control Problems in Manufacturing.

Wand, Y., and R. Weber. 2002. "Research Commentary: Information Systems and Conceptual Modeling?A Research Agenda." *Information Systems Research.* 13(4) (December): 363–376. doi:10.1287/isre.13.4.363.69.

Primary References

Dori, D. 2002. *Object-Process Methodology – A Holistic Systems Paradigm.* New York, NY, USA: Springer-Verlag.

Estefan, J. 2008. *A Survey of Model-Based Systems Engineering (MBSE) Methodologies,* Rev. B. San Diego, CA, USA: International Council on Systems Engineering. INCOSE-TD-2007-003-02.

Golany, B. and A. Shtub. 2001. "Work Breakdown Structure." In Salvendy, G. (ed.) 2001. *Handbook of Industrial Engineering, Technology and Operations Management,* 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, Inc. 1263–1280.

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.

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

Additional References

Kristiansen, B.G. and K. Stolen. 2002. "The CORAS Framework for a Model-based Risk Management Process." In Lecture Notes In, edited by Stuart Anderson, Massimo Felici, and SandroEditors Bologna, 2434:94–105. Springer-Verlag. doi:10.1007/3-540-45732-1_11.

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.1, released 31 October 2019

Modeling Standards

Lead Author: *Sanford Friedenthal*, ***Contributing Authors:*** *Dov Dori, Yaniv Mordecai*

Different types of models are needed to support the analysis, specification, design, and verification of systems. The evolution of modeling standards enables the broad adoption of Model-Based Systems Engineering (MBSE).

Motivation for Modeling Standards

Modeling standards play an important role in defining agreed-upon system modeling concepts that can be represented for a particular domain of interest and enable the integration of different types of models across domains of interest. Modeling standards are extremely important to support MBSE, which aims to integrate various system aspects across various disciplines, products, and technologies.

Standards for system modeling languages can enable cross-discipline, cross-project, and cross-organization communication. This communication offers the potential to reduce the training requirements for practitioners who only need to learn about a particular system and enables the reuse of system artifacts. Standard modeling languages also provide a common foundation for advancing the practice of systems engineering, as do other systems engineering standards.

Types of Modeling Standards

Many different standards apply to systems modeling. Modeling standards include standards for modeling languages, data exchange between models, and the transformation of one model to another to achieve semantic interoperability. Each type of model can be used to represent different aspects of a system, such as representing the set of system components and their interconnections and interfaces, or to represent a system to support performance analysis or reliability analysis.

The following is a partial list of representative modeling standards, which also includes the common acronym, when applicable, and a reference as to where additional information can be found on the topic.

Modeling Languages for Systems

Descriptive Models - These standards apply to general descriptive modeling of systems:

- Functional Flow Block Diagram (FFBD) (Oliver, Kelliher, and Keegan 1997)
- Integration Definition for Functional Modeling (IDEF0) (NIST 1993)
- Object-Process Methodology (OPM) [[1]] [[2]] (Dori 2002; ISO 19450 PAS - Publicly Available Specification in progress)
- Systems Modeling Language (SysML)(OMG 2010a)
- Unified Profile for United States Department of Defense Architecture Framework (DoDAF) and United Kingdom Ministry of Defense Architecture Framework (MODAF) (OMG 2011e)
- Web ontology language (OWL) (W3C 2004b)

Analytical Models and Simulations - These standards apply to analytical models and simulations:

- Distributed Interactive Simulation (DIS) (IEEE 1998)
- High-Level Architecture (HLA) (IEEE 2010)
- Modelica (Modelica Association 2010)
- Semantics of a Foundational Subset for Executable Unified Modeling Language (UML) Models (FUML) (OMG 2011d)

Data Exchange Standards

These standards enable the exchange of information between models:

- Application Protocol for Systems Engineering Data Exchange (ISO 10303-233) (AP-233) (ISO 2005)
- Requirements Interchange Format (ReqIF) (OMG 2011c)
- Extensible Mark-Up Language -(XML) Metadata Interchange (XMI) (OMG 2003a)
- Resource Description Framework (RDF) (W3C 2004a)

Model Transformations

These standards apply to transforming one model to another to support semantic interoperability:

- Query View Transformations (QVT) (OMG 2011b)
- Systems Modeling Language (SysML)-Modelica Transformation (OMG 2010c)
- OPM-to-SysML Transformation (Grobshtein and Dori 2011)

General Modeling Standards

These standards provide general frameworks for modeling:

- Model-driven architecture (MDA®) (OMG 2003b)
- IEEE 1471-2000 - Recommended Practice for Architectural Description of Software-Intensive Systems (ANSI/IEEE 2000) (ISO/IEC 2007)

Other Domain-Specific Modeling Standards

Software Design Models

These standards apply to modeling application software and/or embedded software design:

- Architecture Analysis and Design Language (AADL) (SAE 2009)
- Modeling and Analysis for Real-Time and Embedded Systems (MARTE) (OMG 2009)
- Unified Modeling Language (UML) (OMG 2010b)

Hardware Design Models

These standards apply to modeling hardware design:

- Very-High-Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) (IEEE 2008)

Business Process Models

These standards apply to modeling business processes:

- Business Process Modeling Notation (BPMN) (OMG 2011a)

References

Works Cited

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.

Grobshtein, Y. and D. Dori. 2011. "Generating SysML Views from an OPM Model: Design and Evaluation." *Systems Engineering*, 14 (3), Sept. 2011.

IEEE. 1998. *Distributed Interactive Simulation (DIS).* Washington, DC: Institute for Electrical and Electronic Engineers. IEEE 1278.1-1995. Accessed December 4 2014 at IEEE http://standards.ieee.org/develop/project/1278.2.html.

IEEE. 2008. *VHSIC hardware description language (VHDL).* Washington, DC: Institute of Electrical and Electronics Engineers. IEEE Standard 1076-2008. Accessed December 4 2014 at IEEE http://standards.ieee.org/findstds/standard/1076-2008.html.

IEEE. 2010. *Standard for High Level Architecture*. Washington, DC: Institute for Electrical and Electronic Engineers. IEEE Standard 1516. Accessed December 4 2014 at IEEE http://standards.ieee.org/develop/intl/intlstds.html

ISO. 2005. *Application Protocol for Systems Engineering Data Exchange*. Geneva, Switzerland: International Organization for Standardization. ISO 10303-233. Accessed December 4 2014 at ISO http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=55257.

ISO/IEC/IEEE. 2011. *Systems and Software Engineering — Architecture Description.* Geneva, Switzerland: International Organization for Standardization/International Electrotechnical Commission/Institute of Electrical and Electronics Engineers. December 1, 2011. ISO/IEC/IEEE 42010:2011. Accessed December 4 2014 at ISO http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=50508.

Modelica Association. 2010. *Modelica® - A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification, Version 3.2*. Modelica Association. Accessed December 4 2014 at Modelica https://www.modelica.org/documents/ModelicaSpec32.pdf.

NIST. 1993. *Integration Definition for Functional Modeling (IDEF0).* Gaithersburg, MD: National Institute for Standards and Technologies. Accessed December 4 2014 at IDEF http://www.idef.com/IDEF0.htm.

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

OMG 2003a. *XML Metadata Interchange (XMI),* Version 1.1. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/XML/.

OMG. 2003b. *Model driven architecture (MDA®),* Version 1.0.1. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/mda.

OMG. 2009. *Modeling and Analysis for Real-Time and Embedded Systems (MARTE),* Version 1.0. Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/MARTE/1.0/.

OMG. 2010a. *OMG Systems Modeling Language (SysML),* Version 1.2. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at SysML forum http:/ / www. sysml. org/ docs/ specs/ OMGSysML-v1. 2-10-06-02.pdf.

OMG. 2010b. *Unified Modeling Language™ (UML),* Version 2.. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/UML/.

OMG. 2010c. *SysML-Modelica Transformation Specification*, Beta Version. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/SyM/.

OMG. 2011a. *Business Process Modeling Notation (BPMN),* Version 2.0. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/BPMN/2.0/

OMG. 2011b. *Query View Transformations (QVT),* Version 1.1. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/QVT/1.1/.

OMG. 2011c. *Requirements Interchange Format (ReqIF),* Version 1.0.1. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/ReqIF/.

OMG. 2011d. *Semantics of a Foundational Subset for Executable UML Models (FUML),* Version 1.0. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/FUML/ 1.0/.

OMG. 2011e. *Unified Profile for DoDAF and MODAF (UPDM),* Version 1.1. Needham, MA, USA: Object Management Group. Accessed December 4 2014 at OMG http://www.omg.org/spec/UPDM/.

SAE. 2009. *Architecture Analysis & Design Language (AADL).* Warrendale, PA, USA: SAE International. Accessed December 4 2014 at Society of Automotive Engineers http://standards.sae.org/as5506a/.

W3C. 2004a. *Resource Description Framework (RDF),* Version 1.0. World Wide Web Consortium. Accessed December 4 2014 at World Wide Web Consortium http://www.w3.org/RDF/.

W3C. 2004b. *Web ontology language. (OWL).* World Wide Web Consortium. Accessed December 4 2014 at World Wide Web Consortium http://www.w3.org/2004/OWL.

Primary References

Dori, D. 2002. *Object-Process Methodology – A Holistic Systems Paradigm.* Berlin, Germany: Heidelberg; New York, NY, USA: Springer Verlag.

Friedenthal, S., A. Moore, R. Steiner, and M. Kaufman. 2012. *A Practical Guide to SysML: The Systems Modeling Language*, 2nd Edition. Needham, MA, USA: OMG Press.

Additional References

Fritzon, P. 2004. *Object-oriented modeling and simulation with Modelica 2.1.* New York, NY, USA: Wiley Interscience and IEEE Press.

Bibliowicz, A. and D. Dori. *A Graph Grammar-Based Formal Validation of Object-Process Diagrams*. Software and Systems Modeling, 11, (2) pp. 287-302, 2012.

Blekhman, A. and D. Dori. "Model-Based Requirements Authoring." INCOSE 2011 – the 6th International conference on System Engineering. March, 2011.

Dori, D., R. Feldman, and A. Sturm. *From conceptual models to schemata: An object-process-based data warehouse construction method."* Information Systems. *33: 567–593, 2008.*

Osorio, C.A., D. Dori, and J. Sussman. *COIM: An Object-Process Based Method for Analyzing Architectures of Complex, Interconnected, Large-Scale Socio-Technical Systems*. Systems Engineering 14(3), 2011.

Paredis, C.J.J., Y. Bernard, R.M. Burkhart, H-P. de Koning, S. Friedenthal, P. Fritzson, N.F. Rouquette, W. Schamai. 2010. "An overview of the SysML-modelica transformation specification". Proceedings of the 20th Annual International Council on Systems Engineering (INCOSE) International Symposium, 12-15 July 2010, Chicago, IL.

Reinhartz-Berger, I. and D. Dori. "OPM vs. UML—Experimenting with Comprehension and Construction of Web Application Models." *Empirical Software Engineering*, 10: 57–79, 2005.

Weilkiens, T. 2008. *Systems Engineering with SysML/UML*. Needham, MA, USA: OMG Press.

< Previous Article | Parent Article | Next Article (Part 3) >

SEBoK v. 2.1, released 31 October 2019

References

[1] http://www.amazon.com/gp/product/3540654712/sr=8-1/qid=1146053424/ref=sr_1_1/104-2484506-3323967?_encoding=UTF8

[2] http://esml.iem.technion.ac.il/?page_id=874

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

Foundations of Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=57089 *Contributors*: Alee, Apyster, Asquires, Bkcase, Cdagli, Cnielsen, Dcarey, Ddori, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Mhaas, Radcock, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

Systems Fundamentals *Source*: https://www.sebokwiki.org/d/index.php?oldid=57437 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Janthony, Jdahmann, Mhaas, Radcock, Rmalove, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert

Introduction to System Fundamentals *Source*: https://www.sebokwiki.org/d/index.php?oldid=56795 *Contributors*: Afaisandier, Alee, Apyster, Asquires, Bkcase, Ccalvano, Cnielsen, Ddori, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Radcock, Smenck2, Smurawski, WikiWorks753, Wikiexpert

Types of Systems *Source*: https://www.sebokwiki.org/d/index.php?oldid=57379 *Contributors*: Apyster, Asquires, Bkcase, Bwells, Ccalvano, Cnielsen, Dhenry, Dholwell, Janthony, Jdahmann, Jgercken, Jsnoderly, Mhaas, Radcock, Rmalove, Skmackin, Smenck2, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

Complexity *Source*: https://www.sebokwiki.org/d/index.php?oldid=57277 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dhenry, Dholwell, Janthony, Jgercken, Jsnoderly, Mhaas, Radcock, Rmalove, Skmackin, Smenck2, WikiWorks753, Wikiexpert, Zamoses

Emergence *Source*: https://www.sebokwiki.org/d/index.php?oldid=57278 *Contributors*: Apyster, Bkcase, Cnielsen, Dfairley, Dhenry, Janthony, Jgercken, Mhaas, Radcock, Rmalove, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

Fundamentals for Future Systems Engineering *Source*: https://www.sebokwiki.org/d/index.php?oldid=57279 *Contributors*: Bkcase, Mhaas, Radcock, WikiWorks753

Systems Approach Applied to Engineered Systems *Source*: https://www.sebokwiki.org/d/index.php?oldid=57310 *Contributors*: Apyster, Bkcase, Blawson, Cnielsen, Dcarey, Dhenry, Dholwell, Eleach, Gparnell, Janthony, Jgercken, Mhaas, Radcock, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

Overview of the Systems Approach *Source*: https://www.sebokwiki.org/d/index.php?oldid=57096 *Contributors*: Apyster, Bkcase, Blawson, Bwells, Cnielsen, Dcarey, Dhenry, Eleach, Janthony, Jgercken, Mhaas, Radcock, Sfriedenthal, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

Engineered System Context *Source*: https://www.sebokwiki.org/d/index.php?oldid=57097 *Contributors*: Apyster, Asquires, Bkcase, Bwells, Ccalvano, Cnielsen, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Mhaas, Radcock, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

Identifying and Understanding Problems and Opportunities *Source*: https://www.sebokwiki.org/d/index.php?oldid=57098 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Mhaas, Radcock, Rturner, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

Synthesizing Possible Solutions *Source*: https://www.sebokwiki.org/d/index.php?oldid=57322 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Mhaas, Radcock, Rturner, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

Analysis and Selection between Alternative Solutions *Source*: https://www.sebokwiki.org/d/index.php?oldid=57100 *Contributors*: Afaisandier, Apyster, Bkcase, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Mhaas, Radcock, Rturner, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

Implementing and Proving a Solution *Source*: https://www.sebokwiki.org/d/index.php?oldid=57101 *Contributors*: Apyster, Bkcase, Bwells, Cnielsen, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Mhaas, Radcock, Sjackson, Skmackin, Smenck2, WikiWorks753, Wikiexpert, Zamoses

Deploying, Using, and Sustaining Systems to Solve Problems *Source*: https://www.sebokwiki.org/d/index.php?oldid=57102 *Contributors*: Apyster, Bkcase, Bwells, Cnielsen, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Mhaas, Radcock, Rturner, Sfriedenthal, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

Applying the Systems Approach *Source*: https://www.sebokwiki.org/d/index.php?oldid=57103 *Contributors*: Alee, Apyster, Araher, Bkcase, Cnielsen, Dhenry, Dholwell, Hdavidz, Janthony, Jgercken, Mhaas, Radcock, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

Systems Science *Source*: https://www.sebokwiki.org/d/index.php?oldid=57104 *Contributors*: Apyster, Asquires, Bkcase, Blawson, Cnielsen, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Mhaas, Radcock, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

History of Systems Science *Source*: https://www.sebokwiki.org/d/index.php?oldid=57105 *Contributors*: Apyster, Asquires, Bkcase, Ccalvano, Cnielsen, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Mhaas, Qwang, Radcock, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

Systems Approaches *Source*: https://www.sebokwiki.org/d/index.php?oldid=57443 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Dholwell, Eleach, Janthony, Mhaas, Radcock, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert

Systems Thinking *Source*: https://www.sebokwiki.org/d/index.php?oldid=57510 *Contributors*: Apyster, Asquires, Bkcase, Blawson, Ccalvano, Cnielsen, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Mhaas, Radcock, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

What is Systems Thinking? *Source*: https://www.sebokwiki.org/d/index.php?oldid=57447 *Contributors*: Apyster, Asquires, Bkcase, Blawson, Bwells, Ccalvano, Cnielsen, Dhenry, Eleach, Janthony, Jgercken, Mhaas, Radcock, Rturner, Sjackson, Smenck2, WikiWorks, WikiWorks753, Wikiexpert

Concepts of Systems Thinking *Source*: https://www.sebokwiki.org/d/index.php?oldid=57111 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dhenry, Dholwell, Janthony, Jgercken, Mhaas, Radcock, Rmalove, Sfriedenthal, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

Principles of Systems Thinking *Source*: https://www.sebokwiki.org/d/index.php?oldid=57112 *Contributors*: Bkcase, Cnielsen, Dhenry, Dholwell, Dhybertson, Janthony, Mhaas, Radcock, Rmalove, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert

Patterns of Systems Thinking *Source*: https://www.sebokwiki.org/d/index.php?oldid=57113 *Contributors*: Bkcase, Cnielsen, Dhenry, Eleach, Janthony, Mhaas, Radcock, Smenck2, Smurawski, WikiWorks, WikiWorks753

Representing Systems with Models *Source*: https://www.sebokwiki.org/d/index.php?oldid=57116 *Contributors*: Apyster, Bkcase, Cnielsen, Ddori, Dhenry, Dholwell, Eleach, Gparnell, Janthony, Jgercken, Mhaas, Radcock, Sfriedenthal, Sjackson, Skmackin, Smenck2, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

What is a Model? *Source*: https://www.sebokwiki.org/d/index.php?oldid=57301 *Contributors*: Alee, Apyster, Araher, Bkcase, Cnielsen, Ddori, Dhenry, Gparnell, Janthony, Jgercken, Mhaas, Radcock, Rcloutier, Sfriedenthal, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

Why Model? *Source*: https://www.sebokwiki.org/d/index.php?oldid=57449 *Contributors*: Alee, Apyster, Araher, Bkcase, Cnielsen, Dhenry, Dholwell, Janthony, Jgercken, Mhaas, Radcock, Sfriedenthal, Skmackin, Smenck2, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

Types of Models *Source*: https://www.sebokwiki.org/d/index.php?oldid=57296 *Contributors*: Apyster, Araher, Asquires, Bkcase, Cnielsen, Dhenry, Janthony, Jgercken, Mhaas, Radcock, Sfriedenthal, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

System Modeling Concepts *Source*: https://www.sebokwiki.org/d/index.php?oldid=57506 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Ddori, Dhenry, Eleach, Janthony, Jgercken, Mhaas, Radcock, Sfriedenthal, Sjackson, Skmackin, Smenck2, Smurawski, WikiWorks, WikiWorks753, Wikiexpert, Zamoses

Integrating Supporting Aspects into System Models *Source*: https://www.sebokwiki.org/d/index.php?oldid=57121 *Contributors*: Bkcase, Cnielsen, Eleach, Mhaas, Smenck2, WikiWorks753, Ymordecai

Modeling Standards *Source*: https://www.sebokwiki.org/d/index.php?oldid=57122 *Contributors*: Apyster, Bkcase, Cnielsen, Dcarey, Ddori, Dhenry, Eleach, Janthony, Jgercken, Mhaas, Radcock, Sfriedenthal, Skmackin, Smenck2, Smurawski, WikiWorks753, Wikiexpert, Zamoses

Image Sources, Licenses and Contributors