# Systems development life cycle

**6.1 INTRODUCTION:** The **systems development life cycle (SDLC)**, also referred to as the **application development life-cycle**, is a term used in systems engineering, information systems and software engineering to describe a process for planning, creating, testing, and deploying an information system. The systems development life-cycle concept applies to a range of hardware and software configurations, as a system can be composed of hardware only, software only, or a combination of both.

## 6.2 Overview

A systems development life cycle is composed of a number of clearly defined and distinct work phases which are used by systems engineers and systems developers to plan for, design, build, test, and deliver information systems. Like anything that is manufactured on an assembly line, an SDLC aims to produce high quality systems that meet or exceed customer expectations, based on customer requirements, by delivering systems which move through each clearly defined phase, within scheduled time-frames and cost estimates. Computer systems are complex and often (especially with the recent rise of service-oriented architecture) link multiple traditional systems potentially supplied by different software vendors. To manage this level of complexity, a number of SDLC models or methodologies have been created, such as "waterfall"; "spiral"; "Agile software development"; "rapid prototyping"; "incremental"; and "synchronize and stabilize".

SDLC can be described along a spectrum of agile to iterative to sequential. Agile methodologies, such as XP and Scrum, focus on lightweight processes which allow for rapid changes (without necessarily following the pattern of SDLC approach) along the development cycle. Iterative methodologies, such as Rational Unified Process and dynamic systems development method, focus on limited project scope and expanding or improving products by multiple iterations. Sequential or big-design-up-front (BDUF) models, such as waterfall, focus on complete and correct planning to guide large projects and risks to successful and predictable results. Other models, such as anamorphic development, tend to focus on a form of development that is guided by project scope and adaptive iterations of feature development.

In project management a project can be defined both with a project life cycle (PLC) and an SDLC, during which slightly different activities occur. According to Taylor (2004) "the project life cycle encompasses all the activities of the project, while the systems development life cycle focuses on realizing the product requirements".

SDLC is used during the development of an IT project, it describes the different stages involved in the project from the drawing board, through the completion of the project.

## 6.3 History

The product life cycle describes the process for building information systems in a very deliberate, structured and methodical way, reiterating each stage of the product's life. The systems development life cycle, according to Elliott & Strachan & Radford (2004), "originated in the 1960s, to develop large scale functional business systems in an age of large scale business conglomerates. Information systems activities revolved around heavy data processing and number crunching routines".

Several systems development frameworks have been partly based on SDLC, such as the structured systems analysis and design method (SSADM) produced for the UK government Office of Government Commerce in the 1980s. Ever since, according to Elliott (2004), "the traditional life cycle approaches to systems development have been increasingly replaced with alternative approaches and frameworks, which attempted to overcome some of the inherent deficiencies of the traditional SDLC".

## Phases

The system development life cycle framework provides a sequence of activities for system designers and developers to follow. It consists of a set of steps or phases in which each phase of the SDLC uses the results of the previous one.

The SDLC adheres to important phases that are essential for developers, such as planning, analysis, design, and implementation, and are explained in the section below. It includes evaluation of present system, information gathering, and feasibility study and request approval. A number of SDLC models have been created: waterfall, fountain, and spiral, build and fix, rapid prototyping, incremental, and synchronize and stabilize. The oldest of these, and the best known, is the waterfall model: a sequence of stages in which the output of each stage becomes the input for the next. These stages can be characterized and divided up in different ways, including the following:

- **Preliminary analysis**: The objective of phase 1 is to conduct a preliminary analysis, propose alternative solutions, describe costs and benefits and submit a preliminary plan with recommendations.

  Conduct the preliminary analysis: in this step, you need to find out the organization's objectives and the nature and scope of the problem under study. Even if a problem refers only to a small segment of the organization itself then you need to find out what the objectives of the organization itself are. Then you need to see how the problem being studied fits in with them. Propose alternative solutions: In digging into the organization's objectives and specific problems, you may have already covered some solutions. Alternate proposals may come from interviewing employees, clients, suppliers, and/or consultants. You can also study what competitors are doing. With this data, you will have three choices: leave the system as is, improve it, or develop a new system.

  Describe the costs and benefits.

- **Systems analysis, requirements definition**: Defines project goals into defined functions and operation of the intended application. Analyzes end-user information needs.

- **Systems design**: Describes desired features and operations in detail, including screen layouts, business rules, process diagrams, pseudo-code and other documentation.

- **Development**: The real code is written here.

- **Integration and testing**: Brings all the pieces together into a special testing environment, then checks for errors, bugs and interoperability.

- **Acceptance, installation, deployment**: The final stage of initial development, where the software is put into production and runs actual business.

- **Maintenance**: During the maintenance stage of the SDLC, the system is assessed to ensure it does not become obsolete. This is also where changes are made to initial software. It involves continuous evaluation of the system in terms of its performance.
- **Evaluation**: Some companies do not view this as an official stage of the SDLC, but is it an important part of the life cycle. Evaluation step is an

extension of the Maintenance stage, and may be referred to in some circles as Post-implementation Review. This is where the system that was developed, as well as the entire process, is evaluated. Some of the questions that need to be answered include: does the newly implemented system meet the initial business requirements and objectives? Is the system reliable and fault-tolerant? Does the system function according to the approved functional requirements? In addition to evaluating the software that was released, it is important to assess the effectiveness of the development process. If there are any aspects of the entire process, or certain stages, that management is not satisfied with, this is the time to improve. Evaluation and assessment is a difficult issue. However, the company must reflect on the process and address weaknesses.

- **Disposal:** In this phase, plans are developed for discarding system information, hardware and software in making the transition to a new system. The purpose here is to properly move, archive, discard or destroy information, hardware and software that is being replaced, in a matter that prevents any possibility of unauthorized disclosure of sensitive data. The disposal activities ensure proper migration to a new system. Particular emphasis is given to proper preservation and archival of data processed by the previous system. All of this should be done in accordance with the organization's security requirements.[8]

In the following example these stages of the systems development life cycle are divided in ten steps from definition to creation and modification of IT work products:

The tenth phase occurs when the system is disposed of and the task performed is either eliminated or transferred to other systems. The tasks and work products for each phase are described in subsequent chapters.

Not every project will require that the phases be sequentially executed. However, the phases are interdependent. Depending upon the size and complexity of the project, phases may be combined or may overlap.

## System investigation

The system investigate the IT proposal. During this step, we must consider all current priorities that would be affected and how they should be handled. Before any system planning is done, a feasibility study should be conducted to determine if creating a new or improved system is a viable solution. This will help to determine the costs, benefits, resource requirements, and specific user needs required for completion. The development process can only continue once management approves of the recommendations from the feasibility study.

Following are different components of the feasibility study:

- Operational feasibility
- Economic feasibility
- Technical feasibility
- Human factors feasibility
- Legal/Political feasibility

**System analysis**

The goal of system analysis is to determine where the problem is in an attempt to fix the system. This step involves breaking down the system in different pieces to analyze the situation, analyzing project goals, breaking down what needs to be created and attempting to engage users so that definite requirements can be defined.

**Design**

In systems design, the design functions and operations are described in detail, including screen layouts, business rules, process diagrams and other documentation. The output of this stage will describe the new system as a collection of modules or subsystems.

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts.

Design elements describe the desired system features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo-code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the system in sufficient detail, such that skilled developers and engineers may develop and deliver the system with minimal additional input design.

**Environments**

Environments are controlled areas where systems developers can build, distribute, install, configure, test, and execute systems that move through the SDLC. Each environment is aligned with different areas of the SDLC and is intended to have specific purposes. Examples of such environments include the:

- *Development environment*, where developers can work independently of each other before trying to merge their work with the work of others,
- *Common build environment*, where merged work can be built, together, as a combined system,
- *Systems integration testing environment*, where basic testing of a system's integration points to other upstream or downstream systems can be tested,
- *User acceptance testing environment*, where business stakeholders can test against their original business requirements,
- *Production environment*, where systems finally get deployed to, for final use by their intended end users.

The planning for, provisioning, and operating of such environments is known as practice of IT environment management.

**Testing**

The code is tested at various levels in software testing. Unit, system and user acceptance testing's are often performed. This is a grey area as many different opinions exist as to what the stages of testing are and how much, if any iteration occurs. Iteration is not generally part of the waterfall model, but usually some occur at this stage. In the testing the whole system is tested one by one

Following are the types of testing:

- *Defect testing* the failed scenarios, including defect tracking
- Path testing
- Data set testing
- Unit testing
- System testing
- Integration testing
- Black-box testing
- White-box testing
- Regression testing
- Automation testing
- User acceptance testing
- Software performance testing

## Training and transition

Once a system has been stabilized through adequate testing, the SDLC ensures that proper training on the system is performed or documented before transitioning the system to its support staff and end users.

Training usually covers operational training for those people who will be responsible for supporting the system as well as training for those end users who will be using the system after its delivery to a production operating environment.

After training has been successfully completed, systems engineers and developers transition the system to its final production environment, where it is intended to be used by its end users and supported by its support and operations staff.

## Operations and maintenance

The deployment of the system includes changes and enhancements before the decommissioning or sunset of the system. Maintaining the system is an important aspect of SDLC. As key personnel change positions in the organization, new changes will be implemented. There are two approaches to system development; there is the traditional approach (structured) and object oriented. Information Engineering includes the traditional system approach, which is also called the structured analysis and design technique. The object oriented approach views the information system as a collection of objects that are integrated with each other to make a full and complete information system.

## Evaluation

The final phase of the SDLC is to measure the effectiveness of the application and evaluate potential enhancements.

## Systems analysis and design

The **systems analysis and design (SAD)** is the process of developing information systems (IS) that effectively use hardware, software, data, processes, and people to support the company's businesses objectives. System analysis and design can be considered the meta-development activity, which serves to set the stage and bound the problem. SAD can be leveraged to set the correct balance among competing high-level requirements in the functional and non-functional analysis domains. System analysis and design interacts strongly with distributed enterprise architecture, enterprise I.T. Architecture, and business architecture, and relies heavily on concepts such as partitioning, interfaces, personae and roles, and deployment/operational modeling to arrive at a high-level system description. This high level description is then further broken down into the components and modules which can be analyzed, designed, and constructed separately and integrated to accomplish the business goal. SDLC and SAD are cornerstones of full life cycle product and system planning.

## Object-oriented analysis

Object-oriented analysis (OOA) is the process of analyzing a task (also known as a problem domain), to develop a conceptual model that can then be used to complete the task. A typical OOA model would describe computer software that could be used to satisfy a set of customer-defined requirements. During the analysis phase of problem-solving, a programmer might consider a written requirements statement, a formal vision document, or interviews with stakeholders or other interested parties. The task to be addressed might be divided into several subtasks (or domains), each representing a different business, technological, or other areas of interest. Each subtask would be analyzed separately. Implementation constraints, (e.g., concurrency, distribution, persistence, or how the system is to be built) are not considered during the analysis phase; rather, they are addressed during object-oriented design (OOD).

The conceptual model that results from OOA will typically consist of a set of use cases, one or more UML class diagrams, and a number of interaction diagrams. It may also include some kind of user interface mock-up.

The input for object-oriented design is provided by the output of object-oriented analysis. Realize that an output artifact does not need to be completely developed to serve as input of object-oriented design; analysis and design may occur in parallel, and in practice the results of one activity can feed the other in a short feedback cycle through an iterative process. Both analysis and design can be performed incrementally, and the artifacts can be continuously grown instead of completely developed in one shot.

Some typical input artifacts for object-oriented :

- Conceptual model: Conceptual model is the result of object-oriented analysis, it captures concepts in the problem domain. The conceptual model is explicitly chosen to be independent of implementation details, such as concurrency or data storage.

- Use case: Use case is a description of sequences of events that, taken together, lead to a system doing something useful. Each use case provides one or more scenarios that convey how the system should interact with the users called actors to achieve a specific business goal or function. Use case actors may be end users or other systems. In many circumstances use cases are further elaborated into use case diagrams. Use case diagrams are used to identify the actor (users or other systems) and the processes they perform.

- System Sequence Diagram: System Sequence diagram (SSD) is a picture that shows, for a particular scenario of a use case, the events that external actors generate, their order, and possible inter-system events.

- User interface documentations (if applicable): Document that shows and describes the look and feel of the end product's user interface. It is not mandatory to have this, but it helps to visualize the end-product and therefore helps the designer.

- Relational data model (if applicable): A data model is an abstract model that describes how data is represented and used. If an object database is not used, the relational data model should usually be created before the design, since the strategy chosen for object-relational mapping is an output of the OO design process. However, it is possible to develop the relational data model and the object-oriented design artifacts in parallel, and the growth of an artifact can stimulate the refinement of other artifacts.

## Life cycle

## Management and control

SPIU phases related to management controls. The SDLC phases serve as a programmatic guide to project activity and provide a flexible but consistent way to conduct projects to a depth matching the scope of the project. Each of the SDLC phase objectives are described in this section with key deliverables, a description of recommended tasks, and a summary of related control objectives for effective management. It is critical for the project manager to establish and monitor control objectives during each SDLC phase while executing projects. Control objectives help to provide a clear statement of the desired result or purpose and should be used throughout the entire SDLC process. Control objectives can be grouped into major categories (domains), and relate to the SDLC phases as shown in the figure.[12]

To manage and control any SDLC initiative, each project will be required to establish some degree of a work breakdown structure (WBS) to capture and schedule the work necessary to complete the project. The WBS and all programmatic material should be kept in the "project description" section of the project notebook. The WBS format is mostly left to the project manager to establish in a way that best describes the project work.

There are some key areas that must be defined in the WBS as part of the SDLC policy. The following diagram describes three key areas that will be addressed in the WBS in a manner established by the project manager.

**Work breakdown structured organization**

Work breakdown structure.

The upper section of the work breakdown structure (WBS) should identify the major phases and milestones of the project in a summary fashion. In addition, the upper section should provide an overview of the full scope and timeline of the project and will be part of the initial project description effort leading to project approval. The middle section of the WBS is based on the seven systems development life cycle phases as a guide for WBS task development. The WBS elements should consist of milestones and "tasks" as opposed to "activities" and have a definitive period (usually two weeks or more). Each task must have a measurable output (e.x. document, decision, or analysis). A WBS task may rely on one or more activities (e.g. software engineering, systems engineering) and may require close coordination with other tasks, either internal or external to the project. Any part of the project needing support from contractors should have a statement of work (SOW) written to include the appropriate tasks from the SDLC phases. The development of a SOW does not occur during a specific phase of SDLC but is developed to include the work from the SDLC process that may be conducted by external resources such as contractors.

**Baselines**

Baselines are an important part of the systems development life cycle. These baselines are established after four of the five phases of the SDLC and are critical to the iterative nature of the model. Each baseline is considered as a milestone in the SDLC.

- functional baseline: established after the conceptual design phase.
- allocated baseline: established after the preliminary design phase.

- product baseline: established after the detail design and development phase.
- updated product baseline: established after the production construction phase.

## Complementary methodologies

Complementary software development methods to systems development life cycle are:

- Software prototyping
- Joint applications development (JAD)
- Rapid application development (RAD)
- Extreme programming (XP); extension of earlier work in Prototyping and RAD.
- Open-source development
- End-user development
- Object-oriented programming

## Strengths and weaknesses

Few people in the modern computing world would use a strict waterfall model for their SDLC as many modern methodologies have superseded this thinking. Some will argue that the SDLC no longer applies to models like Agile computing, but it is still a term widely in use in technology circles. The SDLC practice has advantages in traditional models of systems development that lends itself more to a structured environment. The disadvantages to using the SDLC methodology is when there is need for iterative development or (i.e. web development or e-commerce) where stakeholders need to review on a regular basis the software being designed. Instead of viewing SDLC from a strength or weakness perspective, it is far more important to take the best practices from the SDLC model and apply it to whatever may be most appropriate for the software being designed.