System Life Cycle Models

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

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

Contents

Topics

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

- System Life Cycle Process Drivers and Choices
- Vee Life Cycle Model
- Incremental Life Cycle Model
- Integration of Process and Product Models
- Lean Engineering

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

Type of Value Added Products/Services

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

- A manufacturing enterprise produces nuts, bolts, and lock washer products and then sells their products as value added elements to be used by other enterprises; in turn, these enterprises integrate these products into their more encompassing value-added system, such as an aircraft or an automobile. Their requirements will generally be pre-specified by the customer or by industry standards.
- A wholesaling or retailing enterprise offers products to their customers. Its customers (individuals or enterprises) acquire the products and use them as elements in their systems. The enterprise support system will likely evolve opportunistically, as new infrastructure capabilities or demand patterns emerge.
- A commercial service enterprise such as a bank sells a variety of *products* as services to their customers. This includes current accounts, savings accounts, loans, and investment management. These services add value and are incorporated into customer systems of

individuals or enterprises. The service enterprise's support system will also likely evolve opportunistically, as new infrastructure capabilities or demand patterns emerge.

- A governmental service enterprise provides citizens with services that vary widely, but may include services such as health care, highways and roads, pensions, law enforcement, or defense. Where appropriate, these services become infrastructure elements utilized in larger encompassing systems of interest to individuals and/or enterprises. Major initiatives, such as a next-generation air traffic control system or a metropolitan-area crisis management system (hurricane, typhoon, earthquake, tsunami, flood, fire), will be sufficiently complex enough to follow an evolutionary development and fielding approach. At the elemental level, there will likely be pre-specified single-pass life cycles.
- For aircraft and automotive systems, there would likely be a pre-specified multiple-pass life cycle to capitalize on early capabilities in the first pass, but architected to add further value-adding capabilities in later passes.
- A diversified software development enterprise provides software products that meet stakeholder requirements (needs), thus providing services to product users. It will need to be developed to have capabilities that can be tailored to be utilized in different customers' life-cycle approaches and also with product-line capabilities that can be quickly and easily applied to similar customer system developments. Its business model may also include providing the customer with system life-cycle support and evolution capabilities.

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

and number of life cycle models that are deployed as well as the processes that will be used within any particular life cycle.

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

Categories of Life Cycle Model

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

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

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

Missile.

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

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

The emergence of integrated, interactive hardware-software systems made pre-specified processes potentially harmful, as the most effective human-system interfaces tended to emerge with its use, leading to further process variations, such as soft SE (Warfield 1976, Checkland 1981) and human-system integration processes (Booher 2003, Pew and Mavor 2007). Until recently, process standards and maturity models have tried to cover every eventuality. They have included extensive processes for acquisition management, source selection, reviews and audits, quality assurance, configuration management, and document management, which in many instances would become overly bureaucratic and inefficient. This led to the introduction of more lean (Ohno 1988; Womack et al. 1990; Oppenheim 2011) and agile (Beck 1999; Anderson 2010) approaches to concurrent hardware-software-human factors approaches such as the concurrent vee models (Forsberg 1991; Forsberg 2005) and Incremental Commitment Spiral Model (Pew and Mavor 2007; Boehm, et. al. 2014).

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

Systems Engineering Responsibility

Regardless of the life cycle models deployed, the role of the systems engineer encompasses the entire life cycle of the system-of-interest. Systems engineers orchestrate the development and evolution of a solution, from defining requirements through operation and ultimately until system retirement. They ensure that domain

experts are properly involved, all advantageous opportunities are pursued, and all significant risks are identified and, when possible, mitigated. The systems engineer works closely with the project manager in tailoring the generic life cycle, including key decision gates, to meet the needs of their specific project.

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

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

References

Works Cited

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

Press.

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

Boehm, B., J. Lane, S. Koolmanojwong, and R. Turner. 2014. *The Incremental Commitment Spiral Model: Principles and Practices for Successful Systems and Software*. Indianapolis, IN, USA: Addison-Wesley.

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

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

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

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

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

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

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

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

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

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

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

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

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

Primary References

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

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

INCOSE. 2015. *Systems Engineering Handbook*, version 4. San Diego, CA, USA: International Council on Systems Engineering (INCOSE). INCOSE-TP-2003-002-04.

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

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

Additional References

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

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

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

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

This is an excellent book that follows an idea from inception to its ultimately successful implementation. Although systems engineering is not discussed, it is

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

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

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

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

This is the amazing story of the last Antarctic expedition of Shackleton and the *Endurance* in 1914 to 1917. The

systems engineering lesson is the continuous, daily risk assessment by the captain, expedition leader, and crew as they lay trapped in the arctic ice for 18 months. All 28 crew members survived.

Relevant Videos

- NASA's Approach to Systems Engineering- Space Systems Engineering 101 w/ NASA

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