ICS 3105 OBJECTED ORIENTED SOFTWARE ENGINEERING

Chapter 2: SOFTWARE PROCESS

Chapter Objectives

- By the end of this chapter, the learners should be able to:
  - Discuss software development life cycle models.
  - Choose an appropriate model of analysis and design tasks.

Introduction

- There are alternative methodologies for modeling and designing systems:
  - Structured methodologies and
  - Object-oriented development are the most prominent.

Structured Methodologies

- Structured methodologies have been used to document, analyze, and design information systems since the 1970s.
- Structured refers to the fact that the techniques are step by step, with each step building on the previous one.

Structured Methodologies

- Structured methodologies are top-down, progressing from the highest, most abstract level to the lowest level of detail—from the general to the specific.
- Structured development methods are process-oriented, focusing primarily on modeling the processes, or actions that capture, store, manipulate, and distribute data as the data flow through a system.

Structured Methodologies

- These methods separate data from processes.
- A separate programming procedure must be written every time someone wants to take an action on a particular piece of data.
- The procedures act on data that the program passes to them.

Structured Methodologies

- The primary tool for representing a system’s component processes and the flow of data between them is the data flow diagram (DFD).
- The data flow diagram offers a logical graphic model of information flow, partitioning a system into modules that show manageable levels of detail.

Structured Methodologies

- DFDs rigorously specifies the processes or transformations that occur within each module and the interfaces that exist between them.
- DFDs can be used to depict higher-level processes as well as lower-level details.
- Through leveled data flow diagrams, a complex process can be broken down into successive levels of detail.

Structured Methodologies

- An entire system can be divided into subsystems with a high-level data flow diagram.
- Each subsystem, in turn, can be divided into additional subsystems with second-level data flow diagrams, and the lower-level subsystems can be broken down again until the lowest level of detail has been reached.

Structured Methodologies

- Another tool for structured analysis is a data dictionary, which contains information about individual pieces of data and data groupings within a system.
- The data dictionary defines the contents of data flows and data stores so that systems builders understand exactly what pieces of data they contain.

Structured Methodologies

- Process specifications describe the transformation occurring within the lowest level of the data flow diagrams.
- Process specifications express the logic for each process.

Structured Methodologies

- In structured methodology, software design is modeled using hierarchical structure charts.
- The structure chart is a top-down chart, showing each level of design, its relationship to other levels, and its place in the overall design structure.

Structured Methodologies

- The design first considers the main function of a program or system, then breaks this function into sub-functions, and decomposes each sub-function until the lowest level of detail has been reached.

Structured Methodologies

- If a design has too many levels to fit onto one structure chart, it can be broken down further on more detailed structure charts.
- A structure chart may document one program, one system (a set of programs), or part of one program.

Limitations of Structured Methodologies

- Structured methods are useful for modeling processes, but do not handle the modeling of data well.
- They also treat data and processes as logically separate entities, whereas in the real world such separation seems unnatural.

Limitations of Structured Methodologies

- Different modeling conventions are used for analysis (the data flow diagram) and for design (the structure chart).

Object-Oriented Methodology

- Object-oriented development addresses these limitations of structured methodologies.
- Object-oriented development uses the object as the basic unit of systems analysis and design.

Object-Oriented Methodology

- An object combines data and the specific processes that operate on those data.
- Data encapsulated in an object can be accessed and modified only by the operations, or methods, associated with that object.

Object-Oriented Methodology

- Instead of passing data to procedures, programs send a message for an object to perform an operation that is already embedded in it.
- The system is modeled as a collection of objects and the relationships among them.

Object-Oriented Methodology

- Because processing logic resides within objects rather that in separate software programs, objects must collaborate with each other to make the system work.
- Object-oriented modeling is based on the concepts of class and inheritance.

Object-Oriented Methodology

- Objects belonging to a certain class, or general categories of similar objects, have the features of that class.
- Classes of objects in turn can inherit all the structure and behaviors of a more general class and then add variables and behaviors unique to each object.

Object-Oriented Methodology

- New classes of objects are created by choosing an existing class and specifying how the new class differs from the existing class, instead of starting from scratch each time.

Object-Oriented Methodology

- Object-oriented development is more iterative and incremental than traditional structured development.
- During analysis, systems builders document the functional requirements of the system, specifying its most important properties and what the proposed system must do.

Object-Oriented Methodology

- Interactions between the system and its users are analyzed to identify objects, which include both data and processes.
- The object-oriented design phase describes:
  - How the objects will behave and
  - How they will interact with one other.

Object-Oriented Methodology

- Similar objects are grouped together to form a class, and classes are grouped into hierarchies in which a subclass inherits the attributes and methods from its super class.

Object-Oriented Methodology

- The information system is implemented by translating the design into program code, reusing classes that are already available in a library of reusable software objects and adding new ones created during the object-oriented design phase.

Object-Oriented Methodology

- Implementation may also involve the creation of an object-oriented database.
- The resulting system must be thoroughly tested and evaluated.

Benefits of Object-Oriented Methodology

- Because objects are reusable, object-oriented development could potentially reduce the time and cost of writing software because organizations can reuse software objects that have already been created as building blocks for other applications.

Benefits of Object-Oriented Methodology

- New systems can be created by using some existing objects, changing others, and adding a few new objects.
- Object-oriented frameworks have been developed to provide reusable, semi-complete applications that the organization can further customize into finished applications.

Alternative Systems Building Approaches

- Systems differ in terms of their size and technological complexity and in terms of the organizational problems they are meant to solve.
- A number of systems building approaches have been developed to deal with these differences.

Alternative Systems Building Approaches

- The traditional systems life cycle.
- Prototyping.
- Application software packages.
- End-user development, and
- Outsourcing.

Traditional Systems Life Cycle

- The systems life cycle is the oldest method for building information systems.
- The life cycle methodology is a phased approach to building a system, dividing systems development into formal stages.

Traditional Systems Life Cycle

- Systems development specialists have different opinions on how to partition the systems-building stages, but they roughly correspond to the stages of systems development.

Traditional Systems Life Cycle

- The systems life cycle methodology maintains a very formal division of labor between end users and information systems specialists.

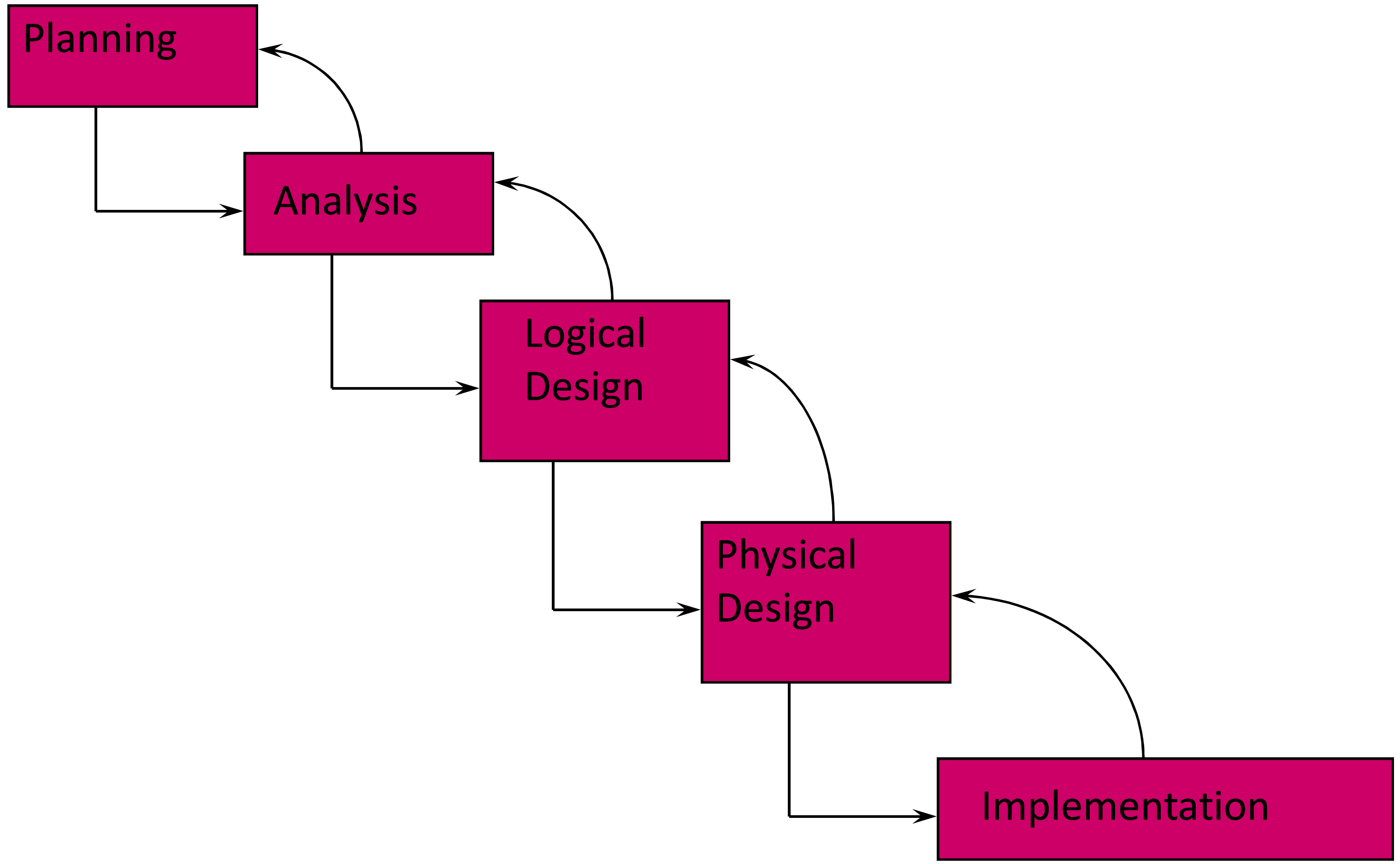
Traditional Systems Life Cycle

- Technical specialists, such as system analysts and programmers, are responsible for much of the systems analysis, design, and implementation work; end users are limited to providing information requirements and reviewing the technical staff's work.

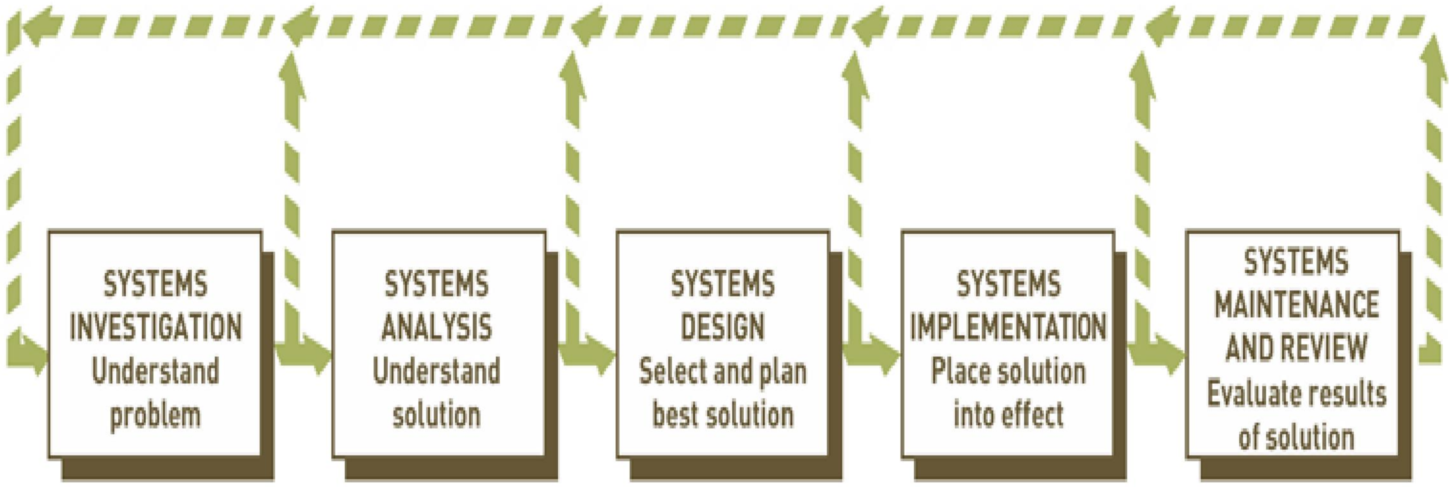
Traditional Systems Life Cycle

- The life cycle also emphasizes formal specifications and paperwork, so many documents are generated during the course of a systems project.

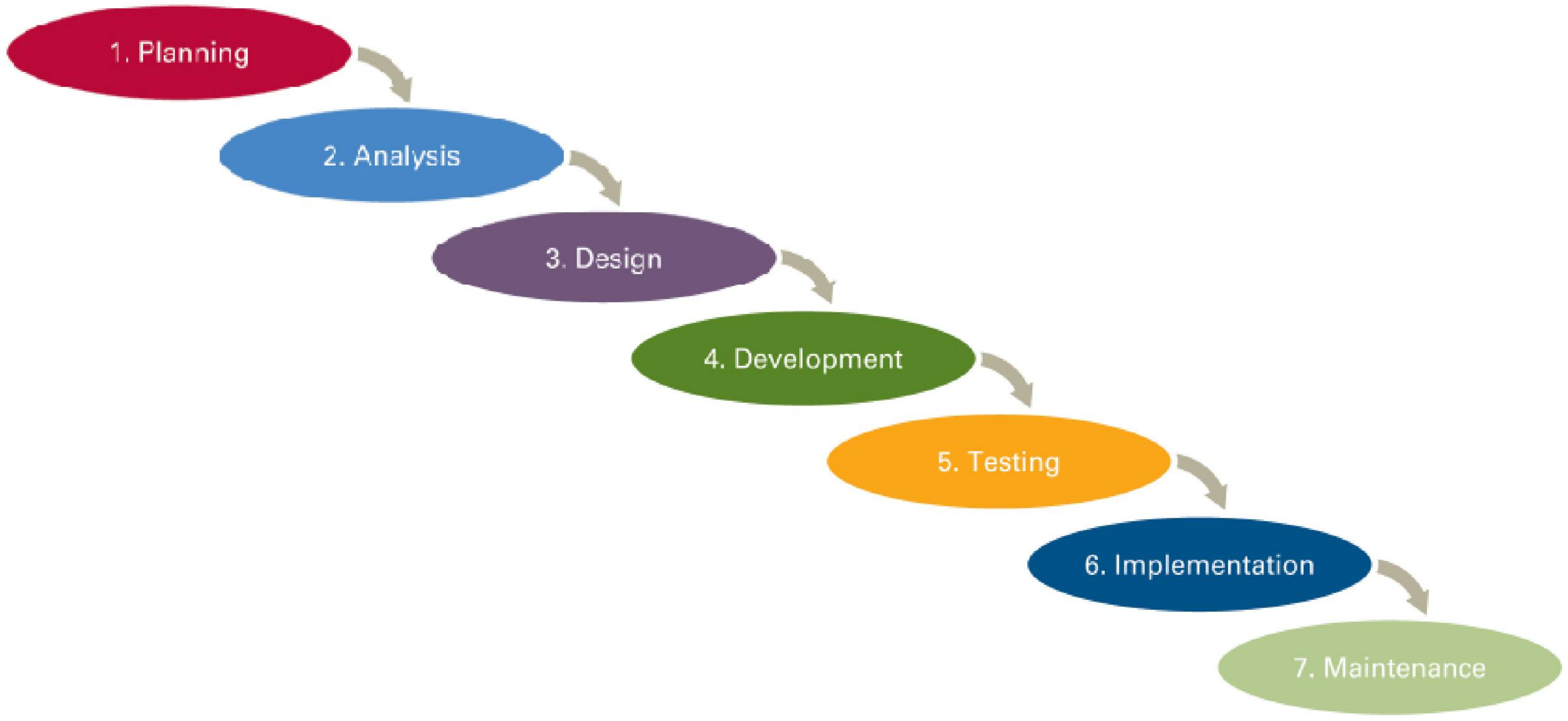
Traditional Systems Development Lifecycle ("*The Waterfall Model*")



Traditional Systems Life Cycle


SYSTEMS INVESTIGATION
Understand problem
SYSTEMS ANALYSIS
Understand solution
SYSTEMS DESIGN
Select and plan best solution
SYSTEMS IMPLEMENTATION
Place solution into effect
SYSTEMS MAINTENANCE AND REVIEW
Evaluate results of solution

Traditional Systems Life Cycle



Traditional Systems Life Cycle

- The systems life cycle is still used for building large complex systems that require a rigorous and formal requirements analysis, predefined specifications, and tight controls over the system-building process.
- However, the systems life cycle approach can be costly, time-consuming, and inflexible.

Traditional Systems Life Cycle

- Although systems builders can go back and forth among stages in the life cycle, the systems life cycle is predominantly a "waterfall" approach in which tasks in one stage are completed before work for the next stage begins.

Limitations of Traditional Systems Life Cycle

- Activities can be repeated, but volumes of new documents must be generated and steps retraced if requirements and specifications need to be revised.
  - This encourages freezing of specifications relatively early in the development process.

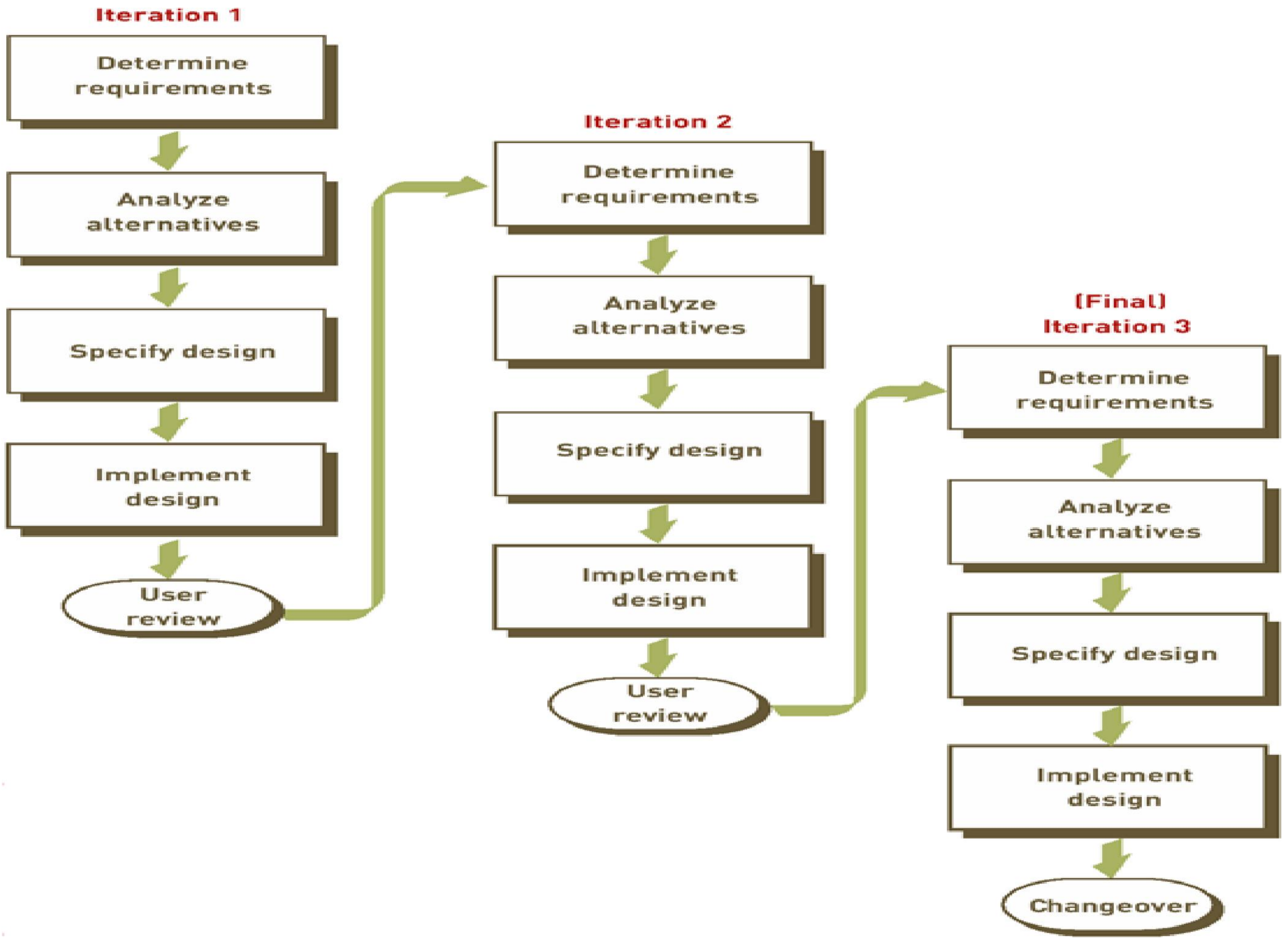
Limitations Traditional Systems Life Cycle

- The life cycle approach is also not suitable for many small desktop systems, which tend to be less structured and more individualized.

Prototyping

- Prototyping consists of building an experimental system rapidly and inexpensively for end users to evaluate.
- By interacting with the prototype, users can get a better idea of their information requirements.
- The prototype endorsed by the users can be used as a template to create the final system.

Prototyping



Prototyping

- The prototype is a working version of an information system or part of the system, but it is meant to be only a preliminary model.
- Once operational, the prototype will be further refined until it conforms precisely to users' requirements.

Prototyping

- Once the design has been finalized, the prototype can be converted to a polished production system.

Prototyping

- The process of building a preliminary design, trying it out, refining it, and trying again has been called an iterative process of systems development because the steps required to build a system can be repeated over and over again.

Prototyping

- Prototyping is more explicitly iterative than the conventional life cycle, and it actively promotes system design changes.
- It has been said that prototyping replaces unplanned rework with planned iteration, with each version more accurately reflecting users' requirements.

4-Step Process in Prototyping

- Step 1:Identify the user’s basic requirements.
- Step 2:Develop an initial prototype.
- Step 3:Use the prototype.
- Step 4:Revise and enhance the prototype.

Step 1:Identify the user’s basic requirements

- The system designer (usually an information systems specialist) works with the user only long enough to capture the user’s basic information needs.

Step 2:Develop an initial prototype

- The system designer creates a working prototype quickly, using tools for rapidly generating software.

Step 3:Use the prototype

- The user is encouraged to work with the system to determine how well the prototype meets his or her needs and to make suggestions for improving the prototype.

Step 4:Revise and enhance the prototype

- The system builder notes all changes the user requests and refines the prototype accordingly.
- After the prototype has been revised, the cycle returns to Step 3.
- Steps 3 and 4 are repeated until the user is satisfied.

Prototyping

- When no more iterations are required, the approved prototype then becomes an operational prototype that furnishes the final specifications for the application.
- Sometimes the prototype is adopted as the production version of the system.

Advantages of Prototyping

- Prototyping is most useful when there is some uncertainty about requirements or design solutions and often used for designing an information system's end-user interface (the part of the system with which end users interact, such as online display and data entry screens, reports, or Web pages).

Advantages of Prototyping

- Because prototyping encourages intense end-user involvement throughout the systems development life cycle, it is more likely to produce systems that fulfill user requirements.

Disadvantages of Prototyping

- However, rapid prototyping can polish over essential steps in systems development.
- If the completed prototype works reasonably well, management may not see the need for reprogramming, redesign, or full documentation and testing to build a polished production system.

Disadvantages of Prototyping

- Some of these hastily constructed systems may not easily accommodate large quantities of data or a large number of users in a production environment.

End-User Development

- Some types of information systems can be developed by end users with little or no formal assistance from technical specialists.
- This phenomenon is called end-user development.

End-User Development

- A series of software tools categorized as fourth-generation languages makes this possible.
- Fourth-generation languages are software tools that enable end users to create reports or develop software applications with minimal or no technical assistance.

End-User Development

- Some of these fourth-generation tools also enhance professional programmers’ productivity.
- Fourth-generation languages tend to be nonprocedural, or less procedural, than conventional programming languages.

End-User Development

- Procedural languages require specification of the sequence of steps, or procedures, that tell the computer what to do and how to do it.
- Nonprocedural languages need only specify what has to be accomplished rather than provide details about how to carry out the task.

Benefits of End-User Development

- End-user-developed systems can be completed more rapidly than those developed through the conventional systems life cycle.
- Allowing users to specify their own business needs improves requirements gathering and often leads to a higher level of user involvement and satisfaction with the system.

Limitations of End-User Development

- Fourth-generation tools still cannot replace conventional tools for some business applications because they cannot easily handle the processing of large numbers of transactions or applications with extensive procedural logic and updating requirements.

Rapid Application Development

- Object-oriented software tools, reusable software, prototyping, and fourth-generation language tools are helping systems builders create working systems much more rapidly than they could using traditional systems-building methods and software tools.

Rapid Application Development (RAD)

- The term rapid application development (RAD) is used to describe this process of creating workable systems in a very short period of time.

Rapid Application Development (RAD)

- RAD can include:
  - The use of visual programming and other tools for building graphical user interfaces
  - Iterative prototyping of key system elements
  - Automation of program code generation, and
  - Close teamwork among end users and information systems specialists.

Rapid Application Development (RAD)

- Simple systems often can be assembled from prebuilt components.
- The process does not have to be sequential, and key parts of development can occur simultaneously.

Rapid Application Development (RAD)

- Sometimes a technique called joint application design (JAD) is used to accelerate the generation of information requirements and to develop the initial systems design.

Rapid Application Development (RAD)

- JAD brings end users and information systems specialists together in an interactive session to discuss the system's design.
- Properly prepared and facilitated JAD sessions can significantly speed up the design phase and involve users at an intense level.

Agile Development

- Agile development focuses on rapid delivery of working software by breaking a large project into a series of small subprojects that are completed in short periods of time using iteration and continuous feedback.

Agile development

- Each mini-project is worked on by a team as if it were a complete project, including planning, requirements analysis, design, coding, testing, and documentation.
- Improvement or addition of new functionality takes place within the next iteration as developers clarify requirements.

Agile Development

- This helps to minimize the overall risk, and allows the project to adapt to changes more quickly.
- Agile methods emphasize face-to-face communication over written documents, encouraging people to collaborate and make decisions quickly and effectively.

Component-based Development

- Object-oriented development can be used for building systems that can respond to rapidly changing business environments, including Web applications.
- A software component is a system element offering a predefined service or event, and able to communicate with other components.

Component-based Development

- To further speed up software creation, groups of objects have been assembled to provide software components for common functions such as a graphical user interface or online ordering capability that can be combined to create large-scale business applications.

Component-based Development

- component-based development enables a system to be built by assembling and integrating existing software components.
- Increasingly, these software components are coming from cloud services.

Component-based Development

- Businesses are using component-based development to create their e-commerce applications by combining commercially available components for:
  - Shopping carts
  - User authentication
  - Search engines, and
  - Catalogs with pieces of software for their own unique business requirements.

End of Chapter 2