# IS 525 – Chapter 2

## Methodology

**Dr. Nesrine Zemirli**

Assistant Professor.

IS Department – CCIS / King Saud University
E-mail: nzemirli@ksu.edu.sa
Web: http://fac.ksu.edu.sa/nzemirli/home

# Chapter Topics

- Fundamental concepts and building blocks of methodology.
- Benefits and risks of methodology.
- Software development methodologies
- Modeling concepts and software development.
- Project management concepts and tools.

# Methodology

- Methodology is:
  - a set of methods, rules, practices, procedures, techniques and tools used to achieve a goal, or
  - the theoretical understating of the principles that determine how such methods, practices, tools, etc., are used.

# Methodology

## The "How" of Building Solutions

**Methods**

A methodology combines methods & techniques into a whole.

**Tools**

A methodology may require devices that help in accomplishing tasks.

**Procedures**

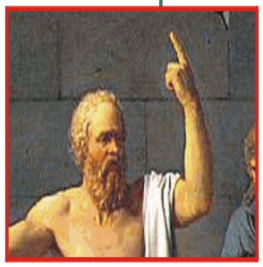
A methodology defines the path to the goal in a systematic manner.

**Rules**

Rules of methodology are abstractions of past experience or new ideas.

**Methodology**

A methodology may include other methodologies for various aspects of development.



**"Philosophy"**

A theoretical framework distinguishes a methodology from its components.

# Methods Versus Methodology

- While "method" defines a **tactic**, "methodology" defines the **strategy**.

- Often, a tactic makes sense only in the context of strategy.

# What Guides Our Actions?

- Cookbook
  - An ordered set of steps.
- Observation
  - We can learn something by observing others do it.
- Anecdotes
  - We might hear or read stories about how somebody has done something.
- Trial and Error
  - Lacking any reliable guidelines, we *imagine* some method that would work, and try it. If it does not work, we try another approach, if we can.
- Experience
  - We rely on the experience. If the results are not satisfactory, then we adjust the "experience."
- Patterns
  - Patterns result from *collective* experience.

Methodology is both the most abstract and the most systematic guide to action. It evolves from the above but cannot be reduced to any of them.

# Challenges of Methodology

- Methodology is needed not only in creating the **solution** but also in understanding the **problem**, organizing the production, assuring quality and the managing the consequences of the solution.
- As creators, we have to confront *three* elements: the **problem**, the **solution as method** or **methodology** (or "how"), and the **solution as answer** (or "what").

# The Problem Domain

- The problem that is to be solved, or the need that is to be satisfied exists in a **context**.

- This context is called the "**problem domain**" (or the "problem *space*")
  - For example, a **headache** is a **symptom**, and needs to be treated within a context of the real problem.

# The Solution Domain

- In building a solution, we also create a new and distinct **context** called the "solution domain" (or the "solution *space*").
  - For example, most cars have fans to cool down the engine, even though the heat produced by the engine has no connection to transportation for which the car is built.

# Methodologies in Parallel

- The process of building a solution may require more than one methodology to succeed.

- A methodology can be **hierarchical**, meaning that it is composed of **sub-methodologies** which, in turn, might consist of other methodologies.

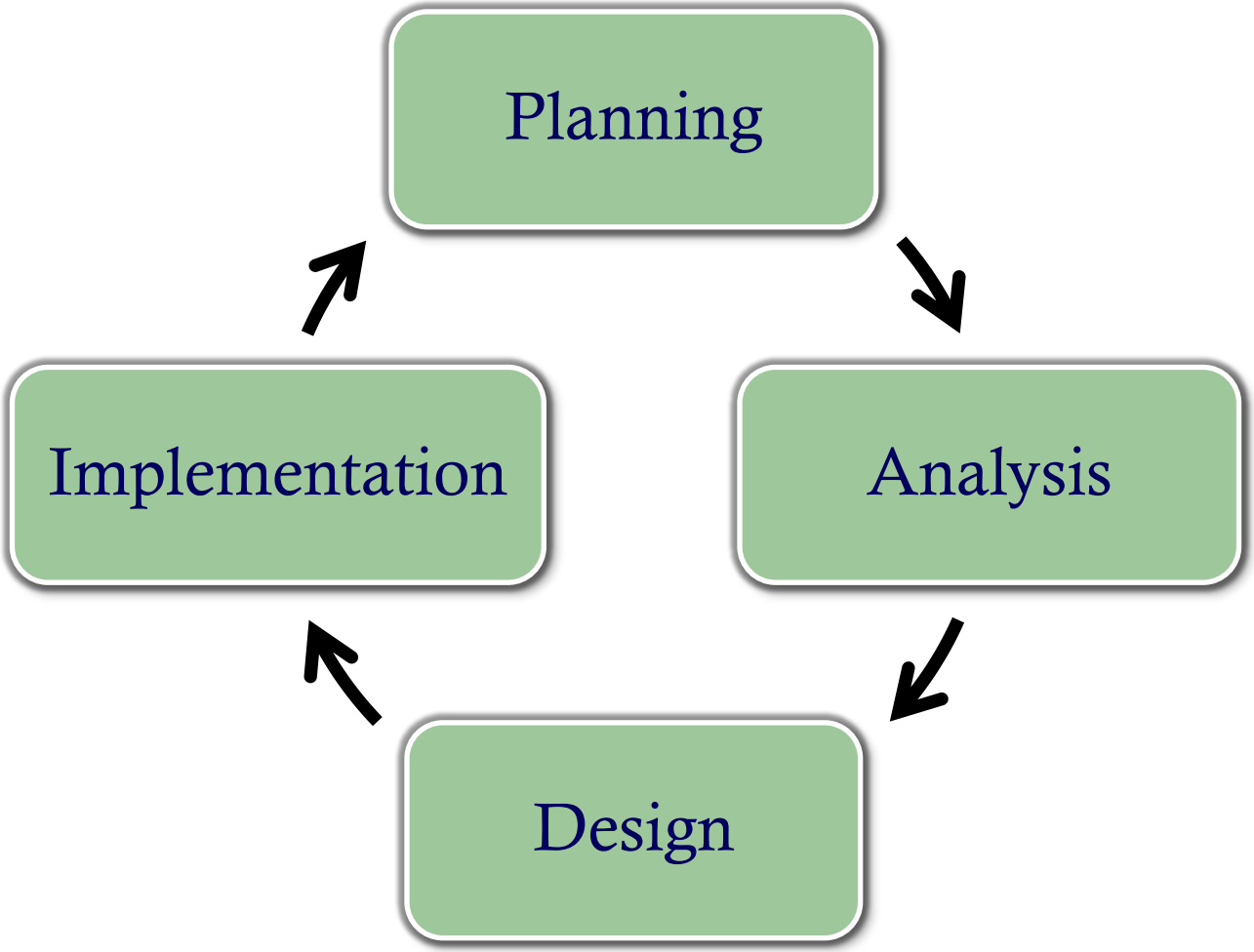
# The *Ad Hoc* Approach

- The **ad hoc** approach is development without an overall theoretical framework
  - To succeed, the ad hoc approach must rely overwhelmingly on:
    - the ingenuity of participants to improvise solutions for unforeseen problems
    - the ability of the participants to coordinate and communicate with each other, and
    - what can be conveniently described as "luck", meaning that the right people hit the right targets under the right circumstances.
- As should be expected, **ad hoc** is a high-risk approach.

# System Development Life Cycle (SDLC)

- SDLC methodologies view software development as primarily a project management process rather than a technical one.
- Each phase is a "**milestone**" and the resulting documents are the "deliverables."

# Systems Development Life Cycle



# SDLC: Planning

1. Project Initiation
   - Develop a system request
   - Conduct a feasibility analysis
2. Project Management
   - Develop work plan
   - Staff the project
   - Control and direct the project

***Why should we build this system?***

# SDLC: Analysis

1. Develop analysis strategy
2. Gather requirements
3. Develop a system proposal

***What should the system do for us?***
***Where and when will it be used?***

# SDLC: Design

1. Develop a design strategy
2. Design architecture and interfaces
3. Develop databases and file specifications
4. Develop the program design

***How will we build the system?***

# SDLC: Implementation

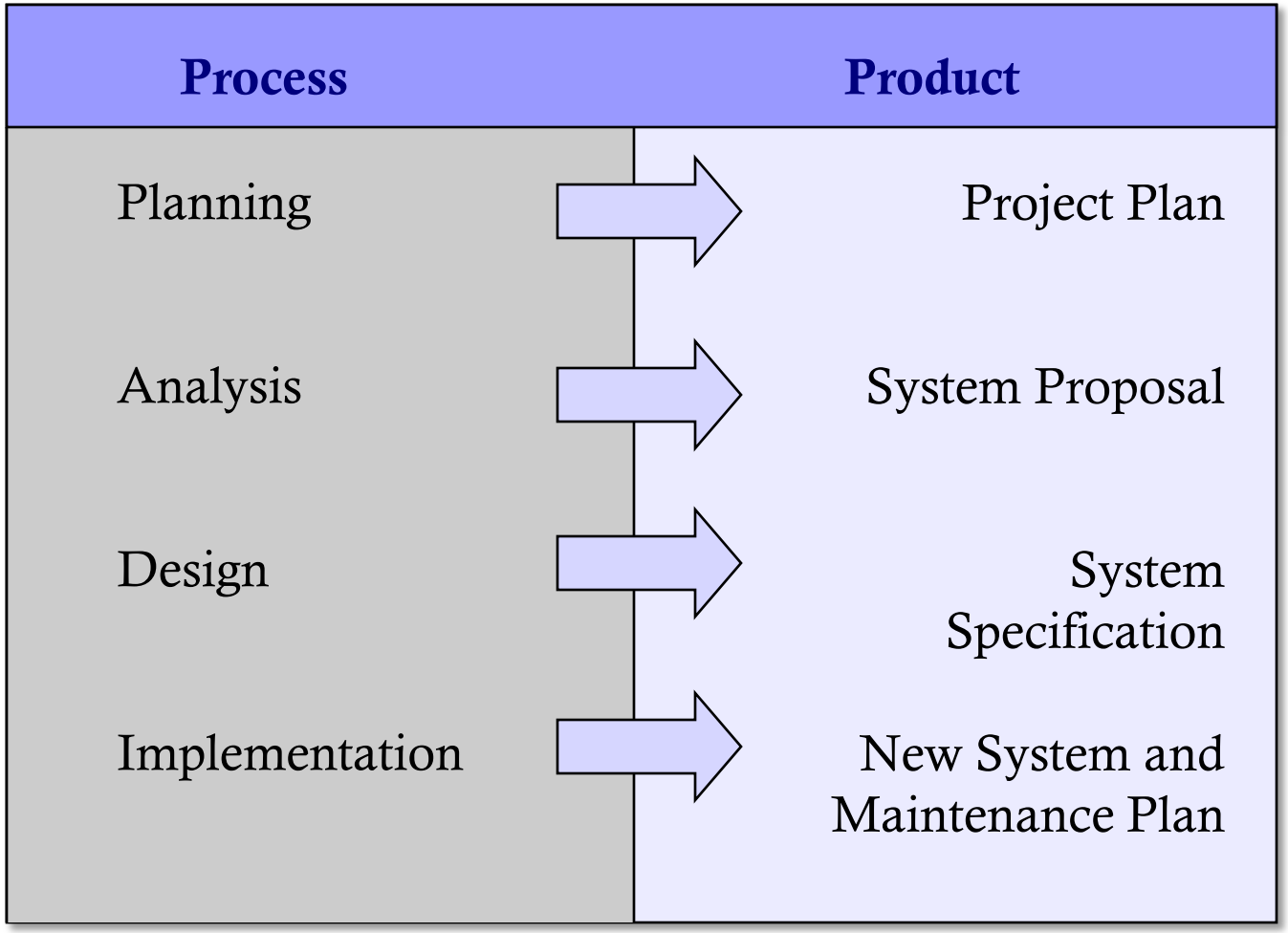
1. Construct system
2. Install system
   - Implement a training plan for the users
3. Establish a support plan

***Build* the system!**

# Putting the SDLC Together

- Each phase consists of steps that lead to specific deliverables
- The system evolves through gradual refinement
- Once the system is implemented, it may go back into a planning phase for its next revision, a follow-on system, or maintenance releases

# Processes and Deliverables



# Software Development Methodologies

- Software development consists of a wide spectrum of activities that individual methodologies cover selectively and from different viewpoints.
- A ***methodology*** is a formalized approach to implementing the SDLC
- Well-known methodologies include:
  - Waterfall development
  - Parallel development
  - Rapid application development
  - Agile development

# Categories of Methodologies

- Structured Design
  - Waterfall Development
  - Parallel Development
- Rapid Application Development
  - Phased
  - Prototyping
  - Throwaway Prototyping
- Agile Development
  - eXtreme Programming

# Software Development Methodologies Address:

- Requirements Gathering
  - This activity determines the requirements that the product must address.
- Feasibility Study
  - Determines whether it is possible — technically, economically, legally or organizationally — to build a certain software.
- Domain Analysis
  - Discovers ❶ the meaning of requirements within the context, ❷ concepts within the domain that are related to the problem and can affect the solution, and possibly ❸ the consequences of the solution on the problem domain.
- Analysis
  - Analyzing the requirements to build a **conceptual** model of the solution (the product).
- Design
  - Transforms the "what" into "how." Design itself consists of several distinct activities; **logical design, physical design**, and **architectural design**.

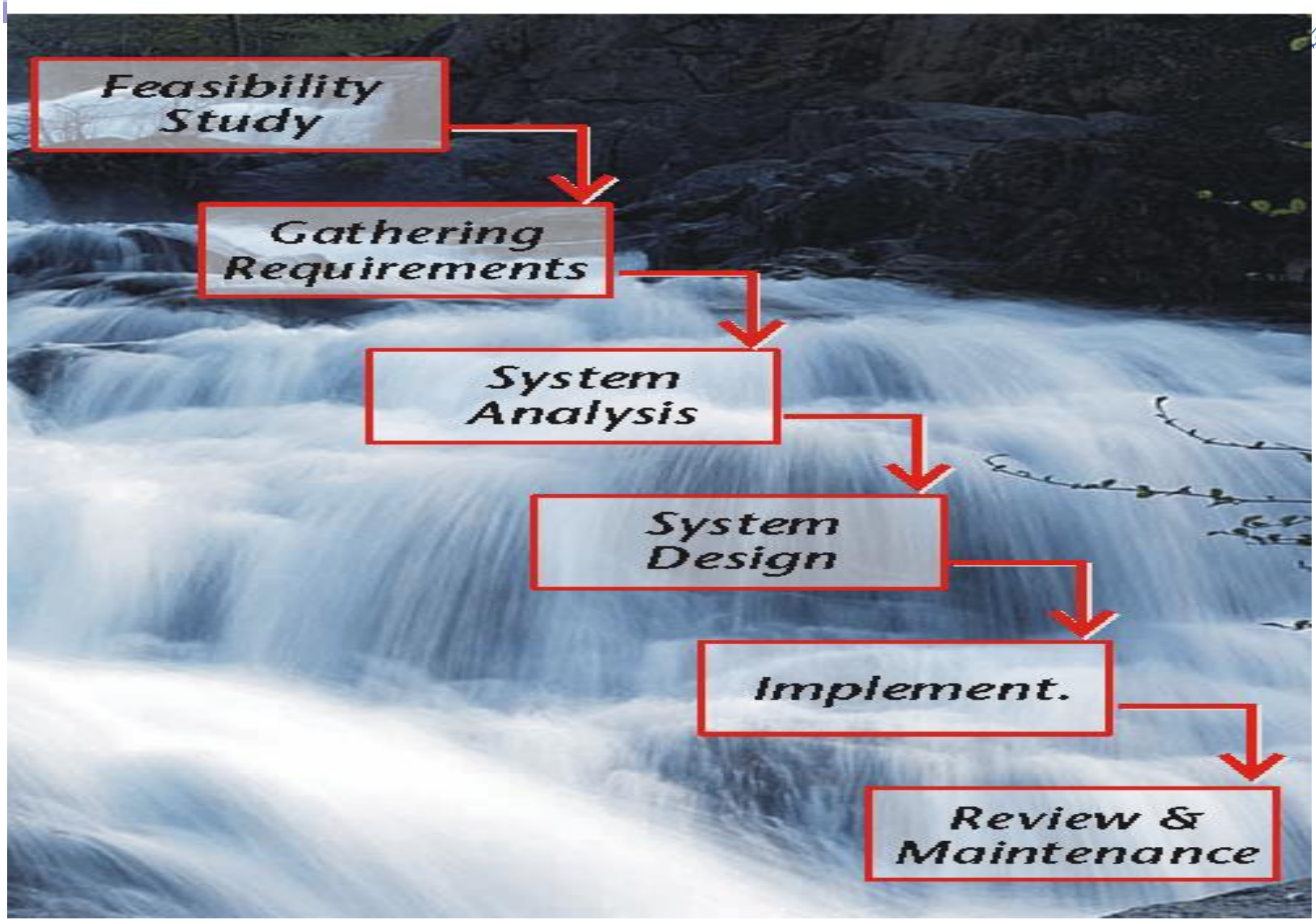
# Software Development Methodologies Address:

- Implementation
  - Turns the blueprints of design into an actual product. Programming is usually the most important component of this activity, but it is not the only one.
- Testing and Quality Control
  - Verifies that the product functions according to specifications.
- Deployment and Training
  - This activity consists of ensuring the correct installation on the target platform, user training, creating help files and user manuals, setting up of Web sites to guide users, packaging, et cetera.
- Maintenance
  - Solving problems that may emerge after the deployment of the software, or changes in the environment.

# The Waterfall Model

- The waterfall model specifies a set of sequential phases for software development
  - Each step cannot begin until the previous step has been completed *and* documented.
  - It is document-driven.
  - Design is conceived as **processes and data**, not as objects.

# The Waterfall Model



# The Limits of Waterfall Model

- Inflexibility
  - Cannot easily swim upstream
- Over-Reliance on Documentation
  - Relies too much on documentation
- Detachment from Technology
  - “One size” methodology cannot fit all technologies.
- Detachment from Marketplace
  - Slow-paced methodology.
- Detachment from the Profession
  - Programming is not the same as assembly of cars or baking breads, nor programmers work the same way as manufacturing workers or bakers.

# Rapid Application Development (RAD)

- Rapid Application Development is selecting techniques, methods, practices and procedures that would result faster development and shorter schedules.

# Characteristics of RAD

- Requirements Planning
  - Aims at eliciting information and requirements from the senior people and verifying the goals.
- Design
  - The design phase begins once the *top-level* requirements of the system are identified. To discover more detailed requirements, RAD relies on Joint Application Development (JAD) workshops.
- Implementation
  - Once the users approve the preliminary design, a detailed design of the system is created and code is generated. Implementation phase is heavily dependent upon CASE tools.
- Enhancements and Maintenance
  - In the framework of RAD, a software is never completed until it is retired.
  - there is no significant difference between development and maintenance. (This position is in opposition to most SDLC methodologies.)

# Prototyping

- Prototyping is the creation of a working model of the essential features of the final product for testing and verification of requirements.
- There are two types of prototyping: incremental or evolutionary and throwaway.

# Incremental and Iterative Approach

- In incremental development, the product is built through successive versions that are refined and expanded with each iteration
  - Three concepts underlie the incremental approach:
    - The Initialization Step
    - The Control List
    - The Iteration Step

# Throwaway Approach

- In the incremental approach,
  - the initial prototype is revised and refined repeatedly until it becomes the final product.
- In the throwaway approach,
  - the prototype is discarded after the stakeholders in the development are confident that they have arrived at the correct specifications and the development on the "real" product can start.

# Problems with Prototyping

- Unbalanced Architecture
  - Since the main thrust of prototyping is towards the user interface, the developers tend to include more and more functionality in the outer layers of the information system, creating a distorted architecture.
- The Illusion of Completeness
  - Since the clients might not understand why the developers insist that a lot more is to be done, the prototype may successfully present the user interface for a complex functionality the feasibility of which is far from certain.
- Diminishing Changeability
  - Since prototyping can leave little trace of how the development evolved, modifying the application can resemble an archeological undertaking to piece together a lost civilization.
- Prototyping can result in too little documentation or, more importantly, too little modeling.

# The Spiral Model

- The spiral model is a risk-oriented lifecycle model that breaks a software project up into mini-projects.

# Agile Methodologies

- Motivated by recognition of software development as fluid, unpredictable, and dynamic
- Three key principles
  - Adaptive rather than predictive
  - Emphasize people rather than roles
  - Self-adaptive processes

# When to use Agile Methodologies

- If your project involves:
  - Unpredictable or dynamic requirements
  - Responsible and motivated developers
  - Customers who understand the process and will get involved

# Extreme Programming (XP)

- A better-known example of agile methods, and one of the earliest ones, is Extreme Programming (XP).

- XP has a set of clear-cut practices that can be grouped in four categories:
  - Planning
  - Designing
  - Coding
  - Testing

# Extreme Programming (XP)

- **Planning**
  - user stories
  - release plan
  - pair programming
- **Designing**
  - CRC cards
  - Spike solutions
  - refactoring
- **Coding**
  - collective ownership
  - sequential integration
  - no overtime
- **Testing**
  - Unit testing
  - Acceptance testing
  - Integration testing

# The Capability Maturity Model (CMM)

- CMM aims to measure the maturity of software development *process* within an organization.

- CMM is essentially a **rating and auditing framework of standards**

# Maturity Levels of CMM

- Initial (or ad hoc).
  - *Focus: Individual Effort*
- Repeatable.
  - *Focus: Project Management*
- Defined.
  - *Focus: Engineering Process*
- Managed.
  - *Focus: Product & Process Quality*
- Optimizing.
  - *Focus: Continuous Process Improvement*

# Modeling

- Modeling, as a methodology, is the systematic representation of the relevant features of a product or a system from particular perspectives.

# Modeling for Software Development

- Software modeling is shaped by four interweaved factors:
  1. how the real world is seen,
  2. how software is defined,
  3. the process of development, and
  4. the modeling language.

# Object-Oriented Development

- A full object-oriented development requires three things:
  - ❶ an object-oriented technology,
  - ❷ an object-oriented analysis and design, and
  - ❸ a project plan adapted to an object-oriented approach

# Object-Oriented Development

- Object-oriented *technology* is mature and rules the market.
- Object-oriented software development is **iterative**

# Project Management Concepts

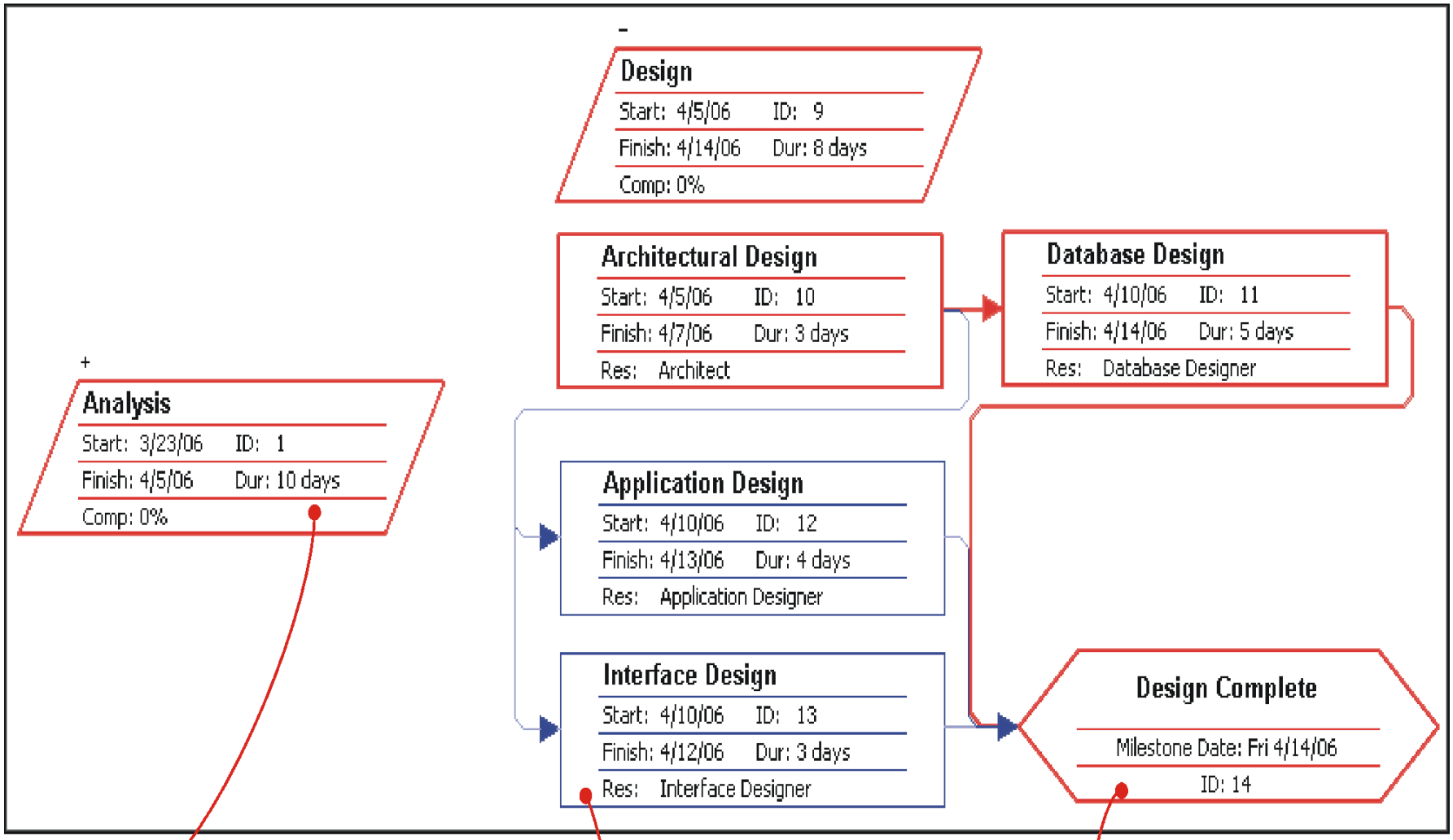
- A project is a collection of related tasks that must be completed in a particular order and within a certain timeframe to achieve a specific goal.

# Concepts Underlying Project

- The Goal
  - The goal of a project must be verifiable
- The Scope
  - The scope of the project defines the boundaries of the goal
- The Timeframe
  - A project's lifetime is finite
- The Lifecycle
  - Lifecycle identifies the phases of the project from its inception to its completion

# The Network Diagram

## The Visual Layout of The Project Workflow



# Tasks and Activity Threads

- A project is composed of related tasks that take place on one or more activity threads or paths.

# Scheduling Tasks

## Project's Plan of Action

| | Task Name | Duration | Start | Finish | Predecessors | Resource Names |
|---|---|---|---|---|---|---|
| 1 | **Analysis** | **10 days** | **Thu 3/23/06** | **Wed 4/5/06** | | |
| 2 | Organize Requirements | 2 days | Thu 3/23/06 | Fri 3/24/06 | | Business Analyst |
| 3 | Domain Analysis | 3 days | Tue 3/28/06 | Thu 3/30/06 | 2 | Business Analyst |
| 4 | Discover Use Cases | 3 days | Mon 3/27/06 | Wed 3/29/06 | 2 | Business Analyst,System Analyst |
| 5 | Elaborate Use Cases | 2 days | Thu 3/30/06 | Fri 3/31/06 | 4 | System Analyst,Business Analyst |
| 6 | Develop Structural Models | 3 days | Mon 4/3/06 | Wed 4/5/06 | 3,5 | System Analyst |
| 7 | Develop Dynamic Models | 2 days | Mon 4/3/06 | Tue 4/4/06 | 5 | System Analyst |
| 8 | Analysis Complete | 0 days | Tue 4/4/06 | Tue 4/4/06 | 7 | |
| 9 | **Design** | **8 days** | **Wed 4/5/06** | **Fri 4/14/06** | | |
| 10 | Architectural Design | 3 days | Wed 4/5/06 | Fri 4/7/06 | 8 | Architect |
| 11 | Database Design | 5 days | Mon 4/10/06 | Fri 4/14/06 | 10 | Database Designer |
| 12 | Application Design | 4 days | Mon 4/10/06 | Thu 4/13/06 | 10 | Application Designer |
| 13 | Interface Design | 3 days | Mon 4/10/06 | Wed 4/12/06 | 10 | Interface Designer |
| 14 | Design Complete | 0 days | Fri 4/14/06 | Fri 4/14/06 | 11,12,13 | |
| 15 | **Implementation** | **3 days** | **Mon 4/17/06** | **Wed 4/19/06** | | |

Summary Element

Terminal Element

Milestone

Dependencies

# Milestones and Deliverables

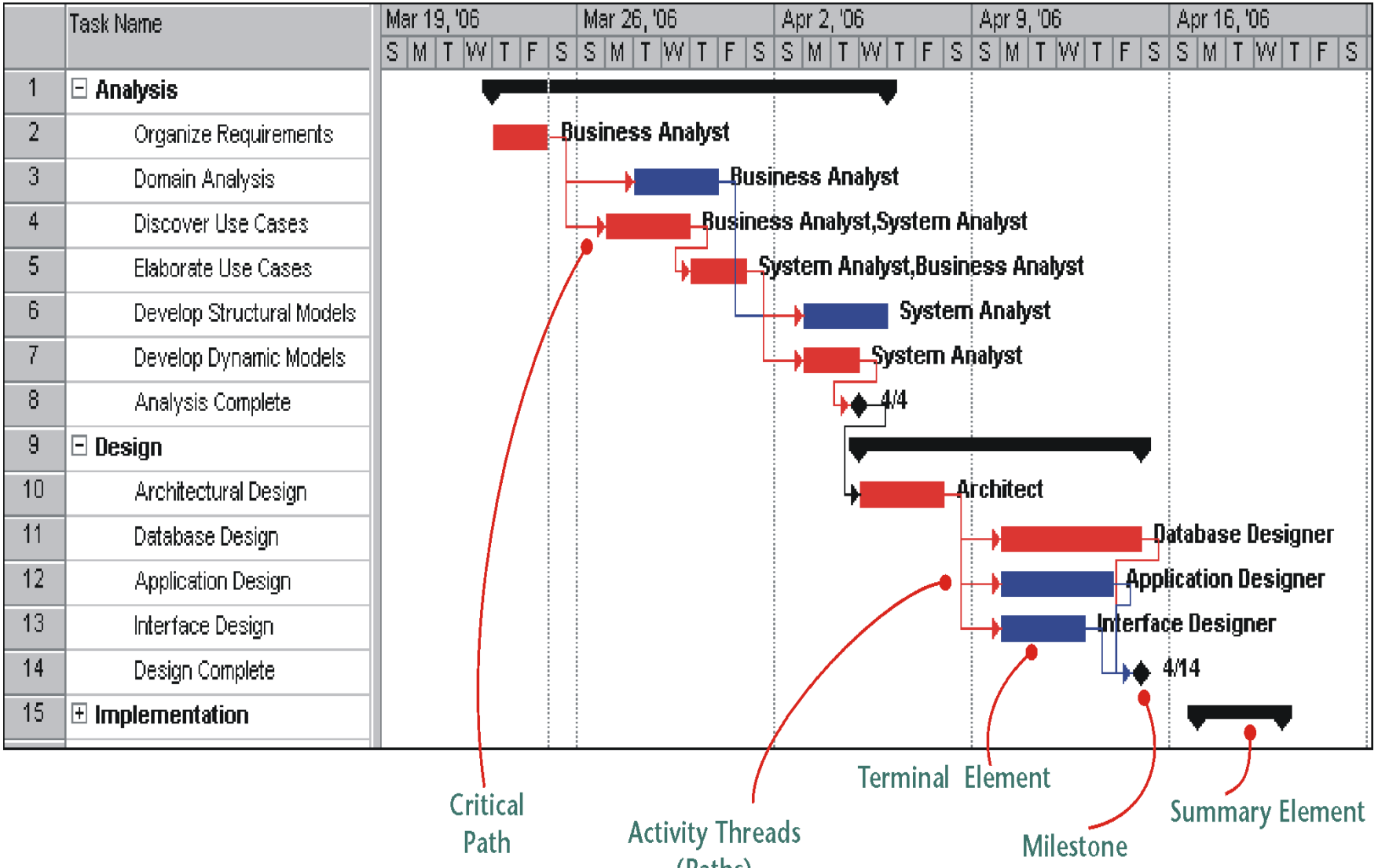
- Milestones are significant events in the life of the project. Deliverables are the verifiable results of tasks.

# Tools and Techniques

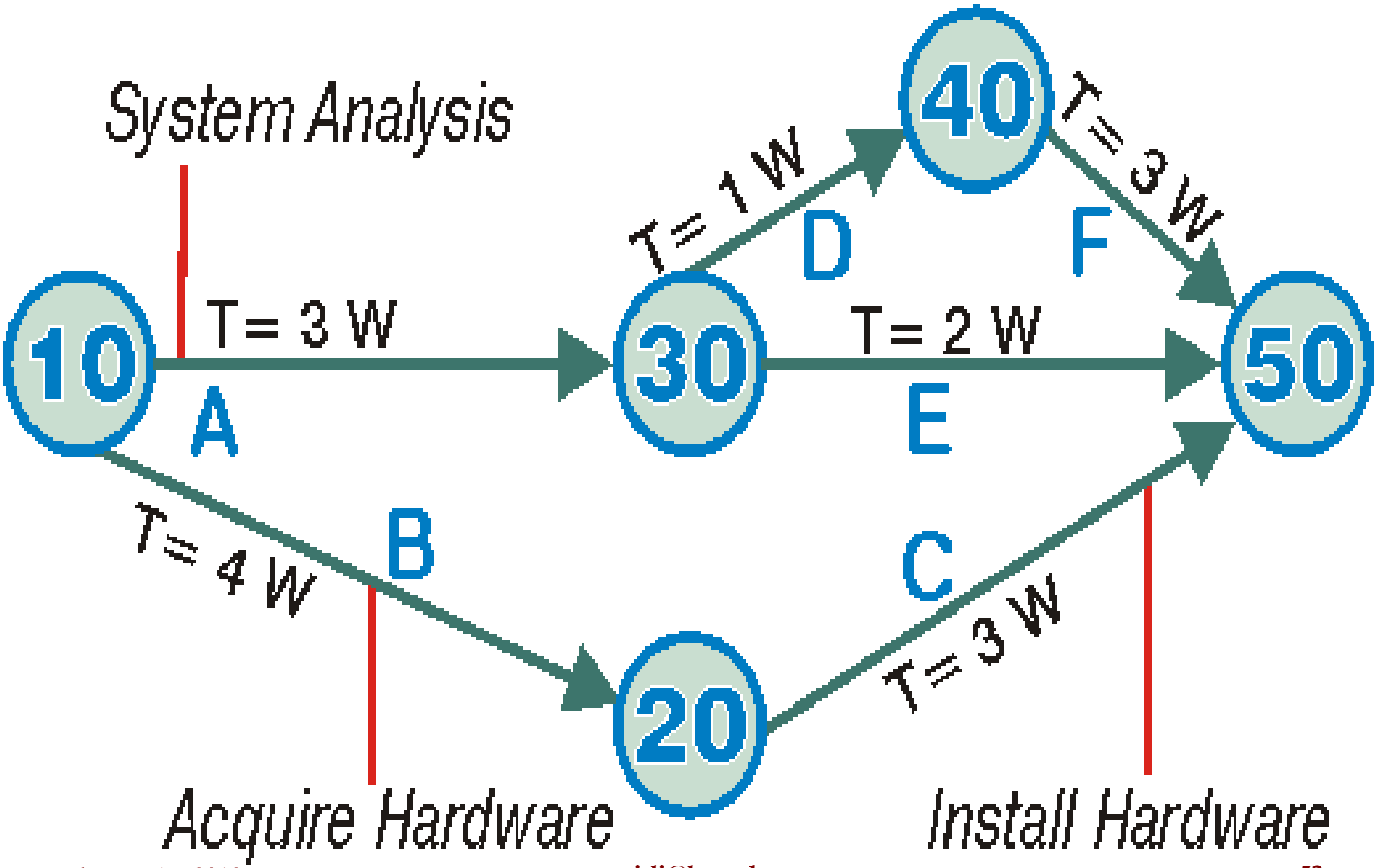
- Project management tools are primarily focused on modeling the components and the flow of the project from various perspectives and with different levels of detail.

# Gantt Chart

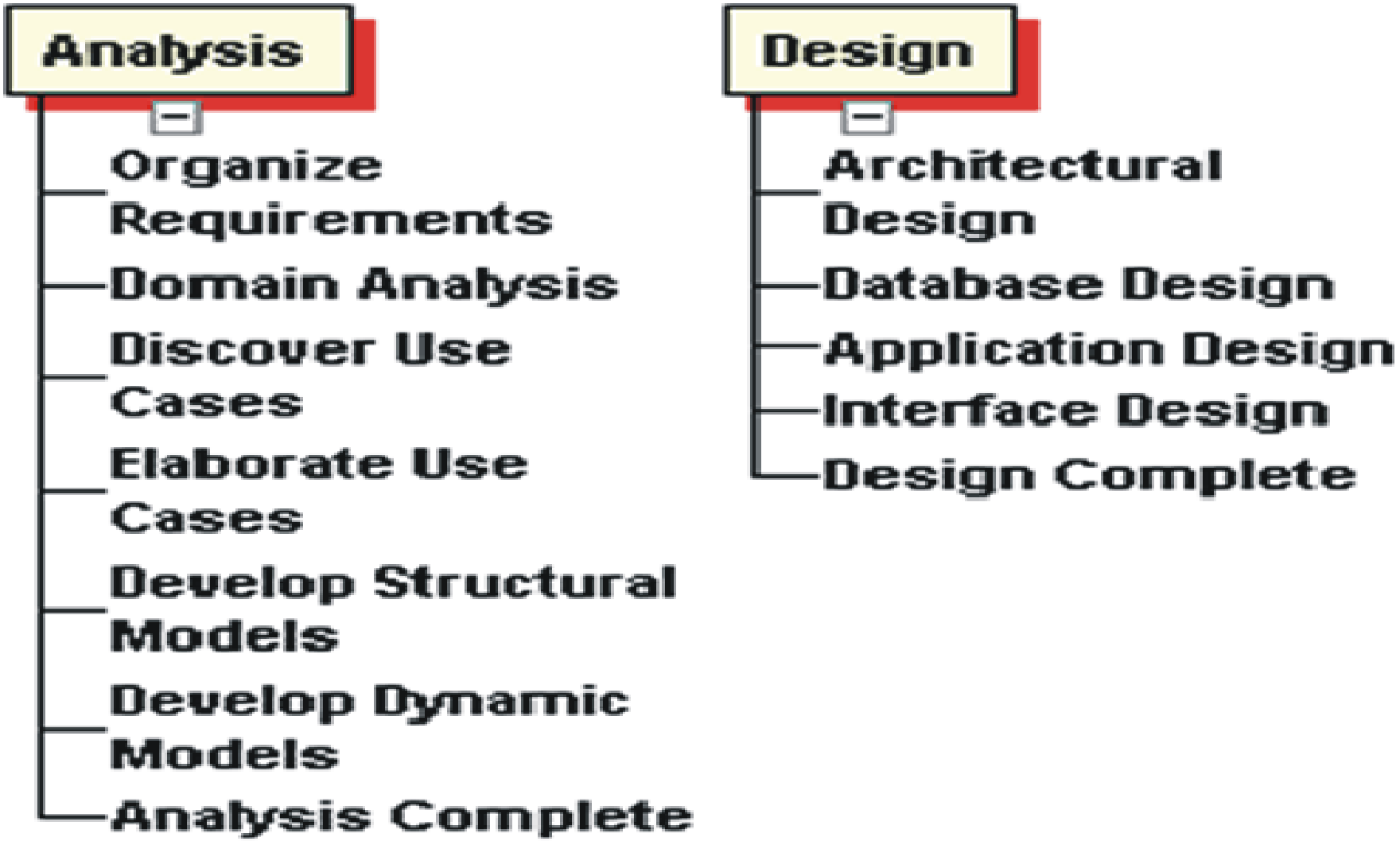
## Tracking the Project



# PERT Chart


System Analysis
10
T= 3 W
A
30
T= 1 W
D
40
T= 3 W
F
T= 2 W
E
50
T= 4 W
B
20
C
T= 3 W
Acquire Hardware
Install Hardware

# Work Breakdown Structure



# Feasibility Studies and Risk Management

- Feasibility studies aim to discover whether the expectations are realistic. Risk management guards against the unexpected.

# The Project Manager

- The job of the project manager depends on how “management” is defined.

# Project Management and Software Development

- Project plan is sequential in concept. Software development is iterative in practice.