# Product as a System Fundamentals

***Lead Author:*** *Ricardo Pineda*

The "Product Systems Engineering" knowledge area is graciously sponsored by PPI.

This article introduces fundamental concepts of product systems.

☐

## Contents

Product Elements and Connections
Core Product and its Enabling Products & Operational Services
Product Architecture, Modeling, and Analysis
Specialty Engineering Integration
References
Works Cited
Primary References
Additional References

## Product Elements and Connections

Product systems consist of product elements and two kinds of connections: connections among elements, and connections between elements and things in the system environment. That portion of the environment that can be influenced by the system or that can influence the system is called the “context.”

Connections between elements contain interactions and relationships (Hybertson 2009). A connection is more than a mere interface.

Interactions occur across *interfaces* between the elements inside or outside the system, and can be defined as exchanges of data, materials, forces, or energy. Connections with an interactive nature can be represented in various engineering artifacts: schematic block diagrams, data flow diagrams, free body diagrams, interface control diagrams, port specifications, energy transfer diagrams, and so on. Product systems engineering (PSE) usually defines interactions in an interface control document, interface design document, interface requirements document, or the equivalent.

Connections also encompass relationships between elements. These relationships can be spatial, motion-related, temporal, or social.

Spatial relationships:

- one element is underneath another
- two elements are *x units* apart
- one element is inside another

Motion-related relationships:

- the relative velocity of two elements is *v units*
- the relative acceleration between two elements is *a units*

Temporal relationships:

- one element exists before another
- two elements must exist at the same time
- two elements must be separated in time by *t units*

Social relationships:

- a human element feels a particular way about a system
- a human element owns another (non-human) element
- a human element understands the operation of a system in a particular way

Relationships that are not about time can still change over time. For example, an element that is inside another element during one mode of operation can be outside of it during a different mode of operation. Therefore, one should not assume that non-temporal relationships are necessarily static in time.

Relationships can be represented in engineering artifacts, including the timing diagram, timeline diagram, mission reference profile, capability road map, and project schedule chart.

Social relationships include the implicit or explicit social obligations or expectations between the roles that human elements play in a system. These roles may be assigned different authorities, responsibilities, and accountabilities. See the discussion on organization behavior in the article Team Dynamics. Organizational behavior theories and human factors may need to be considered when engineering such a product system.

There can also be social relationships between the humans and the non-human elements of the system. This may involve how the human "feels" about things in the system or perhaps even the system as a whole. Humans inside or outside the system-of-interest may have different degrees of "understanding" with respect to how the system operates, its limitations and capabilities, and the best way to operate it safely and effectively. The "ownership" relationship can be important in determining things like who can operate or change some configuration or mode of the system.

There are many such social relationships in a product system that are often ignored or misunderstood when performing PSE. Social relationships can affect the overall performance or behavior of a product system to the point of determining its success or failure.

## Core Product and its Enabling Products & Operational Services

A variety of systems (themselves being products or services) enable the development, delivery, operation and eventual disposal of a product, as shown in Figure 1. The concept of enabling systems is defined in the ISO/IEC 15288 standard (2015).

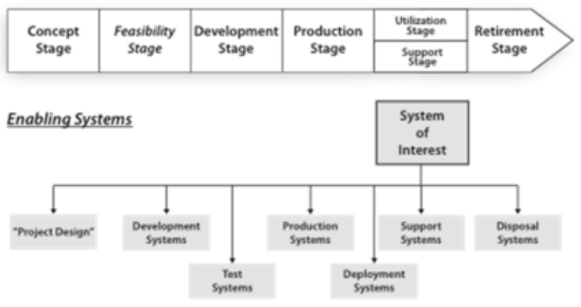

**Figure 1. Example of Enabling Systems (Lawson 2010).** Reprinted with permission of Harold "Bud" Lawson. All other rights are reserved by the copyright owner.

In the figure, the system-of-interest (SoI) goes into operation as a delivered product or offered service in the utilization stage while maintenance and logistics are provided (by a product sustainment system) simultaneously in the support stage. These two stages are commonly executed in parallel, and they offer opportunities to observe any need for changes in the properties of the product or service or how it is operated and supported. Making changes iterates the life cycle and results in new or improved products or features.

The delivered product and its enabling systems collectively form a wider system-of-interest (WSOI). The project design enabling system is an enterprise based system asset that establishes the strategy and means of organizing the projects to be executed along the life cycle. In many larger organizations, this type of enabling system is institutionalized and can be based upon recommendations of the Project Management Institute (PMI).

Product systems should be viewed as enabling service systems. That is, once deployed, a product system provides a service that contributes to an enterprise’s operations. To the acquirer, the SoI provides operational services to users. This is true at several levels:

- Hardware and software products are used to enable the provisioning of service systems,
- Enterprises incorporate products as system assets and use them to enable operations, and

- Provided products are integrated into the system of systems.

## Product Architecture, Modeling, and Analysis

IEEE standard 1471-2000 defines architecture as "the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution" (IEEE 2000). Similarly, ISO/IEC 42010-2011 defines architecture as "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution" (ISO/IEC 2011).

A product's purpose (stakeholder’s need) is realized by a product system (the SoI). Because product systems are composed of different entities (components, assemblies, subsystems, information, facilities, processes, organizations, people, etc.) that together produce the results unachievable by any of the entities alone, architecting the product is based on a whole systems approach. To architect with a whole systems approach means to define, document, design, develop, manufacture, distribute, maintain, improve, and to certify proper implementation of the product’s objectives in terms of the functional (the "what"), the behavioral (the use, or intended operations), the logical (interaction and relationships between entities) and the physical constructs (Wasson 2006; Maier 2009; Blanchard and Fabrycky 2011).

The system architect starts at the highest level of abstraction, concentrating on needs, functions, systems characteristics and constraints (concerns) before identifying components, assemblies, or subsystems. This is the systems view, and it is used to represent the stakeholder’s market service description or the concept of operations (understanding of the opportunity/problem space).

Next to be documented, as needs become better understood, are architectural descriptions at different levels of abstraction, representing various stakeholders interests. These are the architecture models. They define the possible solution spaces for the product purpose in the form of detailed system, operational, behavioral, and physical requirements of the product system.

Different modeling techniques are then used to analyze different types of requirements. For operational scenarios and different modes of operation, there are hierarchical decomposition and allocation, architectural block diagrams (ABD), functional block diagrams (FBD), functional flow block diagrams (FFBD), and use case diagrams. For interactions and relationships among hardware and/or software components there are sequence diagrams, activity diagrams, state diagrams, and data flow diagrams. See (Maier 2009) Chapter 8 for an introduction to models and modeling.

Analysis of the solution space makes it possible to produce detailed technical specs, engineering drawings, blueprints, software architectures, information flows, and so on, that describe the entities in the product system. An entity’s requirements bound its attributes and characteristics, levels of performance, operational capabilities, and design constraints. During design and integration, entity characteristics can be traced back to requirements (requirements traceability being a key aspect of SE). Verification and validation plans created during the requirements phase are the basis of testing certification that the product does what it was intended to do.

Overall, what occurs is the transformation of a set of requirements into products and processes that satisfy the stakeholder's need. The architecture is represented by a set of models that communicate an integrated view of the product's intent and purpose, and the interactions and interfaces required among all the different participating entities. The product's purpose is articulated in terms of business objectives (market, cost, functionality, performance, and time to deliver). The set of models includes sufficient variety to convey information appropriately to the stakeholders, designers/developers, specialty engineering, operations, manufacturers, management, and marketing and sales personnel.

Different architecture frameworks have been developed to guide product teams in defining the product architecture for commercial and for public enterprises. In general, an architecture framework describes a "view," meaning a "collection of models that represent the whole system from the perspective of a set of related stakeholder concerns." Prime examples of architecture frameworks are the Zachman framework (Zachman 1992), The Open Group Architecture Framework (TOGAF) (TOGAF 2011), the Enhanced-Telecom Operations Map (e-TOM), just to mention a few in the commercial sector. In the case of public enterprises a few architecture frameworks include the Department of Defense Architecture Framework (DODAF 2.0) (DoD 2010), the Federal Enterprise Architecture Framework (FEAF) (FEA 2001), the British Ministry of Defense Architecture Framework (MODAF) (MOD 2004), etc.

Differences between acquired products and offered products play an important role in defining product system requirements. Acquired products are life cycle-managed directly by the acquirer; for instance, acquired defense systems are defined, developed, tested, owned, operated, maintained and upgraded by the defense agency. See the article Product Systems Engineering Key Aspects within this KA.

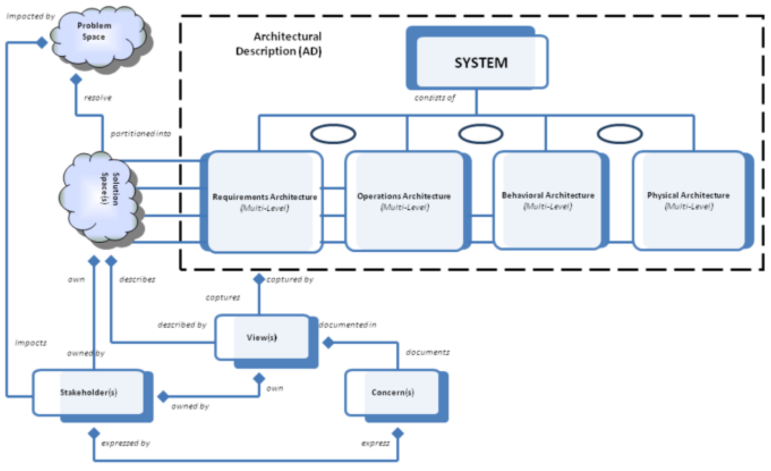

**Figure 2. System Architectural Description Elements (Adapted from Wasson 2006).** Reprinted with permission of John Wiley & Sons Inc. All other rights are reserved by the copyright owner.

## Specialty Engineering Integration

The INCOSE *SE Handbook* defines specialty engineering as:

“Analysis of specific features of a system that requires special skills to identify requirements and assess their impact on the system life cycle.”

Areas of expertise that fall under this umbrella definition include logistics support, electromagnetic compatibility analysis, environmental impact, human factors, safety and health analysis, and training. The unique characteristics, requirements, and design challenges of a system-of-interest all help determine the areas of specialty that apply.

A number of specialty engineering areas are typically important to systems engineers working on the development, deployment, and sustainment of product systems. For example, logistics support is essential for fielded product systems that require maintenance and repair. The delivery of services, materials, parts, and software necessary for supporting the system must all be considered very early in the development activity. These factors should usually be considered before the system requirements and concept definition are complete. To integrate these specialty disciplines sufficiently early on, the systems engineer needs to know what specialties relate to the system under development, how they relate to the systems engineering process, and how to integrate them into the life cycle process.

For product systems with significant hardware content and that operate in challenging environments, the following specialty engineering areas must usually be considered:

- manufacturability,
- reliability and maintainability,
- certification (essential where human safety is an issue),
- logistics support,
- electromagnetic compatibility (if they radiate),
- environmental impact,
- human factors,
- safety and health, and
- training.

The relationship of these specialty areas to the systems engineering process must be understood and considered. The key aspects of the relationship are:

- when the specialty needs to be considered,
- what essential data or information it provides,
- the consequences of not including the specialty in the systems engineering process, and
- how the systems engineers should interact with the specialty engineers.

Grady (2006) provides an overview, with references, for many of the specialty engineering disciplines, including reliability engineering; parts, materials, and process engineering (PMP); maintainability engineering, availability, producibility engineering, design to cost/life cycle cost (DTC/LCC), human factors engineering, corrosion prevention and control (CPC), system safety engineering, electromagnetic compatibility (EMC) engineering, system security engineering, mass properties engineering, and environmental impact engineering.

Eisner (2008) lists specialty engineering as one of the “thirty elements” of systems engineering. “Specialty engineering refers to a set of engineering topics that have to be explored on some, but not all, systems engineering efforts. In other words, some systems involve these special disciplines and some do not. Examples of specialty engineering areas include electromagnetic compatibility and interference, safety, physical security, computer security, communications security, demand forecasting, object-oriented design, and value engineering.” Some of what we consider specialty engineering in the present article, Eisner includes among his “thirty elements” of systems engineering, but not as part of the specialty engineering element.

There is no standard list of specialty engineering disciplines. What is considered specialty engineering varies according to the community to which the systems engineering belongs, and sometimes to the preferences of the customer.

## References

### Works Cited

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.

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

Eisner, H. 2008. "Chapter 7. Essentials of Project and Systems Engineering Management," in *The Thirty Elements of Systems Engineering*, 3rd ed. New York, NY, USA: John Wiley & Sons.

Grady, J. 2006. "Chapter 3.7. System Requirements Analysis," in *Specialty Engineering Requirements*

*Analysis*. New York, NY, USA: Academic Press.

Grady, J. 2006. *System Requirements Analysis*. New York, NY, USA: Academic Press.

Grady, J. 2010. *Systems Synthesis - Product and Process Design*. Boca Raton, FL, USA: CRC Press.

Hybertson, D. 2009. *Model-oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boston, MA, USA: Auerbach Publications.

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

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

ISO/IEC/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.

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

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

MOD. 2004. *Ministry of Defence Architecture Framework (MODAF)*, version 2. London, UK: UK Ministry of Defence.

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

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

Zachman, J.A. 1992. "Extending and Formalizing the Framework for Information Systems Architecture." *IBM*

*Systems Journal*. 31 (3): 590-616.

## Primary References

Eisner, H. 2008. "Chapter 7. Essentials of Project and Systems Engineering Management," in *The Thirty Elements of Systems Engineering*, 3rd ed. New York, NY, USA: John Wiley & Sons.

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

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

## Additional 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.

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

Grady, J. 2006. "Chapter 3.7. System Requirements Analysis," in *Specialty Engineering Requirements Analysis*. New York, NY, USA: Academic Press.

Grady, J. 2006. *System Requirements Analysis*. New York, NY: Academic Press.

Grady, J. 2010. *Systems Synthesis- Product and Process Design*. Boca Raton, FL, USA: CRC Press.

Hybertson, D. 2009. *Model-oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*. Boston, MA, USA: Auerbach Publications.

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

ISO/IEC. 2008. *Systems and Software Engineering -- System Life Cycle Processes*. Geneva, Switzerland: International Organisation for Standardisation /

International Electrotechnical Commissions. ISO/IEC/IEEE 15288:2008.

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.

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

< Previous Article | Parent Article | Next Article >
**SEBoK v. 2.10, released 06 May 2024**