Software Development Life Cycle

Model of the Systems Development Life Cycle with the Maintenance bubble highlighted.

Software Development Life Cycle (SDLC), in systems engineering and software engineering refers to the process of creating or altering systems, and the models and methodologies that people use to develop these systems. The concept generally refers to computer or information systems.

In software engineering the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system the software development process.

Contents

Overview

Software Development Life Cycle (SDLC) is any logical process used by a systems analyst to develop an information system, including requirements, validation, training, and user ownership. An SDLC should result in a high quality system that meets or exceeds customer expectations, reaches completion within time and cost estimates, works effectively and efficiently in the current and planned Information Technology infrastructure, and is inexpensive to maintain and cost-effective to enhance.

Computer systems have become more complex and often (especially with the advent of Service-Oriented Architecture) link multiple traditional systems potentially supplied by different software vendors. To manage this level of complexity, a number of system development life cycle (SDLC) models have been created: "waterfall," "fountain," "spiral," "build and fix," "rapid prototyping," "incremental," and "synchronize and stabilize." Although the term SDLC can refer to various models, it typically denotes a waterfall methodology.

In project management a project has both a life cycle and a "systems development life cycle," during which a number of typical activities occur. The project life cycle (PLC) encompasses all the activities of the project, while the systems development life cycle focuses on realizing the product requirements.

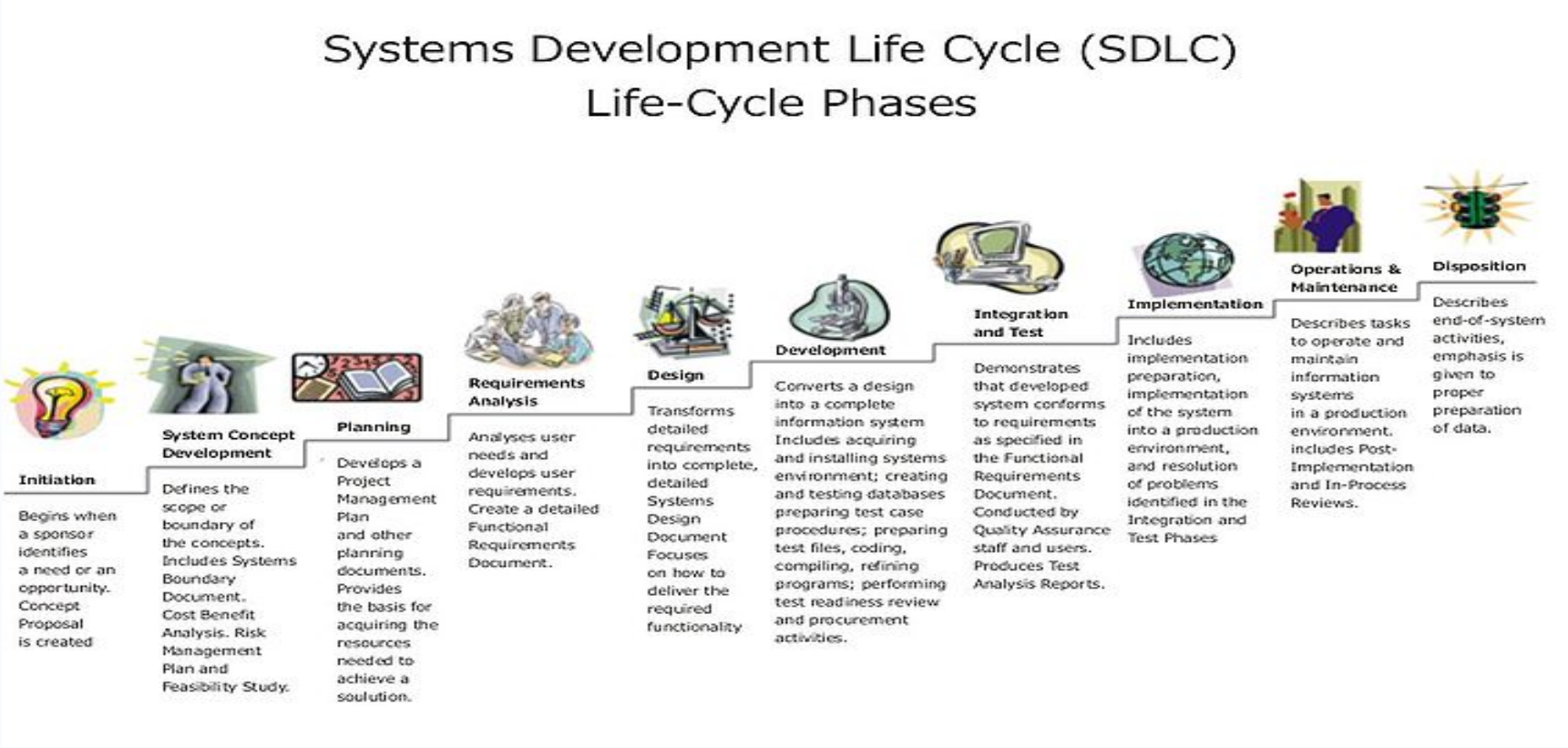
History

Software development life cycle is the oldest formalized methodology for building information systems, intended to develop information systems in a very deliberate, structured and methodical way, reiterating each stage of the life cycle. The traditional systems development life cycle originated in the 1960s to develop large scale functional business systems in an age of large scale business conglomerates. Information systems activities resolved around heavy data processing and number crunching routines

In the 1980s the Structured Systems Analysis and Design Method (SSADM) was based in SDLC. SSADM is a systems approach to the analysis and design of information systems, produced for the Office of Government Commerce, a UK government office concerned with the use of technology in government. Since the 1980s the traditional life cycle approaches to systems development has been increasingly replaced with alternative approaches and frameworks, which attempted to overcome some of the inherent deficiencies of the traditional SDLC.

Software Development Phases

Software Development Life Cycle (SDLC) adheres to important phases that are essential for developers, such as planning, analysis, design, and implementation, and are explained in the section below. There are several Software Development Life Cycle Models in existence. The oldest model, that was originally regarded as "the Systems Development Life Cycle" is the waterfall model: a sequence of stages in which the output of each stage becomes the input for the next. These stages generally follow the same basic steps but many different waterfall methodologies give the steps different names and the number of steps seems to vary between 4 and 7. There is no definitively correct Systems Development Life Cycle model, but the steps can be characterized and divided in several steps.



The SDLC can be divided into ten phases during which defined IT work products are created or modified. 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.

Initiation/Planning

To generate a high-level view of the intended project and determine the goals of the project. The feasibility study is sometimes used to present the project to upper management in an attempt to gain funding. Projects are typically evaluated in three areas of feasibility: economical, operational, and technical. Furthermore, it is also used as a reference to keep the project on track and to evaluate the progress of the MIS team. The MIS is also a complement of those phases. This phase is also called the analysis phase.

Requirements Gatherings and Analysis

The goal of systems analysis is to determine where the problem is in attempt to fix the system. This step involves breaking down the system in different pieces and drawing diagrams to analyze the situation. Analysts project goals, breaking down functions that need to be created, and attempt to engage users so that definite requirements can be defined.

Design

In systems 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.

Build or Coding

Modular and subsystem programming code will be accomplished during this stage. Unit testing and module testing are done in this stage by the developers. This stage is intermingled with the next in that individual modules will need testing before integration to the main project. Planning in software life cycle involves setting goals, defining targets, establishing schedules, and estimating budgets for an entire software project

Testing

The code is tested at various levels in software testing. Unit, system and user acceptance testing 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 occurs at this stage.

Types of testing:

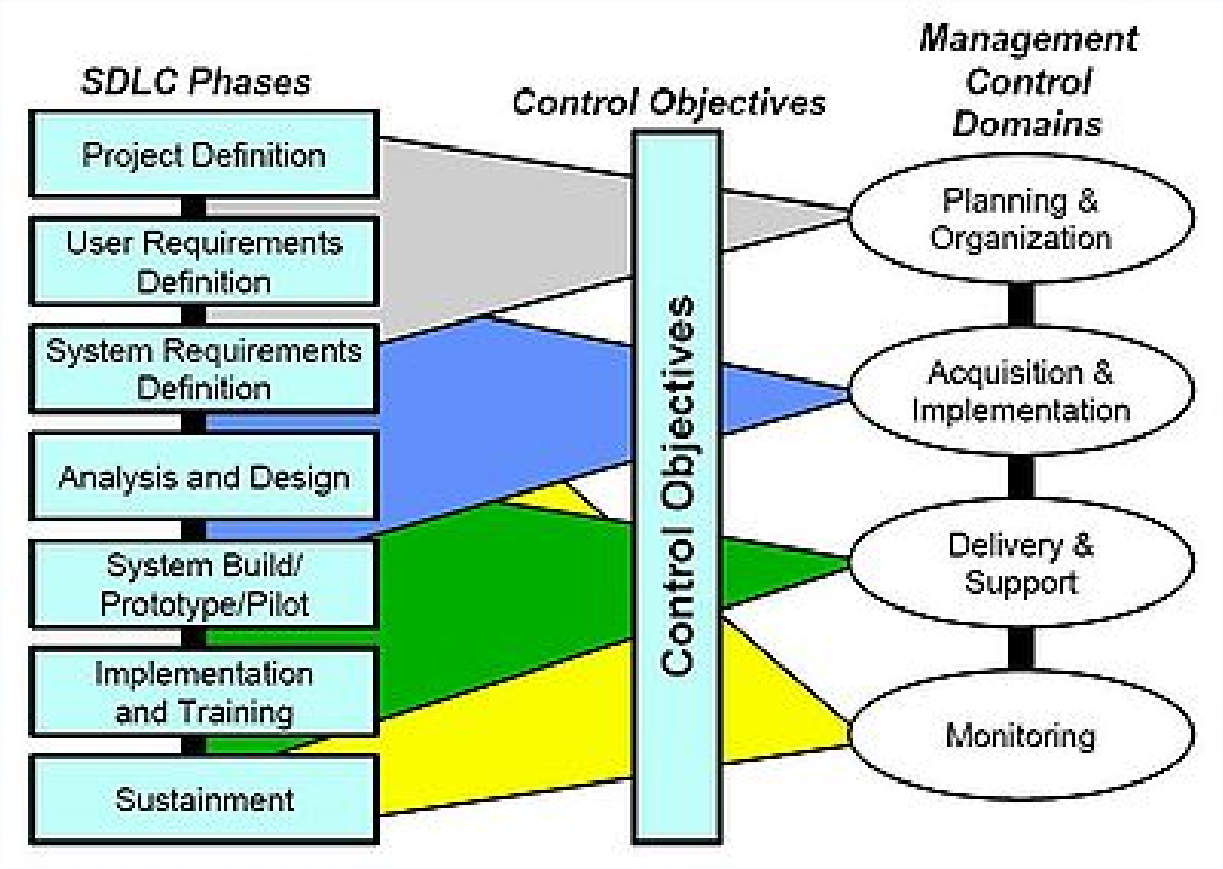
- Data set testing
- Unit testing
- System testing
- Integration testing
- Black box testing
- White box testing
- Module testing
- Regression testing
- Automation testing
- User acceptance testing

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, which will require system updates.

Systems Development Life Cycle topics

Management and control

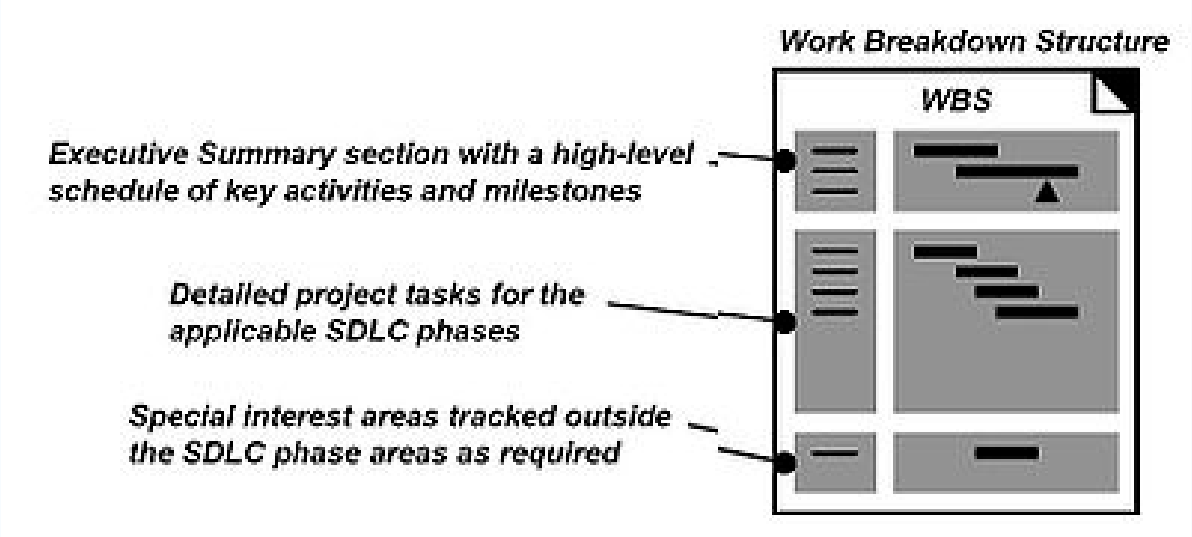


SDLC Phases Related to Management Controls.

The Systems Development Life Cycle (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 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.

To manage and control an 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 Structure 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 (SDLC) 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.g. 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 in the SDLC

Baselines are an important part of the Systems Development Life Cycle (SDLC). 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 to SDLC

Complementary Software development methods to Systems Development Life Cycle (SDLC) are:

- Software Prototyping
- Joint Applications Design (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

Comparison of Methodologies (Post, & Anderson 2006)							
	SDLC	**RAD**	**Open Source**	**Objects**	**JAD**	**Prototyping**	**End User**
Control	Formal	MIS	Weak	Standards	Joint	User	User
Time Frame	Long	Short	Medium	Any	Medium	Short	Short
Users	Many	Few	Few	Varies	Few	One or Two	One
MIS staff	Many	Few	Hundreds	Split	Few	One or Two	None
Transaction/DSS	Transaction	Both	Both	Both	DSS	DSS	DSS
Interface	Minimal	Minimal	Weak	Windows	Crucial	Crucial	Crucial
Documentation & Training	Vital	Limited	Internal	In Objects	Limited	Weak	None
Integrity & Security	Vital	Vital	Unknown	In Objects	Limited	Weak	Weak
Reusability	Limited	Some	Maybe	Vital	Limited	Weak	None

Strength and Weaknesses

Few people in the modern computing world would use a strict waterfall model for their Systems Development Life Cycle (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. A comparison of the strength and weaknesses of SDLC:

Strength and Weaknesses of SDLC	
Strengths	**Weaknesses**
Control.	Increased development time.
Monitor Large projects.	Increased development cost.
Detailed steps.	Systems must be defined up front.
Evaluate costs and completion targets.	Rigidity.
Documentation.	Hard to estimate costs, project overruns.
Well defined user input.	User input is sometimes limited.
Ease of maintenance.	
Development and design standards.	
Tolerates changes in MIS staffing.	

An alternative to the SDLC is Rapid Application Development, which combines prototyping, Joint Application Development and implementation of CASE tools. The advantages of RAD are speed, reduced development cost, and active user involvement in the development process.

It should not be assumed that just because the waterfall model is the oldest original SDLC model that it is the most efficient system. At one time the model was beneficial mostly to the world of automating activities that were assigned to clerks and accountants. However, the world of technological evolution is demanding that systems have a greater functionality that would assist help desk technicians/administrators or information technology specialists/analysts.