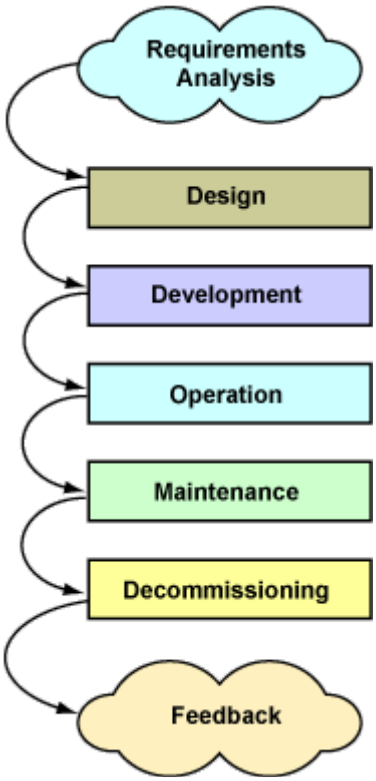
UNIVERSITY OF MARYLAND
University College

Learning Resource

The System Development Life Cycle (SDLC)

What Is the SDLC?

The system development life cycle (SDLC) is a structured methodology and process that guides the development of an information system. SDLC is based on a series of related activities that are combined into phases, sometimes called **life-cycle phases**. The phases represent a state or stage in the life of an information system. Generally speaking, an information system life cycle proceeds from requirements gathering to design and development to operations and maintenance to decommissioning. Each successive phase leverages the documentation and knowledge gained from the previous phases. The figure below shows the general flow of a basic SDLC.



Basic System Development Life Cycle

The main purpose of using SDLC is to promote quality during the design, development, and implementation effort. When SDLC is used properly, an information system is more reliable and cost-effective because project activities are planned, documented, tracked, and

controlled. To ensure that the information system will meet the stated requirements, SDLC also includes predefined reviews, inspections, and audits for the life-cycle processes and deliverables to identify variances and recommend changes.

Using the SDLC Acronym

As with most acronyms, there can be some confusion associated with using SDLC. Within the information technology industry, SDLC may also be used for:

- **Synchronous Data Link Control**—A communications protocol that divides network functions into clearly defined layers.
- **Software development life cycle**—Also known as **software development process (SDP)**, this is the set of life-cycle phases associated with software programs.

For the purposes of this module, SDLC will be used as defined in the first section of this module.

Why Is SDLC Important in the Development of an Information System?

An information system does not consist solely of the software and hardware an organization uses. Effective use of technology is also dependent on having a solid set of processes and procedures for meeting business objectives, delivering products and services, and enabling continuous process improvement. Another important component of an information system is the trained, skilled people who use the technology, processes, and procedures to operate in and manage the organization.

The relationship between the technology, processes, procedures, and people is symbiotic: any change to one component will have some effect on the others. For example, introducing a new human resources information system into an organization without considering how it might affect the organization's processes and procedures could doom the system to failure before it is fully deployed. A key aspect of using SDLC is considering all components of an information system throughout the entire project. This holistic approach is one of the main reasons why using SDLC is increasingly becoming a critical success factor for implementing today's complex, high-stakes information systems.

Because implementing these systems is an expensive, multiyear effort, SDLC is also an important organizational tool to ensure that information system resources are implemented in a fiscally responsible and efficient manner. A life-cycle approach ensures that there is a clear plan and process for:

- identifying and validating organizational requirements early in the project
- designing and developing the system based on the approved requirements

- deploying and transitioning the completed system to the user community
- operating, maintaining, and updating the system once it is deployed
- decommissioning the system when it is no longer required or when it is replaced

The SDLC Phases

The SDLC phases are the sequence of activities associated with the life cycle of an information system. Although the number of SDLC activities can vary depending on the type and complexity of the information-system project or the SDLC model used, there are some common guidelines that allow the activities to be grouped into clearly defined phases. These recommended guidelines are outlined below.

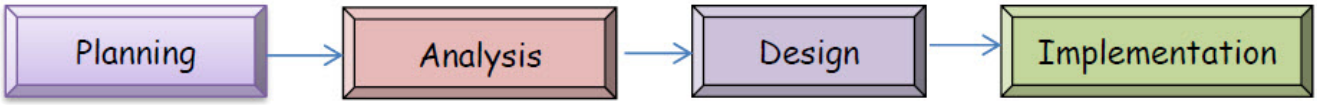
- Complete a preliminary investigation, requirements analysis, and system recommendation.
- Specify a detailed design based on an approved set of requirements.
- Develop the system according to the approved design specification.
- Test the system and gain user acceptance.
- Install, operate, and maintain the accepted system.
- Update or replace the system as organizational goals and requirements change.
- Decommission the system when it is no longer needed.
- Document, report on, and approve each phase of the SDLC before beginning the next phase.

Following these common guidelines helps mitigate the risk that the design and development effort will get out of control either through missed requirements, schedule delays, or cost overruns. Because the guidelines require interaction with stakeholders throughout the project, they also prevent surprises when the system is rolled out to the user community. As you read through the approaches in the following sections, see if you can identify these common steps.

Four-Phase Approach

This approach divides the life cycle into four major phases. It may be used when an organization has a good understanding of its requirements or the type of information system being implemented.

The figure below shows the four phases and some of the key activities associated with each phase.



Four-Phase Approach

Planning	Analysis	Design	Implementation
Those responsible for creating the system must first determine whether the system is actually needed, and what is will do for the organization if it is created. A basic idea of how to build the system is explored. A project plan is created and a project manager assigned.	Again, the responsible group must first decide whether to go ahead with the project and whether the resources needed to complete the work are even available. A feasibility study needs to be completed which justifies the need to replace an existing system, identifies the improvements that will be made if the new system is created, and reviews whether there is budget and resources sufficient to create the system.	In the design phase, a detailed plan for creation of the system is developed. This plan is then implemented and hopefully results in creation of all the components needed to complete the system.	Implementation consists of both the actual product development and installation of the product for the users. This phase should also include documentation, user manuals, training, and actual maintenance of the system, and any future updates or expansion of the system.

Source: Four Phases of SDLC? (n.d.).

Nine-Phase Approach

This approach divides the life cycle into nine phases. The following table shows the phases and some of the key activities associated with each phase. As you read through the table, compare it with the four-phase approach. Note that a more granular approach is taken for

the preliminary investigation, requirements analysis, and system recommendation portions of the project.

Organizations may use this approach when implementing an unfamiliar type of information system. The nine-phase approach is also more appropriate for implementing an information system that will be used across all business units within an organization.

Nine-Phase Approach

SDLC Phases	Key Activities
Initiation phase	• Develop business case • Identify project sponsor • Appoint project manager • Develop concept proposal • Review and approve concept proposal
System concept development phase	• Analyze business need • Form project team • Plan project • Develop project-acquisition strategy • Identify and analyze risks • Obtain funding and resources • Document phase efforts • Review and approve phase documents

SDLC Phases	Key Activities
Planning phase	• Refine acquisition strategy • Analyze project schedule • Document internal processes • Establish agreements with stakeholders • Develop project-management plan • Review and approve project-management plan
Requirements analysis phase	• Define functional requirements • Define technical requirements • Conduct reviews and approve requirements
Design phase	• Design system • Design business processes • Outline operations and maintenance manuals • Outline deployment plan • Conduct design reviews • Approve system design

SDLC Phases	Key Activities
Development phase	• Refine and complete software requirements • Refine and complete software design • Acquire and install hardware • Code and test software • Conduct hardware- and software-qualification testing • Install software • Test system qualification • Complete plans and support documentation • Test and review documentation • Develop deployment plan • Obtain approval and acceptance of all development documentation
Integration and test phase	• Conduct subsystem/system testing • Conduct security testing • Conduct user-acceptance testing • Review and finalize development-phase documentation • Obtain user acceptance
Implementation phase	• Communicate deployment plan • Execute training plan • Perform data entry, migration, and conversion • Install new system • Perform post implementation evaluation • Obtain approval to operate the system

SDLC Phases	Key Activities
Operations and maintenance phase	• Transition project to operations • Operate system • Perform data and software administration • Perform system and software maintenance • Identify problems, recommend modifications, and update the system • Monitor organizational changes, recommend modifications, and update the system

Ten-Phase Approach

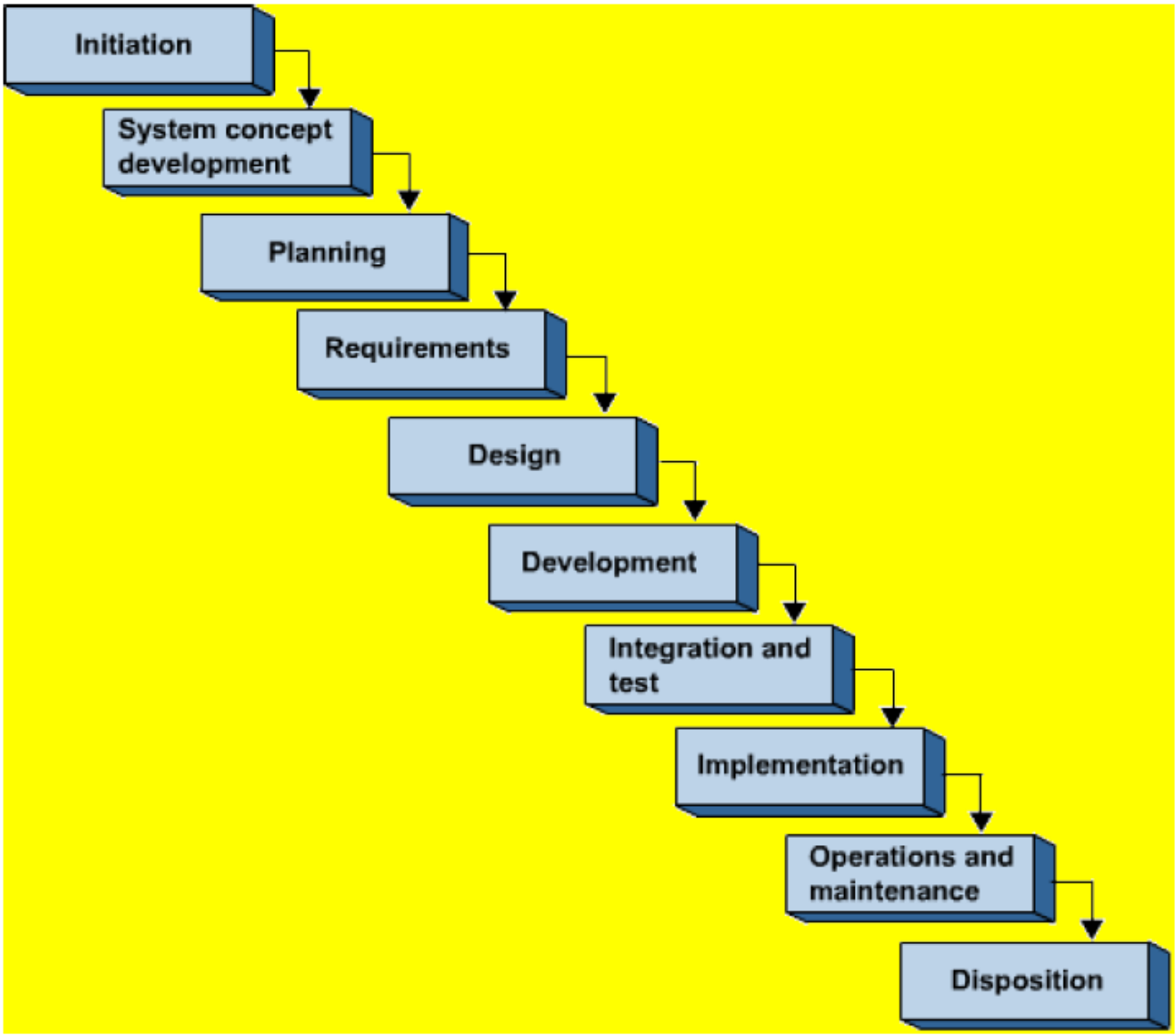
The U.S. Department of Justice (DOJ) uses a 10-phase SDLC approach on its information system implementation projects. Like the nine-phase approach, this approach emphasizes the preliminary investigation, requirements analysis, and system recommendation project activities. The main difference between the DOJ approach and the nine-phase approach is that the DOJ approach also includes a phase to dispose of the information system when it is no longer needed.

SDLC Models

Waterfall Model

The waterfall model is often used to represent the SDLC process. This linear, sequential model is often considered to be the foundation and origin of today's SDLC methodology. Although there is disagreement as to when the model was first introduced, the general consensus is that it has been in existence in one form or another since the 1960s.

Waterfall development is still widely used for software engineering projects because it has distinct goals for each phase of the development and requires each phase to be fully completed before the next phase can begin. Once the decision is made to go to the next phase, there is no turning back. Like a waterfall, once the water goes over the cliff (phase), it cannot flow back. The figure below graphically shows how the waterfall model works using the DOJ's ten-phase approach.



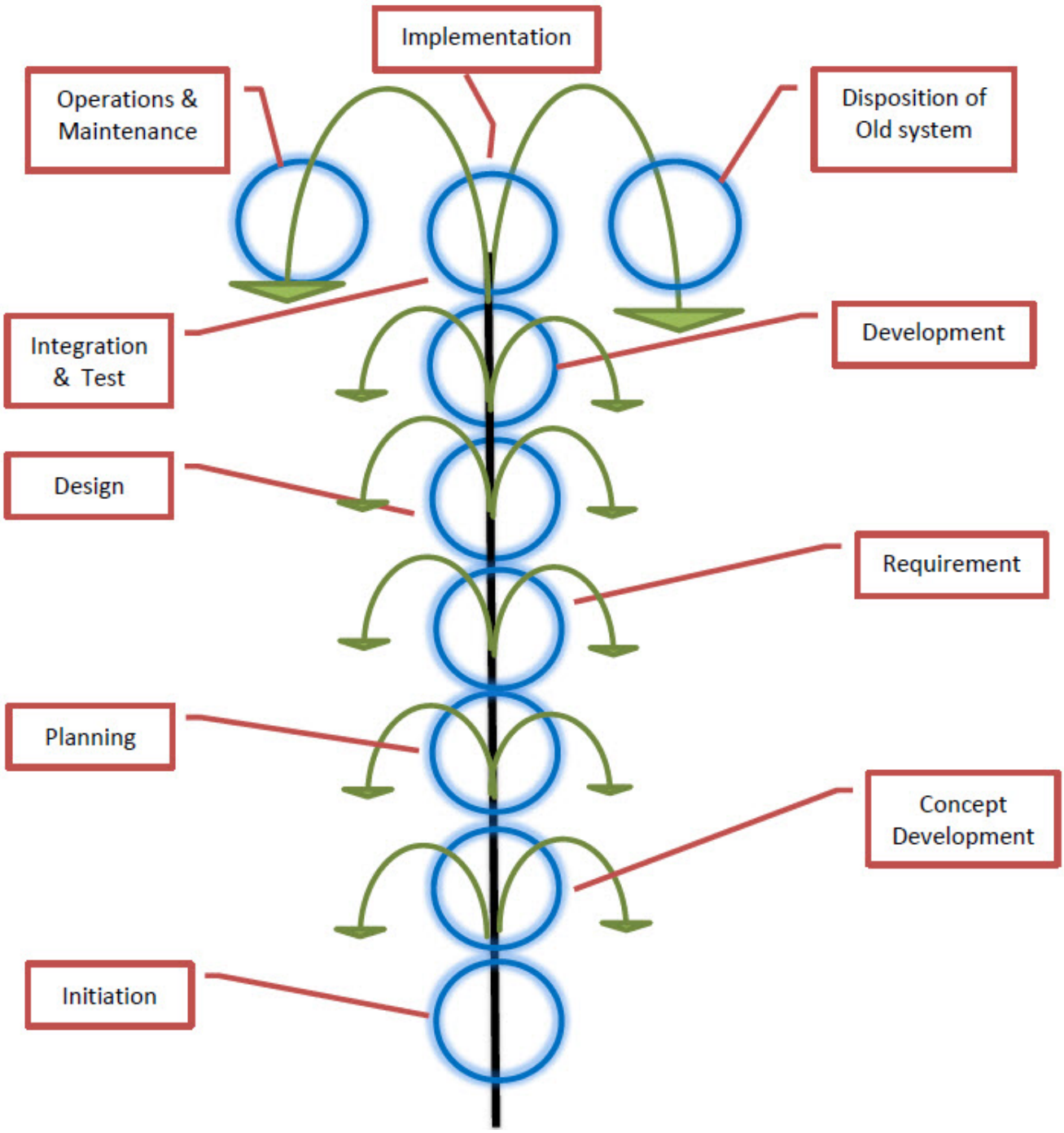
Waterfall SDLC Model

The advantage of waterfall development is that it allows for direct project manager and management control. A timeline can be established with specific deadlines for each phase, and a software solution can proceed through the development process like a product through an assembly line, and if properly managed, be delivered on time. Each phase of development proceeds in a predefined order, without any overlapping steps or turning back.

The disadvantage of waterfall development is that there is no returning to a previous phase. Once the software solution is in the design phase, it is difficult to go back and modify a feature or function that was not well thought out in the requirements phase. Today's complex, cross-functional information systems require a more iterative approach and development effort.

Fountain Model

The fountain model recognizes that overlap may be needed between some development phases, and previous phases may have to be revisited throughout the development cycle. For example, planning may need to be fully completed prior to beginning requirements analysis. Once planning is completed, the requirements analysis, design, and development phases may have activities that must overlap to ensure the system is properly built. Like water in a fountain, details about the information system are pushed up through the phases, but at any time the details may flow back through the previous phases to be refreshed and refined as more is learned about the system. The figure below graphically shows the way the fountain model works, using the DOJ's 10-phase approach.



Fountain SDLC Model Based on DOJ 10-Phase Approach

Source: Janet Zimmer

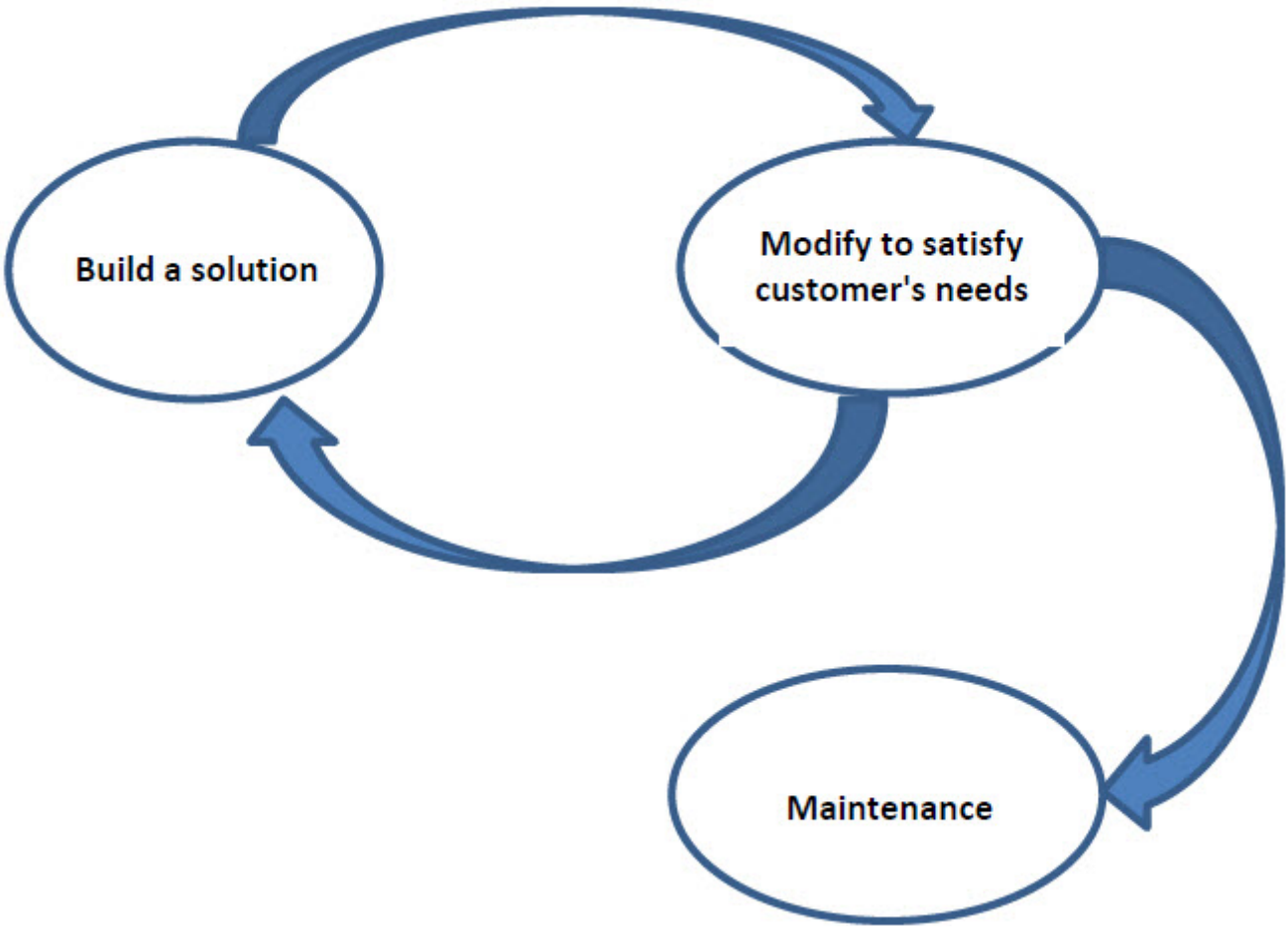
At first, the fountain model can be confusing. If you have never used the model on a project, you may not understand that the overlapping phases and curved arrows demonstrate the highly iterative nature of this life-cycle approach.

The advantage of fountain development is that changes can be made to the components of the information system as the project team learns more about what is actually needed or uncovers gaps in the concept, requirements, or design.

The disadvantage of this model is that it may take more time and cost more to complete the information system. Without strong project management, the information system theoretically may never be completed if the project team gets caught in a loop of ever-increasing scope and continuously changing requirements.

Build-and-Fix Model

Build and fix is recognized as the crudest, least structured model in the SDLC family. In this model, the solution is developed without any proper preliminary investigation, requirements analysis, or design. In essence, the solution is built (think of this as a working model or prototype) and modified as often as necessary until it satisfies the customer's needs. The figure below graphically shows the way this model, which uses only the development and operational phases of the four-phase approach, works. Some of the study and design phase activities may be completed during a highly iterative modify phase.



Build-and-Fix SDLC Model

Source: Janet Zimmer

The advantage of the build-and-fix model is that it provides an efficient framework for extremely small, low-priority development efforts that involve a single customer. In some cases, it may be necessary to use the build-and-fix model when there is not enough time for a more rigorous approach. The highly iterative nature of this model ensures intense and frequent customer involvement in the development of the information system.

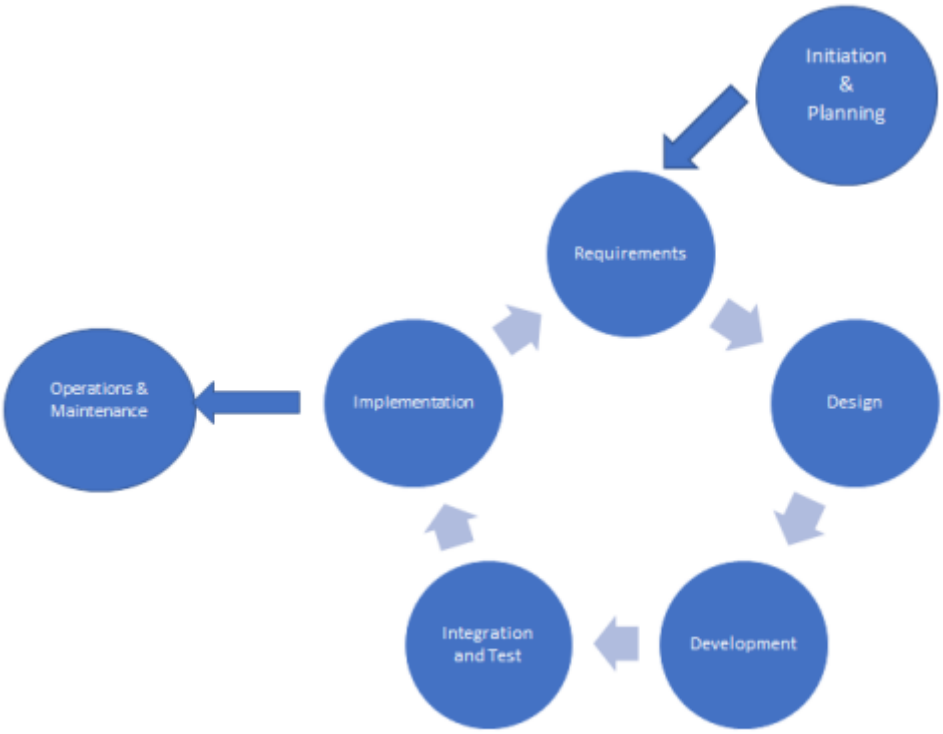
The disadvantage of this approach is that the cost is usually greater than if a preliminary investigation, requirements analysis, and detailed design had been completed. This is an extremely open-ended, risky approach that requires careful management and control. Organizations are strongly discouraged from using this SDLC model except for small, low-priority projects.

Rapid Application Development (RAD) or Rapid Prototyping Model

In most cases, rapid application development (RAD) is used when developing a software solution that is heavily dependent on the organization's business processes and the end users' knowledge and understanding of those processes. In essence, the end users can provide better feedback about the system requirements by examining a live system rather than commenting on the associated documentation.

In a sense, the RAD is also a type of working model or prototype of the solution that allows the end user to see how various requirements have been implemented as the product is developed. It is analogous to working with a tailor who is making a custom suit for you. At various stages of the suit's construction, you may have to return to the tailor's shop, try on the unfinished pieces, and provide feedback that is used to update the measurements and complete the suit. The type of suit and the fit you expect may dictate how many iterations are needed before the tailor's work is done.

RAD is made possible by the significant advances in the software development environment that allow for more rapid code generation and faster modifications to application screens and other user interfaces. The figure below graphically shows how the RAD model works using a modified version of the DOJ's ten-phase approach.



RAD/Rapid Prototyping Model

Source: Janet Zimmer

The advantage of the RAD model is that it can result in a lower level of rejection when the information system is placed into production. End users are given the opportunity to work with the screens online in a production-like environment, which means a significant number of design and development errors can be caught earlier in the process. The model also allows end users to be heavily involved in the software development effort and take ownership of the finished product.

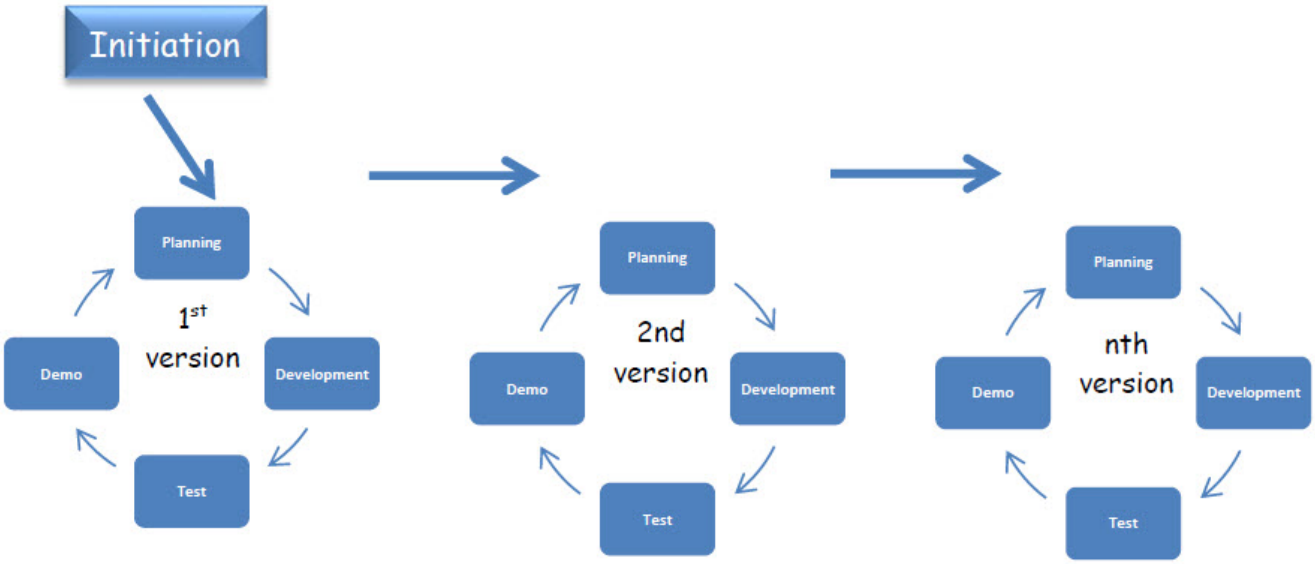
The disadvantage of this model is that RAD could lead to cost and schedule overruns. Another downside is the propensity of the end user to increase the scope and add new requirements during the development effort. Some end users may think that because it is easy for the developer to produce the basic screens that it is just as easy to add extra enhancements. Without strong project leadership, participants can lose sight of the goal of producing an optimal, useful system and instead attempt to develop a gold-plated application that goes beyond the organization's requirements. For this reason, the project team may use a blend of RAD prototyping and the traditional waterfall approach.

Agile Model

The agile model is, in some ways, similar to the build-and-fix and RAD models in that multiple releases of the product are made, each with small incremental additions that lead up to a final product. Each release is tested by the customers and requires a close working relationship between customers, developers, and testers. That interaction with the

customer can also be its downfall if the customer is not sure of or clear about the direction in which the project is heading, potentially resulting in false starts or dramatic changes in requirements as the project progresses.

The figure below graphically shows how the agile model might work.



Agile Model

Source: Janet Zimmer

The SDLC and Testing

Regardless of the model used in the system development, testing should be incorporated into every phase of the life cycle. These are some of the types of testing that should be incorporated into the development cycles:

Unit test: This test focuses on just one of those subsystems to ensure that it operates correctly and produces the results according to specifications for the system.

Integration test: This test uses real data and tests whether each of units continue to work properly with the "live" data used by the various subsystems. This type of test works with multiple units to see that output from one unit is properly applied to another unit and produces expected results.

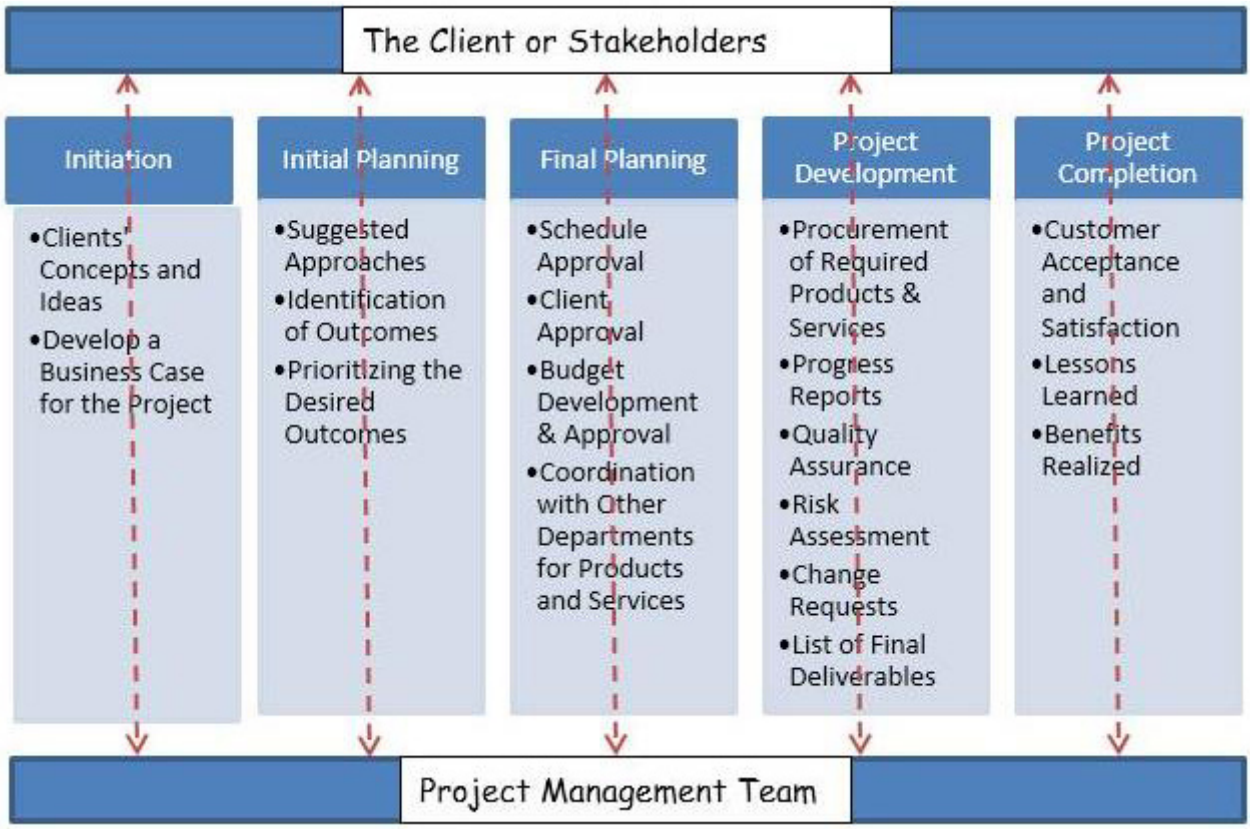
System test: This test determines whether all of the components of the system work together. This is especially important if different work units are creating pieces or subsystems of the project. These subsystems must be able to work seamlessly with all other subsystems to which they are connected. Where the integration test might test closely related units, the system test involves using data across all units to ensure expected results.

Acceptance test: Although this might be used only after the system test is verified, the end users may get involved in smaller versions of unit or integration tests to ensure that the system is working according to the users' specifications. The user may not accept the product if it does not perform according to those specifications.

Relationship Between SDLC and Project Management

Project management is a profession and discipline that uses a systematic process to plan, manage, execute, and control projects. Project managers are found in just about every commercial and noncommercial environment, including construction, education, financial services, government, medicine, manufacturing, nonprofit, technology, and utilities environments.

The project-management process uses the same structure and rigor found in the SDLC phases models. A typical project-management process may include the steps shown in the figure below.



Project Management Process

Source: Janet Zimmer

Although many projects do not require an SDLC approach, most information technology projects do. Specifically, when some form of information system is needed, SDLC is required and project management is needed to plan, schedule, and control the associated

activities. As an information system moves through its life-cycle phases, it may spawn several projects. For example, there may be separate projects to:

- determine the business need,
- find, analyze, evaluate, and select a vendor
- define what needs to be done to update an aging system during the operations and maintenance phase
- dispose of an information system that is no longer required

In most cases, the project ends when the information system moves into the operations and maintenance phase. In all cases, it is project management that brings order and organization to information-system-development efforts.

Summary

SDLC is the progression through a series of stages or states of an information system. It lasts from the conception of the system to its disposition. The number of life-cycle phases can vary from system to system and according to the needs of the organization. SDLC models are tools that allow project and development teams to correctly follow the SDLC stages required to develop the various types of information systems. Project management is used to plan, schedule, and control the SDLC phases associated with a selected model.

Sources

Elliott, R. K. (2006). Sorting out SDLC terminology. Retrieved from http://www.managingsoftwaredevelopment.com/Ed001article2.htm

Four phases of SDLC? (n.d.). Retrieved from http://www.answers.com/Q/Four_phases_of_SDLC

Justice Management Division. (2003, January). *Systems development life cycle guidance document*. Washington, DC: Department of Justice.

Mulcahy, R. (2005). *PMP exam prep* (5th ed.). Lakewood, CO: RMC Publications, Inc.

Office of the Chief Information Officer. (2006, February). *Smithsonian information technology plan FY 2006–FY 2011*. Washington, DC: Smithsonian Institution.

Office of Information Technology. (2006, August). *Systems development life cycle (SDLC)*. Volume 2: SDLC Phases. Annapolis, MD: Maryland Department of Budget & Management.

Project Management Institute. (2004). *A guide to the project management body of knowledge (PMBOK Guide)* (3rd ed.). Newtown Square, PA: Project Management Institute, Inc.

Schwalbe, K. (2005). *Information technology project management* (4th ed.). Boston: Thomson Course Technology.

Shelly, G. B., Cashman, T. J., & Vermaat, M. E. (2007). *Discovering computers 2007: A gateway to information*. Boston: Thomson Course Technology.
