# Software Development

Hamza Ahmed

**Abstract:** Software development is a rigorous process used in the creation and maintenance of software. In this paper the systems development lifecycle (SDLC) has been explored as the universal model upon which other models are based. Examples of software crises emerging from software failures and errors have also been discussed in a bid to aid in the appreciation for the need software development lifecycle process. The SDLC phases i.e. planning, analysis, design and implementation are discussed in much detail and the importance of each of this process right from when the software is conceived as an idea by the project sponsor, to when the software is actually installed and used to add value to the organization.

The paper also depicts the SDLC as a general software development lifecycle model that can be enhanced to create context specific software process models. In fact, most software process models such as waterfall, incremental and prototyping models all have aspects of the four phases in the SDLC but with unique variations to suit software development needs of each project. The development of highly reliable, quality and cost-effective software also relies on the application of good project management practices in the SLDC to ensure all technical, functional and organizational requirements are met.

—————————— ◆ ————————

## Introduction

Software refers to the programs that interact with computer hardware (physical electronic components) to produce the desired output or information. Software is categorized as system or application software. System software is responsible for direct interaction with hardware and offers crucial services to the user. Examples of system software include the computer operating system, device drivers, antivirus software, and utility software that provides tasks such as disk management and data backup. Application software refers to programs that aids in delivering daily business services, user support, and entertainment. In fact, application software is tailored to specific functions to meet particular needs. Examples of such software include word processors, database software, computer games, web browsers and media players (Shelly & Rosenblatt, 2010).

Software development refers to the iterative process of creating and maintaining software systems. The most common technique used in software development is the systems development lifecycle (SDLC) (Kroenke, 2012). The SDLC is a fundamental four-phase process consisting of the planning, analysis, design and implementation phases. It is also the established basic framework used in the general development of information systems. The paper describes the SDLC process and its significance in ensuring delivery of quality and reliable software at the right time and within the budget constraints.

## The importance of the software development lifecycle

The systems development lifecycle (SDLC) is applied to the development of all information systems, and, in this case, software development. The process involves determining how the software will support business needs, the design, building and implementing the design, and how to deliver the final product to the intended users. From a theoretical standpoint, the process seems simple but in real-world software development scenarios, the process is quite complex. In order to appreciate the critical nature of why the SDLC is needed, some business examples are discussed.

In the year 2010 alone, estimates show that organizations and governments spent about $2.4 trillion on hardware, software and IT services globally with a predicted spending increase of 3.5% in the subsequent year (Pettey & Goasduff, 2010). A previous study conducted in 2008 showed that the success of 68% of technology projects, especially those involving software development was improbable. The study showed that most systems/software development projects were either abandoned prior to completion, were delivered over budget or significantly late, or with incomplete features that did not match the requirements (IAG Consulting, 2008). In 2009, a follow-up study was conducted in an attempt to quantify the failure rate, and the costs were found to place a $6.2 trillion toll on the global economy (Sessions, 2009). These specific estimates were rigorously questioned by some quarters but it is evident that even if the figure was halved, the cost of system development failures would still be a staggering number in terms of the high project failure rate, and the costs incurred due to failure (Krigsman, 2009).

Some examples of costly and embarrassing software glitches and failures that faced organizations and governments in the year 2010 only are discussed. US toy distributor, Toys R Us experienced a software glitch that resulted in the double billing of shoppers on Black Friday purchases. Telecommunications giant, Verizon Wireless also had to make refunds worth $50 million to its customers due to glitches in its billing system. Banks too were not left out since a software glitch locked out Chase bank online banking clients from their accounts for 24 hours. A glitch in the McAfee antivirus software caused the clients' computers to lock, and the company had to offer reimbursements for machine repair costs and free 2-year subscription to the antivirus package to affected customers. In matters national security, a software issue caused operators of a U.S. Navy drone (UAV-unmanned aerial vehicle) to lose control over the craft causing it to fly into restricted airspace near Washington DC. The operators regained control of the device after 20 minutes (Krigsman, 2010). However, while there have been countless books, journal articles and methods designed to reduce system/software project failures, it is important to understand that there is no "silver bullet" to guarantee 100% success rate on software development projects since it does not exist (Brooks, 1987). However, by following the core concepts and practical methodologies of the SDLC, there is a higher probability of improving the success rate of software development projects.

**The Systems Development Lifecycle (SDLC)**

Software development using the systems development life cycle is a four-phase process involving planning, analysis, software design, and implementation. Each phase of the SDLC has its own subset of steps that rely on methods that yield deliverables at each stage i.e. documentation, files and/or similar output that explains various system aspects (NARA, 2007). Ideally, the SDLC phases proceed logically from project start to finish which is true for some software projects. However, in most projects, the development teams move through the steps consecutively, iteratively, incrementally or following other patterns to ensure all requirements are met. Different teams and software projects may place particular emphasis on some aspects of the SDLC or approach the SDLC phases differently, but all software development projects have aspects of the four phases of the SDLC (Dennis, Wixom & Roth, 2012).

The paper provides an overview of SDLC phases, stages and techniques used to yield the deliverables in these stages. It is also critical to note that the SDLC is a gradual refinement process, and the general idea of software functionality is realized after observing the deliverables in the analysis phase. The output of the analysis is used in the design which then refines the output to produce a new extended set of deliverables on how exactly the system should be built. The output of the design phase is used to guide actual system creation in the implementation phase. In this regard, each SDLC phase is seen to elaborate, refine and extend work done in previous phases (Dennis, Wixom & Roth, 2012).

**1. Planning Phase:**

This is the first phase of the SDLC, and it involves understanding why the software should be built and how the development team will start working on the project. The planning phase has two major steps which are project initiation and project management. The initiation stage involves identifying the business value of the software to the organization in terms of cost lowering and/or raising revenue. Most new ideas about the software originate from departments other than IT such as market research, accounting, and human resources in the form of system requests. A system request is a document that provides a business need in brief and explains how the software will solve the need and create value for the business. The IT department then works with the department or individual that made the request (project sponsor) to conduct a formal feasibility analysis. The feasibility analysis seeks to assess the key aspects of the project proposed in terms of technical, economic and organizational feasibility. Technical feasibility assesses whether the software can be built from a technical aspect, economic feasibility assesses the business value of the software while organizational feasibility assesses whether the software will be used if built. The deliverables in this stage are the feasibility analysis and system request documents which as submitted to the steering committee for approval (Dennis, Wixom & Roth, 2012).

If the software project is approved, it enters the project management stage where a project manager develops a work plan, provides staffing, and posts techniques to aid the software development team in directing and controlling the project throughout the course of the SDLC. The deliverable in this stage is a project plan describing the path to be followed by the team developing the software.

2. Analysis Phase:

In this phase, the software users and the expected functionality are determined. Other issues that are considered include when and where the software will be used. In this phase, therefore, the software development team evaluates existing software systems, identifies improvement areas, and develops the new software's concept. There are three steps in this phase which include an analysis strategy, requirements gathering and developing a system proposal.

The analysis strategy is created to aid the development team and usually contains information on the current software (if any), its current limitations, and possible ways to design the new software. In the requirements gathering stage, information is collected using questionnaires, interviews, and discussions and analyzed together with input from the project sponsor and other stakeholders. The results of this analysis lead to the development of the new software/system concept. The system concept is used as a basis for developing various business analysis models describing business operations if the new software/system, is installed. The set produced typically consists of models representing the necessary data and processes to support the main underlying business process.

In the third stage of the analysis phase, the system concept and business model are combined into a system proposal document that is presented to the project sponsor and other stakeholders e.g. steering committee members and top management who then decide whether the software project should progress. The initial deliverable is the system proposal since it describes the business requirements to be met by the software, and it is also the first tangible step to developing the new software. The deliverable of this phase is the system analysis and an initial high-level design for the new system (Dennis, Wixom & Roth, 2012).

**3. Design Phase:**

In this phase, decisions are made in terms of how the software will operate in terms of hardware, software dependencies, and network infrastructure. The user interface, form inputs, reports, databases and files required are also specified in this section. While most strategic decisions about the software being developed are made during the development of the system concept, the design stage is the one that determines exactly how the software will work. This phase has four stages i.e. developing a design strategy, architecture development, specifying files and databases, and program design.

The design strategy step involves clarifying whether software development will be done by the company programmers, outsourced to a third party vendors or whether an off-the-shelf package is ideal. The second step involves designing the architecture of the software describing the hardware, network resources and software needed. In most cases, the software/system adds to an already existing system infrastructure within the organization. Interface design will specify how users will navigate while interacting with the software and well as modes of data input such as forms, and output such as reports (Dennis, Wixom & Roth, 2012).

The third step in the design phase involves developing file and database specifications to determine exactly what data will be stored, and how and where it will be stored. The final design step is where the team of analysts develops a program design defining the program modules to be coded and the functionality delivered by each of the programs.

The deliverables in the design phase i.e. interface design, architecture design, file and database specifications, and the program design is the final system specification that the software programmers will use to implement the system. At the end of design, the project plan and feasibility analysis are also reassessed and revised by the steering/approval committee and project sponsor who collectively determine whether to continue with software development of terminate the project entirely (Dennis, Wixom & Roth, 2012).

**4. Implementation:**

This is the final SDLC phase where the software is actually built/coded or purchased (in case an off-the-shelf option was picked). This is the most critical phase of the SDLC, and also the most expensive since it is here that all the system requirements are actualized. The implementation phase has three steps which include: system construction and testing, system installation, and the establishment of the system support plan.

In the system construction and testing stage, the programmers, user interface designers code the program modules and interfaces as specified in the design, and also test them to ensure that the various components work and interact as designed. Due to the high cost of bug fixing, testing is a crucial aspect of the implementation process, and most organizations and developers will give more attention to system testing that on writing software code in the first place. After the system has been tested, and the data inputs and output verified and validated, the system is ready for installation (Dennis, Wixom & Roth, 2012).

In the installation step, the new software is turned on, and the old system turned off. There are several approaches that could be used when converting from the old to the new system but even then one of the crucial deliverables of any conversion technique is a sound training plan to educate users on how to use the new software and help manage new changes introduced by the system. After installation, the final implementation step involves the development of a support plan for the software/system developed. The plan is usually developed by the system analysts, and includes both formal and informal post-implementation reviews, and systematic methods of identifying any minor or changes needed by the system, and ways of documenting these changes to ensure all records are up to date (Dennis, Wixom & Roth, 2012).

**Summary and Conclusion:**

In conclusion, it is evident that software development is a critical business activity that could increase the business value if well implemented or turn into a costly affair in case of failures. In this regard, the development of high quality, reliable and cost-effective software relies heavily on the management processes and careful implementation of steps in the software development lifecycle. The SDLC phases described in this paper are the general phases to be found in any software development models. In fact, there exist different SDLC process models such as the incremental model, waterfall model, iterative model, rapid application development, agile development and other process models which all follow the SDLC lifecycle but with some variations depending on the software type, organization scale, time and budget constraints (ISTQB, 2015; Tutorialspoint.com, 2015). However, due to the scope limitation of this paper, it was prudent to discuss the SDLC in general since all the specific process models inherit the basic SDLC phases in one aspect or another.

**References**

Brooks,. (1987). No Silver Bullet Essence and Accidents of Software Engineering. Computer, 20(4), 10-19. doi:10.1109/mc.1987.1663532

Dennis, A., Wixom, B., & Roth, R. (2012). System analysis and design. Hoboken, NJ: John Wiley.

Freetutes.com, (2015). Software (System Development) life Cycle Models. [online] Freetutes.com. Available at: http://www.freetutes.com/systemanalysis/sa002-software-life-cycle-models.html [Accessed 24 Apr. 2015].

Goldsmith, R. (2015). Software development life cycle phases, iterations, explained step by step. [online] SearchSoftwareQuality. Available at: http://searchsoftwarequality.techtarget.com/answer/Software-development-life-cycle-phases-iterations-explained-step-by-step [Accessed 25 Apr. 2015].

IAG Consulting,. (2008). Press Release: Business Analysis Benchmark 2008 | IAG Consulting. Iag.biz. Retrieved 24 April 2015, from http://www.iag.biz/about-iag/news-events/press-release-business-analysis-benchmark-2008.html

ISTQB,. (2015). What are the Software Development Life Cycle (SDLC) phases?. Istqbexamcertification.com. Retrieved 25 April 2015, from http://istqbexamcertification.com/what-are-the-software-development-life-cycle-sdlc-phases/

Krigsman, M. (2009). Critique: $6.2 trillion global IT failure stats | ZDNet. ZDNet. Retrieved 24 April 2015, from http://www.zdnet.com/article/critique-6-2-trillion-global-it-failure-stats/

Krigsman, M. (2010). Ten great software glitches for 2010 | ZDNet. ZDNet. Retrieved 24 April 2015, from http://www.zdnet.com/article/ten-great-software-glitches-for-2010/?tag=mantle_skin;content

Krigsman, M. (2015). Beyond IT Failure | ZDNet. ZDNet. Retrieved 25 April 2015, from http://www.zdnet.com/blog/projectfailures/

Kroenke, D. (2012). Using MIS. Upper Saddle River, N.J.: Prentice Hall.

NARA,. (2007). Systems Development Life Cycle Checklists (1st ed.). National Archives and Records Administration (NARA). Retrieved from http://www.archives.gov/records-mgmt/initiatives/sdlc-checklist.pdf

Pettey, C., & Goasduff, L. (2010). Gartner Says Worldwide Enterprise IT Spending on Pace to Grow 2.9 Percent in 2010. Gartner.com. Retrieved 24 April 2015, from http://www.gartner.com/newsroom/id/1419513

Sessions, R. (2009). The IT Complexity Crisis: Danger and Opportunity (1st ed.). Object Watch. Retrieved from http://imap.objectwatch.com/whitepapers/ITComplexityWhitePaper.pdf

Shelly, G., & Rosenblatt, H. (2010). Systems analysis and design. Boston, Mass.: Thomson Course Technology.

Tutorialspoint.com,. (2015). SDLC - Overview. Tutorialspoint.com. Retrieved 25 April 2015, from http://www.tutorialspoint.com/sdlc/sdlc_overview.htm