

# INTRODUCTION TO SYSTEM ANALYSIS AND DESIGN



# WHAT IS AN INFORMATION SYSTEM?

An information system is a collection of interrelated components that **collect**, **process** and **store**, and provide as **output** the information needed to complete a business task.

# EXAMPLES OF INFORMATION SYSTEMS

- Course registration system
- Online order system
- Online banking system

# WHAT IS SYSTEM ANALYSIS ABOUT?

- Understanding the goals and strategies of the business.
- Defining the information requirements that support those goals and strategies.
- It is not about programming.

# SYSTEM ANALYSIS VS. SYSTEM DESIGN

## System Analysis:

- Investigation of the problem and requirement rather than solution.

## System Design:

- A conceptual solution that fulfills the requirements, rather than implementation.



# SYSTEM ANALYST

A business professional who uses analysis and design techniques to solve business problems using information technology.

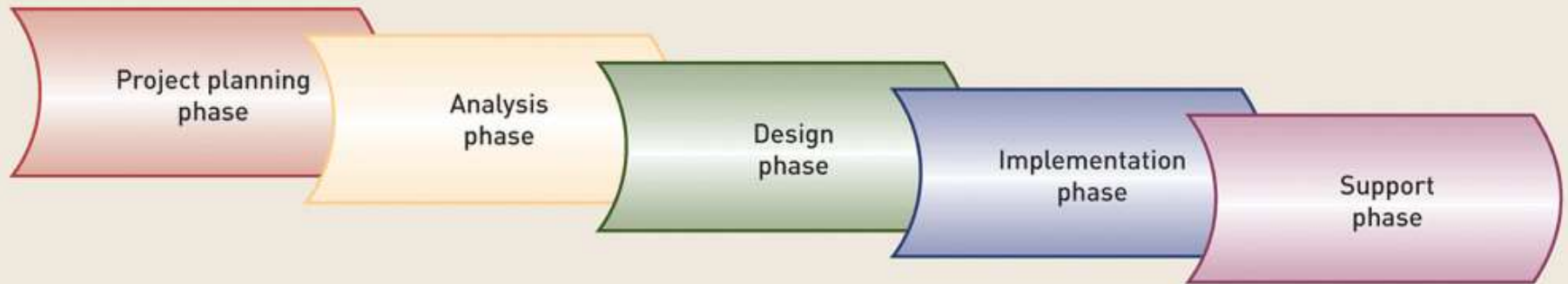
# THE ROLE OF A SYSTEM ANALYST

- Business knowledge.
- Business problem solver.
- Help translate business requirements into IT projects.

# TRADITIONAL SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)

Figure 2-2


Information system development phases





# TRADITIONAL SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)

- **Project planning** – initiate, ensure feasibility, plan schedule, obtain approval for project
- **Analysis** – understand business needs and processing requirements
- **Design** – define solution system based on requirements and analysis decisions
- **Implementation** – construct, test, train users, and install new system
- **Support** – keep system running and improve it



Each of the phases include a set of steps, which rely on techniques that produce specific deliverables that provide understanding about the project.



## To Understand the SDLC:

- Each phase consists of steps that lead to specific deliverables
- The system evolves through gradual refinement



# PLANNING

This phase is the fundamental process of understanding why an information system should be built.

The Planning phase will also determine how the project team will go about building the information system.

The Planning phase is composed of two planning steps.

# PLANNING

1. During **project initiation**, the system's **business value** to the organization is identified (How will it lower costs or increase revenues?) as well as the **feasibility** of the project from different point of views
1. During **project management**, the project manager creates a work plan, staffs the project, and puts techniques in place to help the project team control and direct the project through the entire SDLC.



# ANALYSIS

The analysis phase answers the questions of who will use the system, what the system will do, and where and when it will be used.

During this phase the project team investigates any current system(s), identifies improvement opportunities, and develops a concept for the new system.

This phase has three analysis steps.

# ANALYSIS

1. **Analysis strategy:** This is developed to guide the projects team's efforts. This includes an analysis of the current system.
2. **Requirements gathering:** The analysis of this information leads to the development of a concept for a new system. This concept is used to build a set of analysis models.
3. **System proposal:** The proposal is presented to the project sponsor and other key individuals who decide whether the project should continue to move forward.



# ANALYSIS

The system proposal is the initial deliverable that describes what business requirements the new system should meet.

The deliverable from this phase is both an analysis and a high-level initial design for the new system.





# DESIGN

In this phases it is decided how the system will operate, in terms of the hardware, software, and network infrastructure; the user interface, forms, and reports that will be used; and the specific programs, databases, and files that will be needed.

# DESIGN

1. **Design Strategy:** This clarifies whether the system will be developed by the company or outside the company.
2. **Architecture Design:** This describes the hardware, software, and network infrastructure that will be used.
3. **Database and File Specifications:** These documents define what and where the data will be stored.
4. **Program Design:** Defines what programs need to be written and what they will do.

## UI Design



# IMPLEMENTATION

During this phase, the system is either developed or purchased (in the case of packaged software).

This phase is usually the longest and most expensive part of the process.

The phase has three steps.

# IMPLEMENTATION

**System Construction:** The system is built and tested to make sure it performs as designed.

**Installation:** Prepare to support the installed system.

**Support Plan:** Includes a post-implementation review.

# CATEGORY I OF THE SYSTEM DEVELOPMENT METHODOLOGY: STRUCTURED DESIGN

Structured design methodologies adopt a formal step-by-step approach to the SDLC that moves logically from one phase to the next.

This design methodology introduces the use of formal modeling or diagramming techniques to describe business processes (process models) and data (data models).

# WATERFALL DEVELOPMENT

With waterfall development- based methodologies, the analysts and users proceed sequentially from one phase to the next.

The two key advantages of waterfall development-based methodologies are:

- The system requirements are identified long before programming begins.
- Changes to the requirements are minimized as the project proceeds.

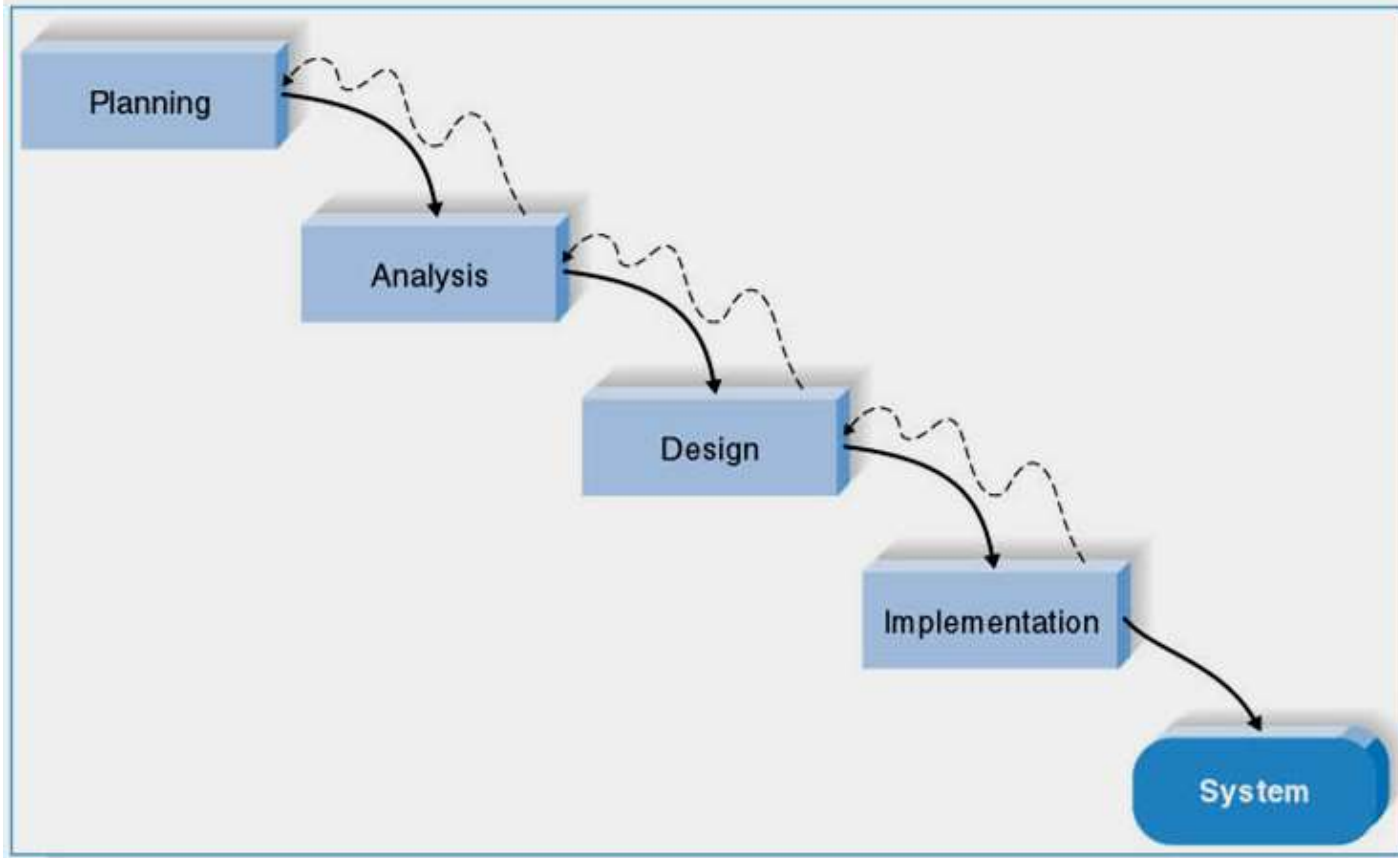
# WATERFALL DEVELOPMENT

The two key disadvantages of waterfall development-based methodologies are:

- The design must be completely specified before programming begins.

- A long time elapses between the completion of the system proposal in the analysis phase and the delivery of the system.

# WATERFALL DEVELOPMENT-BASED METHODOLOGY





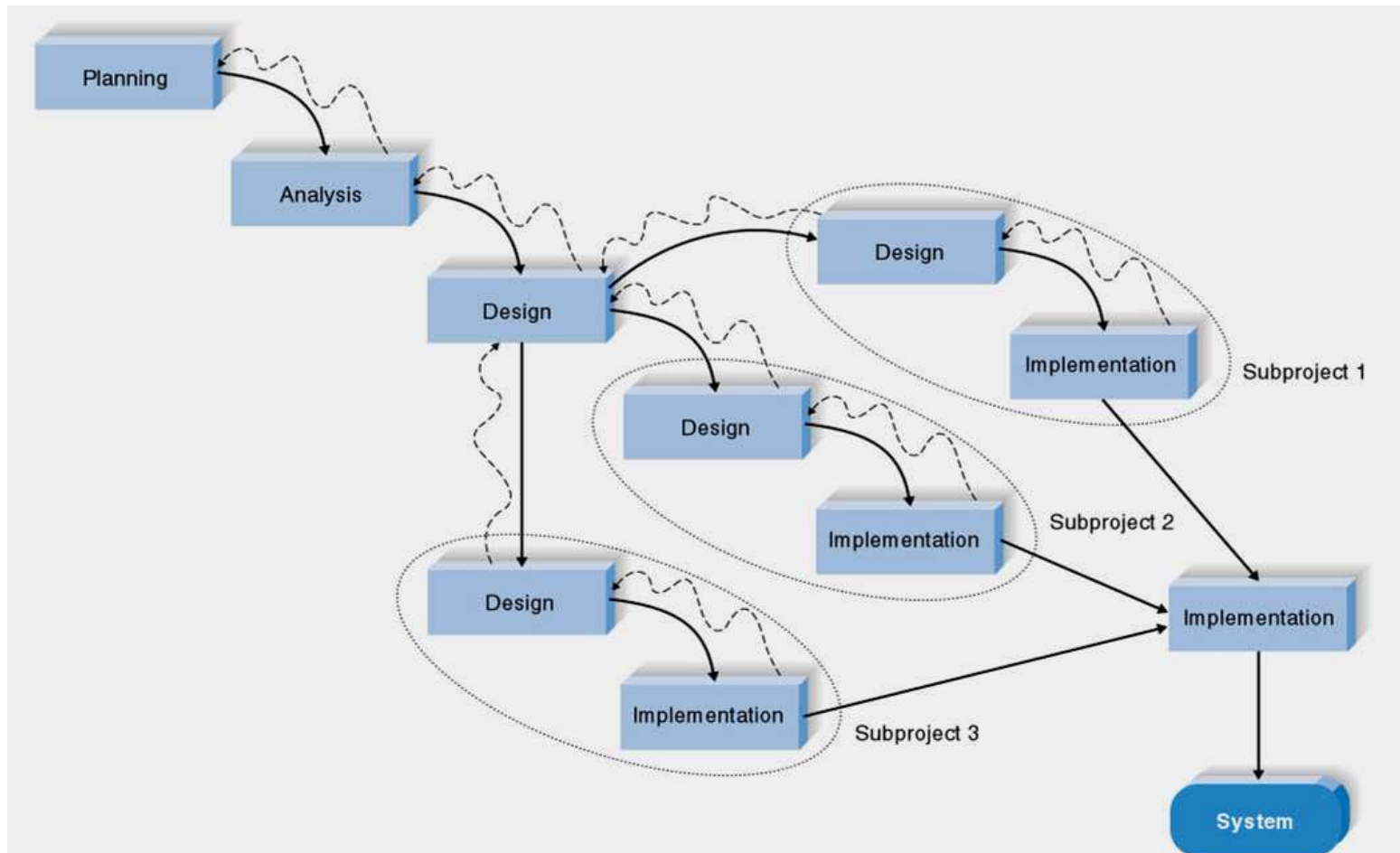
# PARALLEL DEVELOPMENT

This methodology attempts to address the long time interval between the analysis phase and the delivery of the system.

Additional work:

- Project division
- Integration at the end.

A GENERAL ANALYSIS/DESIGN FOR THE ENTIRE SYSTEM IS PERFORMED AND THEN THE PROJECT IS DIVIDED INTO A SERIES OF DISTINCT SUBPROJECTS.



# CATEGORY II OF THE SYSTEM DEVELOPMENT METHODOLOGY: RAPID APPLICATION DEVELOPMENT (RAD)

RAD-based methodologies adjust the SDLC phases to get some part of system developed quickly and into the hands of the users.

Most RAD-based methodologies recommend that analysts use special techniques and computer tools to speed up the analysis, design, and implementation phases, such as CASE (computer-aided software engineering) tools.



# RAD

One possible problem with RAD-based methodologies is managing user expectations.

# RAD: PHASED DEVELOPMENT

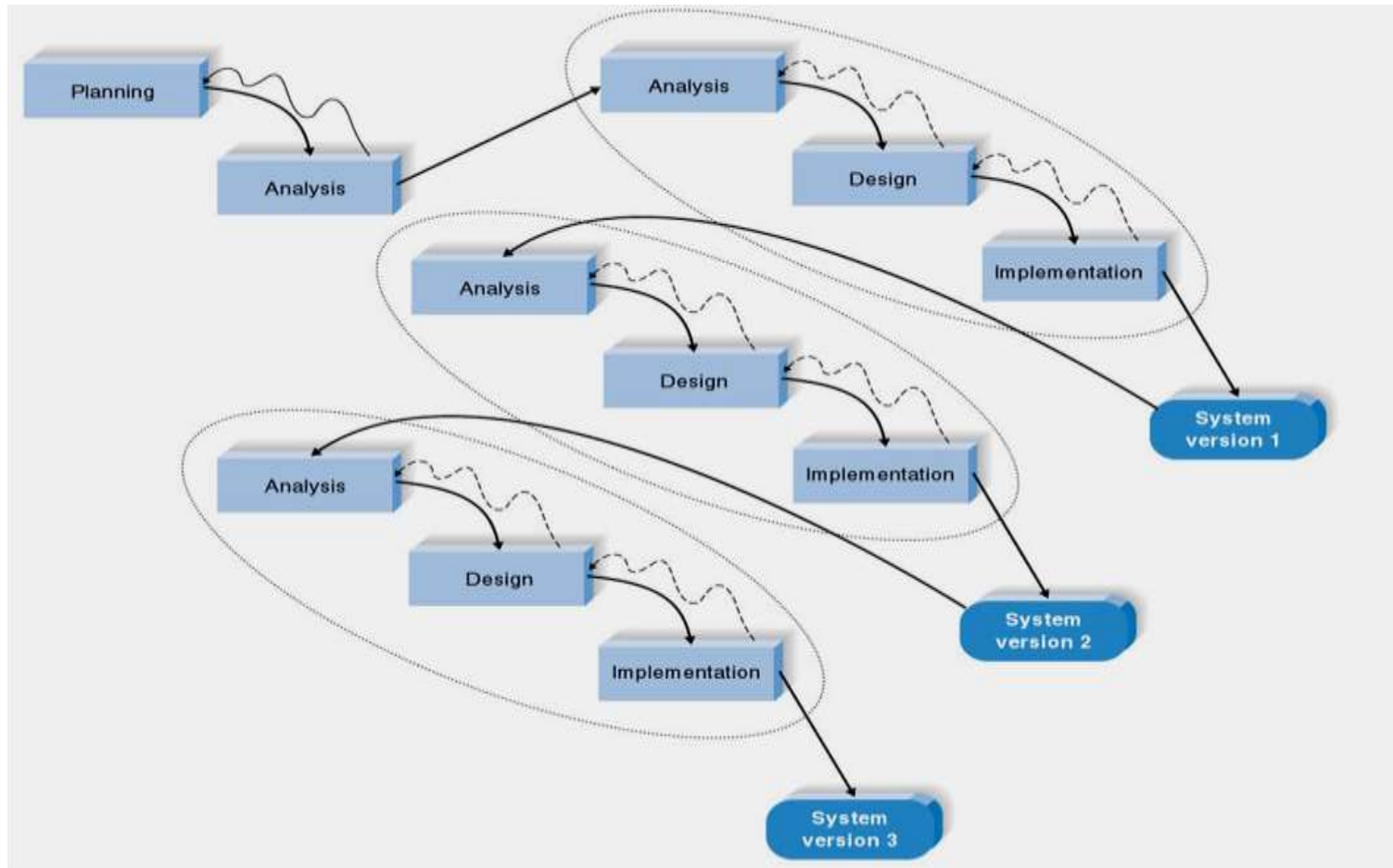
This methodology breaks the overall system into a series of versions that are developed sequentially.

The team categorizes the requirements into a series of versions, then the most important and fundamental requirements are bundled into the first version of the system.

The analysis phase then leads into design and implementation; however, only with the set of requirements identified for version 1.

As each version is completed, the team begins work on a new version.

# PHASED DEVELOPMENT-BASED METHODOLOGY



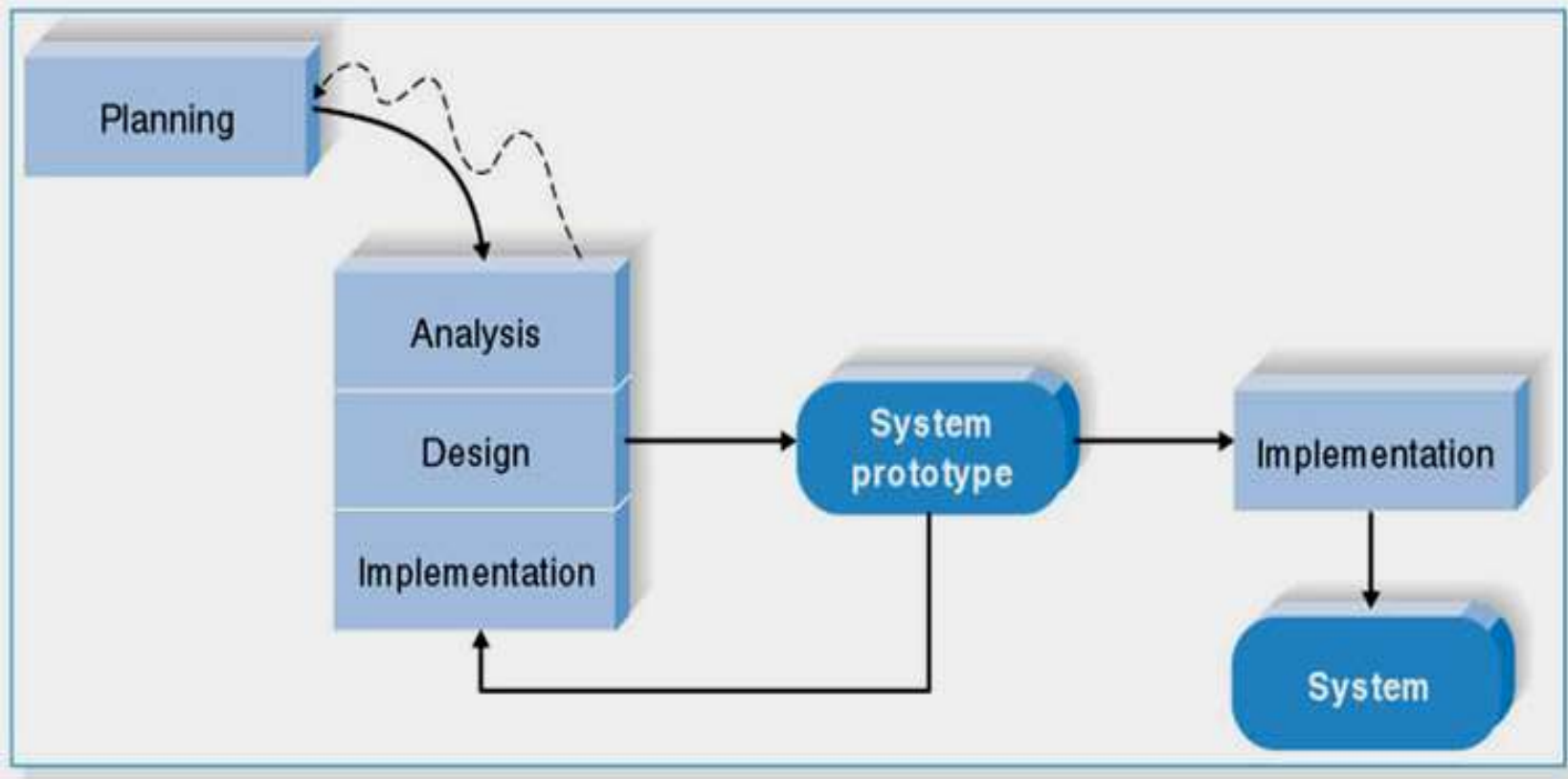
# RAD: PROTOTYPING

Prototyping-based methodologies perform the analysis, design and implementation phases concurrently.

All three phases are performed repeatedly in a cycle until the system is completed.

A prototype is a smaller version of the system with a minimal amount of features.

# PROTOTYPING-BASED METHODOLOGY





# RAD: PROTOTYPING

Advantage: Provides a system for the users to interact with, even if it is not initially ready for use. The user can give feedback.

Disadvantages:

- Manage user expectations.
- Forget some important points since we are prototyping (opposite of careful design)

## CATEGORY III OF THE SYSTEM DEVELOPMENT METHODOLOGY: AGILE DEVELOPMENT

This category focuses on streamlining the SDLC by eliminating much of the modeling and documentation overhead and the time spent on those tasks.

Projects emphasize simple, iterative application development.

This category uses extreme programming, which is described next.

# EXTREME PROGRAMMING (XP)

Extreme Programming (XP) was founded on four core values:

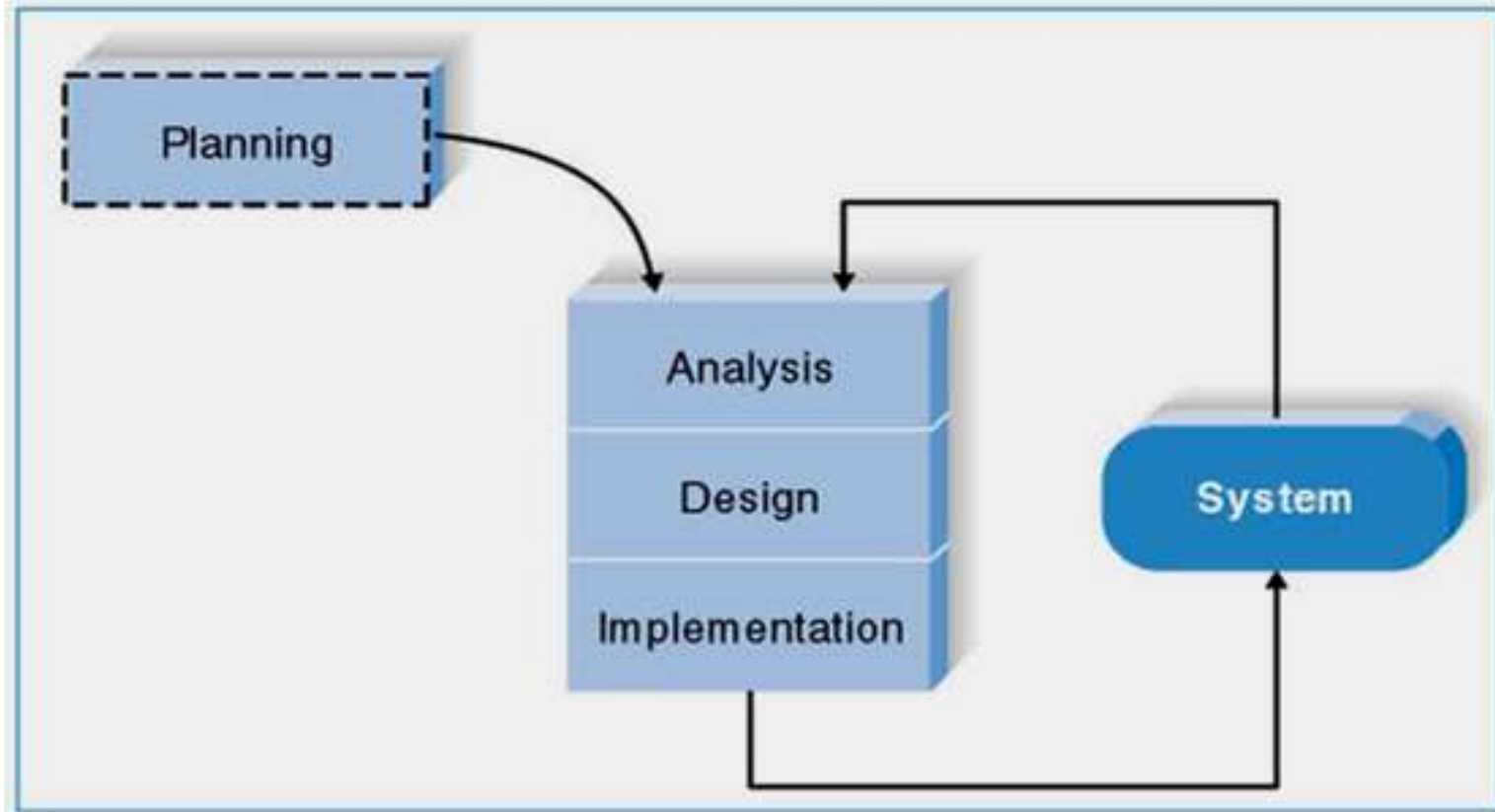
- Communication
- Simplicity
- Feedback
- Courage

# EXTREME PROGRAMMING (XP)

## Key principles of XP include:

- Continuous testing
- Simple coding
- Close interaction with the end users to build systems very quickly

# AN EXTREME PROGRAMMING-BASED METHODOLOGY



# SELECTING THE APPROPRIATE DEVELOPMENT METHODOLOGY

Selecting a methodology is not simple, as no one methodology is always best.

Many organizations have their own standards.

The next figure summarizes some important methodology selection criteria.

# SELECTION CRITERIA

**Clarity of requirements**

**Familiarity with technology**

**System complexity**

**System reliability**

**Short time schedule**

**Schedule visibility**

**Others**

# CLARITY OF USER REQUIREMENTS

RAD methodologies of prototyping is usually more appropriate when user requirements are unclear as they provide prototypes for users to interact with early in the SDLC.



# FAMILIARITY WITH TECHNOLOGY

If the system is designed without some familiarity with the base technology, risks increase because the tools may not be capable of doing what is needed.

# SYSTEM COMPLEXITY

Complex systems require careful and detailed analysis and design.

Project teams who follow phased development-based methodologies tend to devote less attention to the analysis of the complete problem domain than they might if they were using other methodologies.



# SYSTEM RELIABILITY

System reliability is usually an important factor in system development.

# SHORT TIME SCHEDULES

RAD-based methodologies are well suited for projects with short time schedules as they increase speed.

Waterfall-based methodologies are the worst choice when time is essential as they do not allow for easy schedule changes.

# SCHEDULE VISIBILITY

RAD-based methodologies move many of the critical design decisions earlier in the project; consequently, this helps project managers recognize and address risk factors and keep expectations high.

# SELECTING A METHODOLOGY

Ability to Develop Systems	Structured Methodologies		RAD Methodologies			Agile Methodologies
	Waterfall	Parallel	Phased	Prototyping	Throwaway Prototyping	XP
with Unclear User Requirements	Poor	Poor	Good	Excellent	Excellent	Excellent
with Unfamiliar Technology	Poor	Poor	Good	Poor	Excellent	Poor
that are Complex	Good	Good	Good	Poor	Excellent	Poor
that are Reliable	Good	Good	Good	Poor	Excellent	Good
with a Short Time Schedule	Poor	Good	Excellent	Excellent	Good	Excellent
with Schedule Visibility	Poor	Poor	Excellent	Excellent	Good	Good

# TWO APPROACHES TO SYSTEM DEVELOPMENT

## Traditional (Structured) approach

- Also called structured system development
- Structured analysis and design technique (SADT)
- Includes information engineering (IE)

## Object-oriented approach

- Also called OOA, OOD, and OOP
- Views information system as collection of interacting objects that work together to accomplish tasks

# OBJECT-ORIENTED APPROACH

- Completely different approach to information systems
- Views information system as collection of interacting objects that work together to accomplish tasks
  - Objects – things in computer system that can respond to messages
  - Conceptually, no processes, programs, data entities, or files are defined – just objects
- OO languages: Java, C++, C# .NET, VB .NET



# UNIFIED MODELING LANGUAGE (UML)

UML (Unified Modeling Language) is a graphical language that is suitable to express software or system requirements, architecture, and design.

UML used for both database and software modeling

UML modeling also supports multiple views of the same system.

# UML DIAGRAMS

Can be categorized as the following:

- **Structural diagrams:**

Show the building blocks of your system—features that don't change with time.

- Ex: Class diagram

- **Behavioral diagrams:**

Show how your system responds to requests or otherwise evolves over time.

- Ex: Use case diagram

- **Interaction diagrams:**

Is a type of behavioral diagram.

To depict the exchange of messages within a *collaboration* (a group of cooperating objects).

- Ex: Sequence diagram & Collaboration diagram

# UML DIAGRAMS

Another way of categorizing **UML** diagram:

- 1. Static diagrams**

- to show the static features of the system. (no change)

- 2. Dynamic diagrams**

- to show how your system evolves over time.

- 3. Functional diagrams:**

- to show the details of behaviors and algorithms.

# OBJECT-ORIENTED ANALYSIS (OOA)

Trying to figure out what the users and customers of a software effort want the System to do.

Builds a “real-world” model from requirements

- client interviews, domain knowledge, real-world experience collected in use cases and other simple notations

OOA models address three aspects of the system (its objects)

- class structure and relationships
- sequencing of interactions and events
- data transformations and computations