Approaches to System Development

Predictive and Adaptive Approach to SDLC

- **Software Development Life Cycle** is the application of standard business practices to building software applications. It is typically divided into six to eight steps: Planning, Requirements, Design, Build, Document, Test, Deploy, Maintain
- In today's diverse development environment, there are many approaches to developing systems, and they are based on different approaches to the SDLC (see ***Figure 1***).

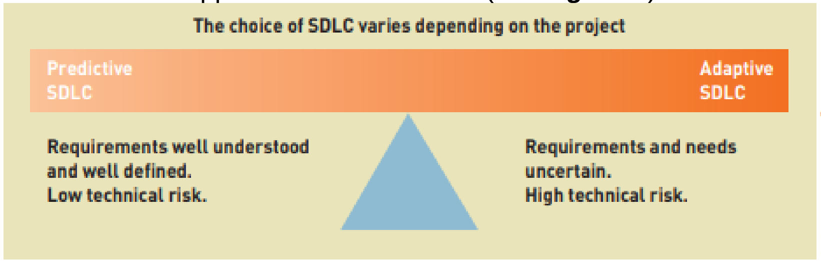

The choice of SDLC varies depending on the project
Predictive SDLC
Adaptive SDLC
Requirements well understood and well defined. Low technical risk.
Requirements and needs uncertain. High technical risk.

Figure 1. Predictive versus adaptive approaches to the SDLC

- A **predictive approach** to the SDLC assumes that the development project can be planned and organized and that the new information system can be developed according to the plan. Predictive SDLCs are useful for building systems that are well understood and defined
- For example, a company may want to convert its old, networked client/server system to a newer Web-based system that includes a smartphone app. In this type of project, the staff already understands the requirements very well, and no new processes need to be added. Thus, the project can be carefully planned, and the system can be built according to the specifications.
- An **adaptive approach** to the SDLC is used when the system's requirements and the users' needs are not well understood. In this situation, the project cannot be planned completely. Some system requirements may need to be determined after preliminary development work. Developers should still be able to build the solution, but they need to be flexible and adapt to the project as it progresses.
- In practice, any project could have, and most do have predictive and adaptive elements. That is why ***Figure 1*** shows them as endpoints along a continuum, not as mutually exclusive categories.
- The predictive approaches are more traditional and were conceived during the 1970s through the 1990s. Many of the newer, adaptive approaches have evolved with object-oriented technology and Web development; they were created during the late 1990s and into the twenty-first century.

Traditional Predictive Approaches to the SDLC

- The development of a new information system requires several different but related sets of activities. In predictive approaches, a group of activities identifies the problem and secures approval to develop a new system; this is called **project initiation.**
- The second group of activities, called **project planning**, involves planning, organizing, and scheduling the project. These activities map out the project's overall structure.
- The third group of activities called **analysis** focuses on discovering and understanding the details of the problem or need. The intent here is to figure out exactly what the system must do to support the business processes.
- The fourth group of activities called **design** focuses on configuring and structuring the new system components. These activities use the requirements that were defined earlier to develop the program structure and the algorithms for the new system.
- The fifth group of activities called **implementation** includes programming and testing the system.
- The sixth group of activities called **deployment** involves installing and putting the system into operation.

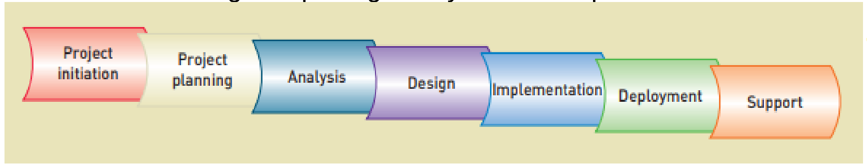


Figure 2. Traditional information system development phases

- These six groups of activities are sometimes referred to as **phases** of the system development project, and they provide the framework for managing the project.
- Another phase, called the **support phase**, includes the activities needed to upgrade and maintain the system after it has been deployed. The support phase is part of the overall SDLC, but it is not normally considered part of the initial development project. ***Figure 2*** illustrates the six phases of a traditional predictive SDLC plus the support phase.

Newer Adaptive Approaches to the SDLC

- In an adaptive approach, project activities, including plans and models, are adjusted as the project progresses. Iterations can be used to create a series of mini-projects that address smaller parts of the application
- Using iterations, the project can adapt to any changes as it proceeds. Also, parts of the system are available early on for user evaluation and feedback, which helps ensure that the application will meet the needs of the users.

Methodologies, Models, Tools, and Techniques

- A **system development methodology** provides guidelines for every facet of the system development life cycle. For example, within a methodology, certain models, such as diagrams, are used to describe and specify the requirements.
- Each project team uses a set of tools, usually computer-based tools, to build models, record information, and write the code. These tools are considered part of the overall methodology used for a given project.

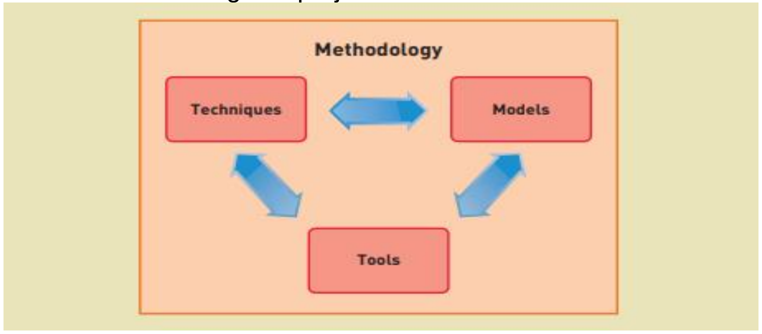
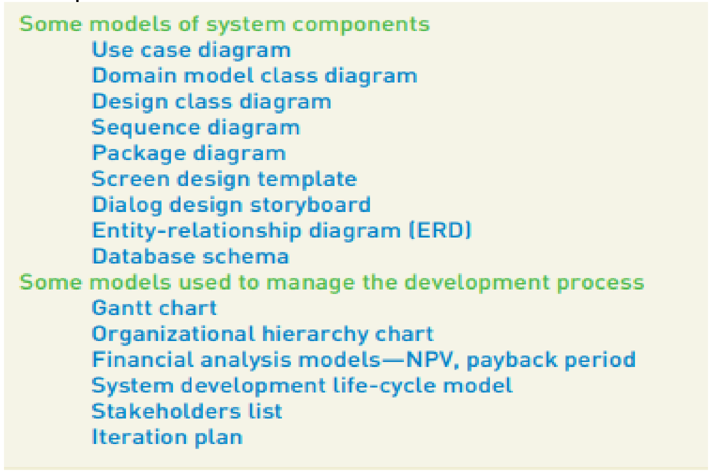


Figure 3. Components of a Methodology

- **Models** used in system development include representations of inputs, outputs, processes, data, objects, object interactions, locations, networks, and devices, among other things
- Most of the models are graphical models, which are drawn representations that employ agreed-upon symbols and conventions. ***Figure 4*** lists some models used in system development.



Some models of system components
- Use case diagram
- Domain model class diagram
- Design class diagram
- Sequence diagram
- Package diagram
- Screen design template
- Dialog design storyboard
- Entity-relationship diagram (ERD)
- Database schema

Some models used to manage the development process
- Gantt chart
- Organizational hierarchy chart
- Financial analysis models—NPV, payback period
- System development life-cycle model
- Stakeholders list
- Iteration plan

Figure 4. Example models used in system development

- In the context of system development, a **tool** is software support that helps create models or other components required in the project. Tools might be simple drawing programs for creating diagrams. They might also include an application that stores information about the project, such as data definitions, use case descriptions, and other artifacts
- Tools have been specifically designed to help system developers. Programmers should be familiar with integrated development environments (IDEs), which include many tools to help with programming tasks. For example, smart editors, context-sensitive help, and debugging tools.
- **Visual modeling tools** are available to systems analysts to help them create and verify important system models. These tools are used to draw such diagrams as class diagrams or

activity diagrams. Other visual modeling tools help design the database or even generate program code. ***Figure 5*** lists the types of tools used in system development.

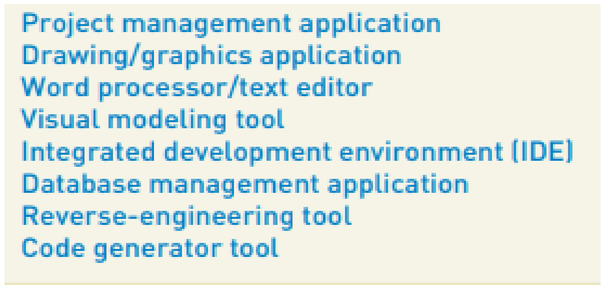

Project management application
Drawing/graphics application
Word processor/text editor
Visual modeling tool
Integrated development environment (IDE)
Database management application
Reverse-engineering tool
Code generator tool

***Figure 5*.** Types of tools used in system development

- In system development, a **technique** is a collection of guidelines that helps an analyst complete an activity or task. It often includes step-by-step instructions for creating a model, or it might include more general advice on collecting information from system users
- Examples include data-modeling techniques, software testing techniques, user-interviewing techniques, and relational database design techniques. ***Figure 6*** lists some techniques commonly used in system development.

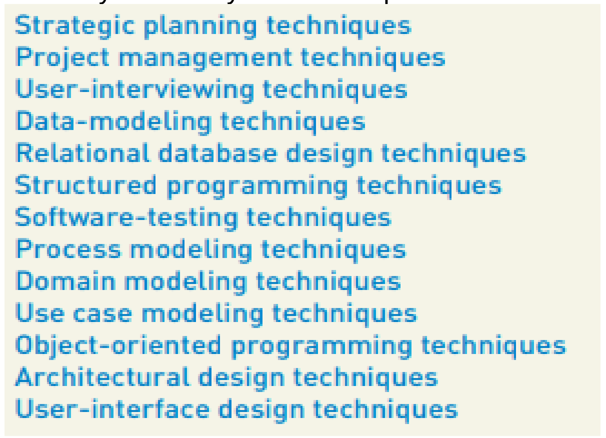

Strategic planning techniques
Project management techniques
User-interviewing techniques
Data-modeling techniques
Relational database design techniques
Structured programming techniques
Software-testing techniques
Process modeling techniques
Domain modeling techniques
Use case modeling techniques
Object-oriented programming techniques
Architectural design techniques
User-interface design techniques

***Figure 6*.** Some techniques used in system development

REFERENCES:

Tilley, S. (2020). *System analysis and design.* USA: Cengage Learning.
Coronel,C. and Morris, S. (2017). *Database Systems: Design, implementation, and management* (12th ed.). USA: Cengage Learning.
Satzinger, J., Jackson, R., and Burd, S. (2016). *Systems analysis and design in a changing world-course technology.* USA.