

Comprehensive Question Four

By

Michael J. LeHew

A Paper Presented in Partial Fulfillment

Of the Requirements of

OM9995 Comprehensive Examination II

April 2004

Address:	3212 Orbison
City, State, Zip:	Tucson, AZ 85742
Phone:	520-256-3291
E-mail:	jefflehw@aol.com
Instructor:	Dr. Leibe
Mentor:	Dr. Jelena Vucetic

The Systems Development Life Cycle has often been used, modified, redefined and in the development of software systems. Analyze and compare the various models used in Systems Development Methodologies, as they impact the development of systems; Investigation, Analysis, Design, Implementation and Maintenance.

Table of Contents

Comprehensive Question	i
Table of Contents	ii
Executive Summary	4
Introduction	4
Literature Review	5
Discussion	9
Conclusion	11

Executive Summary

The Systems Development Life Cycle (SDLC), as originally conceived was not simply a means to develop software, or provide a cure all through the use of some technological means. On the contrary, it is first and foremost a methodology or approach to solving problems through a systematic and recursive process which allows the design teams the means and tools to, at various stages, iteratively assess the project, program or system prior to implementation.

This examination of some of the various historical evolution of the concept, in conjunction with some of the most current and applied practices in Systems methodologies will provide a synthesis of complimentary albeit not exact duplicates of the concept. Complimentary literature review, case analysis, and academic texts will be presented to present a broad analysis of the concepts, practicality and current focus on Systems.

Introduction

One of the earliest references to the Systems Development Life Cycle is provided by Daniels and Yeates (1971), in their book *Basic Training in Systems Analysis*. The name itself provides some insight into the intentions of the authors, and the subsequent modifications and customizations that have followed. At the foundation of the methodology is the development of systems, whether these systems are based or enhanced by technology is irrelevant. The system developed could impact only a workgroup or it may be as grand in scope to affect the Business Process through optimization. Through the years there have been a variety of evolutionary changes to the basic foundation concepts, but the core concept has remained relatively unchanged.

Systems development is an iterative process, which has a variety of steps that are accomplished in parallel or in order, but through the iterative and recursive nature allow the developer to return to a previous step to address issues that arise during the process. There are almost as many variations on the concept as there are systems, or companies that implement them. The basic foundation remains the same and adding steps or combining steps has very little effect on the outcome, as the entire process is repeated at intervals depending on the life span of the system in question.

The Systems Analyst using such a methodology is better equipped than ones plying their trade on an ad hoc basis. It is the nature of the formal methodology that is of key interest, and the variations of such methods is possibly good fodder for debate, analysis or study, but feasibly has

little effect on the complicated nature of corporate development and the use of in-house "shop rules" that we are faced with today. The global concept of the SDLC is the focus of this work. Several contemporary practices will be looked at, those not only incorporates the basic foundational principles of the SDLC, but add some complimentary capabilities that would otherwise not be available.

Project management principles and product development, both of a technical nature and of a manual nature, all use some form of methodology. The difference between the methods used in practice, and the theoretical model of the steps taken, differ in application, documentation and procedural steps that are followed. The primary root or core of the SDLC remains the same. The SDLC is a procedural process with which to solve problems. Not the answer in and of itself, but a means to derive the solution and provide a checks and balances aspect to systems design.

It is some importance at this stage to indicate that one of the most important aspects of each stage of the SDLC is documentation. There are reporting requirements that may or may not make sense in our daily lives, but when generating systems changes, it is imperative to document each step of the way. The use of prior knowledge and lessons learned has proved an invaluable tool in the development of new systems in the past, and a nightmare for those of us who have faced systems projects, with little or no legacy documentation to work from.

The iterative nature of the SDLC is based upon basic Boolean Logic and Critical Thinking Skills, applied to Systems Development. Is the project feasible? Is it in our interest to produce or

develop the systems change? If the answer is yes, go on to the next step. If not, return to the Feasibility Study and either look for another solution, or address a different problem. Not all systems problems are feasible to solve based upon Time, Quality and or Money variables. Not all feasible systems changes are actually in the long term interest to implement within a system. There are so many variables that go into the process itself, from which the macro "steps" can lose meaning, or get ignored in the rush to implementation.

The focus on this paper is to evaluate a portion of the literature, texts, and work done to develop and enhance the SDLC, while presenting the core intention of the process, while examining the overall effectiveness of the SDLC as a tool to develop systems. The development of the SDLC, and in fact is continually being refined, will be examined, while maintaining a macro view of the process as a whole. Each organization, consortium, institution or academic group seems to have a variation on the basic SDLC, which precludes addressing each in turn. Each of these however, contains the same basic concepts, although labels and categories are expanded or collapsed to suit the detail requirements of the organization.

Literature Review

O'Brien, (2002) defines the SDLC along with Systems Development as "conceiving, designing and implementing a system; developing information systems by a process of investigation, analysis, design, implementation and maintenance". Early reference to the life-cycle composes the methodology into the following steps, originally coined as the "waterfall model". (Daniels & Yeates, 1971)

- Feasibility study
 - Needs assessment
 - Feasibility
 - Available resources (Monetary and Physical)
- System investigation
 - Current System
 - Competing Systems
 - Technological Advancements
- Systems analysis
 - Possible Solutions
 - Analysis of returns based upon courses of action
- Systems design
 - Systems Modeling
 - CASE
 - Documentation

- Specification
- Implementation
 - Development
 - Prototyping
 - Piloting
 - Implementing
 - Testing
- Review and maintenance
 - Support
 - Documentation
 - Performance Tuning
 - Reporting

(Daniels & Yeates, 1971)

The feasibility study establishes the beginning of the SDLC by determining the need for improvement or change, the practicality or feasibility of success and the available resources to accomplish the task. Feasibility and profitability should not be confused at this point, as there are a number of projects that may have a high feasibility, but do not actually offer any strategic interest to the corporation in question.

Systems investigation begins with a thorough understanding of the capabilities of the current system and the hidden value that said system may provide. It is imperative that all subject matter

experts, end users and key stake holders in a system be thoroughly probed for information in order to gain a comprehensive understanding of the system, prior to any prospective change analysis being conducted.

Review and maintenance, combined into a single phase, is focused on the deployed system. It may be the iterative implantation of a portion of the enterprise or it may be reached only when the entire system is in place. This phase is continually be assessed by managers watching the daily performance results and industry trends to ensure that the system in place is providing maximum return for the investment, while maintaining a competitive advantage when compared to other systems.

Analysis of the current system, its benefits and shortcomings coupled with a modeling of alternative scenarios follows the investigation, to produce a working idea of what the new system will look like. This analysis should include a strategic, rather than short sighted viewpoint of where any changes will affect the corporate return on investing in the new system. Empirical results of proposed scenarios, while not always feasible can be gathered from internal review of similar changes, or from the experiences of external users of a proposed systems change.

Designing systems today is quite complex, but does have the advantage of a myriad of Computer Assisted Software Engineering (CASE) tools, that not only help to model systems changes, but allow the design team to model load tolerance, run simulations against the systems changes while gathering statistical information based upon these simulations. The time saved in documenting the changes, and producing searchable documentation, standardized data naming and structures is by far one of the major advantages of using CASE tools. Most high end, and I

might add high priced, CASE tools allow for sharing in a virtual teaming environment to track changes in real time, while preventing the duplication of effort on a component of a system by way of a repository, requiring designers to "check in and out" the component of the system that they are working on.

Implementation is a complex phase of the SDLC that includes building a prototype, or mock-up of the intended system change in order to further test, and receive feedback from users, stakeholders and management along the process track. The creation of prototypes allows for a variety of alternatives to be tested, and a best case solution for the scenario in question to be selected. Testing, and the accompanying documentation of the results, is key to the implementation phase and the entire SDLC in general. Often confused as a portion of only the implementation phase, "testing" is inevitably the driving force behind beginning the cycle once more, upon delivery. When a system fails various performance tests, it is returned to the feasibility phase to determine an alternative, enhancement, change or total redesign.

The need for a formal methodology therefore is not a new concept, and has been researched and documented by other researchers. Rockart and Flannery (1983) presented that at the time between 40-50 percent of user applications were developed ad hoc, for work group or local use. Locally developed applications today with the ability enhancements of twenty one years of increased performance in personal computing, and desktop applications, would suggest that a greater percentage than previously reported are in place and actively used daily. Not surprisingly, more resources are spent on maintaining software than for its development.

Maintenance costs for large scale software systems can amount to somewhere between 40% and 67% of the total system life cycle cost (Carr and Wagner, 2002).

One corporations approach, Iterative Development in the Field (IDF), by IBM, is characterized by repeated evaluation and redesign cycles that are carried out throughout the product life cycle, from initial discovery and gathering of requirements to beyond deployment in the field (Green, et al, 2002). The use of IDF is characterized by the SDLC principles being applied by developers in conjunction with customers or end users. This approach is intended to speed the process of delivering systems enhancements or changes to existing systems.

The Systems Development Life Cycle is a structured model for managing the development of almost any project. One model (Shelly, Cashman, & Rosenblatt, 1998) which will be used for further discussion, proposes that there are five phases or stages during which specific functions are defined and assessed as part of the iterative process.

1. The systems planning phase begins a request for the initiation of a project by managers at some level through the use of various documentation, specific to the industry, organization or activity. This document triggers a preliminary investigation or feasibility study which will determine the continuance of project development. Stakeholders in the form of managers, users, internal and external customers for example, are provided a report, describing the impact of the project, or systems implementation, based upon the technical, economic, or other operational impacts. This reporting process determines if the project will continue and at what level of funding or resource allocation. There is a prioritization based upon risk and

benefit analysis that generally accompanies the assignment of project resources and their relative importance to the organization from a strategic standpoint.

2. The systems analysis process investigates the various options available based upon the current system, possible alternatives and expected outcomes. The investigator analyzes the current system through a variety of means, and then documents the necessary procedures used to accomplish the tasks. This results in proposed solutions or alternatives available to the stakeholders, and is included in the process as an iterative component. Often, a systems Requirement Document, is developed, which will not limit the delivery of the requested changes, but, on the contrary, define what exactly is needed in order to reach various solutions to the problem or task in question.
3. Gaining approval from the preceding phases, the Analyst (team) can proceed into the design of the proposed solutions in the form of a Systems Design Document, often accompanied with models, prototypes and iterative implementations of the system. The research and modeling of the system, generally through the use of Computer Assisted Software Engineering tools (CASE) today, will generate a valid set of documentation that allows the implementation team a wide range of complexity, but with uniform structure and compliance to project specifications.
4. The implementation stage addresses the actual development of the system, by following the software design documents developed in the preceding phase. Software developers may write the code, Database Architects may build the database, or Network and Telecommunications Specialists would install and configure new hardware and software.. The primary goal of the implementation phase is to establish functional implementation of the designed system,

which may be in the form of the entire system at once, or a phased implementation that utilizes a cluster, or portion of the implementation until final acceptance and testing is accomplished.

5. Once the system is functioning, the system enters the support phase, where information systems personnel must maintain, document and continually monitor performance of the system. It is of some importance that the documentation of issues, and the subsequent fixes are accounted for, as well as performance tuning issues. It is by way of the maintenance phase that the SDLC begins once more, as systems maintenance or performance issues exceed the worth of the system. The process would then begin with feasibility once more.

Discussion

The variety and magnitude of systems in place in the global information systems mandate that Information Systems Professionals follow the tenets of the basic SDLC, in the spirit of the original design. Shop rules, localizations, and modifications, are inevitable, but must not interfere with the foundations of systems design. Continued research and refinement into the most efficient method to modify a system, should remain in flux. The very nature of information systems advancements has allowed the developer greater flexibility in customization of deployed systems to suit the local needs.

Often lost in the rush to bring product to market or to enhance existing systems is the necessary documentation to make the next iteration more seamless improvement. It has been my experience that, although the SDLC may be followed albeit informally, there are a large number of organizations that do not adhere to the need for documenting the steps taken along the way.

It is by way of adequate documentation that continuity is maintained, and repeated mistakes are not costly to an organization. The use of the SDLC and thoroughly documenting all iterations, lessons learned and the pitfalls determined along the way, will not only allow for greater knowledge management for the developers at present, but add to the body of knowledge for future developers, when the current system becomes a legacy system.

References

- Benamati, J. & Lederer, (2001). The effect of rapid IT change on the demand for training. *Communications of the Association of Computing Machinery* (ACM No. 1-58113-363-1/01/04)
- Brancheau, J., Janz, B., & Wetherbe, J. (1996, June). Key issues in information systems management: 1994-95 SIM delphi results. *MIS Quarterly* 20 (2) 225-242.
- Carr, M., & Wagner, C. (2002). A Study of Reasoning Processes in Software Maintenance Management. *Information Technology and Management*, 3(1-2), 181.
- Cheney, P. (1988). Information systems skills requirements: 1980 & 1988. *Communications of the Association of Computing Machinery* (ACM No. 0-89791-262-4/88/0400-0001)
- Daniels, A. and Yeates, D. A. (1971). *Basic Training in Systems Analysis*. London: Pitman
- Englishman, C. & Isaacson, G. (1995). Information Protection Awareness. In Z. G. Ruthberg &
- Gallivan, M. (1994). Changes in the management of the information system organization: An exploratory study. *Communications of the Association of Computing Machinery* (ACM No. 0-89791-652-2/94/0003)
- Grindler, G. (1993). Business continuity planning. In Z. G. Ruthberg & H. F. Tipton (Eds.) *Handbook of information security management* (pp. 179-190). Boston: Auerbach.
- Greene, S. L., Jones, L., Matchen, P., & Thomas, J. C. (2003). Iterative development in the field. *IBM Systems Journal*, 42(4), 594.
- Levinson, N. (1988). Repositioning the information systems management function: Implications for information systems personnel. *Communications of the Association of Computing Machinery* (ACM No. 0-89791-262-4/88/0400-0167)
- Luftman, J., Papp, R., & Brier, T. (1999, March). Enablers and inhibitors of business-IT alignment. *Communications of the Association of Information Systems*, 1 (11).
- Maxwell, J. (1999). *The 21 indispensable qualities of a leader*. Nashville, TN: Thomas Nelson Publishers.
- McLeod, R. & Schell, G. (2001). *Management information systems* (8th ed.). Upper Saddle River, NJ: Prentice Hall.

- Morgan, G. (1997). *Images of organization*. Thousand Oaks, CA: Sage Publications.
- Rockart, J. & Flannery, L. (1983, October). The management of end-user computing. *Communications of the Association of Computing Machinery*, 26 (10).
- Senn, J. (1998). *Information technology in business: Principles, practices, and opportunities*. Upper Saddle River, NJ: Prentice Hall.
- Shelly, G., Cashman, T., & Rosenblatt, H. (1998). *Systems analysis and design* (3rd ed.). Cambridge, MA: Course Technology.
- Tipton, H. & Krause, M. (Eds.). (2000). *Information security management handbook*. Boca Raton, FL: Auerbach Publications.
- Westerback, L. (1999). Toward best practices for strategic information systems management. *ProQuest Digital Dissertations* (UMI No. 9