System Architecture

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

The purpose of system architecture activities is to define a comprehensive solution based on principles, concepts, and properties logically related to and consistent with each other. The solution architecture has features, properties, and characteristics which satisfy, as far as possible, the problem or opportunity expressed by a set of system requirements (traceable to mission/business and stakeholder requirements) and life cycle concepts (e.g., operational, support) and which are implementable through technologies (e.g., mechanics, electronics, hydraulics, software, services, procedures, human activity).

System Architecture is abstract, conceptualization-oriented, global, and focused to achieve the mission and life cycle concepts of the system. It also focuses on high-level structure in systems and system elements. It addresses the architectural principles, concepts, properties, and characteristics of the system-of-interest. It may also be applied to more than one system, in some cases forming the common structure, pattern, and set of requirements for classes or families of similar or related systems.

☐

Contents

General Concepts and Principles

Notion of Structure

The SEBoK considers systems engineering to cover all aspects of the creation of a system, including system architecture.

The majority of interpretations of system architecture are based on the fairly intangible notion of *structure* (i.e. relationships between elements). Some authors limit the types of structure considered to be architectural; for example, restricting themselves to *functional* and

physical structure. Recent practice has extended consideration to include *behavioral*, *temporal* and other dimensions of structure.

ISO/IEC/IEEE 42010 *Systems and Software Engineering - Architecture Description* (ISO 2011) provides a useful description of the architecture considering the stakeholder concerns, architecture viewpoints, architecture views, architecture models, architecture descriptions, and architecting throughout the life cycle.

A discussion of the features of systems architectures can be found in (Maier and Rechtin 2009).

An attempt to develop and apply a systematic approach to characterizing architecture belief systems in systems engineering has been described by the INCOSE UK Architecture Working Group (Wilkinson et al. 2010, Wilkinson 2010).

Architecture Description of the System

An architecture framework contains standardized viewpoints, view templates, meta-models, model templates, etc. that facilitate the development of the views of a system architecture (see architecture framework for examples). ISO/IEC/IEEE 42010 (ISO 2011) specifies the normative features of architecture frameworks, viewpoints, and views as they pertain to architecture description. A viewpoint addresses a particular stakeholder concern (or set of closely related concerns). The viewpoint specifies the kinds of model to be used in developing the system architecture to address that concern (or set of concerns), the ways in which the models should be generated, and how the models are related and used to compose a view.

Logical and physical models (or views) are often used for representing fundamental aspects of the system architecture. Other complementary viewpoints and views are necessarily used to represent how the system architecture addresses stakeholder concerns, for example, cost models, process models, rule models, ontological models, belief models, project models, capability models, data models, etc.

Classification of Principles and Heuristics

Engineers and architects use a mixture of mathematical principles and heuristics (heuristics are lessons learned through experience, but not mathematically proven).

When an issue is identified and defined through system requirements, principles and heuristics may or may not be able to address it. Principles and heuristics that are used in system views/models can be classified according to the domains in which those system views/models are used, as follows:

1. **Static domain** relates to physical structure or organization of the SoI broken down into systems and system elements. It deals with partitioning systems, system elements, and physical interfaces.
2. **Dynamic domain** relates to logical architecture models, particularly to the representation of the behavior of the system. It includes a description of functions (i.e. transformations of input flows into output flows) and interactions between functions of the system and between those of the external objects or systems. It takes into account reactions to events that launch or stop the execution of functions of the system. It also deals with the effectiveness (i.e. performances, operational conditions) of the system.
3. **Temporal domain** relates to temporal invariance levels of the execution of functions of the system. This means that every function is executed according to cyclic or synchronous characteristics. It includes decisional levels that are asynchronous characteristics of the behavior of some functions.
4. **Environmental domain** relates to enablers (production, logistics support, etc.), but also to the survivability of the system in reaction to natural hazards or threats and to the integrity of the system in reaction to internal potential hazards. This includes, for example, climatic, mechanical, electromagnetic, and biological aspects.

More detailed classification of heuristics can be found in (Maier and Rechtin 2009).

Transition from System Requirements to Logical and Physical Architecture Models

The aim of the approach is to progress from system requirements (representing the problem from a supplier/designer point of view, as independent of technology as possible) through an intermediate model of logical architecture to allocate the elements of the logical architecture model to system elements of

candidate physical architecture models.

(System requirements and logical architecture models share many characteristics, as they are both organized on functional lines, independently of the implementation. Some authors (Stevens et al. 1998) go so far as to conflate the two, which simplifies the handling of multiple simultaneous views. Whether this approach is adopted depends on the specific practices of the development organization and where contractual boundaries are drawn.)

Design decisions and technological solutions are selected according to performance criteria and non-functional requirements, such as operational conditions and life cycle constraints (e.g., environmental conditions, maintenance constraints, realization constraints, etc.), as illustrated in Figure 1. Creating intermediate models, such as logical architecture models, facilitates the validation of functional, behavioral, and temporal properties of the system against the system requirements that have no major technological influence impacts during the life of the system, the physical interfaces, or the technological layer without completely questioning the logical functioning of the system.

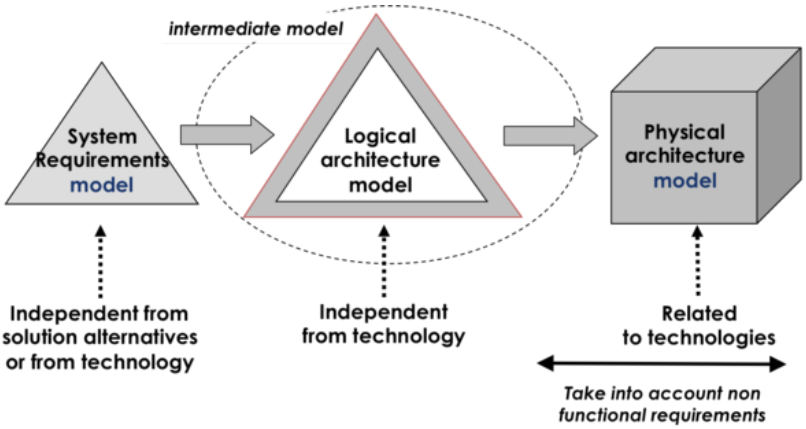

Figure 1. Usage of Intermediate Logical Architecture Models During Architecture and Design (Faisandier 2012).

Iterations between Logical and Physical Architecture Model Development

As discussed in system requirements, the exact approach taken in the synthesis of solutions will often depend on whether the system is an evolution of an already understood product or service, or a new and unprecedented solution (see Synthesizing Possible Solutions).

Whatever the approach, architecture activities require spending several iterations between logical architecture models development and physical architecture models development, until both logical and physical architecture models are consistent and provide the necessary level of detail. One of the first architecture activities is the creation of a logical architecture model based on nominal scenarios (of functions). The physical architecture model is used to determine main system elements that could perform system functions and to organize them.

Subsequent logical architecture model iterations can take into account allocations of functions to system elements and derived functions coming from physical solution choices. It also supplements the initial logical architecture model by introducing other scenarios, failure analyses, and operational requirements not previously considered. Derived functions are allocated to system elements; in turn, this affects the physical architecture models.

Additional iterations are focused on producing complete and consistent logical and physical views of the solution.

During system design, technological choices can potentially lead to new functions, new input/output and control flows, and new physical interfaces. These new elements can lead to creation of new system requirements, called *derived requirements*.

Notion of Interface

The notion of interface is one of the most important to consider when defining the architecture of a system. The fundamental aspect of an interface is functional and is defined as inputs and outputs of functions. As functions are performed by physical elements (system elements), inputs/outputs of functions are also carried by physical elements; these are called physical interfaces. Consequentially, both functional and physical aspects are considered in the notion of interface. A detailed analysis of an interface shows the function *"send"* located in one system element, the function *"receive"* located in the other one, and the function *"carry*" as being performed by the physical interface that supports the input/output flow (see Figure 2).

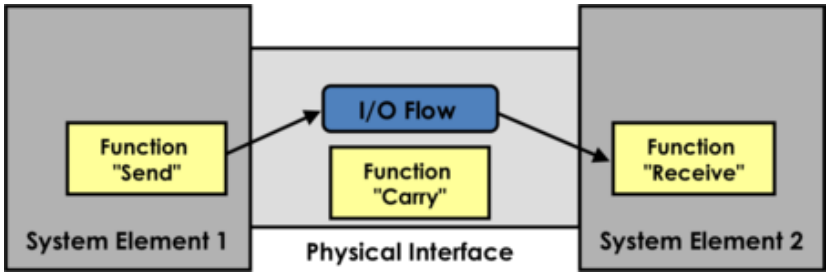

Figure 2. Complete Interface Representation (Faisandier 2012).

In the context of complex exchanges between system elements, particularly in software-intensive systems, a protocol is seen as a physical interface that carries exchanges of data. However, the input/output flows can include many other exchanges than data, such as energy.

Emergent Properties

The overarching architecture of a system may have design properties or operational effects that emerge from the arrangement and interaction between system elements, but which may not be properties of any individual element or intended for the system as a whole.

The elements of an engineered system interact among themselves and can create desirable or undesirable phenomena, such as inhibition, interference, resonance, or the reinforcement of any property. The definition of the system includes an analysis of interactions between system elements in order to prevent undesirable properties and reinforce desirable ones.

A property which emerges from a system can have various origins, from a single system element to the interactions among several elements (Thome, B. 1993). The term emergent properties is used by some authors to identify any property which emerges from a system, while other may refer to this as synergy and reserve emergent property for explaining unexpected properties or properties not considered fully during system development, but have emerged during operation. The system concept of emergence is discussed in SEBoK Part 2 (see Emergence).

Table 1. Properties and Emergent Properties.(SEBoK Original)

Broad Categories of Properties	Description and Examples

Local Property	The property is located in a single system element – e.g. the capacity of a container is the capacity of the system.
Accumulative System Property	The property is located in several system elements and is obtained through the simple summation of elemental properties – e.g. the weight of the system results from the sum of the weights of its system elements.
Emergent Property Modified by Architecture and/or Interactions.	The property exists in several system elements and is modified by their interactions – e.g. the reliability/safety of a system results from the reliability/safety of each system element and the way they are organized. Architectural steps are often critical to meeting system requirements.
Emergent Property Created by Interactions	The property does not exist in system elements and results only from their interactions – e.g. electromechanical interfaces, electromagnetism, static electricity, etc.
Controlled Emergent Property	Property controlled or inhibited before going outside the system – e.g.: unbalance removed by the addition of a load; vibration deadened by a damper.

Physical architecture design will include the identification of likely synergies and emergent properties and the inclusion of derived functions, components, arrangements, and/or environmental constraints in the logical or physical architectures models to avoid, mitigate or restrain them within acceptable limits. Corresponding *derived requirements* should be added to the system requirements baseline when they impact the system-of-interest(SoI). This may be achieved through the knowledge and experience of the systems engineer or through the application of system patterns. However, it is generally not possible to predict, avoid, or control all emergent properties during the architecture development. Fully dealing with the consequences of emergence can only be done via iteration between system definition, system realization and system deployment and use (Hitchins, 2008)

The notion of emergence is applied during architecture and design to highlight necessary derived functions; additionally, internal emergence is often linked to the notion of complexity. This is the case with complex adaptive systems (CAS), in which the individual elements act independently, but behave jointly according to common constraints and goals (Flood and Carson 1993). Examples of CAS include: the global macroeconomic

network within a country or group of countries, stock market, complex web of cross border holding companies, manufacturing businesses, geopolitical organizations, etc. (Holland, J. 1999 and 2006).

Reuse of System Elements

Systems engineers frequently utilize existing system elements. This reuse constraint has to be identified as a system requirement and carefully taken into account during architecture and design. One can distinguish three general cases involving system element reuse, as shown in Table 2.

Table 2. System Element Re-use Cases (Faisandier 2012). Permission granted by Sinergy'Com. All other rights are reserved by the copyright owner.

Re-use Case	Actions and Comments
Case 1: The requirements of the system element are up-to-date and it will be re-used with no modification required.	• The system architecture to be defined will have to adapt to the boundaries, interfaces, functions, effectiveness, and behavior of the re-used system element. • If the system element is not adapted, it is probable that costs, complexity, and risks will increase.
Case 2: The requirements of the system element are up-to-date and it will be re-used with possible modifications.	• The system architecture to be defined is flexible enough to accommodate the boundaries, interfaces, functions, effectiveness, and behavior of the re-used system element. • The design of the reused system element, including its test reports and other documentation, will be evaluated and potentially redesigned.
Case 3: The requirements are not up-to-date or do not exist.	• It is necessary to reverse engineer the system element to identify its boundaries, interfaces, functions, performances, and behavior. This is a difficult activity, since the extant documentation for the re-used system element is likely unavailable or insufficient. • Reverse engineering is expensive in terms of both time and money, and brings with it increased risk.

There is a common idea that reuse is *free*; however, if not approached correctly, reuse may introduce risks that can be significant for the project (costs, deadlines, complexity).

Process Approach

Purpose

The purpose of the System Architecture process is to generate system architecture alternatives, to select one or more alternative(s) that frame stakeholder concerns and meet system requirements, and to express this in a set of consistent views. (ISO 2015).

It should be noted that the architecture activities below overlap with both system definition and concept definition activities. In particular, key aspects of the operational and business context, and hence certain stakeholder needs, strongly influence the approach taken to architecture development and description. Also, the architecture activities will drive the selection of, and fit within, whatever approach to solution synthesis has been selected.

Activities of the process

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

1. Initialize the definition of the system architecture

- Build an understanding of the environment/context of use for which a system is needed in order to establish insight into the stakeholder concerns. To do this, analyze relevant market, industry, stakeholder, enterprise, business, operations, mission, legal and other information that help to understand the perspectives that could guide the definition of the system architecture views and models.
- Capture stakeholder concerns (i.e., expectations or constraints) that span system life cycle stages. The concerns are often related to critical characteristics of the system that relate to the stages; they should be translated into or incorporated into system requirements.
- Tag system requirements that deal with operational conditions (e.g., safety, security, dependability, human factors, interfaces, environmental conditions) and life cycle constraints (e.g., maintenance, disposal, deployment) that would influence the definition of the

architecture elements.

- Establish an architecture roadmap and strategy that should include methods, modeling techniques, tools, need for any enabling systems, products or services, process requirements (e.g., measurement approach and methods), evaluation process (e.g., reviews and criteria).
- Plan enabling products or services acquisition (need, requirements, procurement).

2. Define necessary architecture viewpoints

- Based on the identified stakeholder concerns, identify relevant architecture viewpoints and architecture frameworks that may support the development of models and views.

3. Develop candidate architectures models and views

- Using relevant modeling techniques and tools, and in conjunction with the Stakeholder Needs and Requirements process and the System Requirements process, determine the system-of-interest context including boundary with elements of the external environment. This task includes the identification of relationships, interfaces or connections, exchanges and interactions of the system-of-interest with external elements. This task enables definition or understanding of the expected operational scenarios and/or system behaviors within its context of use.
- Define architectural entities (e.g., functions, input/output flows, system elements, physical interfaces, architectural characteristics, information/data elements, containers, nodes, links, communication resources, etc.), which address the different types of system requirements (e.g., functional requirements, interface requirements, environmental requirements, operational conditions [dependability, human factors, etc.], constraints [physical dimensions, production, maintenance, disposal]).
- Relate architectural entities to concepts, properties, characteristics, behaviors, functions, and/or constraints that are relevant to decisions of the system-of-interest architecture. This gives rise to

architectural characteristics (e.g., generality, modularity, operability, efficiency, simplicity).

- Select, adapt, or develop models of the candidate architectures of the system, such as logical and physical models (see **Logical Architecture Model Development** and **Physical Architecture Model Development**). It is sometimes neither necessary nor sufficient to use logical and physical models. The models to be used are those that best address key stakeholder concerns.
- From the models of the candidate architectures, compose views that are relevant to the stakeholder concerns and critical or important requirements.
- Define derived system requirements induced by necessary instances of architectural entities (e.g., functions, interfaces) and by structural dispositions (e.g., constraints, operational conditions). Use the system requirements definition process to define and formalize them.
- Check models and views consistency and resolve any identified issues. ISO/IEC/IEEE 42010, 2011 may be used for this.
- Verify and validate the models by execution or simulation, if modeling techniques and tools permit. Where possible, use design tools to check feasibility and validity, and/or implement partial mock-ups, or use executable architecture prototypes or simulators.

4. Relate system architecture to system design

- Define the system elements that reflect the architectural characteristics (when the architecture is intended to be design-agnostic, these system elements may be notional until the design evolves). To do this, partition, align, and allocate architectural characteristics and system requirements to system elements. Establish guiding principles for the system design and evolution. Sometimes, a “reference architecture” is created using these notional system elements as a means to convey architectural intent and to check for design feasibility.
- Define interfaces for those that are necessary for the level of detail and understanding of the architecture. This includes the internal interfaces between the system elements and the external interfaces with

other systems.

- Determine the design properties applicable to system elements in order to satisfy the architectural characteristics.
- For each system element that composes the system, develop requirements corresponding to allocation, alignment, and partitioning of design properties and system requirements to system elements. To do this, use the stakeholder needs and requirements definition process and the system requirements definition process.

5. Assess architecture candidates and select one

- Assess the candidate architectures using the architecture evaluation criteria. This is done through application of the System Analysis, Measurement, and Risk Management processes.
- Select the preferred architecture(s). This is done through application of the Decision Management process.

6. Manage the selected architecture

- Establish and maintain the rationale for all selections among alternatives and decisions for the architecture, architecture framework(s), viewpoints, kinds of models, and models of the architecture.
- Manage the maintenance and evolution of the architecture description, including the models, and views. This includes concordance, completeness, changes due to environment or context changes, and technological, implementation, and operational experiences. Allocation and traceability matrices are used to analyze impacts onto the architecture. The present process is performed at any time evolutions of the system occur.
- Establish a means for the governance of the architecture. Governance includes the roles, responsibilities, authorities, and other control functions.
- Coordinate reviews of the architecture to achieve stakeholder agreement. The stakeholder requirements and system requirements can serve as references.

Artifacts, Methods and Modeling Techniques

This process may create several artifacts, such as system architecture description documents and system justification documents (traceability matrices and architectural choices).

The content, format, layout, and ownership of these artifacts may vary depending on the person creating them and the domains in which they are being used. The outputs of the process activities should cover the information identified in the first part of this article.

Practical Considerations

Pitfalls

Some of the key pitfalls encountered in planning and performing system architecture are provided in Table 3.

Table 3. Pitfalls with System Architecture Definition.(SEBoK Original)

Pitfall	Description
Problem Relevance	If the architecture is developed without input from the stakeholders' concerns or cannot be understood and related back to their issues it might lose the investments of the stakeholder community.
Reuse of System Elements	In some projects, for industrial purposes, existing products or services are imposed very early as architecture/design constraints in the stakeholder requirements or in the system requirements, without paying sufficient attention to the new context of use of the system in which they are also included. It is better to work in the right direction from the beginning. Define the system first, note other requirements, and then see if any suitable non-developmental items (NDI) are available. Do not impose a system element from the beginning, which would reduce the trade-space. The right reuse process consists of defining reusable system elements in every context of use.

Proven Practices

Some proven practices gathered from the references are provided in Table 4.

Table 4. Proven practices with System Architecture Definition.(SEBoK Original)

Practice	Description
Emerging properties	Control the emergent properties of the interactions between the systems or the system elements; obtain the required synergistic properties and control or avoid the undesirable behaviors (vibration, noise, instability, resonance, etc.).

References

Works Cited

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

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

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

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

Wilkinson, M., A. James, M. Emes, P. King, P. Bryant. 2010. “Belief Systems in Systems Architecting: Method and Preliminary Applications." Presented at the IEEE SMC Society’s 5th International Conference on System of Systems Engineering (SoSE). 22nd-24th June 2010. Loughborough University, Leicestershire, UK.

Flood, R.L., and E.R. Carson. 1993. *Dealing with Complexity: An Introduction to the Theory and Application of Systems Science*, 2nd ed. New York, NY, USA: Plenum Press.

Holland, J.H. 1999. *Emergence: From Chaos to Order*. Reading, MA, USA: Perseus Books.

Hitchins, D. 2008. "Emergence, Hierarchy, Complexity, Architecture: How Do They All Fit Together? A Guide for

Seekers after Enlightenment." Self-published white paper. Accessed 4 September 2012. Available at: http://www.hitchins.net/EmergenceEtc.pdf.

Holland, J.H. 2006. "Studying complex adaptive systems." *Journal of Systems Science and Complexity,* vol. 19, no. 1, pp. 1-8. Available at:http://hdl.handle.net/2027.42/41486

Thome, B. 1993. *Systems Engineering, Principles & Practice of Computer-Based Systems Engineering*. New York, NY, USA: Wiley.

Primary References

ANSI/IEEE. 2000. *Recommended Practice for Architectural Description for Software-Intensive Systems*. New York, NY, USA: American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE), ANSI/IEEE 1471-2000.

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

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

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

Blanchard, B.S., and W.J. Fabrycky. 2005. *Systems Engineering and Analysis.* 4th ed. Prentice-Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.
ISO/IEC. 2007. *Systems Engineering – Application and Management of The Systems Engineering Process*. Geneva, Switzerland: International Organization for Standards (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 26702:2007.

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

Martin, J.N. 1997. *Systems Engineering Guidebook: A Process for Developing Systems and Products,* 1st ed. Boca Raton, FL, USA: CRC Press.

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

Additional References

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

OMG. 2010. *OMG Systems Modeling Language specification*, version 1.2, July 2010. Available at: http://www.omg.org/technology/documents/spec_catalog.htm.

Sillitto H. 2014. *Architecting Systems - Concepts, Principles and Practice.* London, UK: College Publications.

Wilkinson, M.K. 2010. "Z8: Systems Architecture," in Z-guide series. Foresgate, UK: INCOSE UK. Available at: http://www.incoseonline.org.uk/Program_Files/Publications/zGuides.aspx?CatID=Publications.

Relevant Videos

- What is Systems Architecture (PART 1)

< Previous Article | Parent Article | Next Article >

SEBoK v. 2.9, released 20 November 2023
