LIFE CYCLES FOR SYSTEM ACQUISITION

Patterson, F.G., Jr.
George Mason University, Fairfax, VA., USA

Keywords: life cycle, acquisition, process, systems engineering

Contents

1. Introduction
1.1. Approaches
1.1.1. Grand Design
1.1.2. Incremental Design
1.1.3. Evolutionary Design
2. Commonly Used Life Cycles
2.1. Waterfall Cycles
2.2. Variations of the Waterfall Model
2.3. Concurrent Engineering Cycles
3. Current and Future Trends
3.1. ISO 9000
3.2. The Capability Maturity Models
3.3. SPICE
3.4. Acquisition Strategies
3.5. Concurrent Design
3.5.1. Deferring the Commitment
3.5.2. Shortening the Learning Curve
Glossary
Bibliography
Biographical Sketch

Summary

In this paper we look at a number of approaches for modeling the system acquisition process. Beginning with the waterfall model of Winston Royce, we continue with derivation that includes more intricate and detailed waterfall models with feedback, including the “V” and spiral models. In addition we look at several acquisition strategies. In reviewing the various models we discuss our view that all of these process models are stages in the evolution of the system acquisition process. As technology improves, simplification of the process is becoming increasingly possible and affordable. Our evidence of this trend includes the increased use of concurrent engineering and concurrent design. In *The Planning and Marketing Life Cycle* we will examine models for this cycle and their relationships to the acquisition and the research, development, test, and evaluation cycles.

1. Introduction

This paper is concerned with life cycles designed to provide guidance for the management of resources applied to the engineering development of products. As such,

life cycles provide the basis for a consistent acquisition or development strategy and are management tools. However, since the engineering of systems is routinely confronted with emergent properties that derive from the combination of their constituent parts, the study of life cycles is subject to many caveats that relate to the discovery, analysis, and treatment of these properties. Indeed, most variations and innovations in life cycle design seem to be addressed to this ubiquitous, challenging, and potentially fortunate problem.

1.1. Approaches

Three approaches are now widely recognized as strategies for product development under specific conditions: “grand design,” “incremental design,” and “evolutionary design.”

1.1.1. Grand Design

Grand design denotes a strategy for product design in which the product is designed in such a way that deployment is deferred until all development has been completed. Grand design is contrasted with incremental design and evolutionary design, both of which are described below. The grand design approach may be used for a product for which there is little expectation of requirements growth in the foreseeable future. Requirements growth may be dramatically reduced—or perhaps eliminated—by anticipating future needs. For example, in designing a new house, unused capacity can be intentionally built in, obviating the need for new design and construction at a later date. Requirements growth may also be limited by environmental, economic, political, or operational factors. An example would be rebuilding the Taj Mahal, which is constructed entirely of white marble—an unforgiving substance that is difficult to extend or change. Another example would be a single-use rocket, for which the lifetime of the operations phase would be a few minutes, quickly followed by the disposal phase. Yet another example would be a government-funded product for which future incremental or evolutionary funding might be extremely difficult to assure.

1.1.2. Incremental Design

Incremental development is a variation of the divide-and-conquer strategy in which a system is built in increments of functional capability. In this approach, as defined by Boehm, the first increment will be a basic, working system. Each successive increment will add functionality to yield a more capable working system. The several advantages of this approach include ease of testing, usefulness of each increment, and availability during development of user experiences with previous increments. Boehm’s modification of the waterfall to allow incremental development provides a number of successive copies of each section of the waterfall model, starting with detailed design; followed by code, integration and product verification, implementation and system test; and ending with deployment and revalidation. The spiral model provides a more succinct representation in which each successive increment is assigned to the next higher level of the spiral. An example of incremental development is the International Space Station.

1.1.3. Evolutionary Design

The evolutionary development model is an attempt to achieve incremental development of products whose requirements are not known in advance. Boehm discusses a process that may be used with iterative rapid prototyping—especially automatic program generation—and user feedback to develop a full-scale prototype. This prototype may be refined and delivered as a production system or it may serve as a *de facto* specification for new development. More generally, where evolutionary development is possible, it can be represented using a waterfall model in the fashion used by Boehm to represent incremental development. However, representing evolutionary development entails repetition of the requirements design and specification development activities, and so substantially more of the waterfall must be repeated. Again, the spiral model is a more natural representation, since deployment for a given level of the spiral is directly linked to the definition portion of the next higher level, allowing necessary requirements growth in an orderly fashion.

Many examples of evolutionary design are found in software, such as accounting software, which must change with every new tax regulation. Another example of evolutionary design is the Icehotel in Jukkasjärvi, Sweden, which is entirely rebuilt out of ice each November and melts each May. In 1989 it was a one-room igloo, but it has evolved into a 60-room design and grows with the hotel's requirements each year. It is interesting to note that the Icehotel could also serve as an example of grand design or incremental design, depending on several parameters. For example, since the structure melts each year, each "new" Icehotel is a grand design. However, since the design is reused from year to year, while perhaps adding rooms to the modular design, and since a part of the hotel is usable while the remainder is under construction, the hotel is also an example of incremental design. Finally, since many design decisions may be changed over time to add new functionality, such as the addition of a dining room or a chapel, the case for evolutionary design can be well made.

2. Commonly Used Life Cycles

The life cycle concept emerged as a model of the software process in the late 1960s, but it is now used more generally for the development of all types of systems. It is often argued that software engineering is a subdiscipline of systems engineering. Some authors call this "software systems engineering." Software engineering, with its beginning in computer programming, has been slow to emerge as a discipline at all and, despite many improvements, is still regarded by many as more art than engineering. Nonetheless, software engineering standards have continued to evolve.

-
-
-

Bibliography

Ahern D., Turner R., and Clouse A. (2001). *CMMI Distilled: A Practical Introduction to Integrated Process Improvement*, 336 pp. Boston: Addison-Wesley. [This guide captures the essential elements and basic spirit of the integrated CMM.]

Boehm B.W. (1981). *Prentice-Hall Advances in Computing Science and Technology Series. Software Engineering Economics*, 767 pp. Englewood Cliffs, NJ: Prentice-Hall. [This classic software engineering reference book addresses many subjects, including life cycles.]

Crosby P.B. (1979). *Quality is Free*, pp. 1–75. New York: McGraw-Hill. [The five-stage model introduced in this book is the basis of the family of CMMs.]

Johnson R.A., Kast F.E., and Rosenzweig J.E. (1967). *The Theory and Management of Systems*, pp. 99–147. New York: McGraw-Hill. [This early text on systems theory discusses several development life cycles.]

Patterson F.G., Jr. (1999). Systems engineering life cycles: life cycles for research, development, test, and evaluation; acquisition; and planning and marketing. *Handbook of Systems Engineering and Management* (ed. A.P. Sage and W.B. Rouse), pp. 59–112. New York: John Wiley. [Focuses on many life cycle architectures and provides many references.]

Sage A.P. (1992). *Systems Engineering*, 624 pp. New York: John Wiley. [Sage's textbook, containing extensive references, uses life cycles as a fundamental structure for addressing systems engineering.]

Biographical Sketch

F.G. Patterson Jr. received a B.A. degree from Rhodes College, two M.S. degrees from the University of Memphis, a Sc.M. degree from Brown University, and a Ph.D. degree from George Mason University: the most recent in 1996. He has been a faculty member at several universities, in departments of computer science and systems engineering; has worked as a systems engineer in the aerospace industry, and has worked for the National Aeronautics and Space Administration (NASA). His experience includes extensive assignments on large, complex systems undertaken by the Federal Aviation Administration and by NASA. His interests include systems engineering and systems management, especially the areas of process improvement, software reuse, and new paradigms for rapid system design.