Software Engineering in the Systems Engineering Life Cycle

From SEBoK
Software Engineering in the Systems Engineering Life Cycle
Jump to navigation Jump to search
The printable version is no longer supported and may have rendering errors. Please update your browser bookmarks and please use the default browser print function instead.

Lead Authors: *Tom Hilburn, Dick Fairley,* ***Contributing Author:*** *Alice Squires*

This article describes how software engineeringsoftware engineering (SwE) life cycle processes integrate with the SE life cycle. A joint workshop organized by INCOSE, the Systems Engineering Research Center and the IEEE Computer Society was held to consider this relationship (Pyster et al. 2015). This workshop concluded that:

> *Software is fundamental to the performance, features, and value of most modern engineering systems. It is not merely part of the system, but often shapes the system architecture; drives much of its complexity and emergent behavior; strains its verification; and drives much of the cost and schedule of its development. Given how significant an impact software has on system development and given how complex modern systems are, one would expect the relationship between the disciplines of systems engineering (SE) and software engineering (SWE) to be well defined. However, the relationship is, in fact, not well understood or articulated.*

In this article we give some of the basic relationships between SwE and SE and discuss how these can be related to some of the SEBoK knowledge areas.

☐

Contents

Systems Engineering and Software Engineering Life Cycles

The Guide to the Software Engineering Body of Knowledge (SWEBoK) (Bourque and Fairley, 2014) describes the life cycle of a software product as:

- analysis and design,
- construction,
- testing,
- operation,
- maintenance, and eventually
- retirement or replacement.

This life cycle is common to most other mature engineering disciplines.

In Part 3 of the SEBoK, SE and Management, there is a discussion of SE life cycle modelslife cycle models and life cycle processeslife cycle processes. A Generic Life Cycle Model is described and reproduced in Fig. 1 below. This is used to describe necessary stages in the life cycle of a typical engineered systemengineered system.

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

Part 3 defines a collection of generic SE life cycle processes which define the activities and information needed across the SE life cycle. These processes include activities which contribute across the whole life cycle, with peaks of focused activity in certain stages (see Applying Life Cycle Processes for details).

The following sections provide a brief discussion of how SwE life cycle processes fit into SE life cycle process models. In practice, the details of this relationship are a key part of how a system life cycle is planned and delivered. The relationship will be shaped by the operating domain practice and solution type. Some examples of this are provided in the Implementation Examples.

Systems Engineering and Software Engineering Standards

The Systems Engineering life cycle processes described in Part 3, SE and Management, are largely based on those defined in the ISO/IEC/IEEE SE Life Cycle Processes 15288 Standard (2015).

The SWEBoK references the equivalent ISO/IEC/IEEE Software Engineering Life Cycle Processes 12207 Standard (2008), which defines a very similar set of processes for software systems. Figure 2 shows the relationship between the Enabling, Acquisition, Project and Technical Systems and Software processes in both 15288 and 12207 and the software specific processes of 12207. This alignment is from the last updates of both 12207 and 15288 in 2008. The SE processes have been further updated in 15288:2015, see Systems Engineering and Management for details. This change has not yet been applied to 12207. An update of 12207 was released in 2017, in which the alignment to 15288 was reviewed. See Alignment and Comparison of the Standards for more discussion of the relationships between the standards.

Figure 2. Aligned Process Models for ISO/IEC/IEEE 15288 & 12207: 2008 (Adapted from Roedler 2011). Reprinted with permission of Garry Roedler. All other rights are reserved by the copyright owner.

Systems Engineering and Software Engineering Life Cycle Relationships

Pyster et al. (2015) define two technical dimensions of engineered systemsengineered systems and of the engineering disciplines associated with them. The vertical dimensions of a system are those that modularize around technically focused engineering concerns involving specific elements of the system; the horizontal dimensions of a system involve cross-cutting concerns at the systems level. Examples of vertical concerns include quality attributes and performance effectiveness; and cost, schedule and risk of physical, organizational or human system elements associated with a particular technology domain. Examples of horizontal concerns include addressing evolving customer preferences that drive systems-level quality attributes, trade-off and optimization; resolving system architecture, decomposition and integration issues; implementing system development processes; and balancing system economics, cost, risk and schedule.

In complex systems projects, SE has a horizontal role while traditional engineering disciplines such as electrical, mechanical, and chemical engineering have vertical roles. To the extent that it is responsible for all aspects of the successful delivery of software related elements, SwE can be considered as one of the vertical disciplines. All of these traditional vertical disciplines will have some input to the horizontal dimension. However, the nature of software and its role in many complex systems makes SwE a critical discipline for many horizontal concerns. This is discussed further below.

The ISO/IEC/IEEE 12207 software engineering standard (2008) considers two situations:

- The life cycle of software products, containing minimal physical hardware, should use software specific processes and a simple life cycle
- The life cycle of systems with a significant software content (sometimes called software intensive systems) should integrate the software processes into the SE life cycle

The second of these situations is the one relevant to the practice of SE and requires a significant horizontal contribution from SwE.

The relationship central to this is the way **SwE Implementation Processes** (see Fig 2) are used in the SE life cycle to support the implementation of software intensive system elementssystem elements. This simple relationship must be seen in the context of the concurrencyconcurrency, iterationiteration and recursionrecursion relationship between SE life cycle processes described in Applying Life Cycle Processes. This means that, in general, software requirements and architecture processes will be applied alongside system requirements and architecture processes; while software integration and test processes are applied alongside system integration, verification and validation processes. These interrelationships help with vertical software concerns, ensuring detailed software design and construction issues are considered at the system level. They also help with horizontal concerns, ensuring whole system issues are considered and are influenced by an understanding of software. See the Nature of Software for more details.

The ways these related processes work together will depend on the systems approachsystems approach to solution synthesissynthesis used and how this influences the life cycle. If a top down approach is used, problem needs and system architecture will drive software implementation and realization. If a bottom up approach is used, the architecture of existing software will strongly influence both the system solution and the problem which can be considered. In Applying Life Cycle Processes, a "middle-out" approach is described which combines these two ideas and is the most common way to develop systems. This approach needs a two-way relationship between SE and SwE technical processes.

The **SW Support Processes** may also play these vertical and horizontal roles. Part 3 contains knowledge areas on both System Deployment and Use which includes operation, maintenance and logistics; and Systems Engineering Management which covers the project processes shown in Figure 2. SwE support processes focus on the successful vertical deployment and use of software system elements and the management needed to achieve this. They also support their equivalent horizontal SE processes in contributing to the success of the whole system life cycle. The **Software Reuse Processes** have a particularly important role to play in deployment and use and Product and Service Life Management processes. The latter considers Service Life Extension; Capability Updates, Upgrades, and Modernization; and system Disposal and Retirement. All of these horizontal software engineering activities rely on the associated SE activities having a sufficient understanding of the strengths and limitations of software and SwE (see Key Points a Systems Engineer Needs to Know about Software Engineering).

The Life Cycle Models knowledge area also defines how Vee and Iterative life cycle models provide a framework to tailor the generic life cycle and process definitions to different types of system development. Both models, with some modification, apply equally to the development of products and services containing software. Thus, the simple relationships between SE and SwE processes will form the basis for tailoring to suit project needs within a selected life cycle model.

Software and Systems Challenges

Pyster et al. (2015) define three classes of software intensive systems distinguished by the primary sources of novelty, functionality, complexity and risk in their conception, development, operation and evolution. These are briefly described below:

- **Physical Systems** operate on and generate matter or energy. While they often utilize computation and software technologies as components, those components are not dominant in the horizontal dimension of engineering. Rather, in such systems, they are defined as discrete system elements and viewed and handled as vertical concerns.
- **Computational Systems** include those in which computational behavior and, ipso facto, software are dominant at the systems level. The primary purpose of these systems is to operate on and produce data and information. While these systems always include physical and human elements, these are not the predominant challenges in system development, operation and evolution.
- **Cyber-Physical Systems** are a complex combination of computational and physical dimensions. Such systems are innovative, functionally complex and risky in both their cyber and physical dimensions. They pose major horizontal engineering challenges across the board. In cyber-physical systems, cyber and physical elements collaborate in complex ways to deliver expected system behavior.

Some of the challenges of physical and computational systems are well known and can be seen in many SE and SwE case studies. For example, physical system life cycles often make key decisions about the system architecture or hardware implementation which limit the subsequent development of software architecture and designs. This can lead to software which is inefficient and difficult or expensive to change. Problems which arise later in the life of such systems may be dealt with by changing software or human elements. This is sometimes done in a way which does not fully consider SwE design and testing practices. Similarly, computational systems may be dominated by the software architecture, without sufficient care taken to consider the best solutions for enabling hardware or people. In particular, operator interfaces, training and support may not be considered leading to the need for expensive organizational fixes once they are in use. Many computational systems in the past have been developed without a clear view of the user need they contribute to, or the other systems they must work with to do so. These and other related issues point to a need for system and software engineers with a better understanding of each other's disciplines. Pyster et al. (2015) consider how SE and SwE education might be better integrated to help achieve this aim.

Examples of cyber-physical systems increasingly abound – smart automobiles, power grids, robotic manufacturing systems, defense and international security systems, supply-chain systems, the so-called internet of things, etc. In these systems there is no clear distinction between software elements and the whole system solution. The use of software in these systems is central to the physical outcome and software is often the integrating element which brings physical elements and people together. These ideas are closely aligned with the Service System EngineeringService System Engineering approach described in Part 4.

SEBoK Part 3 includes a Business and Mission Analysis process which is based on the equivalent process in the updated ISO/IEC/IEEE 15288 (2015). This process enables SE to be involved in the selection and bounding of the problem situation which forms the starting point for an engineered system life cycle. For cyber physical systems, an understanding of the nature of software is needed in the formulation of the problem, since this is often fundamentally driven by the use of software to create complex adaptive solution concepts. This close coupling of software, physical and human system elements across the system of interest continues throughout the system life cycle making it necessary to consider all three in most horizontal system level decisions.

The life cycle of cyber physical systems cannot be easily partitioned into SE and SwE achieving their own outcomes but working together on horizontal system issues. It will require a much more closely integrated approach, requiring systems and software engineers with a complementary set of competencies, and changes how the two disciplines are seen in both team and organizational structures. See Enabling Systems Engineering.

References

Works Cited

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org.

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

ISO/IECIEEE. 2008. Systems and Software Engineering — Software Life Cycle Processes. Geneva, Switzerland: International Organization for Standards (ISO)/Institute of Electrical & Electronics Engineers (IEEE) Computer Society, ISO//IECIEEE 12207:2008(E).

Pyster, A., Adcock, R., Ardis, M., Cloutier, R., Henry, D., Laird, L., Lawson, H. 'Bud'., Pennotti, M., Sullivan, K., Wade J. 2015. "Exploring the relationship between systems engineering and software engineering." 13th Conference on Systems Engineering Research (CSER). In Procedia Computer Science, Volume 44, 2015, pp. 708-717.

Roedler, G. 2011. "Towards Integrated Systems and Software Engineering Standards." National Defense Industrial Association (NDIA) Conference, San Diego, CA, USA.

Primary References

Pyster, A., Adcock, R., Ardis, M., Cloutier, R., Henry, D., Laird, L., Lawson, H. ‘Bud’., Pennotti, M., Sullivan, K., Wade J. 2015. Exploring the relationship between systems engineering and software engineering. 13th Conference on Systems Engineering Research (CSER). In Procedia Computer Science, Volume 44, 2015, pp. 708-717.

Additional References

Roedler, G. 2010. *An overview of ISO/IEC/IEE 15288, system life cycle processes*. Asian Pacific Council on Systems Engineering (APCOSE) Conference.

< Previous Article | Parent Article | Next Article >
SEBoK v. 2.5, released 15 October 2021

Categories:

- Part 6
- Topic
- Systems Engineering and Software Engineering

Navigation menu

Personal tools

- Log in

Namespaces

- Page
- Discussion

☐

Variants

Views

- Read
- View source
- View history
- PDF Export

☐

More

Search

Search Go

Stewards

- Use the SEBoK
  - Download SEBoK PDF
  - Copyright Information
  - Cite the SEBoK
  - About the SEBoK
- Additional Information
  - Examples
  - Glossary of Terms
  - Acronyms
  - Recommended References
- Toolbox
  - Recent Changes
  - Random Page
  - What Links Here
  - Special Pages

Quicklinks

- Main Page
- Note to Reviewers
- How to Read the SEBoK
- Acknowledgements
- Copyright Information
- About the SEBoK
- Download SEBoK PDF

Outline

- Table of Contents
- Part 1: Introduction
- Part 2: Systems
- Part 3: SE and Management
- Part 4: Applications of SE
- Part 5: Enabling SE
- Part 6: Related Disciplines
- Part 7: Examples

Navigation

- Knowledge Areas
- Topics
- Use Cases
- Case Studies
- Vignettes