UNIVERSITAS INDONESIA

Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

TOPIC 1

INTRODUCTION TO INFORMATION SYSTEMS ANALYSIS AND DESIGN

ANALISIS DAN PERANCANGAN SISTEM INFORMASI

CSIM603183

Learning Objectives

1. Able to elaborate what **computer-based information systems** is.
2. Able to explain the **system development lifecycle** (SDLC) principles and the deliverables for each phase.
3. Able to explain the evolution of **information systems development methodology.**
4. Able to elaborate the **role** of each member in **IS project team.**
5. Able to elaborate the **characteristics** of **OO Systems and OO Systems Design and Analysis.**
6. Able to elaborate the component of **UML Diagram**

Session Outline

1. Computer-based Information Systems
2. The Principle of Information Systems Development Lifecycle (SDLC) and the Role of System Analyst
3. IS Development Methodology
4. Building Team in IS Development Project

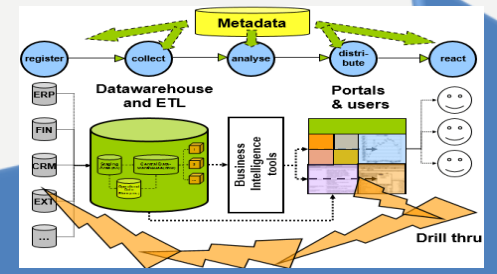
1.1 COMPUTER-BASED INFORMATION SYSTEMS AND THE ROLE OF SYSTEM ANALYST

What are the characteristics of Information Systems?

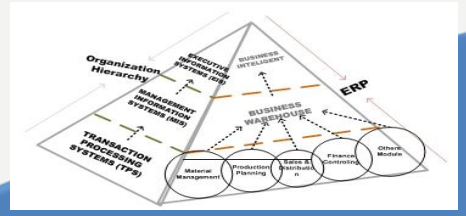
What is Computer Based Information Systems?

- Information System is a set of interrelated **elements or components** that collect (input), manipulate (process), and disseminate (output) data and information and provide a feedback mechanism to meet **an objective.**
  - What are the elements? **People, Data, Processes and Procedures, Technology, Tools, etc.**
  - What objective(s) need to be achieve? **Organization objectives**
- We focus on Computer-based Information System.
  - By definition, Information System is not necessarily needs computer.
  - It's a computer application to perform a certain tasks.

Information Systems Evolution



Business Intelligence Systems



Integrated Information Systems: ERP

Stand alone Information System

Manual

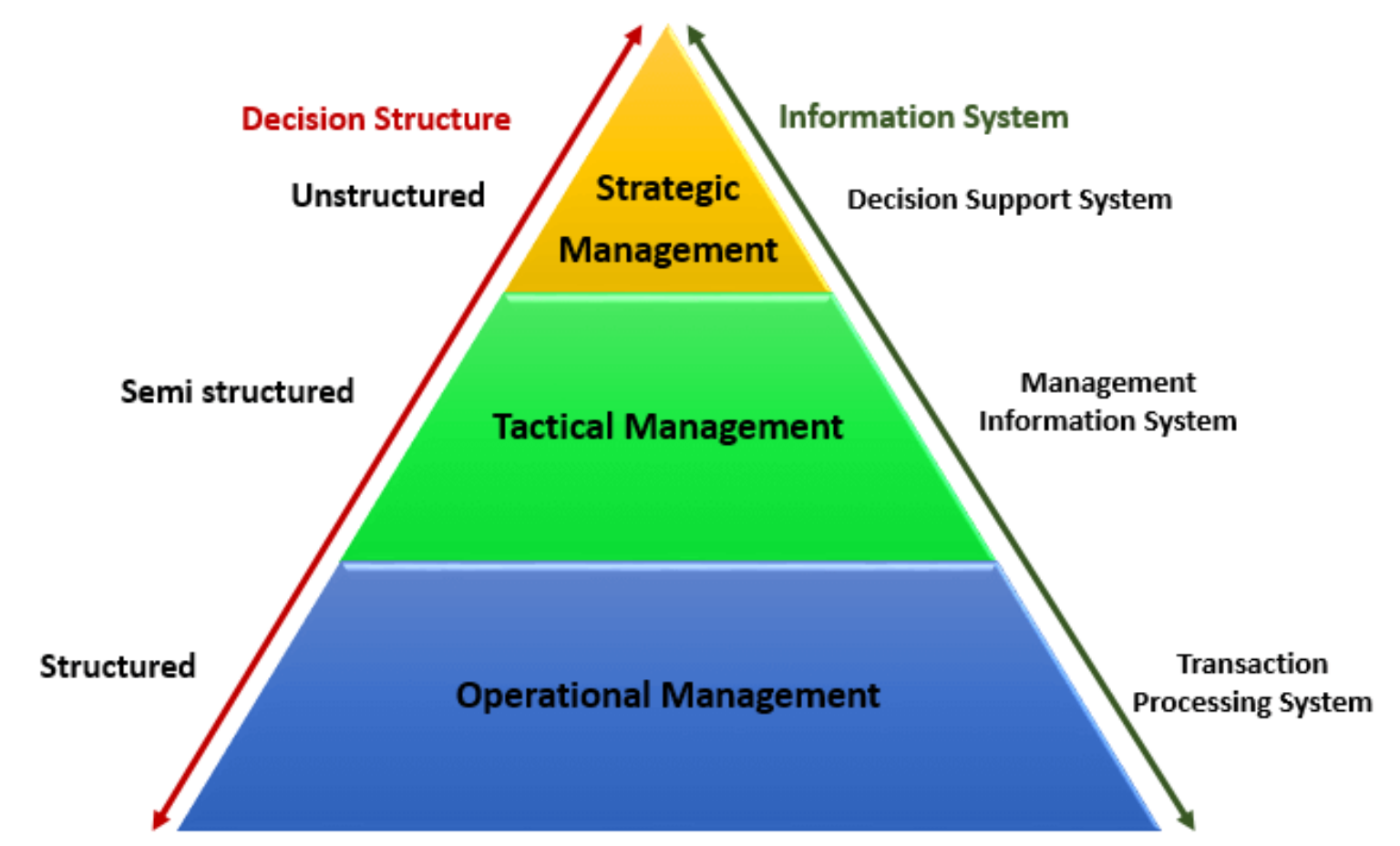
Cloud Computing, Big Data, and Mobile Technology have changed the utilization of IT Services.

Information Systems Evolution

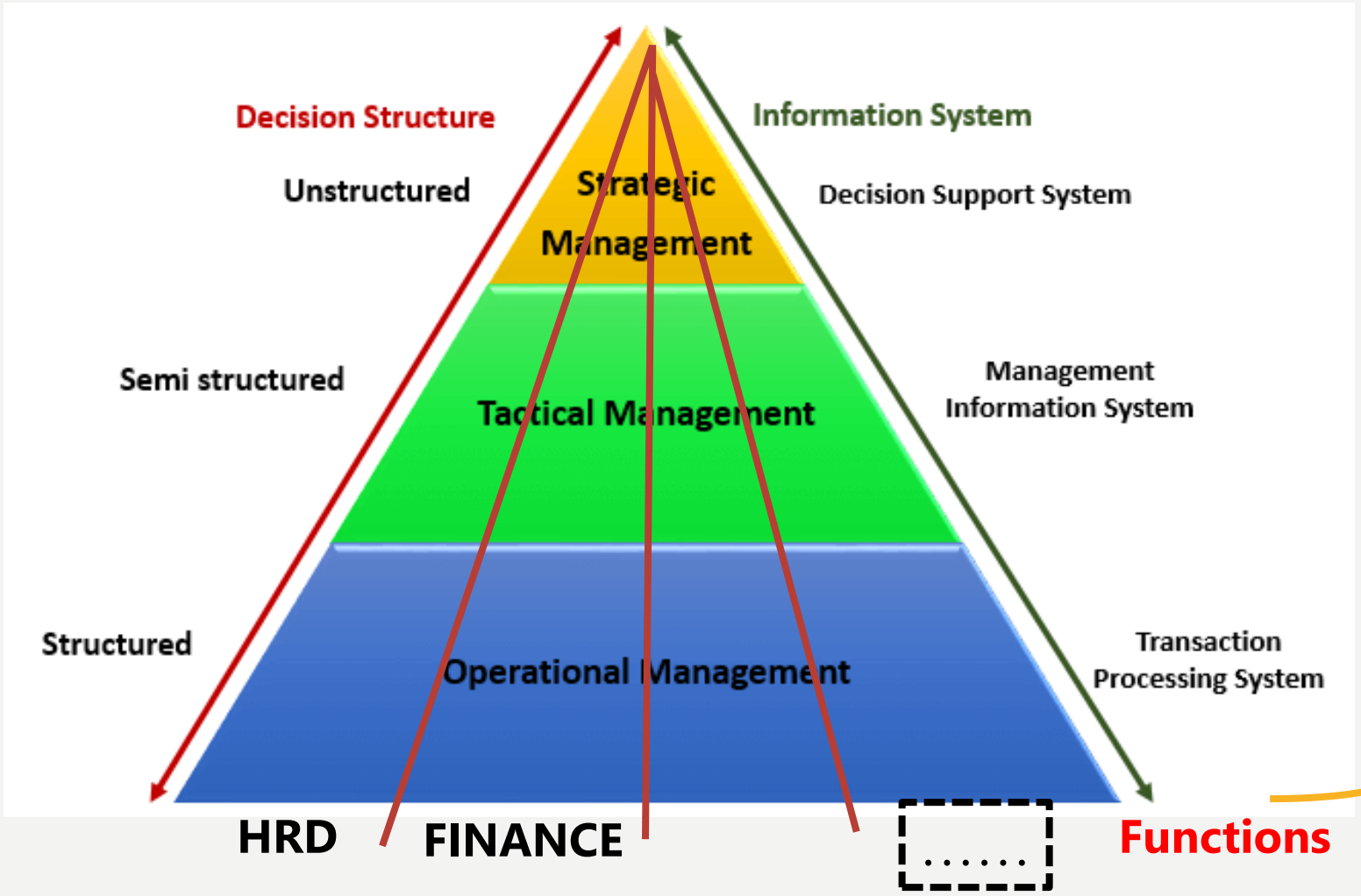
Era	Information System	Data Processing
1960	Transaction Processing System	Automate repeating activity
1970	Management Information System	Recap weekly, monthly, etc. functional data
1980	Strategic Information System	Process External and Internal data
1990	e-Business, e-Commerce, e-Gov.	Process Multi-stakeholder Online Data
2000	Enterprise Resource Planning	Integrate Functional Data
2006	Business Intelligent	Process Temporal and Spatial Data become Big Data

What are the differences among Transaction Processing Systems, Management Information Systems, and Strategic IS

Basic Type of IS



Basic Type of IS

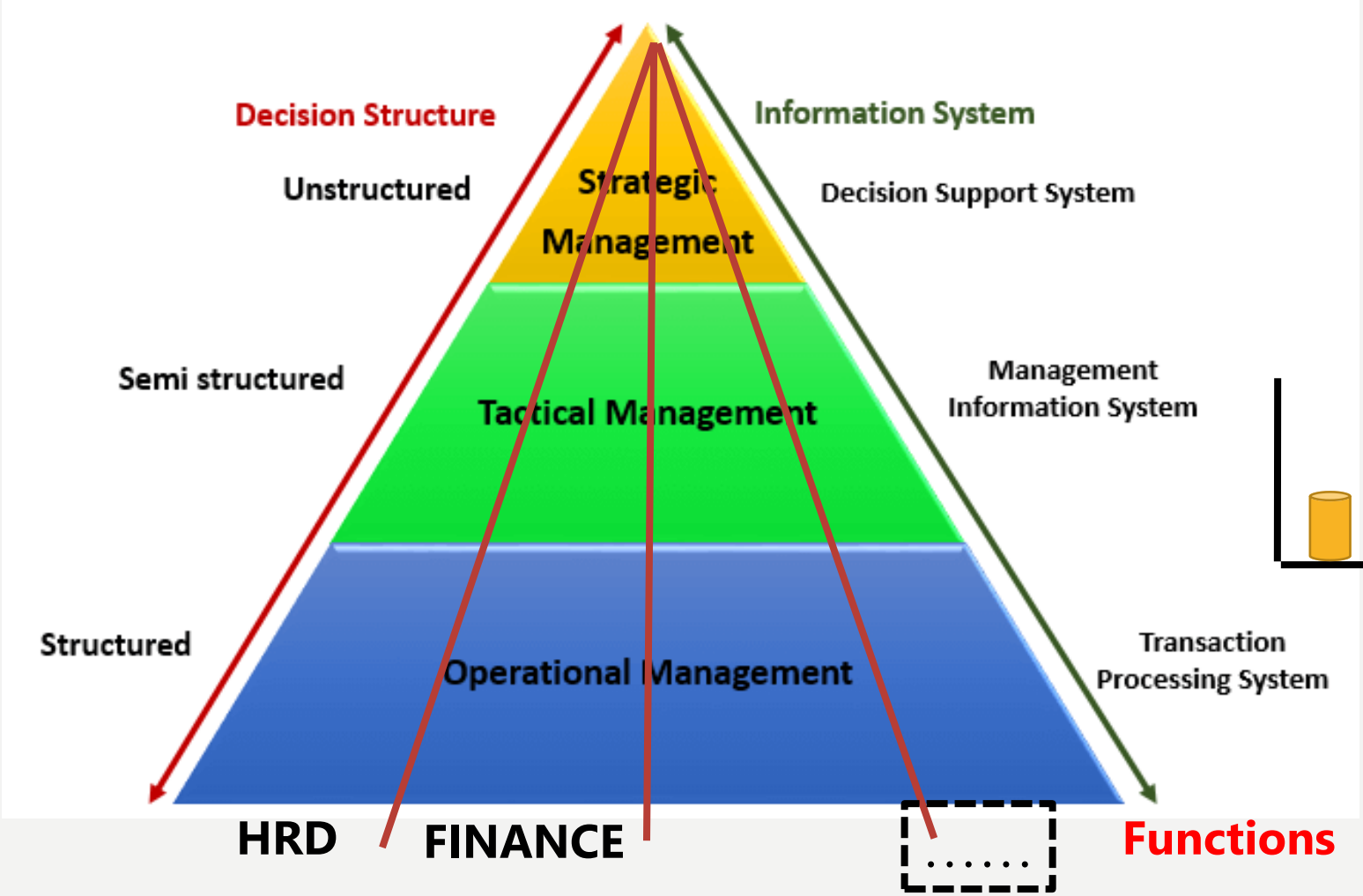


Business Intelligence

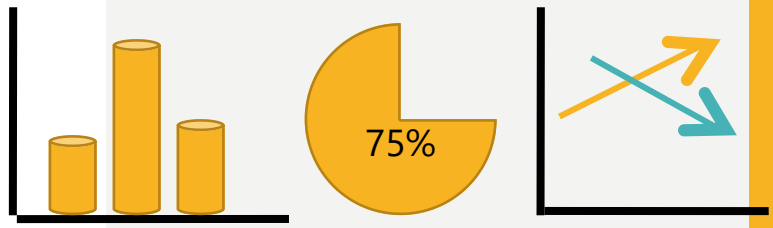
ERP

IS

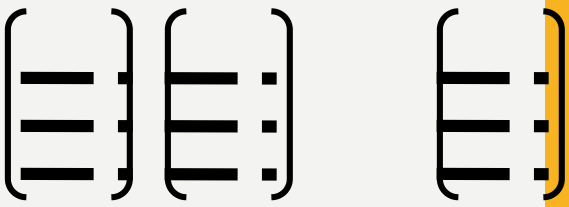
Basic Type of IS



Dashboard



A Set of Graphics



A Set of Tables

1.2 THE SYSTEMS DEVELOPMENT LIFE CYCLE (SDLC)

(Information) Systems Development Lifecycle: SDLC

- The process of understanding how an information system (IS) **can support business needs** by designing a system, building it, and delivering it to users.
- Sounds pretty simple…
- Is it? …

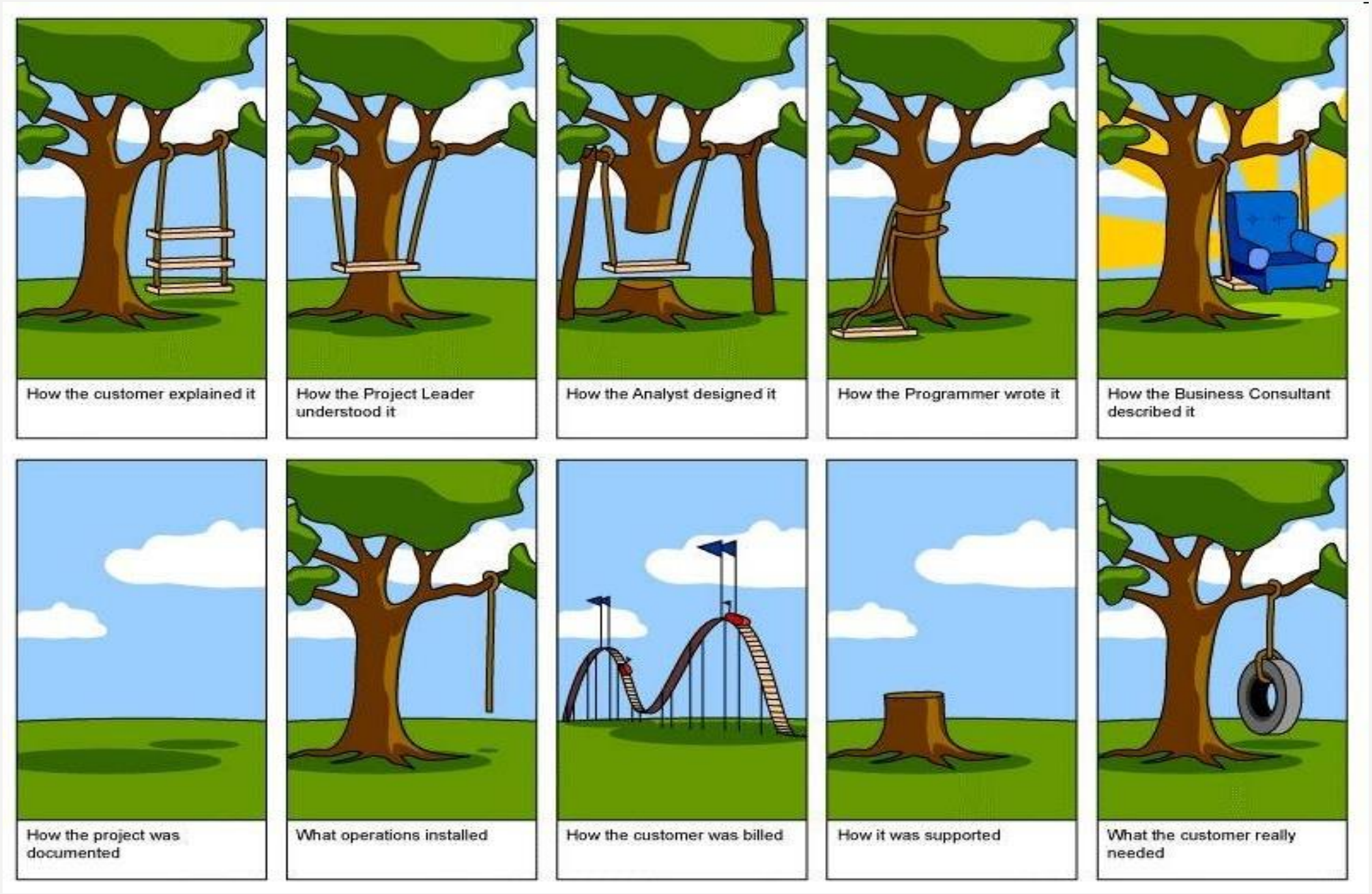
What are the importance of SDLC?

The Importance of SDLC

- The Standish Group (1996)
  - 42% were abandoned before completion
- General Accounting Office (1996)
  - 53% all US Government IS projects were abandoned
  - Many of the remaining were delivered very late, over budget, fewer features than planned.
- IAG Consulting
  - 80% were over time
  - 72% were over budget
  - 55% delivered less than the full functionality
- Panorama Consulting Solutions
  - 54% ERP projects were over time
  - 56% were over budget
  - 48% delivered less than 50% of full functionality
- IBM
  - 59% projects missed one or more of on time, within budget and quality constraints.

Key Person in SDLC

System Analyst


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Key Issues

- Many failed systems were abandoned because analysts tried to build wonderful systems **without understanding how the system would fit with the organization's goals**
- The primarily **goal** of information system is to **create value** for the organization → profit for most organization/company

Key Issues: What should system analyst do?

To Create Value for the Organization!!!

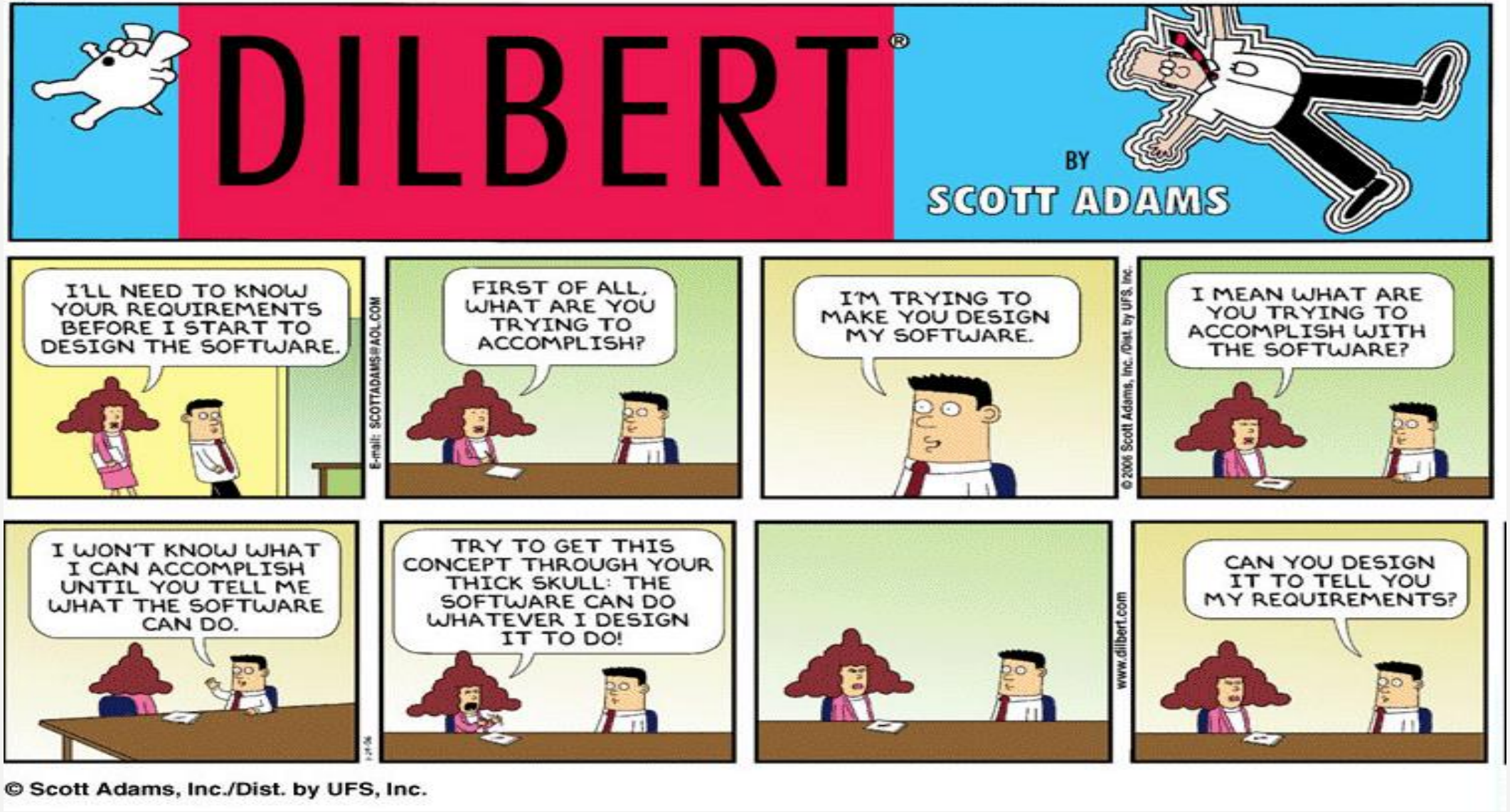
1. Analyze business situation
2. Identify opportunities for improvements
3. Design an IS to implement them

Key Issues

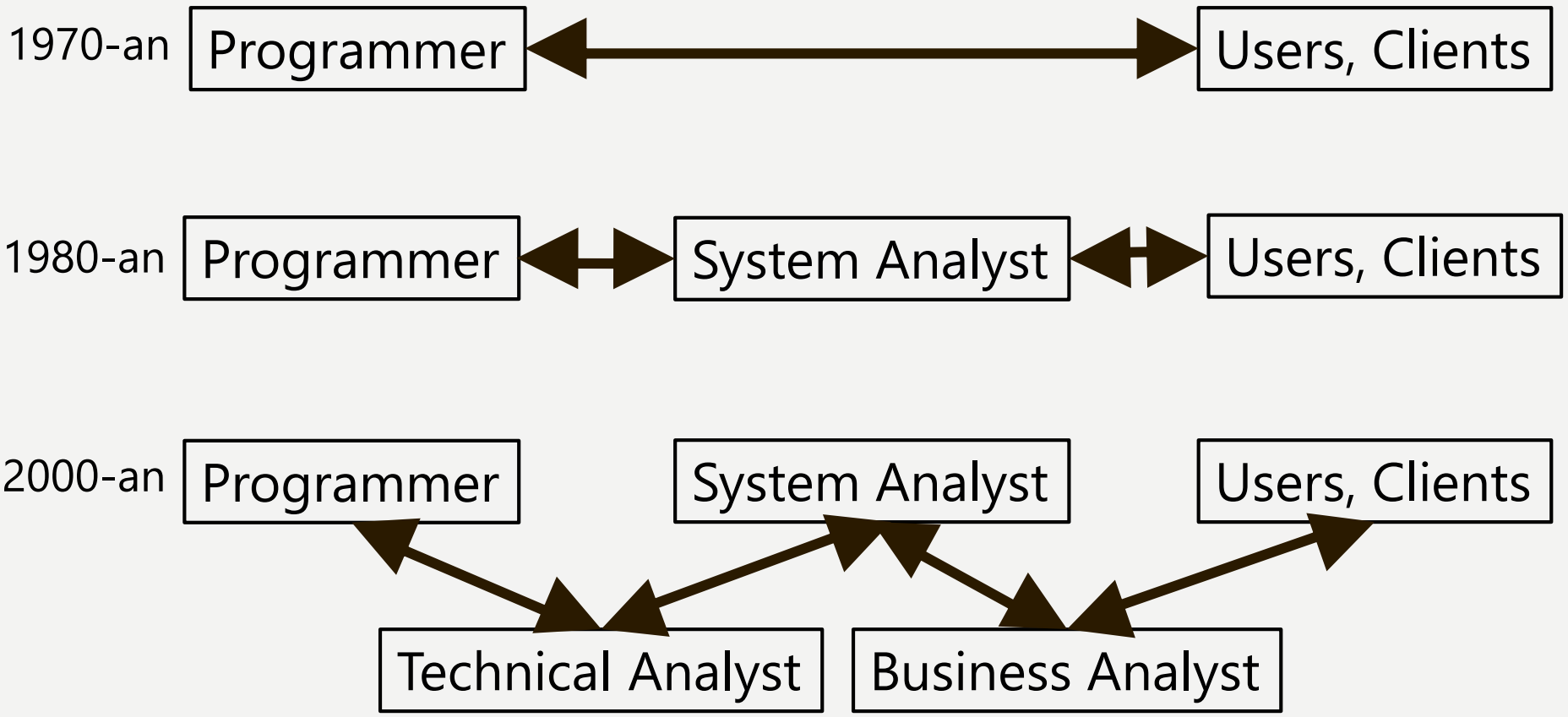
- The **systems analyst** is a key person analyzing the business, identifying opportunities for improvement, and designing information systems to implement these ideas.
- It is important to understand and develop through practice the skills needed to successfully design and implement new information systems.

Dilbert

by Scott Adams



The Changing Roles of System Analyst



Building Information Systems

- Building IS is similar to build a house
  - Starts with basic idea
  - Transformed into a simple drawing and shown to customer and refined until customer agree
  - A set of blueprint is designed
  - The house is built following the blueprint
- In IS, it is called System Development Life Cycle (SDLC) which has a similar of 4-fundamental phases (**Planning**, **Analysis**, **Design** and **Implementation**)
- Each phases is composed of **a series of steps**, which rely upon **techniques** that produce **deliverables** (specific documents and files that provide understanding about the project)

Information Systems Development Lifecycle

- In many project, SDLC phases and steps proceed in a logical path from start to finish, while in other project, the project teams move through the steps consecutively, incrementally, iteratively, or in other pattern.
- SLDC is a **gradual refinement** (each phase refines and elaborates on the work done previously)
  - Deliverables produced in the analysis phase provide a general idea of the shape of the new system
  - These are used as input to the design phase which then refines them to produce a set of more detailed deliverables.
  - These deliverables, in turn, are used in the implementation phase to produce the actual system.

Analyze for each phase of SDLC steps and how they are related each other!

Phases of SDLC

- **Planning**
  - Why build the system?
  - How the project team will go to build it?
- **Analysis**
  - Who, what, when, where will the system be?
- **Design**
  - How will the system will operate, in terms of the hardware, software and infrastructure?
- **Implementation**
  - The system is actually built or purchased
  - System delivery

#1 Planning

- Project Initiation
  - Identify business value (how will the IS lower costs or increase revenue ?)
    - A system request presents a brief summary of business need, and explain how the system will create business value
  - Analyze feasibility (technical, economic and organizational)
- Project Management
  - Develop work plan
  - Staff the project
  - Control and direct project

#2 Analysis

- Analysis Strategy
  - Analyze current system (as-is system) and new system (to-be system)
- Requirement gathering
  - Interview or questionnaires or other method
  - Analysis Model (Process and Data)
- System Proposal
  - Describe what business requirements of the new system should met.

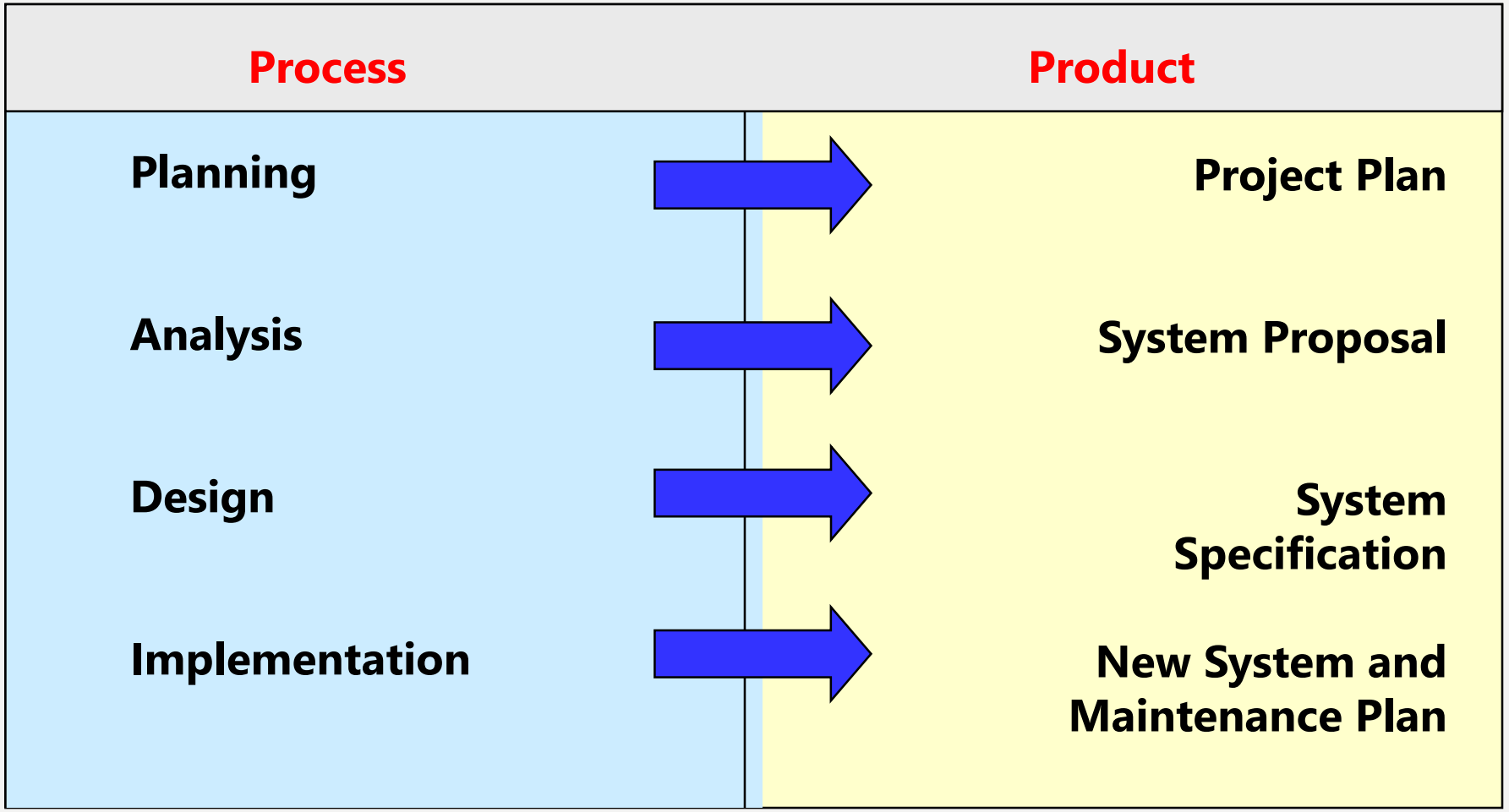
#3 Design

- Design Strategy
  - Build it, outsource or buy ?
- Architectural & Interface Design
  - Describe h/w, s/w, network infrastructure
  - How the users will move through the system
- Database and file specification
- Program design

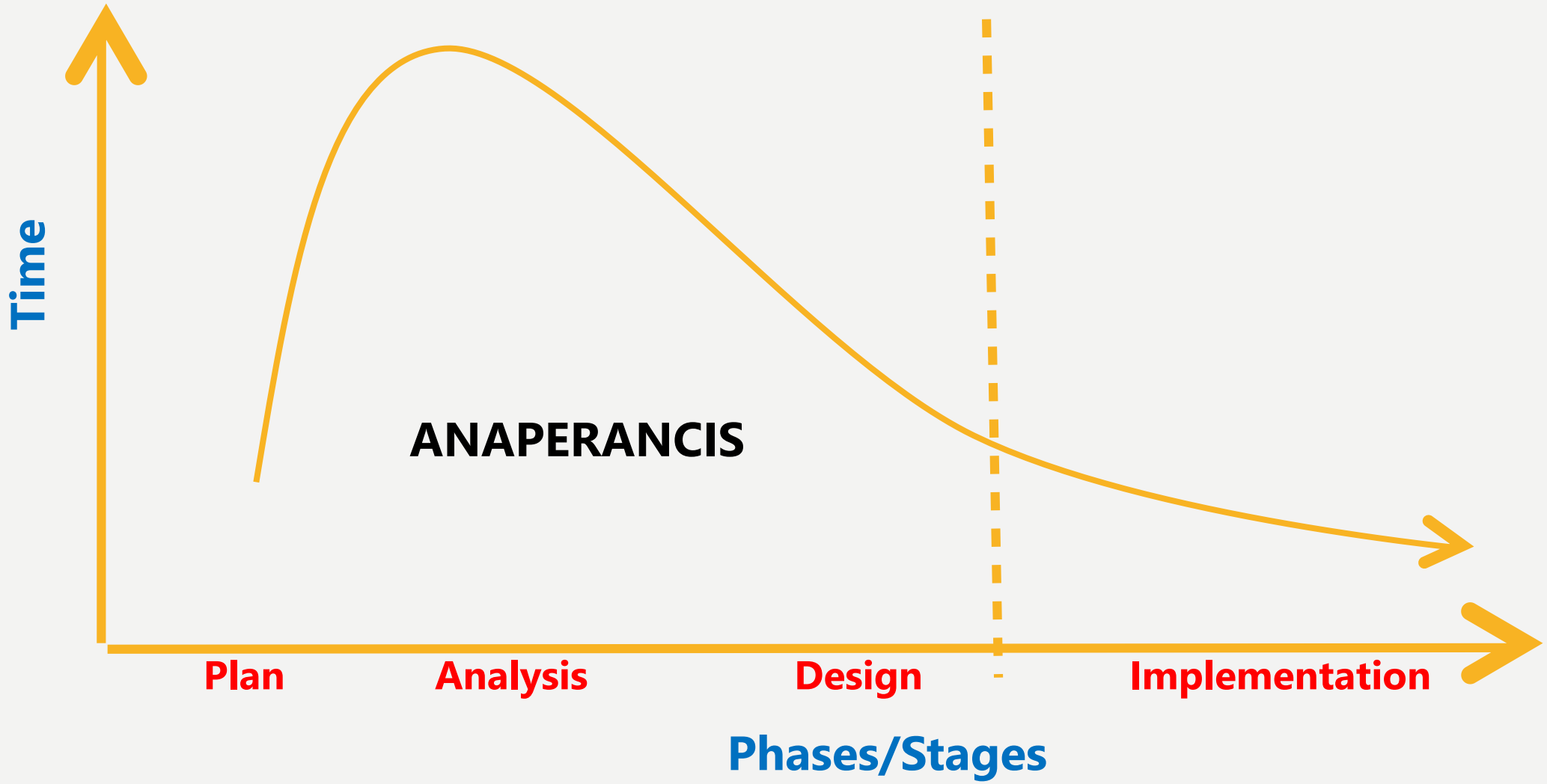
#4 Implementation

- 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.

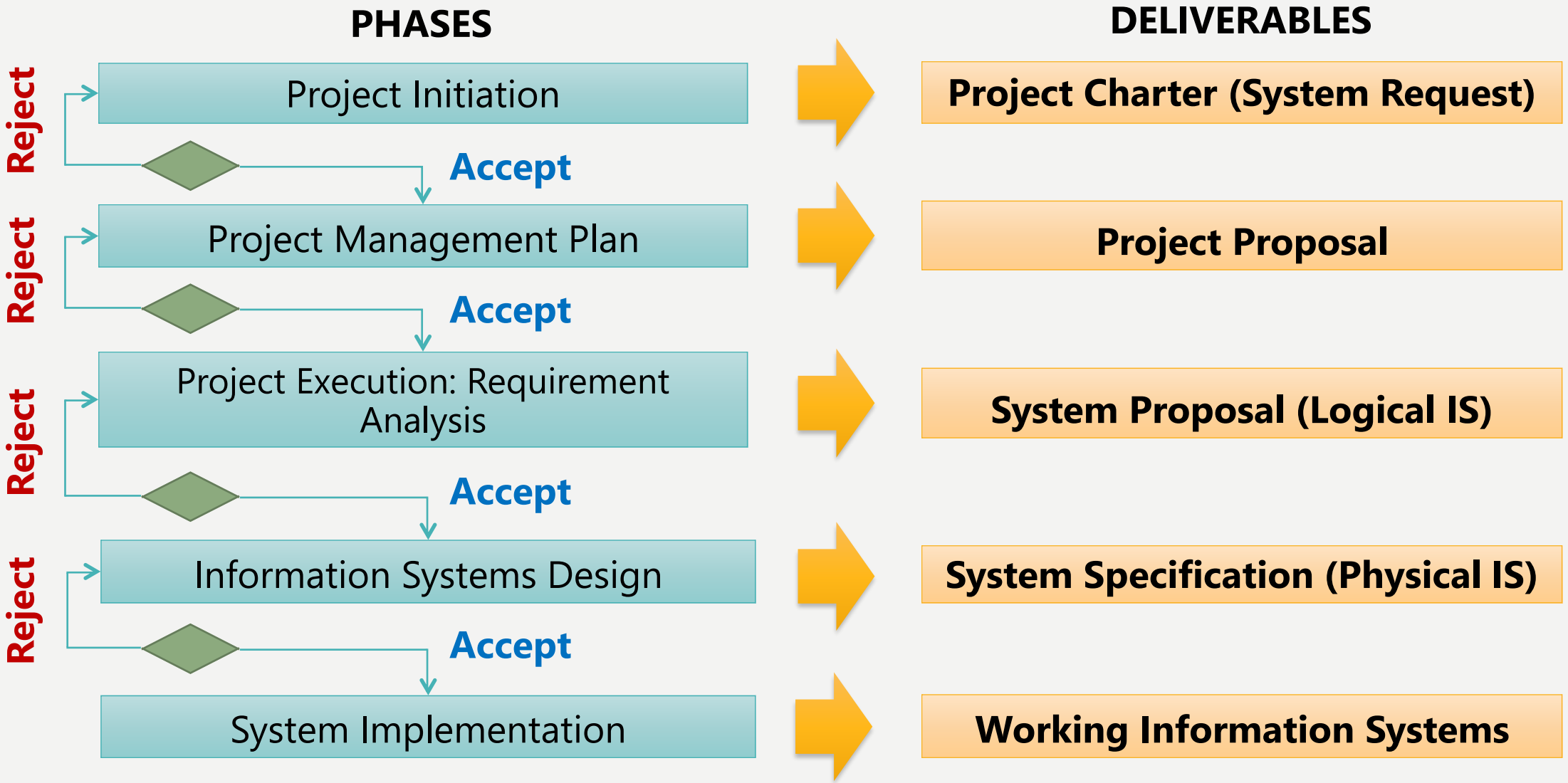
Processes and Deliverables



SDLC Effort Allocation



Basic Information Systems Development Project Life Cycle



1.3 IS DEVELOPMENT METHODOLOGY

What is methodology?

- A formalized approach or series of steps
- A methodology is a formalized approach to implementing the SDLC.
  - The methodology will vary depending on whether the emphasis is on businesses processes or on the data that supports the business.

Writing code without a well-thought-out system request may work for small programs, but rarely works for large ones.

SDLC methodology categorization

- Methodology that focuses on business process or data that support the business
  - Process-centered methodology
  - Data methodology
  - Object-oriented methodology
- Methodology that focuses on the sequencing of SDLC phases and the amount of time and effort.
  - Structured Development
  - Rapid Application Development
  - Agile Development

Process-Centered Methodology

- With this methodology, the focus is on defining the **activities** associated with the system.
- The concentration is on representing the system concept as a **set of processes with information flowing into and out of the processes.**

Data-Centered Methodology

- This methodology focuses on defining the **content of the data** storage containers and how they are organized.
- Data-centered methodologies utilize **data models as the core of the system concept.**

Object Oriented Methodology

- Attempts to balance emphasis on data and process
- Uses Unified Modeling Language (UML) for diagramming

Compare and contrast Structured Development, Rapid Application Development, and Agile Development methodology

Type of Systems Development Methodology

- Structured Design
  - Waterfall
  - Parallel
- Rapid Application Development (RAD)
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum

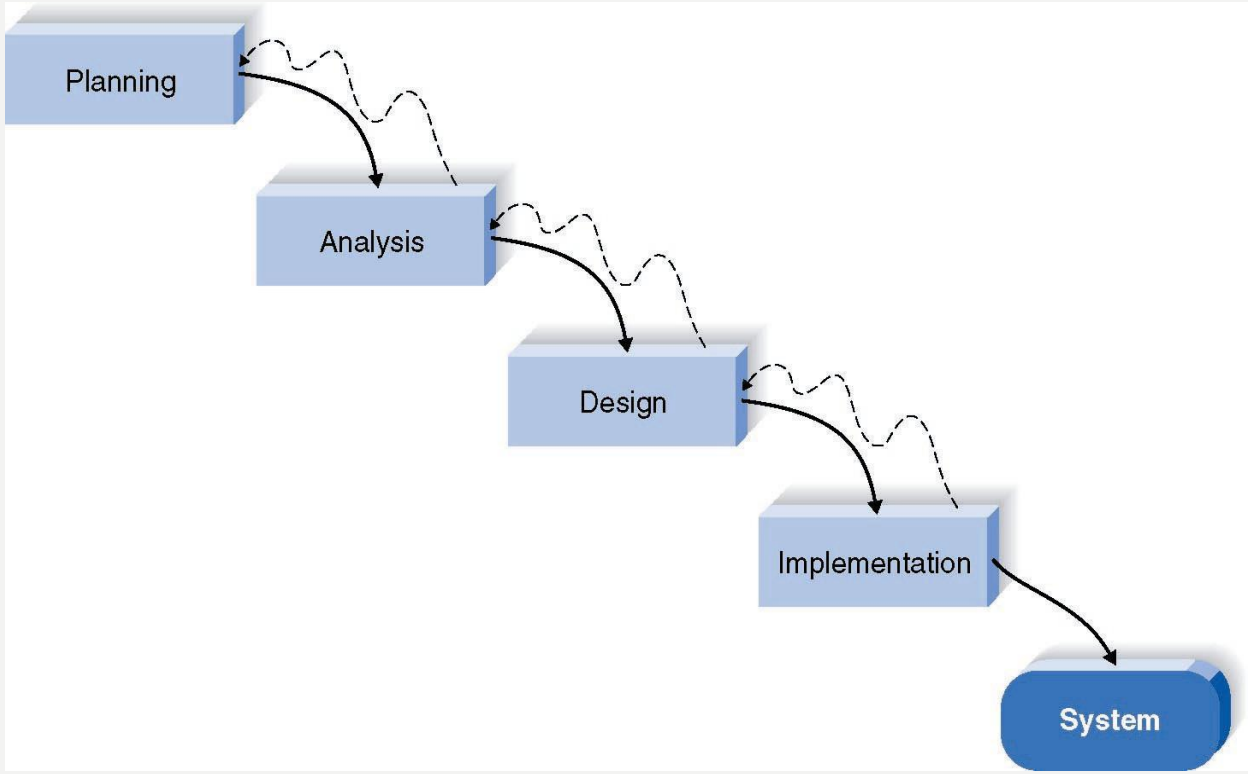
Structured Design

- **Structured Design**
  - Waterfall
  - Parallel
- Rapid Application Development (RAD)
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum

- *Projects move methodically from one to the next step*
- *Generally, a step is finished before the next one begins*
- This design methodology introduces the use of formal modeling or diagramming techniques to describe a system's basic business processes and follows a basic approach of two structured design categories.

Structured Design: Waterfall

- **Structured Design**
  - Waterfall
  - Parallel
- Rapid Application Development (RAD)
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum



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

Structured Design: Waterfall

- **Structured Design**
  - Waterfall
  - Parallel
- Rapid Application Development (RAD)
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum

- **Advantages**:
  - The system requirements are identified long before programming begins.
  - Changes to the requirements are minimized as the project proceeds.
- **Disadvantages**:
  - The length of key deliverables
  - Highest time gap between analysis and the delivery of the system
  - High post implementation effort for a missed requirement
  - Significant rework as the consequences of business environment changes

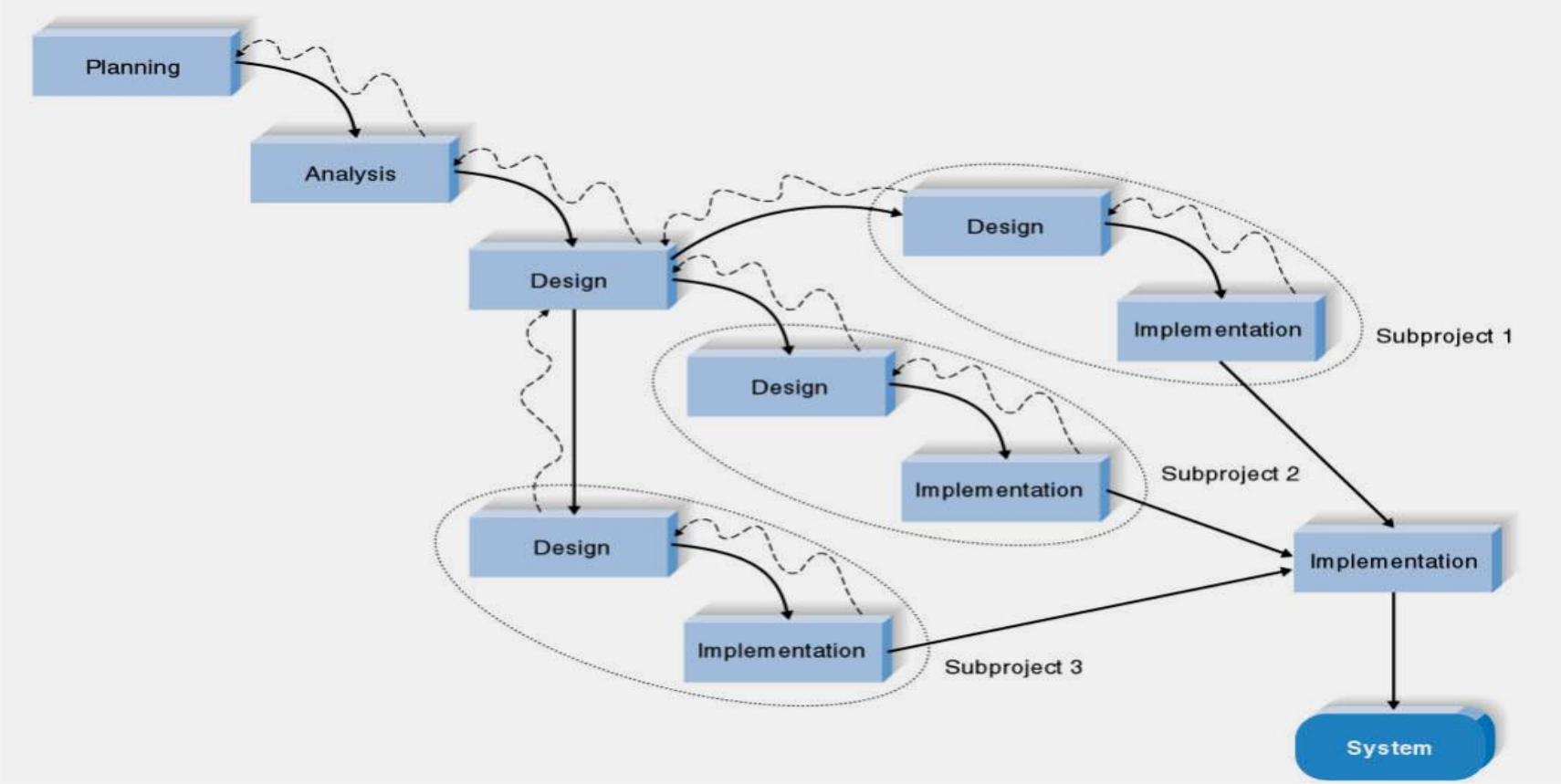
Structured Design: Parallel

- **Structured Design**
  - Waterfall
  - Parallel
- Rapid Application Development (RAD)
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum

- This methodology attempts to address the long time interval between the analysis phase and the delivery of the system
- A general design for the entire system is performed and then the project is divided into a series of distinct subprojects.
- **Advantage**:
  - Reduce time delivery
  - Less changes in the business environment causing rework
- **Disadvantage**:
  - Interdependency of subprojects → significant integration efforts

Structured Design: Parallel

- **Structured Design**
  - Waterfall
  - Parallel
- Rapid Application Development (RAD)
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum



Rapid Application Development (RAD)

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum

- 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.
  - CASE tools
  - JAD sessions
  - Fourth generation/visualization programming languages
  - Code generators

Rapid Application Development (RAD)

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum

- One possible subtle problem with RAD-based methodologies is managing user expectations.
- RAD Categories:
  - Phased development
    - A series of versions
  - Prototyping
    - System prototyping
  - Throw-away prototyping
    - Design prototyping

RAD: Phased Development

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum

- 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.

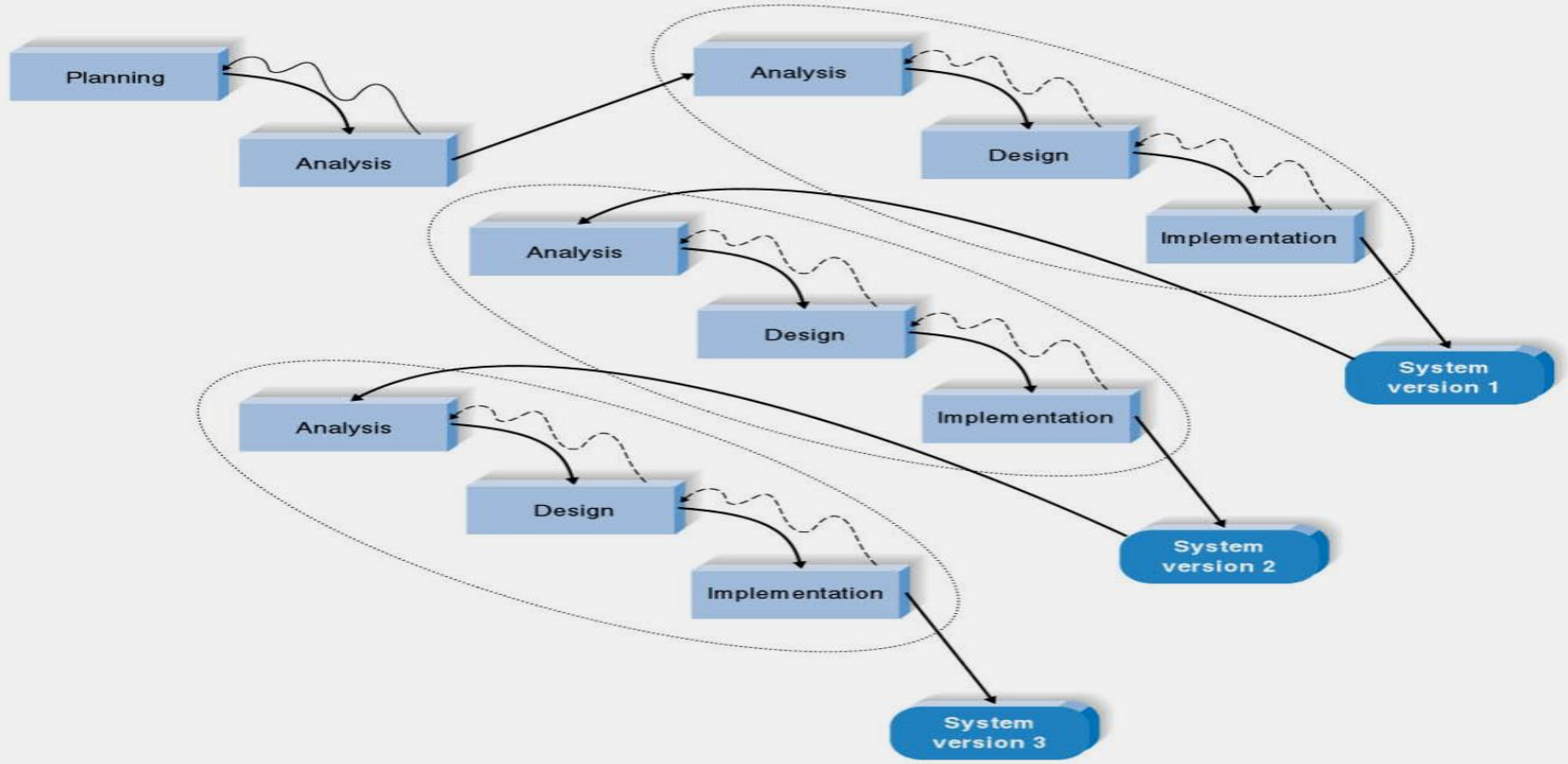
RAD: Phased Development

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum

- **Advantage**:
  - The users quickly use the system, although it does not perform all the functions the users need
- **Disadvantage**:
  - The users begin to work with systems that intentionally incomplete → critical to define the priority of user requirements

RAD: Phased Development

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum



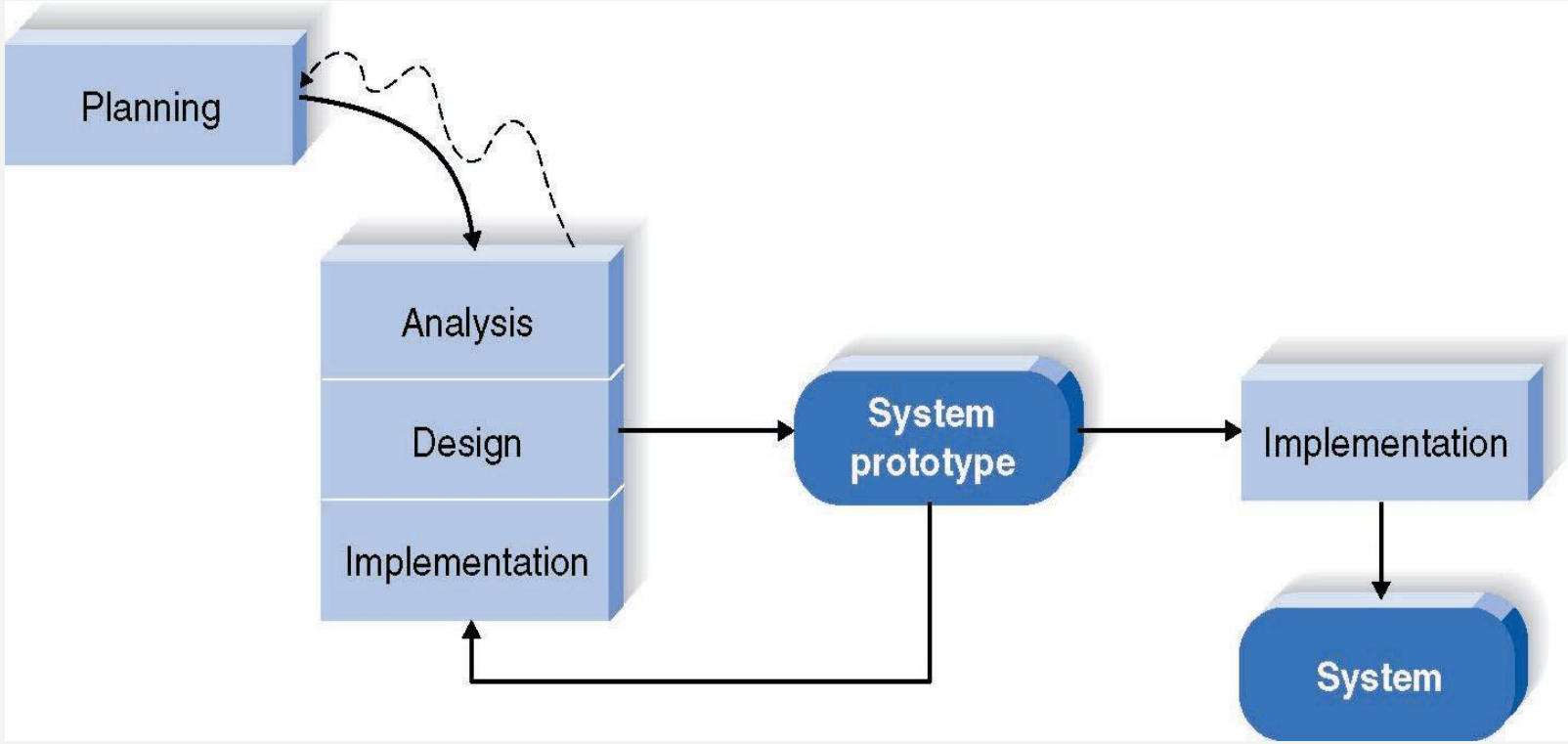
RAD: Prototyping

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum

- 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.
- Quick and dirty program → provides minimal amount of features
- Fits for users with unknown requirements

RAD: Prototyping

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum



RAD: Prototyping

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum

- **Advantage**
  - Quickly provide a system for the users to interact with, even if it is not initially ready for use.
  - Help to more quickly refine real requirements
- **Disadvantage**
  - Often the prototype undergoes such significant changes that many initial design decisions prove to be poor ones.

RAD: Throw-Away Prototyping

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum

- Throw-away prototyping methodologies are similar to prototyping based methodologies.
- The main difference is that **throwaway prototyping IS completed during a different point in the SDLC.**
- Has relatively thorough analysis phase.
- Design prototype is not working systems
- Fit for users who are not completely clear on how the systems work

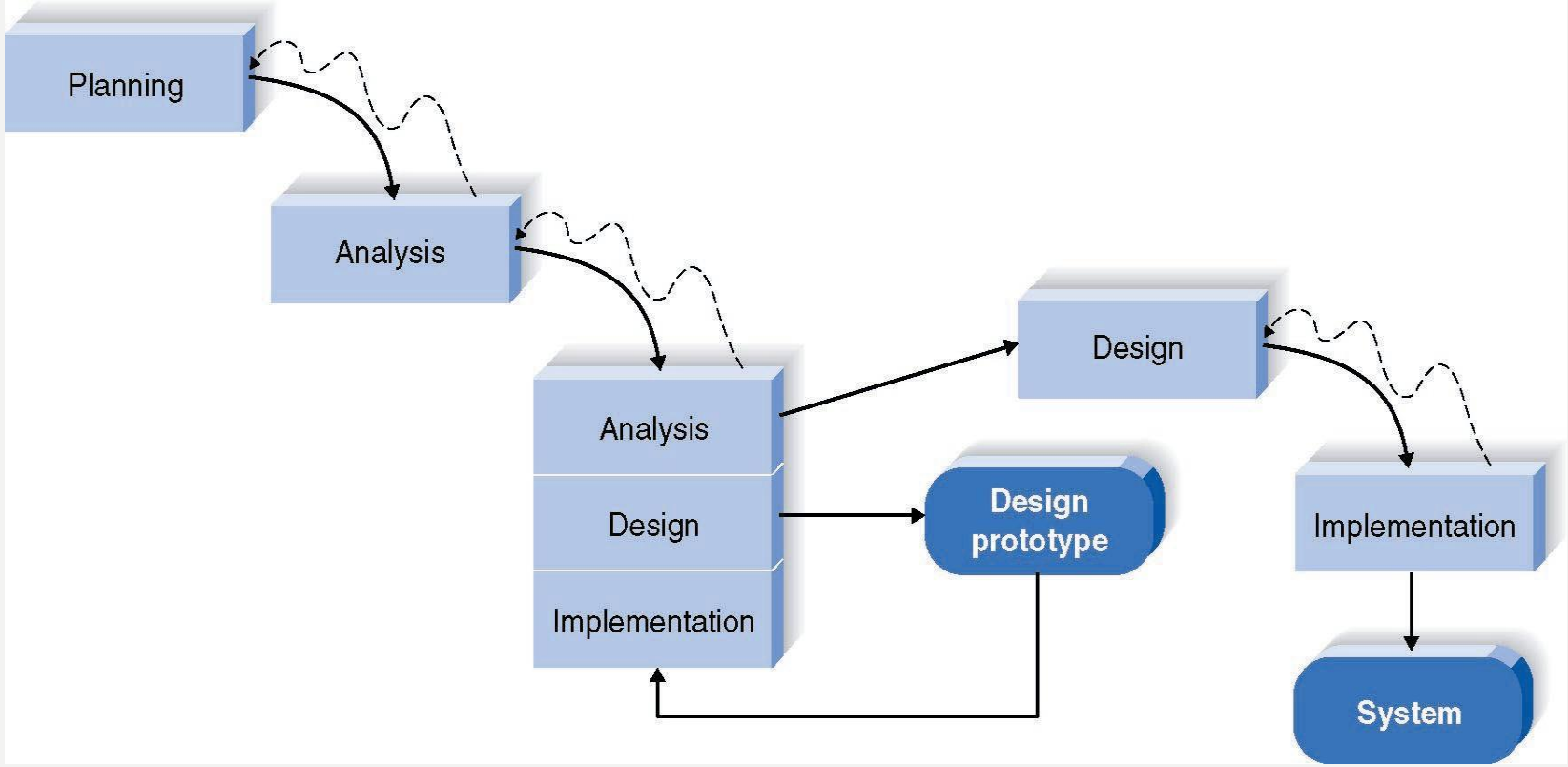
RAD: Throw-Away Prototyping

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum

- **Advantages**
  - Each prototype is used to minimize the risk associated with the system
  - Important issues are understood before the real system is built
  - Produce more stable and reliable systems
- **Disadvantages**:
  - Take longer delivery time compare to prototyping-based methodologies

RAD: Throw-Away Prototyping

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- Agile Development
  - Extreme Programming
  - Scrum



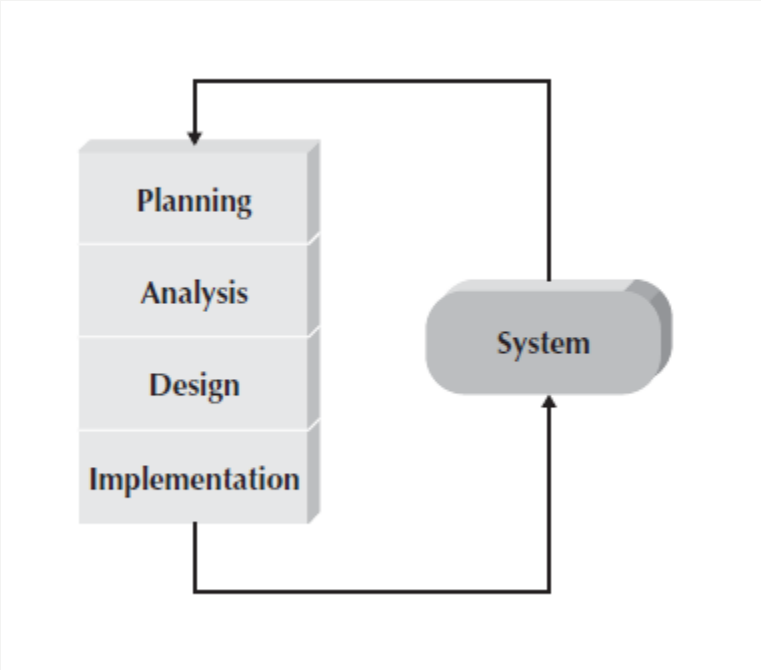
Agile Development

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- **Agile Development**
  - Extreme Programming
  - Scrum

- 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.

Agile Development

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- **Agile Development**
  - Extreme Programming
  - Scrum



Typical agile development methodology

Agile Development: Extreme Programming

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- **Agile Development**
  - Extreme Programming
  - Scrum

- Extreme Programming (XP) was founded on four core values:
  - Communication
  - Simplicity
  - Feedback
  - Courage
- All programming is done in pairs, a shared responsibility for each soft ware component develops among the programmers.
- Key principles of XP include:
  - Continuous testing
  - Simple coding
  - Close interaction with the end users to build systems very quickly

Agile Development: Scrum

- Structured Design
  - Waterfall
  - Parallel
- **Rapid Application Development (RAD)**
  - Phased Development
  - Prototyping
  - Throw-Away Prototyping
- **Agile Development**
  - Extreme Programming
  - Scrum

- Scrum is the most chaotic of all systems development approaches
- Teams are self-organized and self-directed.
- Unlike other approaches, Scrum teams do not have a designated team leader. Instead, teams organize themselves in a symbiotic manner and set their own goals for each sprint (iteration).
- The team members attend the scrum meetings, but anyone can attend.

Selecting Methodology

- Selecting a methodology is not simple, as no one methodology is always best.
- Many organizations have their own standards.

What is the best methodology in SDLC?

What are the criterias for choosing the appropriate methodology?

Selecting Methodology

1. Clarity of User Requirements
2. Familiarity with Technology
3. System Complexity
4. System Reliability
5. Short Time Schedules
6. Schedule Visibility

Selecting Methodology

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

Criterion 1: Clarity of User Requirements

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

Criterion 2: 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.

Criterion 3: 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.

Criterion 4: System Reliability

- System reliability is usually an important factor in system development.
- Throwaway prototyping-based methodologies are most appropriate when system reliability is a high priority.
- Prototyping-based methodologies are generally not a good choice as they lack careful analysis and design phases.

Criterion 5: 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.

Criterion 6: 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.

1.4 BUILDING TEAM IN IS DEVELOPMENT PROJECT

Project Team Skills and Roles

- Projects should consist of a variety of skilled individuals in order for a system to be successful.
- Six major skill sets an analyst should have include:
  - Technical
  - Business
  - Analytical
  - Interpersonal
  - Management
  - Ethical

Categories of Analysts

- Business analyst
- System analyst
- Infrastructure analyst
- Change management analyst
- Project manager

Project Team Roles

Role	Responsibilities
Business analyst	Analyzing the key business aspects of the system Identifying how the system will provide business value Designing the new business processes and policies
Systems analyst	Identifying how technology can improve business processes Designing the new business processes Designing the information system Ensuring that the system conforms to information systems standards
Infrastructure analyst	Ensuring the system conforms to infrastructure standards Identifying infrastructure changes needed to support the system
Change management analyst	Developing and executing a change management plan Developing and executing a user training plan
Project manager	Managing the team of analysts, programmers, technical writers, and other specialists Developing and monitoring the project plan Assigning resources Serving as the primary point of contact for the project

1.5 BASIC CHARACTERISTICS OF O-O SYSTEMS

Characteristics of OO Systems

Object-oriented systems focus on capturing the structure and behavior of information systems in little modules that encompass both data and process.

Familiar Terms

1. Classes and Objects
2. Methods and Messages
3. Encapsulation and Information Hiding
4. Inheritance
5. Polymorphism and Dynamic Binding

Characteristics of OO Systems

1. **Classes and Objects**
   - A class is the general template we use to define and create specific instances, or objects.
   - An object is an instantiation of a class.
   - Each object has attributes that describe information about the object
   - Each object also has behaviors, that specify what the object can do.
2. **Methods and Messages**
   - Methods implement an object's behavior.
   - A method is nothing more than an action that an object can perform.
   - Messages are information sent to objects to trigger methods.
   - A message is essentially a function or procedure call from one object to another object.

Characteristics of OO Systems

3. **Encapsulation and Information Hiding**
   - Encapsulation is simply the combination of process and data into a single entity.
   - The principle of information hiding suggests that only the information required to use a software module be published to the user of the module.
   - We really do not care how the object performs its functions, as long as the functions occur.
   - In object-oriented systems, combining encapsulation with the information-hiding principle supports treating objects as black boxes.

Characteristics of OO Systems

4. **Inheritance**
   - using inheritance to identify higher-level, or more general, classes of objects.
5. **Polymorphism and Dynamic Binding**
   - Polymorphism means that the same message can be interpreted differently by different classes of objects.
   - Polymorphism is made possible through dynamic binding. Dynamic, or late, binding is a technique that delays typing the object until run-time. The specific method that is actually called is not chosen by the object-oriented system until the system is running.

Characteristics of OO Systems Analysis and Design

The primary difference between a traditional approach like structured design and an object-oriented approach is **how a problem is decomposed.**

- In traditional approaches, the problem-decomposition process is either process-centric or data-centric.
- However, processes and data are so closely related that it is difficult to pick one or the other as the primary focus.
- Based on this lack of congruence with the real world, new object-oriented methodologies have emerged that use the RAD-based sequence of SDLC phases but attempt to balance the emphasis between process and data by focusing the decomposition of problems on objects that contain both data and processes.

Characteristics of OO Systems Analysis and Design

1. **Use-Case Driven**
   - Use cases are the primary modeling tools defining the behavior of the system.
   - A use case describes how the user interacts with the system to perform some activity, such as placing an order, making a reservation, or searching for information.
   - The use cases are used to identify and to communicate the requirements for the system to the programmers who must write the system.
   - Use cases are inherently simple because they focus on only one business process at a time.

Characteristics of OO Systems Analysis and Design

1. **Use-Case Driven**
   - In contrast, the process model diagrams used by traditional structured and RAD methodologies are far more complex because they require the systems analyst and user to develop models of the entire system.
   - With traditional methodologies, each system is decomposed into a set of subsystems, which are, in turn, decomposed into further subsystems, and so on.
   - This goes on until no further process decomposition makes sense, and it often requires dozens of pages of interlocking diagrams. In contrast, a use case focuses on only one business process at a time, so developing models is much simpler.

Characteristics of OO Systems Analysis and Design

2. **Architecture-Centric**
   - Any modern approach to systems analysis and design should be architecture-centric.
   - Architecture-centric means that the underlying software architecture of the evolving system specification drives the specification, construction, and documentation of the system.

Characteristics of OO Systems Analysis and Design

2. Architecture-Centric

- Modern object-oriented systems analysis and design approaches should support at least three separate but interrelated architectural views of a system: functional, static, and dynamic.
  - **a)** **The functional, or external, view** describes the behavior of the system from the perspective of the user.
  - **b)** **The structural, or static, view** describes the system in terms of attributes, methods, classes, and relationships.
  - **c)** **The behavioral, or dynamic, view** describes the behavior of the system in terms of messages passed among objects and state changes within an object.

Characteristics of OO Systems Analysis and Design

3. Iterative and Incremental

- Modern object-oriented systems analysis and design approaches emphasize iterative and incremental development that undergoes continuous testing and refinement throughout the life of the project.
- This implies that the systems analysts develop their understanding of a user's problem by building up the three architectural views little by little.
- The systems analyst does this by working with the user to create a functional representation of the system under study.
- Next, the analyst attempts to build a structural representation of the evolving system.

Characteristics of OO Systems Analysis and Design

3. Iterative and Incremental

- Using the structural representation of the system, the analyst distributes the functionality of the system over the evolving structure to create a behavioral representation of the evolving system.
- As an analyst works with the user in developing the three architectural views of the evolving system, the analyst iterates over each of and among the views.
- That is, as the analyst better understands the structural and behavioral views, the analyst uncovers missing requirements or misrepresentations in the functional view. This, in turn, can cause changes to be cascaded back through the structural and behavioral views.

1.6
UML

Characteristics of OO Systems Analysis and Design

- Until 1995, object concepts were popular but implemented in many different ways by different developers. Each developer had his or her own methodology and notation (e.g., Booch, Coad, Moses, OMT, OOSE, SOMA).23 Then in 1995, Rational Software brought three industry leaders together to create a single approach to object-oriented systems development.
- **The objective of UML** was to provide a common vocabulary of object-oriented terms and diagramming techniques rich enough to model any systems development project from analysis through implementation.

UML Diagram

1. Structure diagrams

- provide a way to represent the data and static relationships in an information system.
- the structure diagrams include class, object, package, deployment, component, composite structure, and profile diagrams.

2. Behavior diagrams

- provide the analyst with a way to depict the dynamic relationships among the instances or objects that represent the business information system.
- they also allow modeling of the dynamic behavior of individual objects throughout their lifetime.
- the behavior diagrams support the analyst in modeling the functional requirements of an evolving information system.
- the behavior modeling diagrams include activity, sequence, communication, interaction overview, timing, behavior state machine, protocol state machine, and use-case diagrams.

UML Diagram

Diagram Name	Used to...	Primary Phase
Structure Diagrams		
Class	Illustrate the relationships between classes modeled in the system	Analysis, Design
Object	Illustrate the relationships between objects modeled in the system; used when actual instances of the classes will better communicate the model	Analysis, Design
Package	Group other UML elements together to form higher-level constructs	Analysis, Design, Implementation
Deployment	Show the physical architecture of the system; can also be used to show software components being deployed onto the physical architecture	Physical Design, Implementation
Component	Illustrate the physical relationships among the software components	Physical Design, Implementation
Composite Structure Design	Illustrate the internal structure of a class, i.e., the relationships among the parts of a class	Analysis, Design
Profile	Used to develop extensions to the UML itself	None
Behavioral Diagrams		
Activity	Illustrate business workflows independent of classes, the flow of activities in a use case, or detailed design of a method	Analysis, Design
Sequence	Model the behavior of objects within a use case; focuses on the time-based ordering of an activity	Analysis, Design
Communication	Model the behavior of objects within a use case; focus on the communication among a set of collaborating objects of an activity	Analysis, Design
Interaction Overview	Illustrate an overview of the flow of control of a process	Analysis, Design
Timing	Illustrate the interaction among a set of objects and the state changes they go through along a time axis	Analysis, Design
Behavioral State Machine	Examine the behavior of one class	Analysis, Design
Protocol State Machine	Illustrate the dependencies among the different interfaces of a class	Analysis, Design
Use-Case	Capture business requirements for the system and illustrate the interaction between the system and its environment	Analysis

SUMMARY

Summary

- Basic type of computer-based information systems consist of **Transaction Processing System, Management Information System, and Strategic Information System**, which process data and information for different type of users.
- The development of information systems have to consider the **technological development**, such as cloud computing, big data, and also mobile technology.
- The primarily goal of information system is to **create value for the organization.**
- SDLC helps team member to understand **how an information system (IS) can support and enable business needs** by designing a system, building it, and delivering it to users.

Summary

- SDLC consists of **4-fundamental phases** (Planning, Analysis, Design and Implementation), which each phase refines and elaborates on the work done previously).
- The systems analyst is a key person analyzing the business, **identifying opportunities for improvement**, and designing information systems to implement these ideas.
- SDLC methodology that focuses on business process or data that support the business consists of **process-centered methodology, data methodology, and object-oriented methodology**
- SDLC methodology that focuses on the sequencing of SDLC phases and the amount of time and effort consists of **Structured Development, Rapid Application Development, and Agile Development**

Summary

- Criterias for selecting SDLC methodology: **clarity of user requirements, familiarity with technology, system complexity, system reliability, short time schedules, schedule visibility**

References

1. Systems Analysis and Design: An Object Oriented Approach with UML 5th ed. Alan Dennis, Barbara Haley Wixom, and Roberta M. Roth © 2015 – Chapter 1
2. http://emojipedia.org/whatsapp/
3. For complete description of UML see:
4. www.rational.com/uml