System Lifecycle Process Drivers and Choices

Lead Authors: *Kevin Forsberg, Rick Adcock*

As discussed in the Generic Life Cycle Model article, there are many organizational factors that can impact which life cycle processes are appropriate for a specific system. Additionally, technical factors will also influence the types of life cycle models appropriate for a given system. For example, system requirements can either be predetermined or they can be changing, depending on the scope and nature of the development for a system. These considerations lead to different life cycle model selections. This article discusses different technical factors which can be considered when selecting a life cycle process model and provides examples, guidance and tools from the literature to support life cycle model selection. The life cycle model selected can impact all other aspects of system design and development. (See the knowledge areas in Part 3 for a description of how the life cycle can impact systems engineering (SE) processes.)

□

Contents

Fixed-Requirements and Evolutionary Development Processes

Aside from the traditional, pre-specified, sequential, single-step development process (identified as Fixed Requirements), there are several models of evolutionary development processes; however, there is no one-size-fits-all approach that is best for all situations. For rapid-fielding situations, an easiest-first, prototyping approach may be most appropriate. For enduring systems, an easiest-first approach may produce an unscalable system, in which the architecture is incapable of achieving high levels of performance, safety, or security. In general, system evolution now requires much higher sustained levels of SE effort, earlier and continuous integration and testing, proactive approaches to address sources of system change, greater levels of concurrent engineering, and achievement reviews based on evidence of feasibility versus plans and system descriptions.

Evolutionary development processes or methods have been in use since the 1960s (and perhaps earlier). They allow a project to provide an initial capability followed by successive deliveries to reach the desired system-of-interest (SoI). This practice is particularly valuable in cases in which

- rapid exploration and implementation of part of the system is desired;
- requirements are unclear from the beginning, or are rapidly changing;
- funding is constrained;
- the customer wishes to hold the SoI open to the possibility of inserting new technology when it becomes mature; and
- experimentation is required to develop successive versions.

In evolutionary development a capability of the product is developed in an increment of time. Each cycle of the increment subsumes the system elements of the previous increment and adds new capabilities to the evolving

product to create an expanded version of the product in development. This evolutionary development process, that uses increments, can provide a number of advantages, including

- continuous integration, verification, and validation of the evolving product;
- frequent demonstrations of progress;
- early detection of defects;
- early warning of process problems; and
- systematic incorporation of the inevitable rework that may occur.

Primary Models of Incremental and Evolutionary Development

The primary models of incremental and evolutionary development focus on different competitive and technical challenges. The time phasing of each model is shown in Figure 1 below in terms of the increment (1, 2, 3, ...) content with respect to the definition (Df), development (Dv), and production, support, and utilization (PSU) stages in Figure 1 (A Generic System Life Cycle Model) from the Life Cycle Models article.

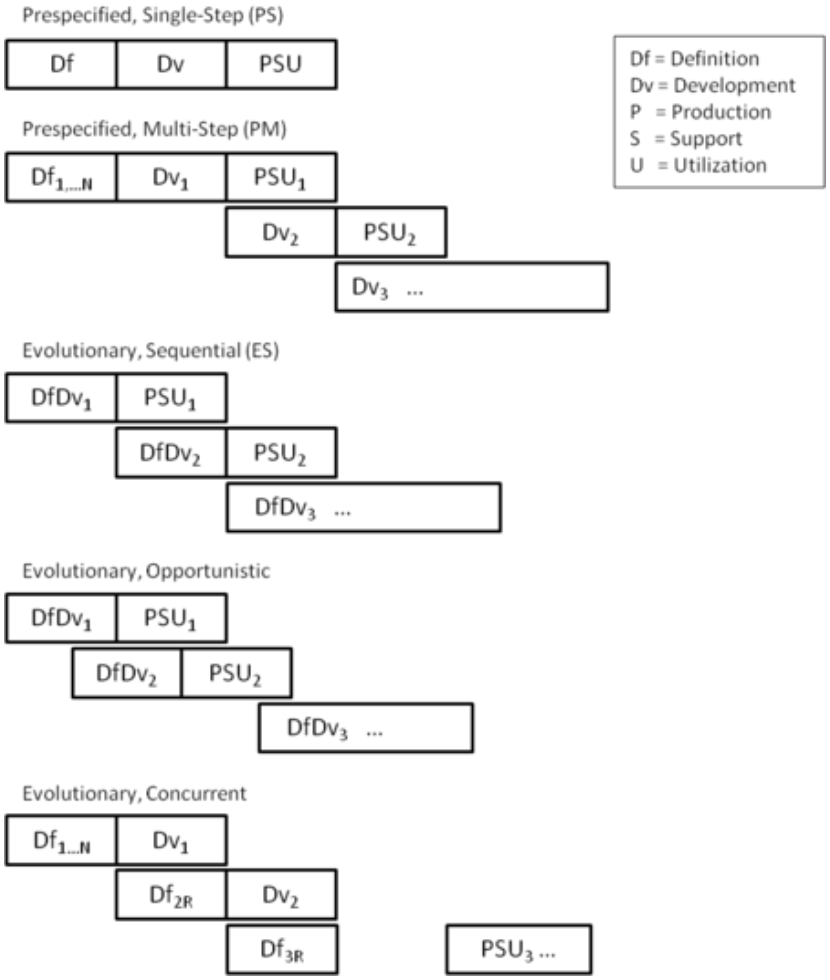

Figure 1. Primary Models of Incremental and Evolutionary Development. (SEBoK Original)

The Figure 1 notations (Df1..N and Dv1..N) indicate that their initial stages produce specifications not just for the first increment, but for the full set of increments. These are assumed to remain stable for the pre-specified sequential model but are expected to involve changes for the evolutionary concurrent model. The latter's notation (Dv1 and Df2R) in the same time frame, PSU1, Dv2 and Df3R in the same time frame, etc.) indicates that the plans and specifications for the next increment are being re-baselined by a systems engineering team concurrently with the development of the current increment and the PSU of the previous increment. This offloads the work of handling the change traffic from the development team and significantly improves its chances of finishing the current increment on budget and schedule.

In order to select an appropriate life cycle model, it is important to first gain an understanding of the main archetypes and where they are best used. Table 1 summarizes each of the primary models of single-step, incremental and evolutionary development in terms of examples, strengths, and weaknesses, followed by explanatory notes.

Table 1. Primary Models of Incremental and Evolutionary Development (Boehm, et. al. 2014, page 73).

Model	Examples	Pros	Cons
Pre-specified Single-step	Simple manufactured products: Nuts, bolts, simple sensors	Efficient, easy to verify	Difficulties with rapid change, emerging requirements (complex sensors, human-intensive systems)
Pre-specified Multi-step	Vehicle platform plus value-adding pre-planned product improvements (PPPIs)	Early initial capability, scalability when stable	Emergent requirements or rapid change, architecture breakers
Evolutionary Sequential	Small: Agile Larger: Rapid fielding	Adaptability to change, smaller human-intensive systems	Easiest-first, late, costly fixes, systems engineering time gaps, slow for large systems
Evolutionary Opportunistic	Stable development, Maturing technology	Mature technology upgrades	Emergent requirements or rapid change, SysE time gaps
Evolutionary Concurrent	Rapid, emergent development, systems of systems	Emergent requirements or rapid change, stable development increments, SysE continuity	Overkill on small or highly stable systems

The *Pre-specified Single-step* and *Pre-specified Multi-step* models from Table 1 are not evolutionary. Pre-specified multi-step models split the development in order to field an early initial operational capability, followed by several pre-planned product improvements (P3Is). An alternate version splits up the work but does not field the intermediate increments. When requirements are well understood and stable, the pre-specified models enable a strong, predictable process. When requirements are emergent and/or rapidly changing, they often require expensive rework if they lead to undoing architectural commitments.

The *Evolutionary Sequential* model involves an approach in which the initial operational capability for the system is rapidly developed and is upgraded based on operational experience. Pure agile software development

fits this model. If something does not turn out as expected and needs to be changed, it will be fixed in thirty days at the time of its next release. Rapid fielding also fits this model for larger or hardware-software systems. Its major strength is to enable quick-response capabilities in the field. For pure agile, the model can fall prey to an easiest-first set of architectural commitments which break when, for example, system developers try to scale up the workload by a factor of ten or to add security as a new feature in a later increment. For rapid fielding, using this model may prove expensive when the quick mash-ups require extensive rework to fix incompatibilities or to accommodate off-nominal usage scenarios, but the rapid results may be worth it.

The *Evolutionary Opportunistic* model can be adopted in cases that involve deferring the next increment until: a sufficiently attractive opportunity presents itself, the desired new technology is mature enough to be added, or until other enablers such as scarce components or key personnel become available. It is also appropriate for synchronizing upgrades of multiple commercial-off-the-shelf (COTS) products. It may be expensive to keep the SE and development teams together while waiting for the enablers, but again, it may be worth it.

The *Evolutionary Concurrent* model involves a team of systems engineers concurrently handling the change traffic and re-baselining the plans and specifications for the next increment, in order to keep the current increment development stabilized. An example and discussion are provided in Table 2, below.

Incremental and Evolutionary Development Decision Table

The Table 2 provides some criteria for deciding which of the processes associated with the primary classes of incremental and evolutionary development models to use.

Table 2. Incremental and Evolutionary Development Decision Table. (Boehm, et. al., 2014, page 74).

Model	Stable, pre-specifiable requirements?	OK to wait for full system to be developed?	Need to wait for next-increment priorities?	Need to wait for next-increment enablers*?
Pre-specified Single-step	Yes	Yes		
Pre-specified Multi-step	Yes	No		

Evolutionary Sequential	No	No	Yes	
Evolutionary Opportunistic	No	No	No	Yes
Evolutionary Concurrent	No	No	No	No

Example enablers: Technology maturity; External-system capabilities; Needed resources; New opportunities

The *Pre-specified Single-step* process exemplified by the traditional waterfall or sequential Vee model is appropriate if the product's requirements are pre-specifiable and have a low probability of significant change and if there is no value or chance to deliver a partial product capability. A good example of this would be the hardware for an earth resources monitoring satellite that would be infeasible to modify after it goes into orbit.

The *Pre-specified Multi-step* process splits up the development in order to field an early initial operational capability and several P3I's. It is best if the product's full capabilities can be specified in advance and are at a low probability of significant change. This is useful in cases when waiting for the full system to be developed incurs a loss of important and deliverable incremental mission capabilities. A good example of this would be a well-understood and well-prioritized sequence of software upgrades for the on-board earth resources monitoring satellite.

The *Evolutionary Sequential* process develops an initial operational capability and upgrades it based on operational experience, as exemplified by agile methods. It is most needed in cases when there is a need to obtain operational feedback on an initial capability before defining and developing the next increment's content. A good example of this would be the software upgrades suggested by experiences with the satellite's payload, such as what kind of multi-spectral data collection and analysis capabilities are best for what kind of agriculture under what weather conditions.

The *Evolutionary Opportunistic* process defers the next increment until its new capabilities are available and mature enough to be added. It is best used when the increment does not need to wait for operational feedback, but it may need to wait for next-increment enablers such as technology maturity, external system capabilities, needed resources, or new value-adding opportunities. A good example of this would be the need to wait for agent-based satellite anomaly trend analysis

and mission-adaptation software to become predictably stable before incorporating it into a scheduled increment.

The *Evolutionary Concurrent* process, as realized in the incremental commitment spiral model (Pew and Mavor 2007; Boehm, et.al., 2014, page 75) and shown in Figure 2, has a continuing team of systems engineers handling the change traffic and re-baselining the plans and specifications for the next increment, while also keeping a development team stabilized for on-time, high-assurance delivery of the current increment and employing a concurrent verification and validation (V&V) team to perform continuous defect detection to enable even higher assurance levels. A good example of this would be the satellite's ground-based mission control and data handling software's next-increment re-baselining to adapt to new COTS releases and continuing user requests for data processing upgrades.

The satellite example illustrates the various ways in which the complex systems of the future, different parts of the system, and its software may evolve in a number of ways, once again affirming that there is no one-size-fits-all process for software evolution. However, Table 2 can be quite helpful in determining which processes are the best fits for evolving each part of the system. Additionally, the three-team model in Figure 2 provides a way for projects to develop the challenging software-intensive systems of the future that will need both adaptability to rapid change and high levels of assurance.

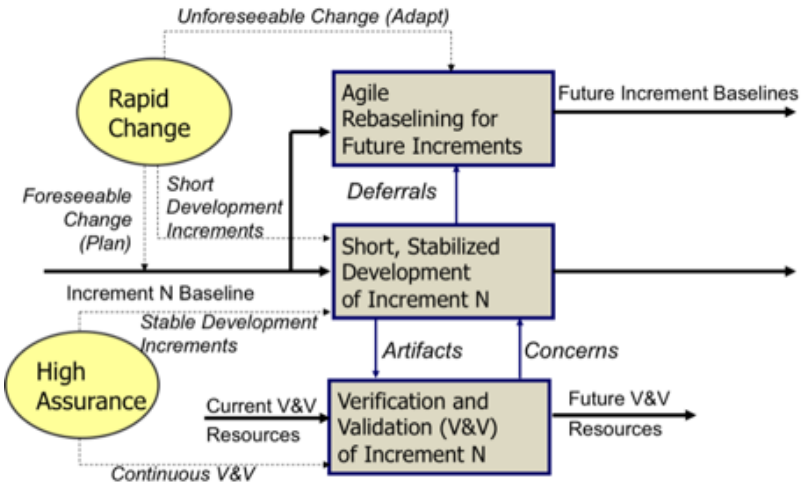

Figure 2. Evolutionary-Concurrent Rapid Change Handling and High Assurance (Pew and Mavor 2007, Figure 2-6).

References

Works Cited

Boehm, B. 2006. “Some Future Trends and Implications for Systems and Software Engineering Processes.” *Systems Engineering.* 9(1): 1-19.

Boehm, B. and J. Lane. 2007. “Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering.” *CrossTalk*. October 2007: 4-9.

Boehm, B. and J. Lane. 2010. *DoD Systems Engineering and Management Implications for Evolutionary Acquisition of Major Defense Systems.* SERC RT-5 report, March 2010. USC-CSSE-2010-500.

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.

Cusumano, M. and D. Yoffee. 1998. *Competing on Internet Time: Lessons from Netscape and Its Battle with Microsoft.* New York, NY, USA: Free Press.

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.

Primary References

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

None.

< Previous Article | Parent Article | Next Article >
SEBoK v. 2.9, released 20 November 2023

cess_Drivers_and_Choices&oldid=70055"
