Software Engineering

Part 1. Overview

- Introduction to Software Engineering
- Socio-Technical Systems
- Critical Systems
- Software Processes
- Project Management

JUNBEOM YOO
KONKUK University
http://dslab.konkuk.ac.kr

Ver. 1.8

Chapter 1.

Introduction to Software Engineering

Objectives

- To introduce software engineering
- To explain software engineering's importance
- To answer key questions about software engineering

Software Engineering

- Software engineering is something
  - concerned with theories, methods and tools for professional software development.
  - concerned with cost-effective software development.

- Let's define software engineering through 11 FAQs as follows.

FAQs about Software Engineering

1. What is software?
2. What is software engineering?
3. What is the difference between software engineering and computer science?
4. What is the difference between software engineering and system engineering?
5. What is a software process?
6. What is a software process model?
7. What are the costs of software engineering?
8. What are software engineering methods?
9. What is CASE (Computer-Aided Software Engineering) ?
10. What are the attributes of good software?
11. What are the key challenges facing software engineering?

1. What is Software?

- Software is computer programs and associated documentation such as requirements, design models and user manuals

- Software products may be developed for a particular customer or for a general market.
  - Generic : developed to be sold to a range of different customers. e.g. PC software such as Excel or Word
  - Bespoke (custom) : developed for a single customer according to their specification. e.g. Software used in a hospital

2. What is Software Engineering?

- Software engineering is
  - An engineering discipline that is concerned with all aspects of software production.
  - All things concerned with a successful development of software

- Software engineers should
  - adopt a systematic and organised approach
  - use
    - appropriate tools,
    - techniques depending on the problem to be solved,
    - development constraints,
    - resources available.

3. What is the Difference between Software Engineering and Computer Science?

- Computer science is concerned with theory and fundamentals.

- Software engineering is concerned with the practicalities of developing and delivering useful software.

- Computer science theories are insufficient to act as a complete underpinning for software engineering (unlike physics and electrical engineering), since it is practiced/performed by people.

4. What is the Difference between Software Engineering and System Engineering?

- System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering.

- Software engineering is part of system engineering process concerned with developing the software infrastructure, control, applications and databases in the system.

5. What is a Software Process?

- Software process is a set of activities whose goal is the development or evolution of software.

- Generic activities in all software processes
  - Specification : what the system should do
  - Development : production of the software system
  - Validation : checking that the software is really what the customer wants
  - Evolution : changing the software in response to changing demands

6. What is a Software Process Model?

- A simplified representation of a software process, presented from a specific perspective.

- Examples of process perspectives
  - Workflow perspective : sequence of activities
  - Data-flow perspective : information flow
  - Role/action perspective : who does what

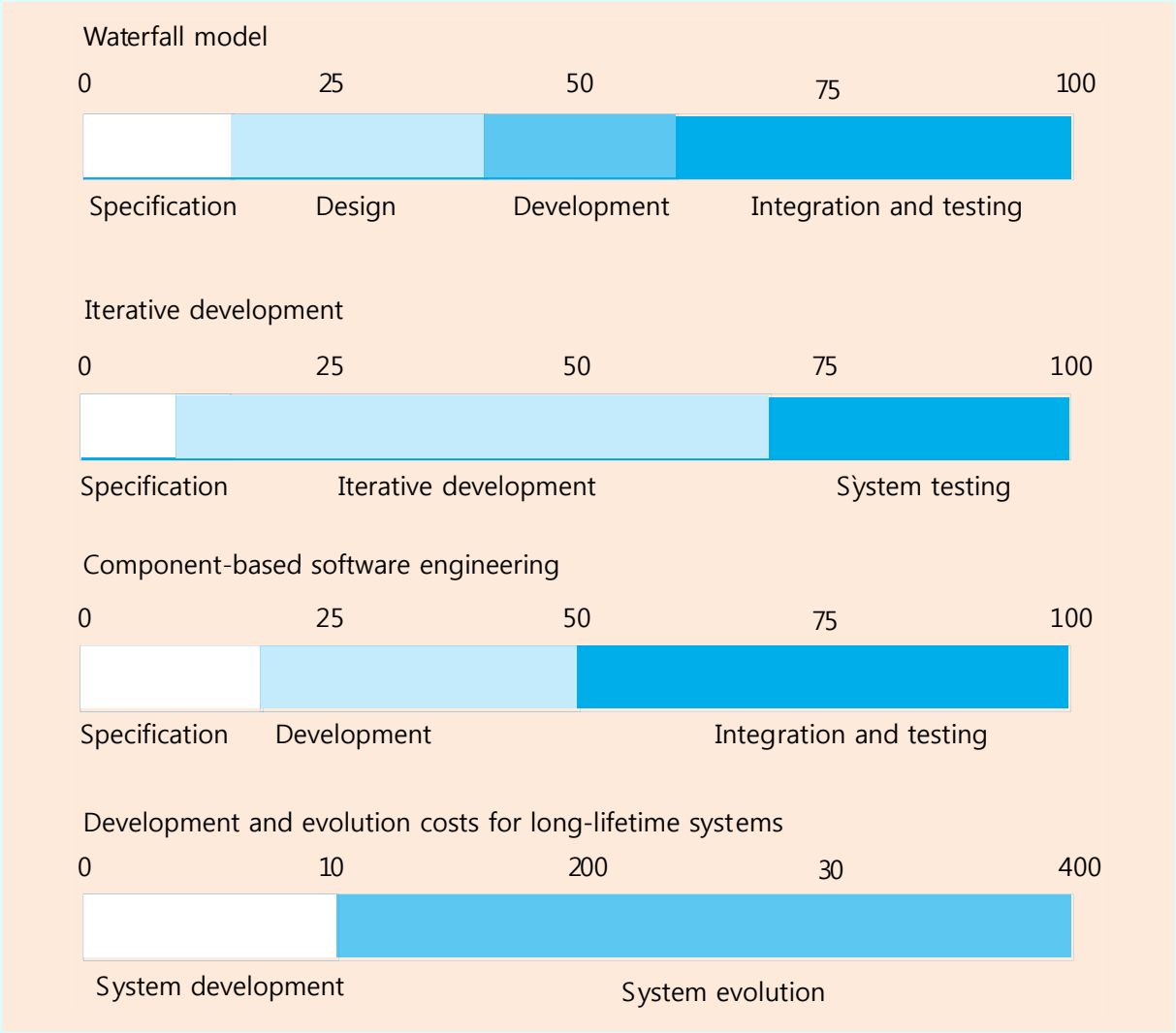
- Generic process models
  - Waterfall
  - Iterative development
  - Component-based software engineering

7. What are the Costs of Software Engineering?

- Development costs are roughly
  - 60% : development costs
  - 40% : testing costs
  - For custom (long-lifetime) software, evolution costs often exceed development costs.

- Costs can vary depending on
  - the type of system being developed
  - the requirements of system attributes (performance and system reliability)

- Therefore, distribution of costs depends on the development model that is used.



Activity-cost distribution varying depending on software process models

8. What are Software Engineering Methods?

- Organized way of producing software according to the process

- Structured approaches to software development which include system models, notations, rules, design advice and process guidance.
  - Model descriptions
    - Descriptions of graphical models which should be produced
  - Rules
    - Constraints applied to system models
  - Recommendations
    - Advice on good design practice
  - Process guidance
    - What activities to follow

9. What is CASE (Computer-Aided Software Engineering) ?

- CASEs are Software systems that are intended to provide automated support for software process activities and software engineering methods.
  - Requirements and design
  - Programming and debugging
  - Testing

10. What are the Attributes of Good Software?

- The good software should
  - deliver the required functionality and performance to the user
  - be maintainable, dependable and acceptable.

- Maintainability
  - Software must evolve to meet changing needs.
- Dependability
  - Software must be trustworthy.
- Efficiency
  - Software should not make wasteful use of system resources.
- Acceptability
  - Software must be accepted by the users for which it was designed.
  - It must be understandable, usable and compatible with other systems.

Summary

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software products consist of developed programs and associated documentation.
- Essential product attributes are maintainability, dependability, efficiency and usability.
- The software process consists of activities that are involved in developing software products. Basic activities are software specification, development, validation and evolution.
- Methods are organized ways of producing software. They include suggestions for the process to be followed, the notations to be used, rules governing the system descriptions produced, and design guidelines.
- CASE tools are software systems which are designed to support routine activities in the software process such as editing design diagrams, checking diagram consistency and keeping track of program tests which have been run.

Chapter 2.

Socio-technical Systems

Objectives

- To explain what a socio-technical system is
- To explain the distinction between a socio-technical system and a computer-based system
- To introduce the concept of emergent system properties
- To explain about system engineering
- To discuss legacy systems and why they are critical to many businesses

What is a System?

- A purposeful collection of inter-related components working together to achieve some common objective
  - May include software, mechanical, electrical and electronic hardware and be operated by people.

- Technical computer-based systems
  - Include hardware and software, but where the operators and operational processes are not normally considered to be part of the system.
  - The system is not self-aware. (e.g. Lap-top, MP3 player, cell phones, etc.)

- Socio-technical systems
  - Systems that include technical systems but also operational processes and people who use and interact with the technical system.
  - Socio-technical systems are governed by organisational policies and rules. (e.g. flight control system, transportation reservation system, etc.)

Characteristics of Socio-technical System

- Emergent properties
  - Properties of the system as a whole, depending on the system components and their relationships
  - Features :
    - non-deterministic
    - complex relationship with organizational objectives

  - Non-deterministic
    - They do not always produce the same output when presented with the same input, because the system's behaviour is partially dependent on human operators.

  - Complex relationships with organisational objectives
    - The extent to which the system supports organisational objectives does not just depend on the system itself.

Emergent Properties

- Emergent properties are a consequence of the relationships between system components.
- Therefore, they can only be assessed and measured once the components have been integrated into a system.

Property	Description
Volume	The volume of a system (the total space occupied) varies depending on how the component assemblies are arranged and connected.
Reliability	System reliability depends on component reliability but unexpected interactions can cause new types of failure and therefore affect the reliability of the system.
Security	The security of the system (its ability to resist attack) is a complex property that cannot be easily measured. Attacks may be devised that were not anticipated by the system designers and so may defeat built-in safeguards.
Repairability	This property reflects how easy it is to fix a problem with the system once it has been discovered. It depends on being able to diagnose the problem, access the components that are faulty and modify or replace these components.
Usability	This property reflects how easy it is to use the system. It depends on the technical system components, its operators and its operating environment.

Types of Emergent Properties

- Functional properties
  - These appear when all the parts of a system work together to achieve some objective.
  - For example, a bicycle has the functional property of being a transportation device once it has been assembled from its components.

- Non-functional properties
  - These relate to the behaviour of the system in its operational environment.
  - Examples are reliability, performance, safety, and security.

Reliability of Systems

- We need to consider the reliability from aspect of systems
  - Even if we have reliable software components,
  - System failures often occur dus to unforeseen interactions between reliable components.
- Therefore, we need system engineering.

- Influences on reliability
  - Hardware reliability
    - What is the probability of a hardware component failing and how long does it take to repair that component?
  - Software reliability
    - How likely is it that a software component will produce an incorrect output.
  - Operator reliability
    - How likely is it that the operator of a system will make an error?

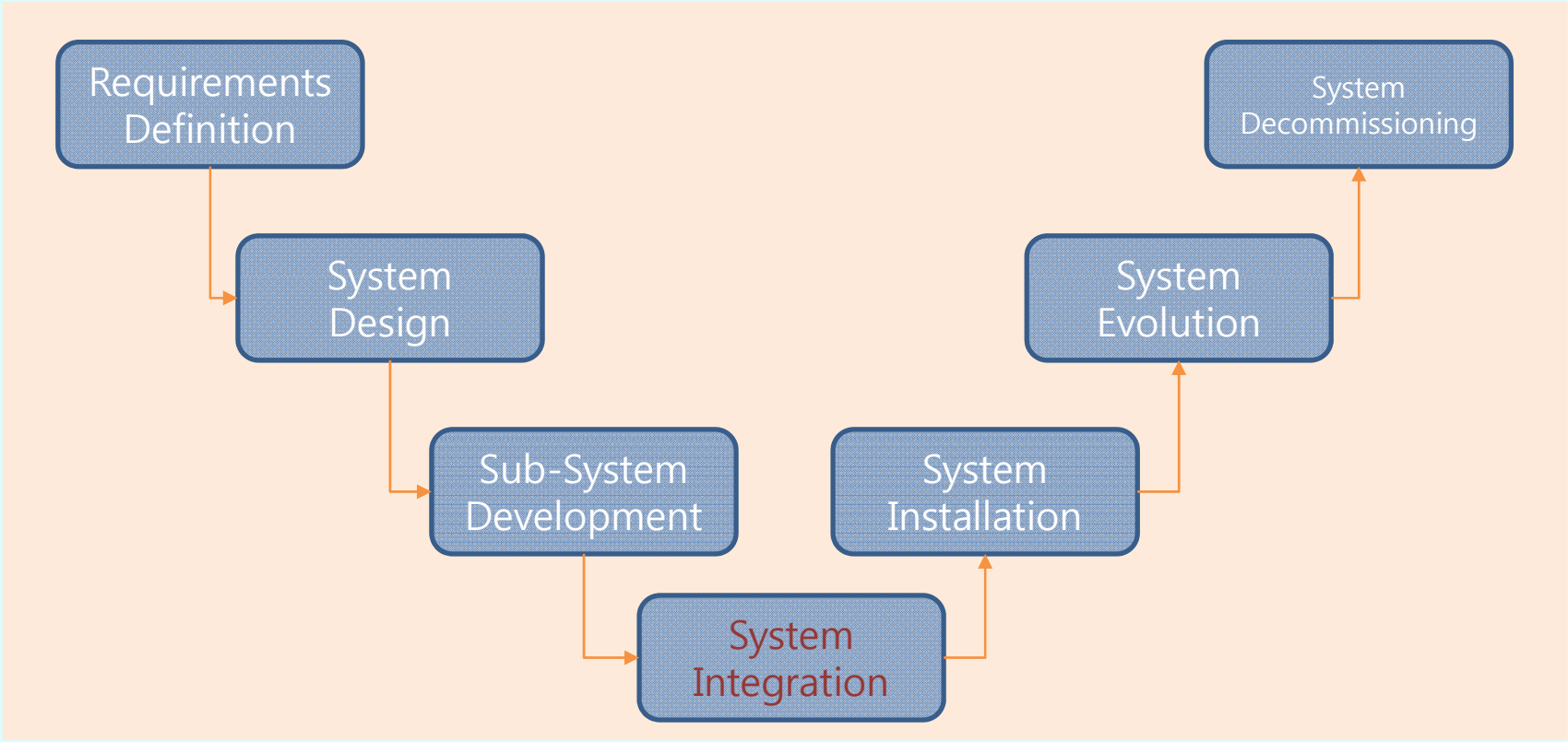
Systems Engineering

- System engineering is concerned with
  - specifying, designing, implementing, validating,
  - deploying and maintaining socio-technical systems

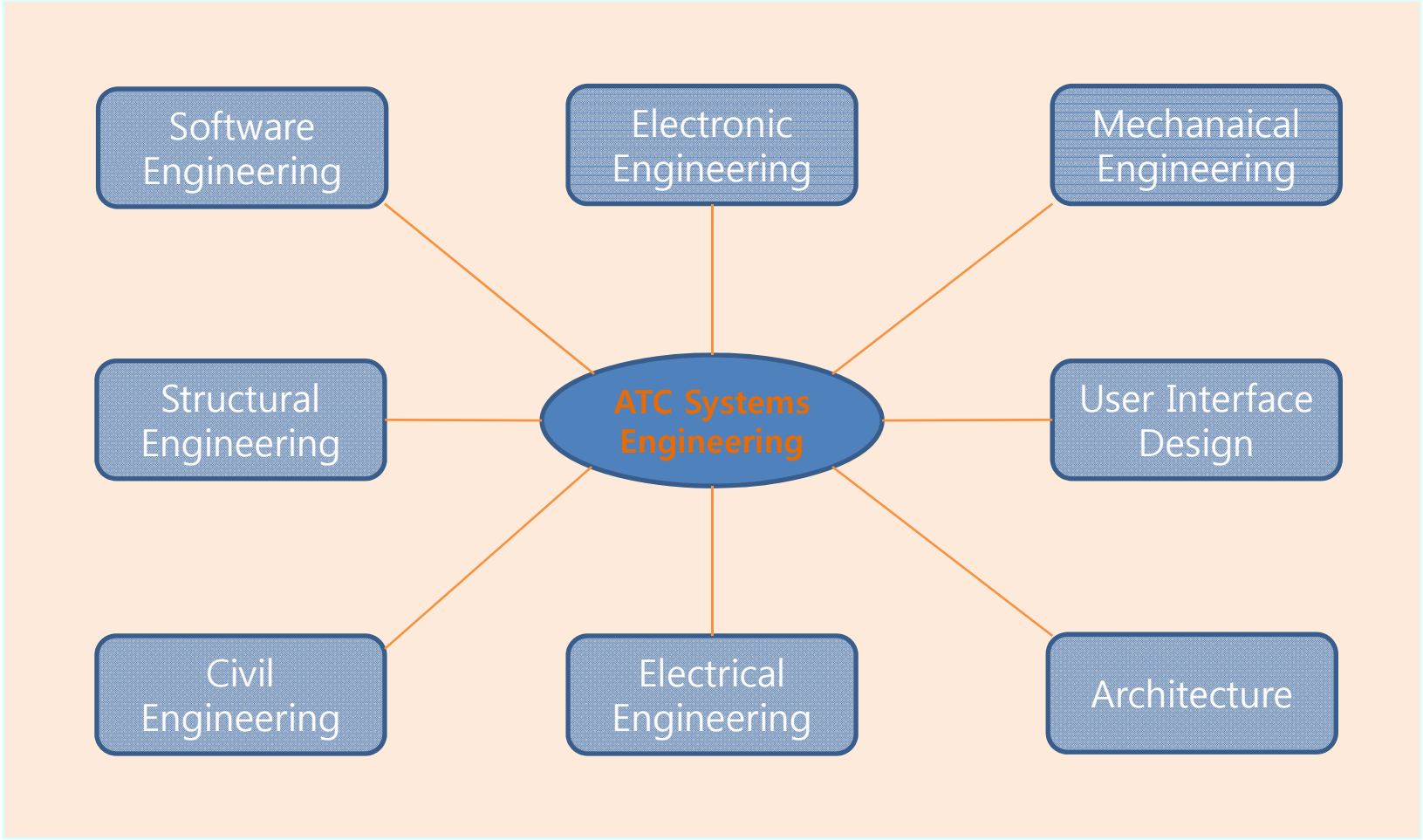
- System engineering is also concerned with
  - services provided by the system,
  - constraints on its construction,
  - operation and the ways in which it is used

Systems Engineering Process

- System engineering process
  - Usually follows a "Waterfall" model.
  - Involves engineers from different disciplines who must work together, and misunderstanding occurs here.



Example: Inter-Disciplinary Involvement in System engineering



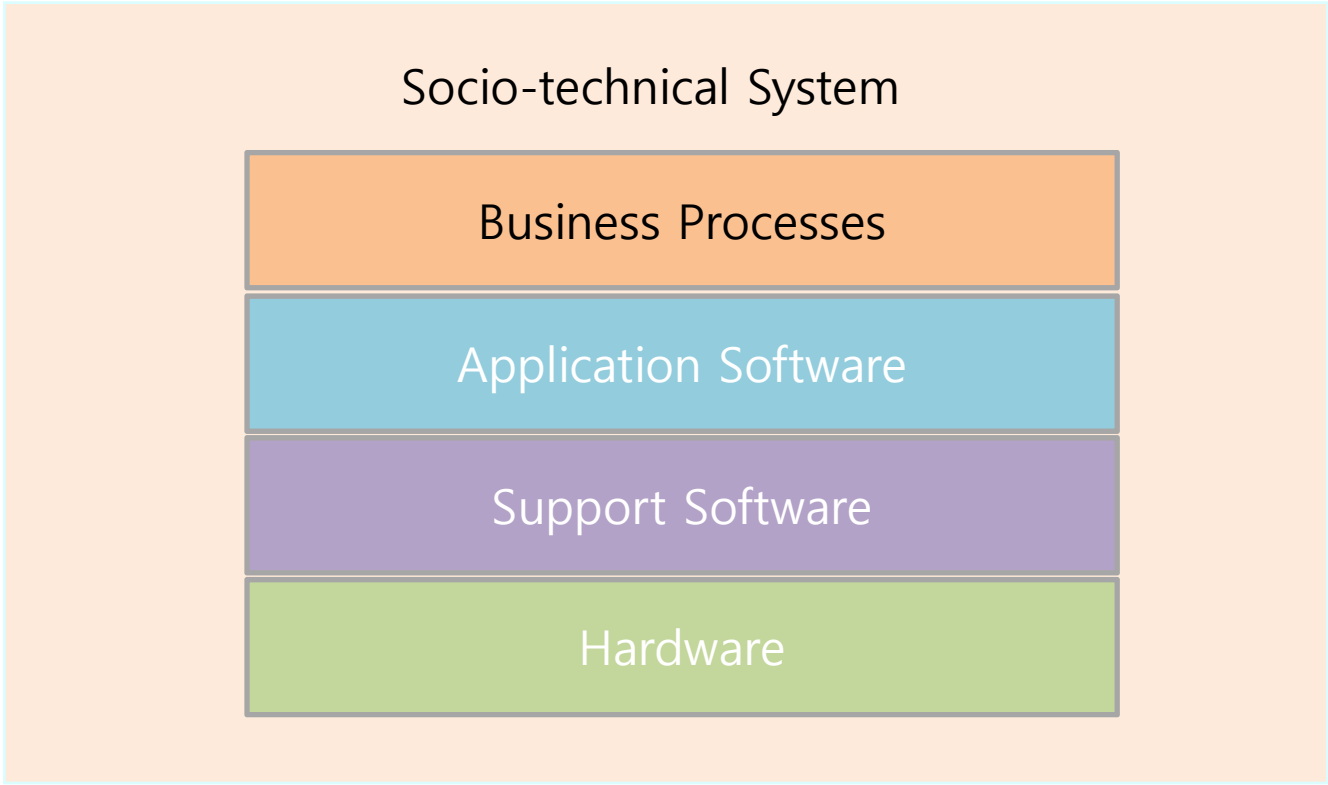
Legacy Systems

- Socio-technical systems that
  - Developed 10~20 years ago.
  - Have been in a stable manner up to now.
  - However, new business needs require a new efficient system.

- Crucial to the operation of a business
- Often too risky to change it with new ones
  - Bank customer accounting system
  - Aircraft maintenance system

- Legacy systems constrain new business processes and consume a high proportion of company budgets to maintain it..

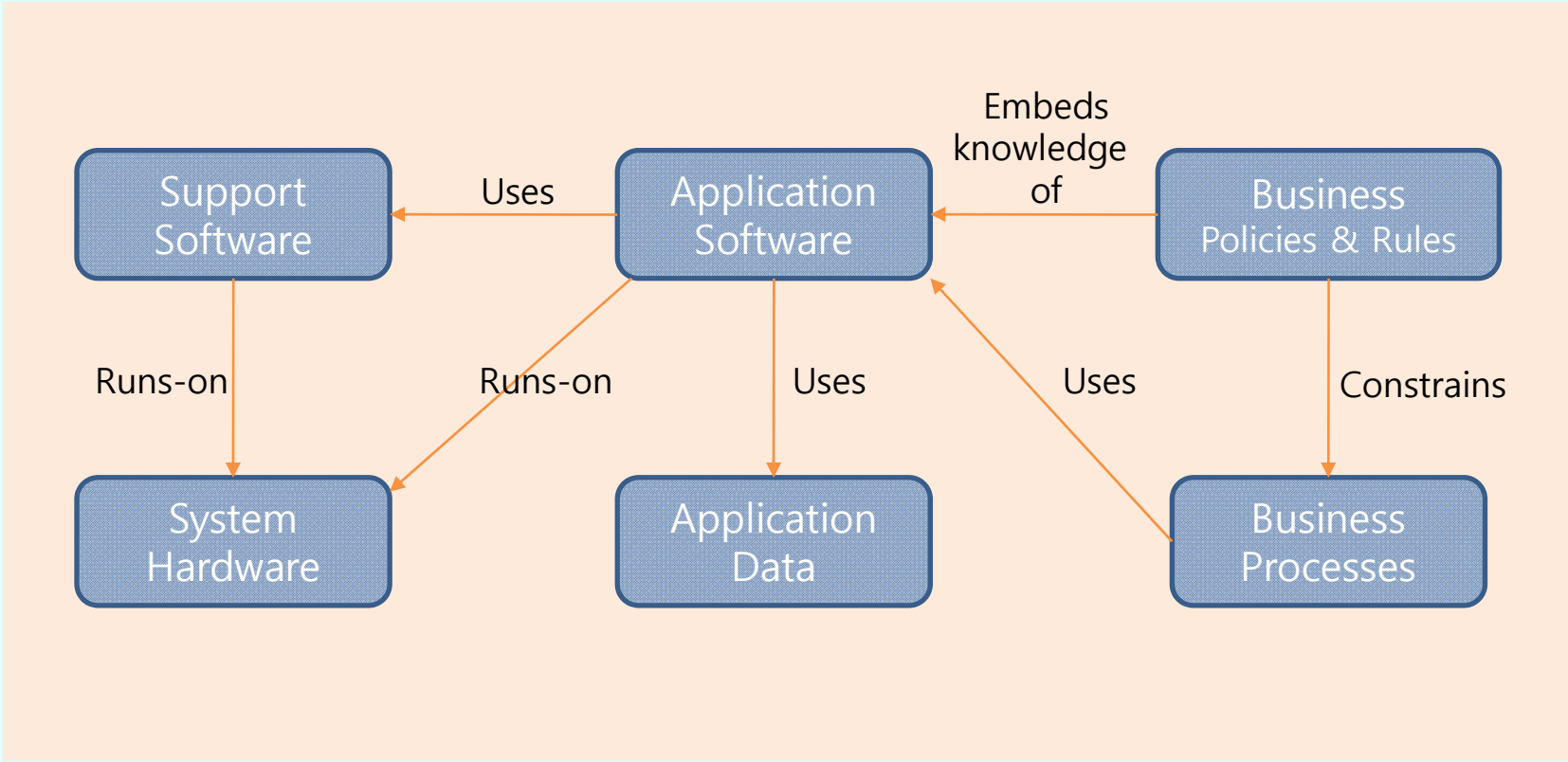
Socio-technical Legacy System



Legacy System Components

- Hardware
  - may be obsolete mainframe hardware.
- Support software
  - may rely on support software from suppliers who are no longer in business.
- Application software
  - may be written in obsolete programming languages.
- Application data
  - often incomplete and inconsistent.
- Business processes
  - may be constrained by software structure and functionality.
- Business policies and rules
  - may be implicit and embedded in the system software.

Relationship between Legacy System Components



Summary

- Socio-technical systems include computer hardware, software and people, and are designed to meet some business goal.
- Emergent properties are properties that are characteristic of the system as a whole.
- The systems engineering process includes specification, design, development, integration and testing. System integration is particularly critical.
- Human and organisational factors have a significant effect on the operation of socio-technical systems.
- A legacy system is an old system that continues to provide essential services.``

Chapter 3.

Critical Systems

Objectives

- To explain what a critical system is
- To explain four dimensions of dependability - availability, reliability, safety and security
- To explain why, for achieving dependability, you need to avoid mistakes, detect and remove errors and limit damage caused by failure (a mid-term problem)

Critical Systems

- Safety-critical systems
  - Failure results in loss of life, injury or damage to environment
  - Ex) Chemical plant protection system
- Mission-critical systems
  - Failure results in failure of some goal-directed activities
  - Ex) Spacecraft navigation system
- Business-critical systems
  - Failure results in high economic losses
  - Ex) Customer accounting system in bank

Development Methods for Critical Systems

- The costs of critical system failure are so high.
- Therefore, development methods for critical systems are not cost-effective for other types of system.

- Examples of development methods
  - Formal methods (specification and verification)
  - Static analysis
  - External quality assurance

System Dependability

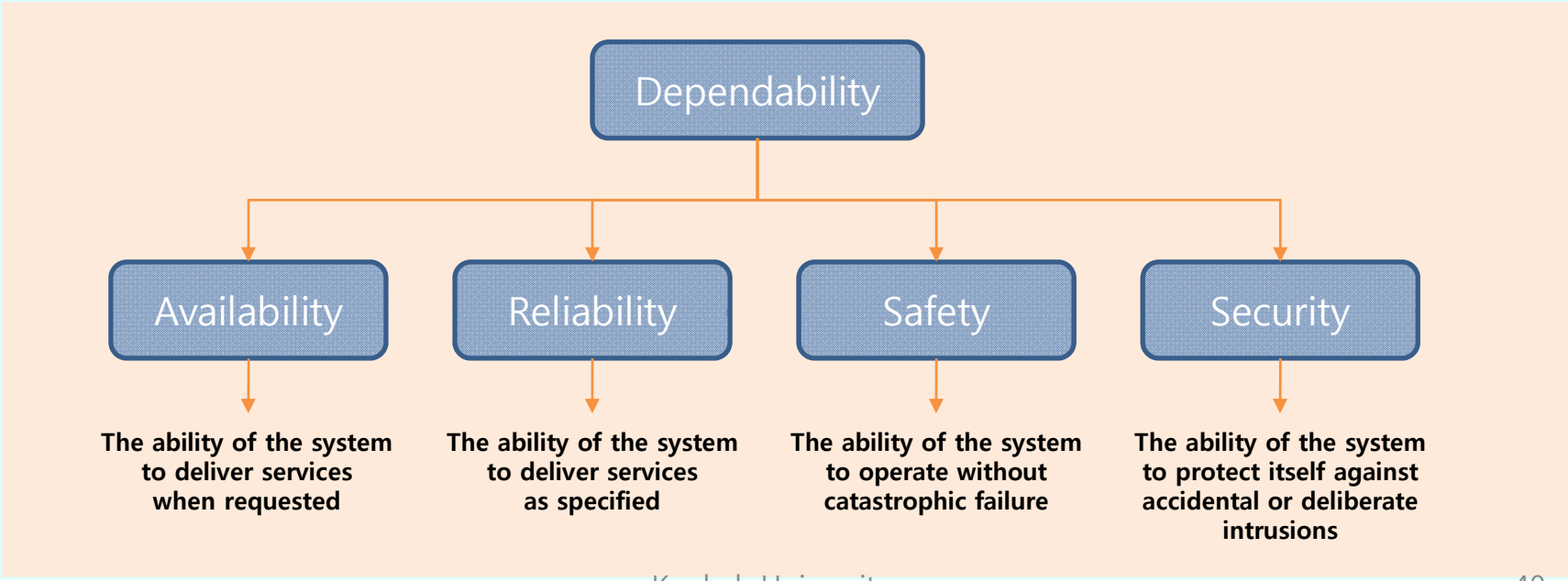
- For critical systems, it is usually the case that the most important system property is the dependability of the system.

- It reflects the extent of the user's confidence that it will operate as users expect and that it will not 'fail' in normal use.

Dependability

- Dependability of system equates to its trustworthiness.
- Dependable system is a system that is trusted by its users.

- Principal dimensions of dependability
  - Availability, Reliability, Safety, Security

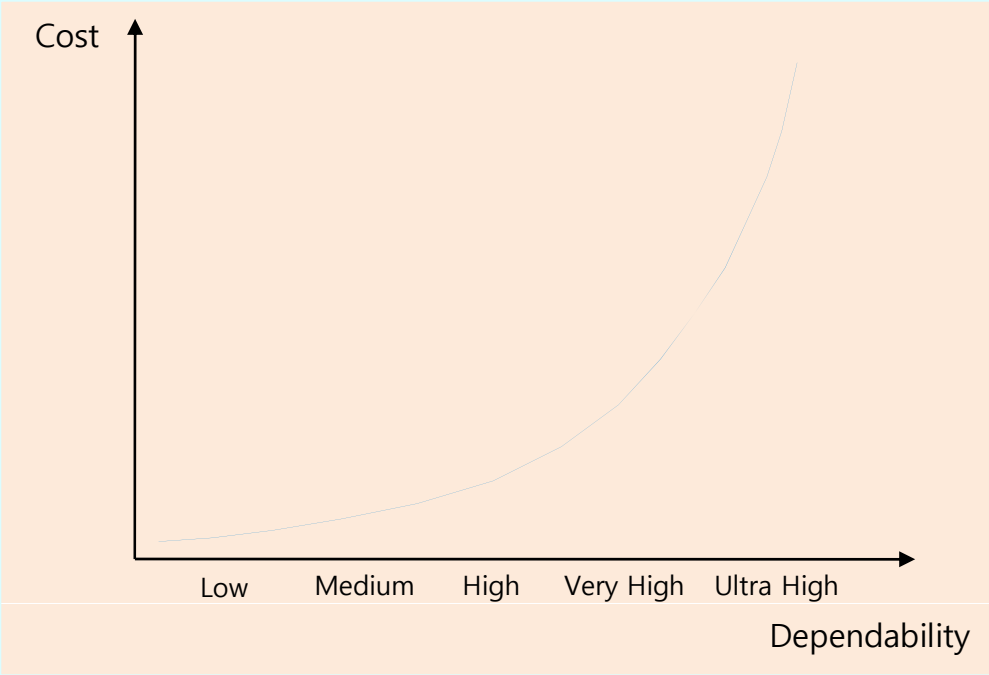


Other Dependability Properties

- Reparability
  - To which extent the system can be repaired in the event of a failure
- Maintainability
  - To which extent the system can be adapted to new requirements
- Survivability
  - To which extent the system can deliver services whilst under hostile attack
- Error tolerance
  - To which extent user input errors can be avoided and tolerated

Dependability Costs

- Dependability costs tend to increase exponentially as required levels of dependability increase.
  - More expensive development techniques and hardware are required.
  - Increased testing and system validation are also required.



Dependability Economics

- Because of very high costs of dependability achievement
- It may be more cost effective to accept untrustworthy systems and pay for failure costs.

- However, it depends on
  - Social and political factors
    - Poor reputation for products may lose future business.
  - System type
    - For business systems, modest levels of dependability may be adequate.

Availability and Reliability

- Availability
  - The probability that a system will be operational and able to deliver the requested services, at a point in time
- Reliability
  - The probability of failure-free system operation over a specified time in a given environment for a given purpose
- Both of these attributes can be expressed quantitatively.
- This class considers them as the same.

Reliability Terminology

Term	Description
System failure	An event that occurs at some point in time when the system does not deliver a service as expected by its users
System error	An erroneous system state that can lead to system behavior that is unexpected by system users.
System fault	A characteristic of a software system that can lead to a system error. For example, failure to initialize a variable could lead to that variable having the wrong value when it is used.
Human error or mistake	Human behavior that results in the introduction of faults into a system.

Reliability Achievement

- Fault avoidance
  - Use development technique which either minimize the possibility of mistakes or trap mistakes before they result in the introduction of system faults
- Fault detection and removal
  - Use verification and validation techniques which increase probability of detecting and correcting errors before system goes into service
- Fault tolerance
  - Use run-time techniques to ensure that system faults do not result in system errors and/or to ensure that system errors do not lead to system failures

Safety

- Safety is a system property that reflects the system's ability to operate, normally or abnormally, without danger of causing human injury or death and without damage to the system's environment.

- Exclusive requirements
  - Exclude undesirable situations rather than specify required system services.
  - "Should not" property

Safety Terminology

Term	Description
Accident (mishap)	An unplanned event or sequence of events which results in human death or injury, damage to property or to the environment. A computer-controlled machine injuring its operator is an example of an accident.
Risk	This is a measure of the probability that the system will cause an accident. The risk is assessed by considering the hazard probability, the hazard severity and the probability that a hazard will result in an accident.
Damage	A measure of the loss resulting from a mishap. Damage can range from many people killed as a result of an accident to minor injury or property damage.
Hazard	A condition with the potential for causing or contributing to an accident. A failure of the sensor that detects an obstacle in front of a machine is an example of a hazard.
Hazard severity	An assessment of the worst possible damage that could result from a particular hazard. Hazard severity can range from catastrophic where many people are killed to minor where only minor damage results.
Hazard probability	The probability of the events occurring which create a hazard. Probability values tend to be arbitrary but range from probable (say 1/100 chance of a hazard occurring) to implausible (no conceivable situations are likely where the hazard could occur).

Safety Achievement

- Hazard avoidance
  - Design the system so that some classes of hazard simply cannot arise
- Hazard detection and removal
  - Design the system so that hazards are detected and removed before they result in an accident
- Damage limitation
  - Includes protection features that minimise the damage that may result from an accident

Security

- Security is a system property that reflects the system's ability to protect itself from accidental or deliberate external attack.
- Security is becoming increasingly important as systems are networked so that external access to the system through the Internet is possible.
- Security is an essential pre-requisite for availability, reliability and safety.

Security Terminology

Term	Description
Exposure	Possible loss or harm in a computing system. This can be loss or damage to data or can be a loss of time and effort if recovery is necessary after a security breach.
Vulnerability	A weakness in a computer-based system that may be exploited to cause loss or harm.
Attack	An exploitation of a system vulnerability. Generally, this is from outside the system and is a deliberate attempt to cause some damage.
Threats	Circumstances that have potential to cause loss or harm. You can think of these as a system vulnerability that is subjected to an attack.
Control	A protective measure to reduce a system vulnerability. Encryption would be an example of a control that reduced a vulnerability of a weak access control system.

Security Assurance

- Vulnerability avoidance
  - Design the system so that vulnerabilities do not occur
  - For example, if there is no external network connection, any external attack is impossible.

- Attack detection and elimination
  - Design the system so that attacks on vulnerabilities are detected and neutralised before they result in an exposure
  - For example, virus checkers find and remove viruses before they infect a system.

- Exposure limitation
  - Design the system so that the adverse consequences of a successful attack are minimized
  - For example, a backup policy allows damaged information to be restored.

Summary

- A critical system is a system where failure can lead to high economic loss, physical damage or threats to life.
- Dependability of a system reflects user's trust in that system.
- Availability is the probability that it will be available to deliver services when requested.
- Reliability is the probability that system services will be delivered as specified.
- Safety is a system attribute that reflects the system's ability to operate without threatening people or the environment.
- Security is a system attribute that reflects the system's ability to protect itself from external attack.

Chapter 4.

Software Processes

Objectives

- To introduce software process models
- To describe three generic process models
- To describe common process activities
- To explain the Rational Unified Process(RUP) model

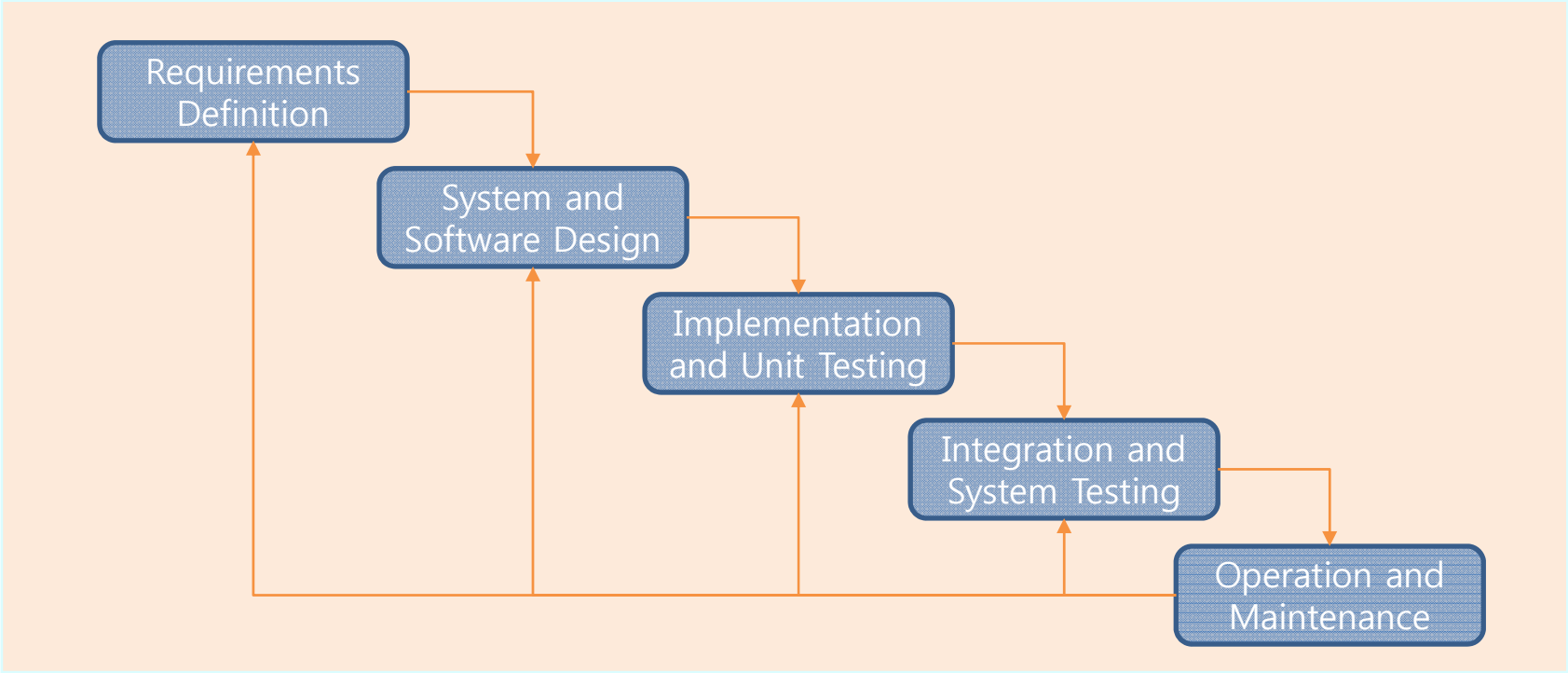
Software Process

- A structured set of activities required to develop a software system
  - Specification
  - Design
  - Validation
  - Evolution

- A software process model is an abstract representation of a process.
  - Waterfall model
  - Evolutionary development
  - Component-based software engineering
  - Many variants

Waterfall Model

- A classic life cycle model
  - Suggests a systematic, sequential approach to software development
  - The oldest paradigm
  - Separate and distinct phases of specification and development
  - Useful in situations where
    - requirements are fixed and work is to proceed to completion in a linear manner



Waterfall Model

- Inflexible partitioning of project into distinct stages makes it difficult to respond to changing customer requirements.

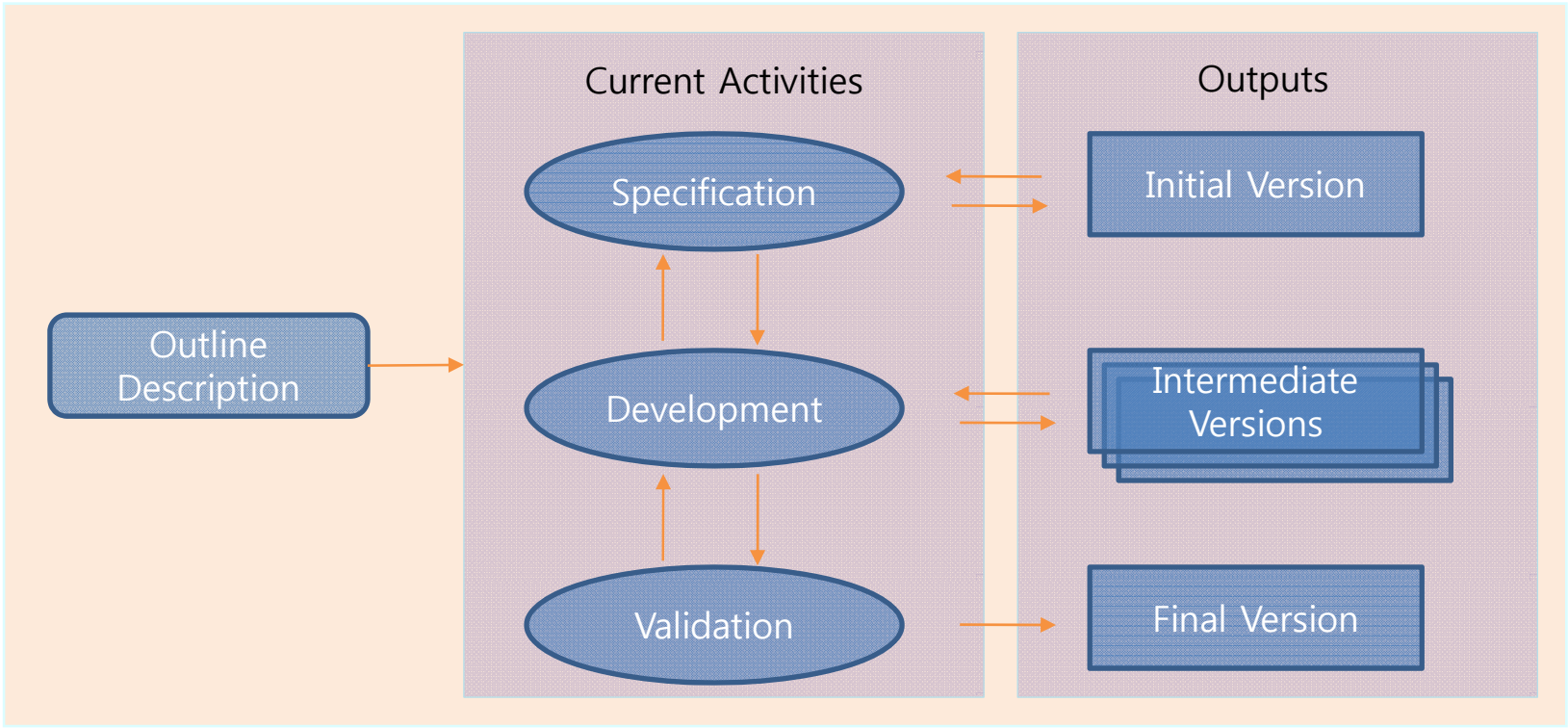
- Therefore, it is only appropriate when
  - Requirements are well-understood.
  - Changes will be fairly limited during design process.

- Waterfall model is mostly used for large system engineering projects where a system is developed at several sites.

- However, few business systems have stable requirements.

Evolutionary Development

- Exploratory development
  - Evolve a final system from an initial outline specification to work with customers.
  - Start with well-understood requirements and add new features as proposed by the customer.



Evolutionary Development

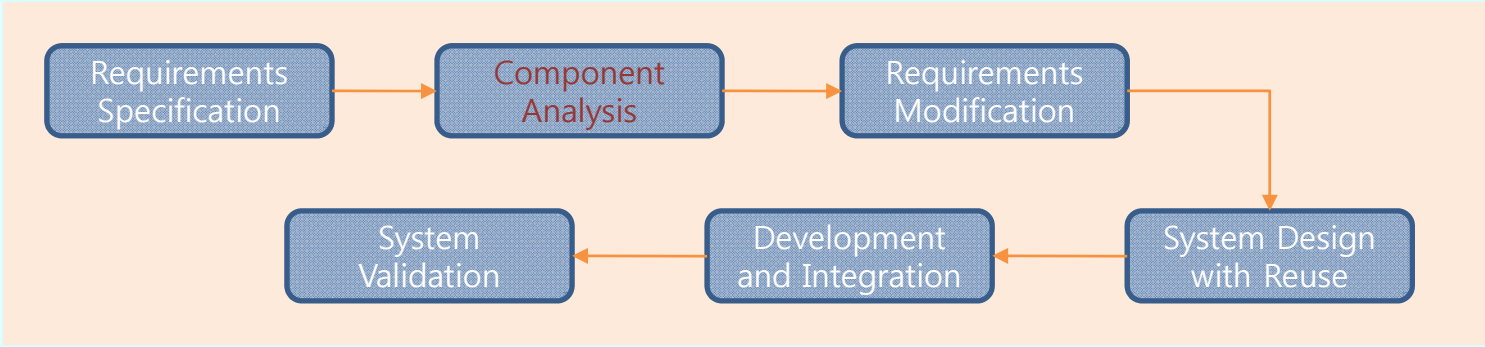
- Problems
  - Lack of process visibility
  - Systems are often poorly structured.
  - Special skills (e.g. in languages for rapid prototyping) may be required.

- Applicability
  - For small or medium-size interactive systems
  - For parts of large systems (e.g. the user interface)
  - For short-lifetime systems

Component-Based Software Engineering

- Systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Based on systematic reuse

- Process stages



Process Iteration

- System requirements always evolve in the course of a project.
- Process iteration itself is often a part of the process for large systems.

- Iteration can be applied to any of the generic process models.

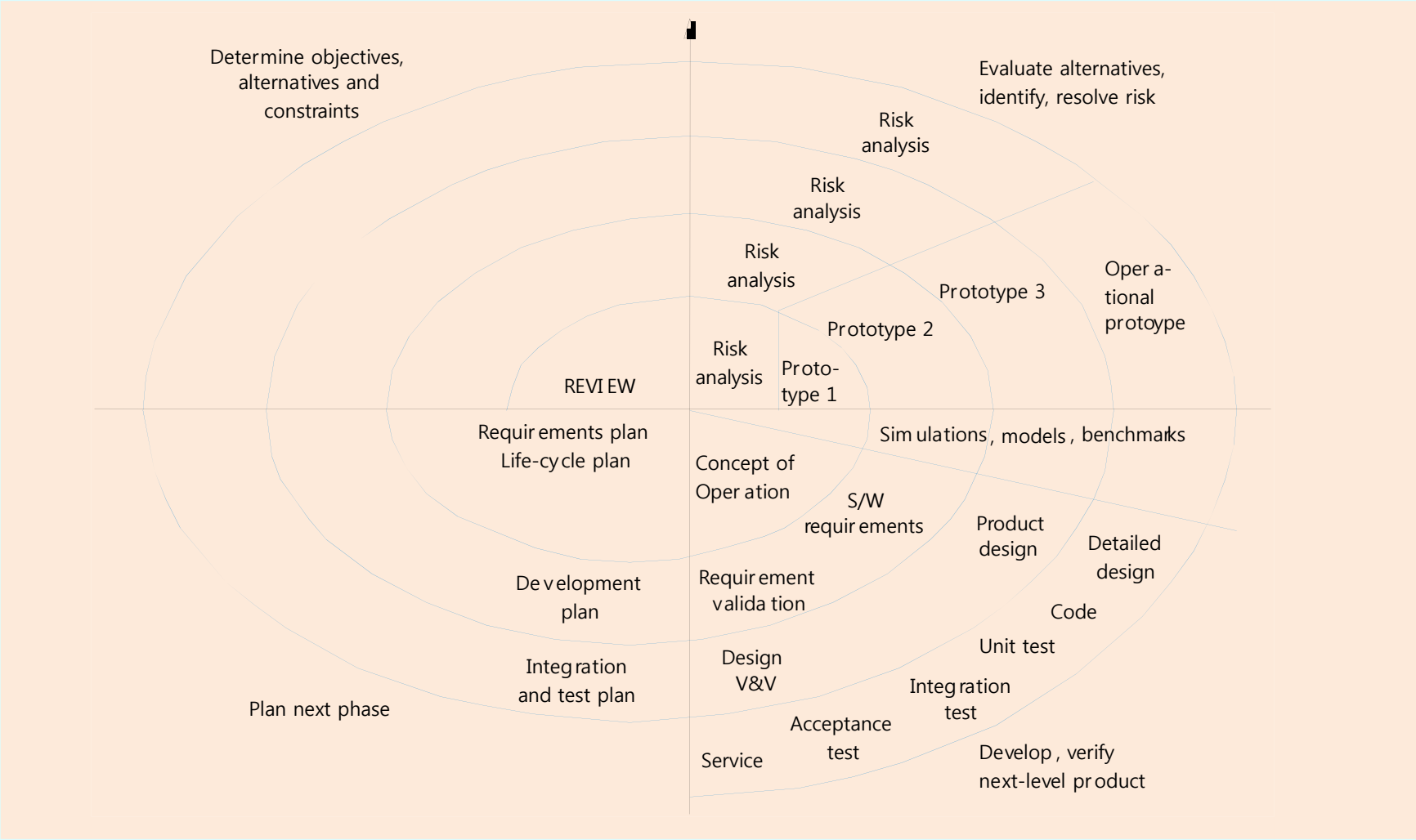
- Two (related) approaches
  - Incremental delivery (→ evolutionary development)
  - Spiral development

Spiral Development

- Represented as a spiral.
- No fixed phases - loops in the spiral are chosen depending on what is required.
- Risks are explicitly assessed and resolved throughout the process.

- Spiral model sectors
  - Objective setting
    - Specific objectives for the phase are identified.
  - Risk assessment and reduction
    - Risks are assessed and activities put in place to reduce the key risks.
  - Development and validation
    - A development model for the system is chosen which can be any of the generic models.
  - Planning
    - The project is reviewed and the next phase of the spiral is planned.

Spiral Development Model



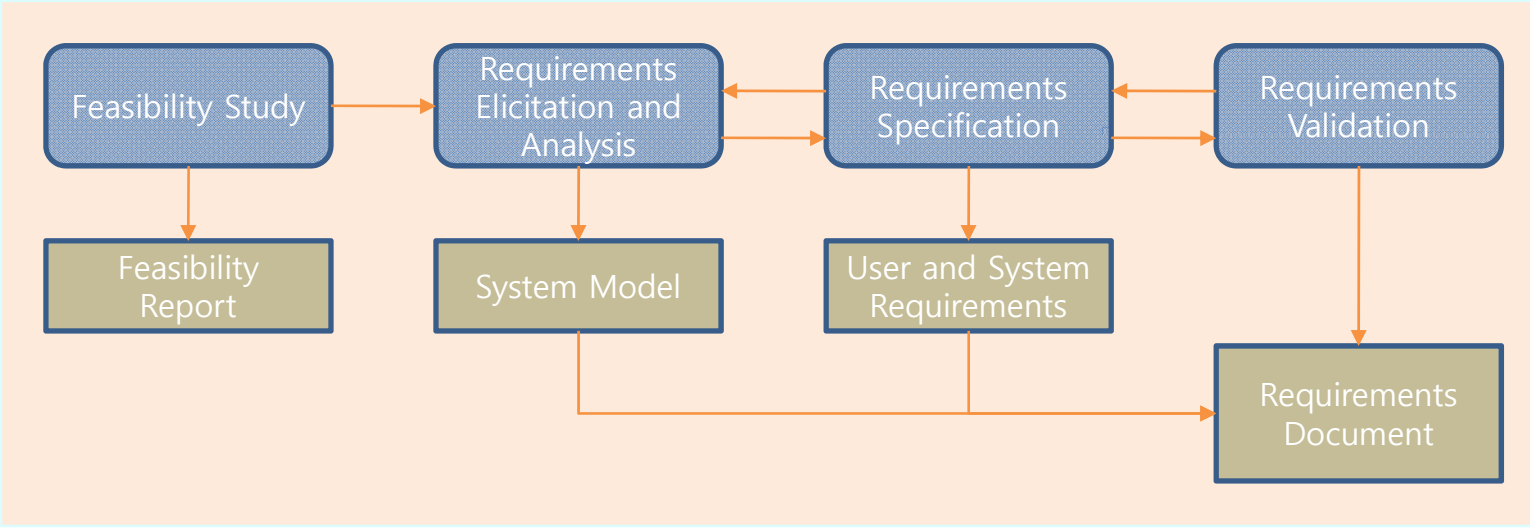
4 Common Process Activities

1. Software specification
2. Software design and implementation
3. Software validation
4. Software evolution

1. Software Specification

- Process of establishing
  - What services are required and
  - Constraints on the system's operation and development
  - Called "Requirements Engineering"

- Requirements engineering process

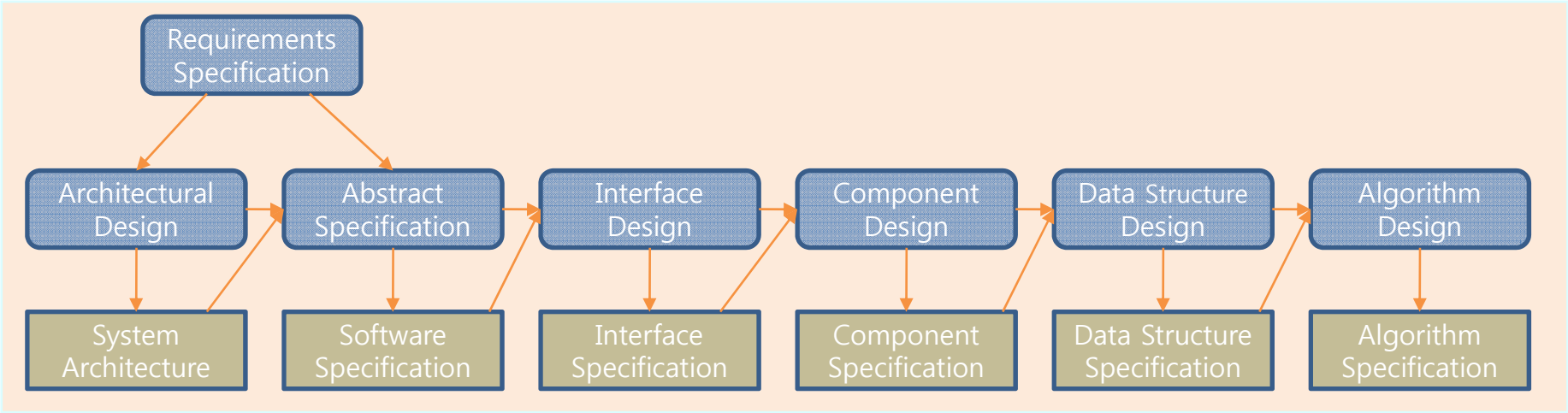


2. Software Design and Implementation

- Process of converting system specification into executable system.

- Software design
  - Design a software structure to realize the specification
- Implementation
  - Translate the design structure into an executable program.
  - Programming is a personal activity. No generic programming process.

- Software design process



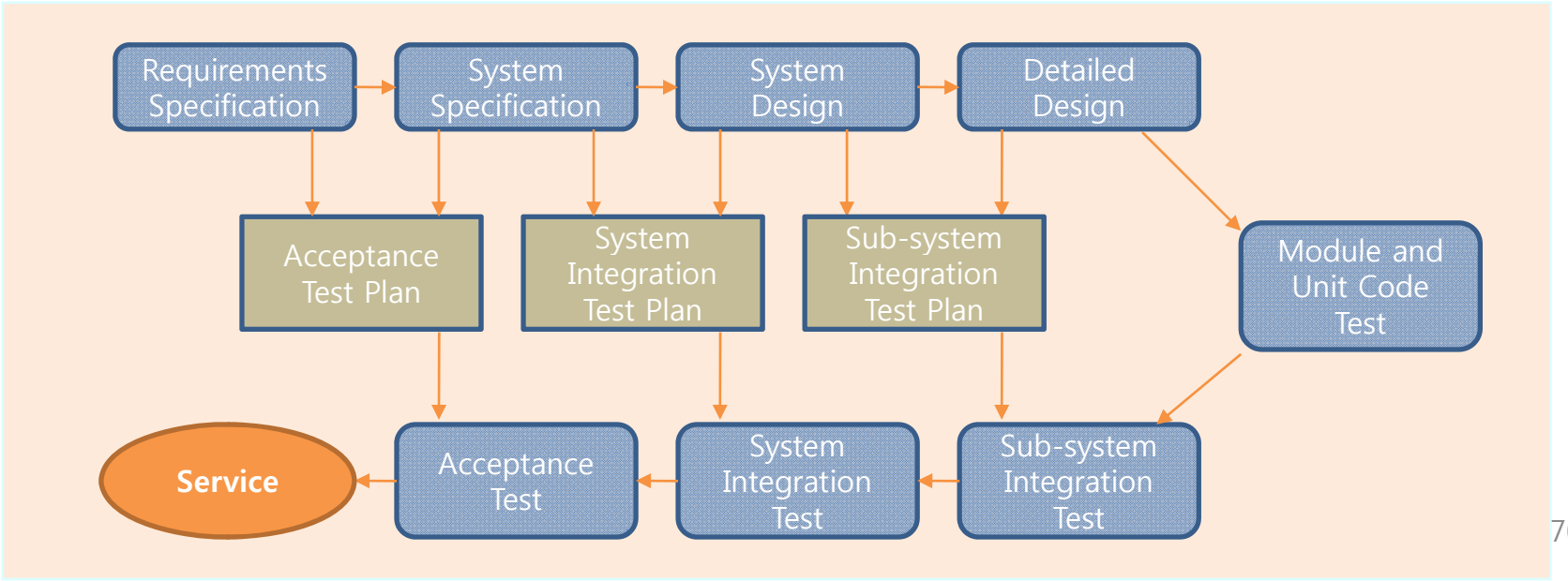
3. Software Validation

- Verification and validation (V & V) is intended to show that
  - System conforms to its specification.
  - System meets requirements of the system customer.
  - Involves
    - Checking (Formal/Informal)
    - Review processes
    - System testing

- System testing involves executing the system with test cases derived from its specification.

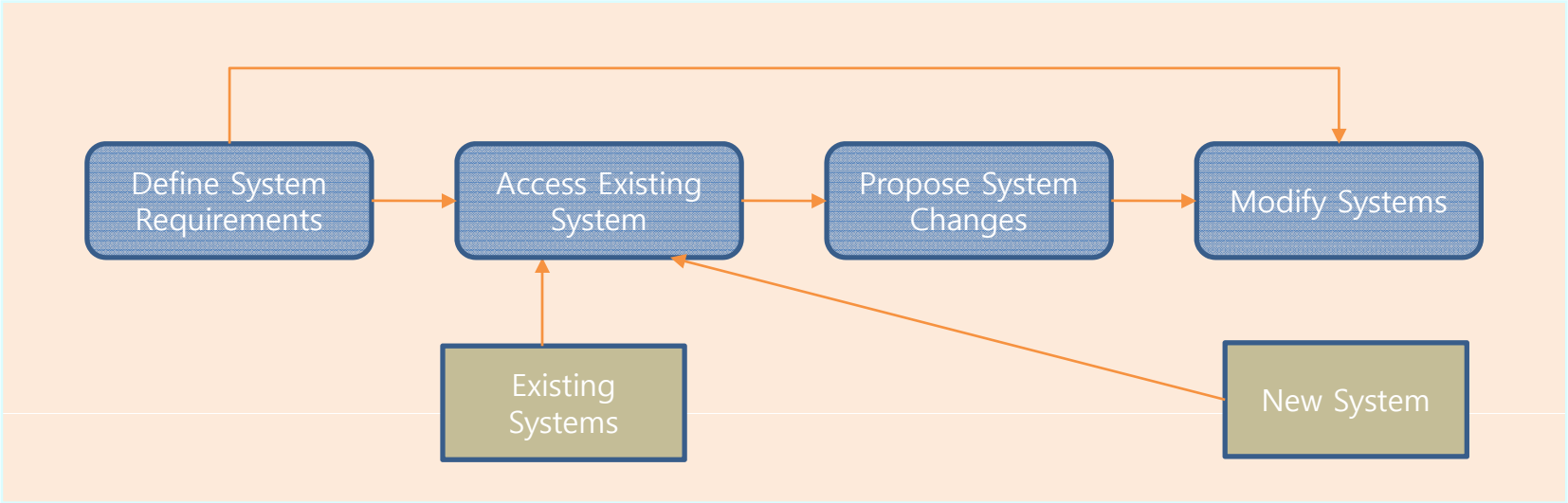
Testing Stages and Phases

- Unit or Component testing
  - Individual components are tested independently.
- System testing
  - Testing of the system as a whole.
- Acceptance testing
  - Testing with customer data to check whether the system meets the customer's needs.



4. Software Evolution

- Software is inherently flexible and can change.
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change.

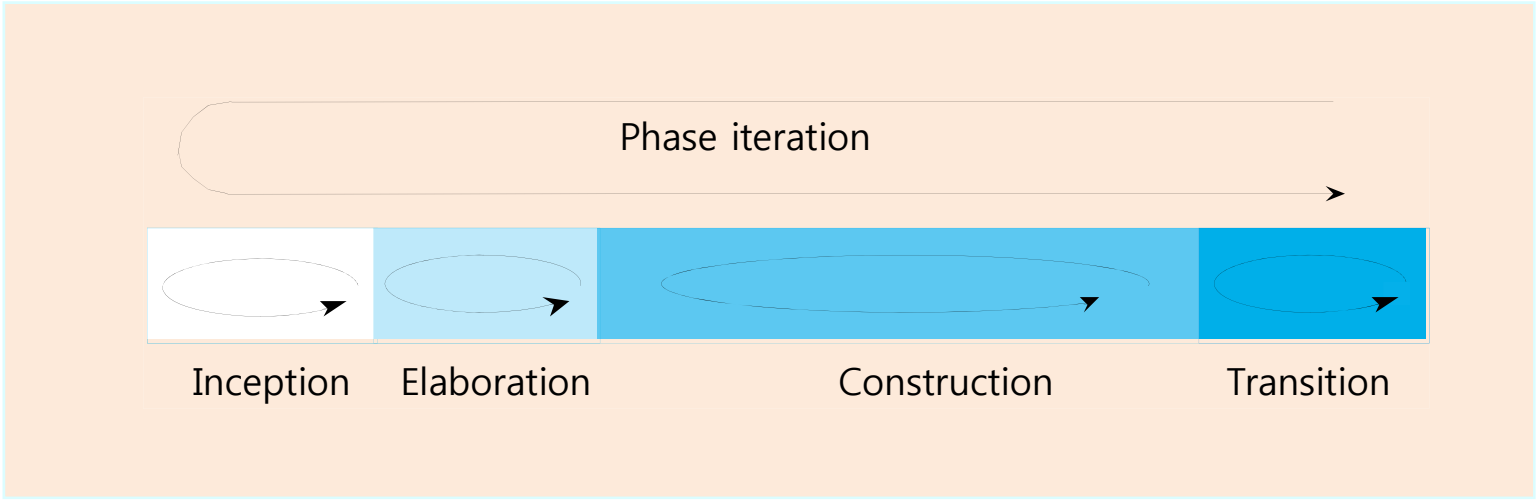


The Rational Unified Process

- A modern process model
  - Derived from working groups on the UML

- Normally described from 3 perspectives
  - Dynamic perspective : shows phases over time
  - Static perspective : shows process activities
  - Practice perspective : suggests good practice.

4 Phases of RUP (Dynamic Perspective)



Workflows(Activities) of the RUP (Static Perspective)

Workflow	Description
Business Modeling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and Design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.
Test	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and Change Management	This supporting workflow managed changes to the system (see Chapter 29).
Project Management	This supporting workflow manages the system development (see Chapter 5).
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

RUP Good Practice (Practice Perspective)

- Suggestions:
  - Develop software iteratively
  - Manage requirements
  - Use component-based architectures
  - Model software Visually
  - Verify software quality
  - Control changes to software

- More than 1,00 best practices.

Summary

- Software processes are the activities involved in producing and evolving a software system.
- Software process models are abstract representations of these processes.
- Generic process models describe organization of software processes. Examples include the waterfall model, evolutionary development and component-based software engineering.
- General software process activities are specification, design and implementation, validation and evolution.
- The Rational Unified Process is a generic process model based on UML.

Chapter 5.

Project Management

Objectives

- To explain main tasks undertaken by project managers
- To introduce software project management and to describe its distinctive characteristics
- To discuss project planning and planning process
- To discuss the notion of risks and risk management process

Software Project Management

- Concerned with activities involved in ensuring that software is delivered
  - on time and
  - on schedule and
  - in accordance with the requirements of the organizations developing and procuring the software.

- Needed because software development is always subject to budget and schedule constraints that are set by the organization developing the software.

Project Management Activities

- Proposal writing
- Project staffing
- Project planning and scheduling
- Project costing
- Project monitoring and reviews
- Personnel selection and evaluation
- Report writing and presentations

Project Staffing

- May not be possible to appoint ideal people to work on a project
  - Project budget may not allow for the use of highly-paid staff.
  - Staff with appropriate experience may not be available.
  - Organization may wish to develop employee skills through performing software projects.

- Managers have to work within these constraints especially when there are shortages of trained staff.

Project Planning

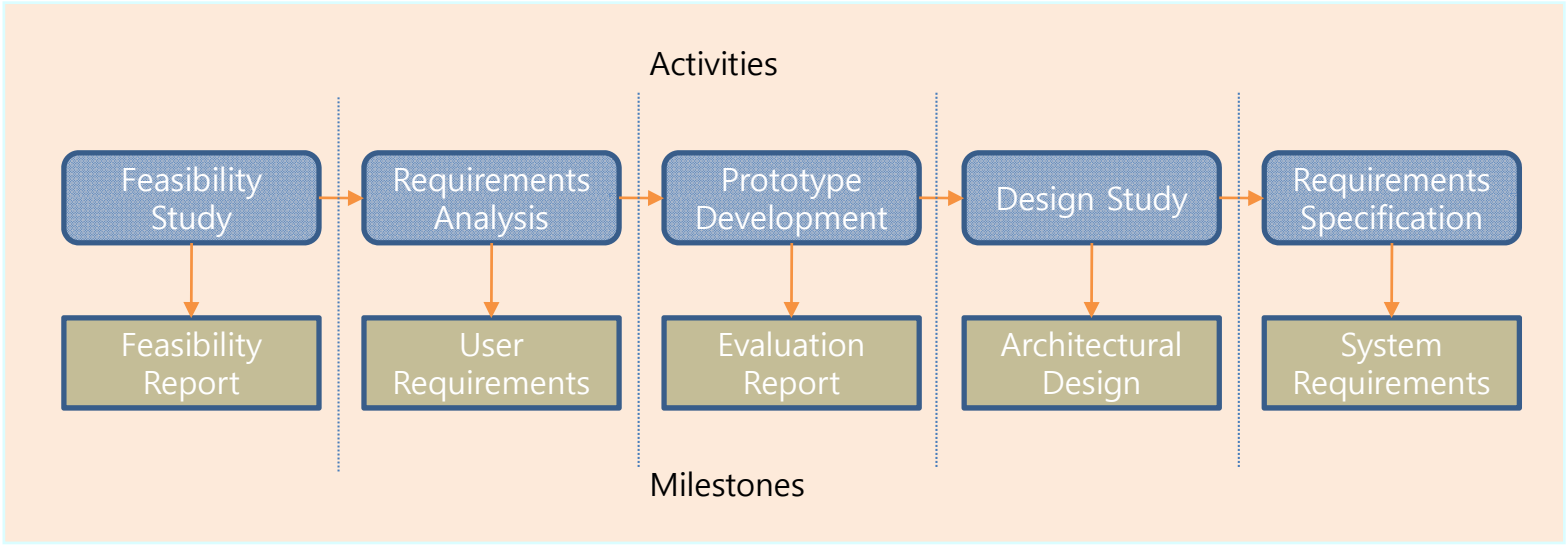
- Probably the most time-consuming project management activity
  - Continuous activity from initial concept through to system delivery
  - Plans must be regularly revised as new information becomes available.
- Various different types of plan may be developed to support main software project plan.

Plan	Description
Quality Plan	Describes the quality procedures and standards that will be used in a project. See Chapter 27.
Validation Plan	Describes the approach, resources and schedule used for system validation. See Chapter 22.
Configuration Management Plan	Describes the configuration management procedures and structures to be used. See Chapter 29.
Maintenance Plan	Predicts the maintenance requirements of the system, maintenance costs and effort required. See Chapter 21.
Staff Development Plan	Describes how the skills and experience of the project team members will be developed. See Chapter 25.

Project Planning Process

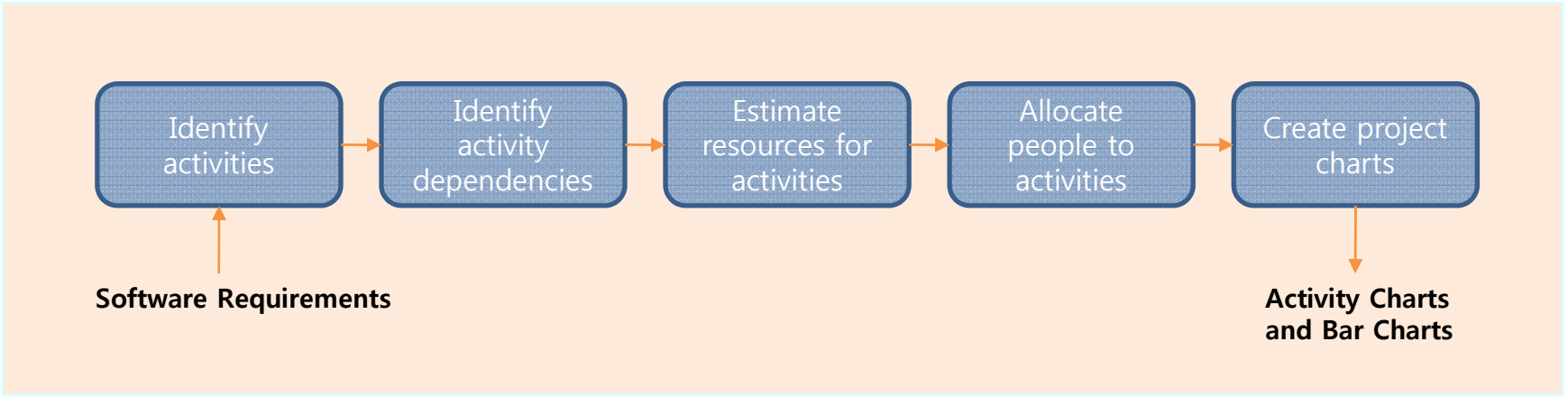
- Activities: produce tangible outputs for management to judge progress
- Milestones : end-point of a process activity
- Deliverables : project results delivered to customers

- Waterfall process allows straightforward definition of progress milestones.

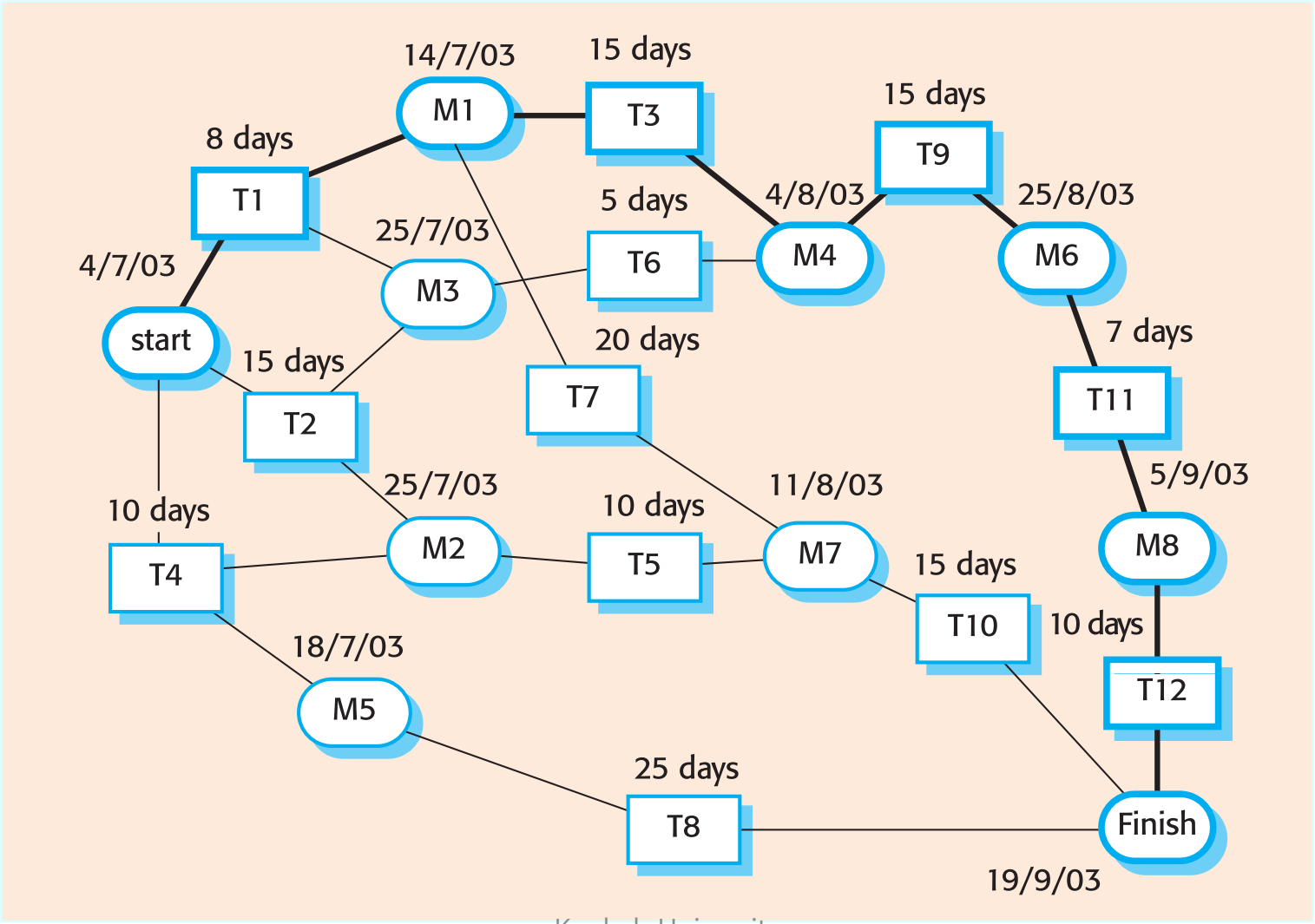


Project Scheduling Process

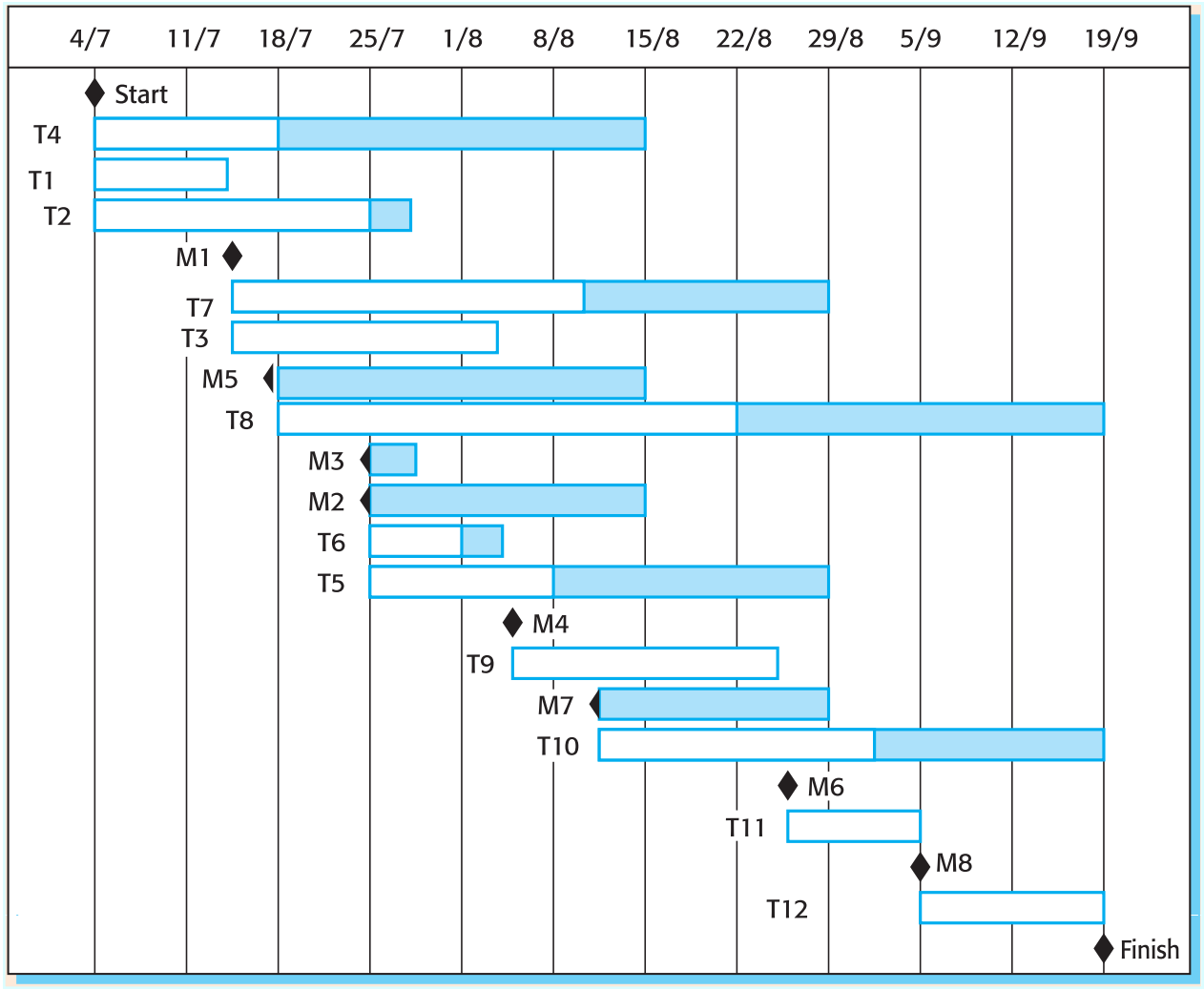
- Split project into tasks and estimate time and resources required to complete each task.
  - Organize tasks concurrently to make optimal use of workforce.
  - Minimize task dependencies to avoid delays caused by one task waiting for another to complete.
- Depend on project manager's intuition and experience.



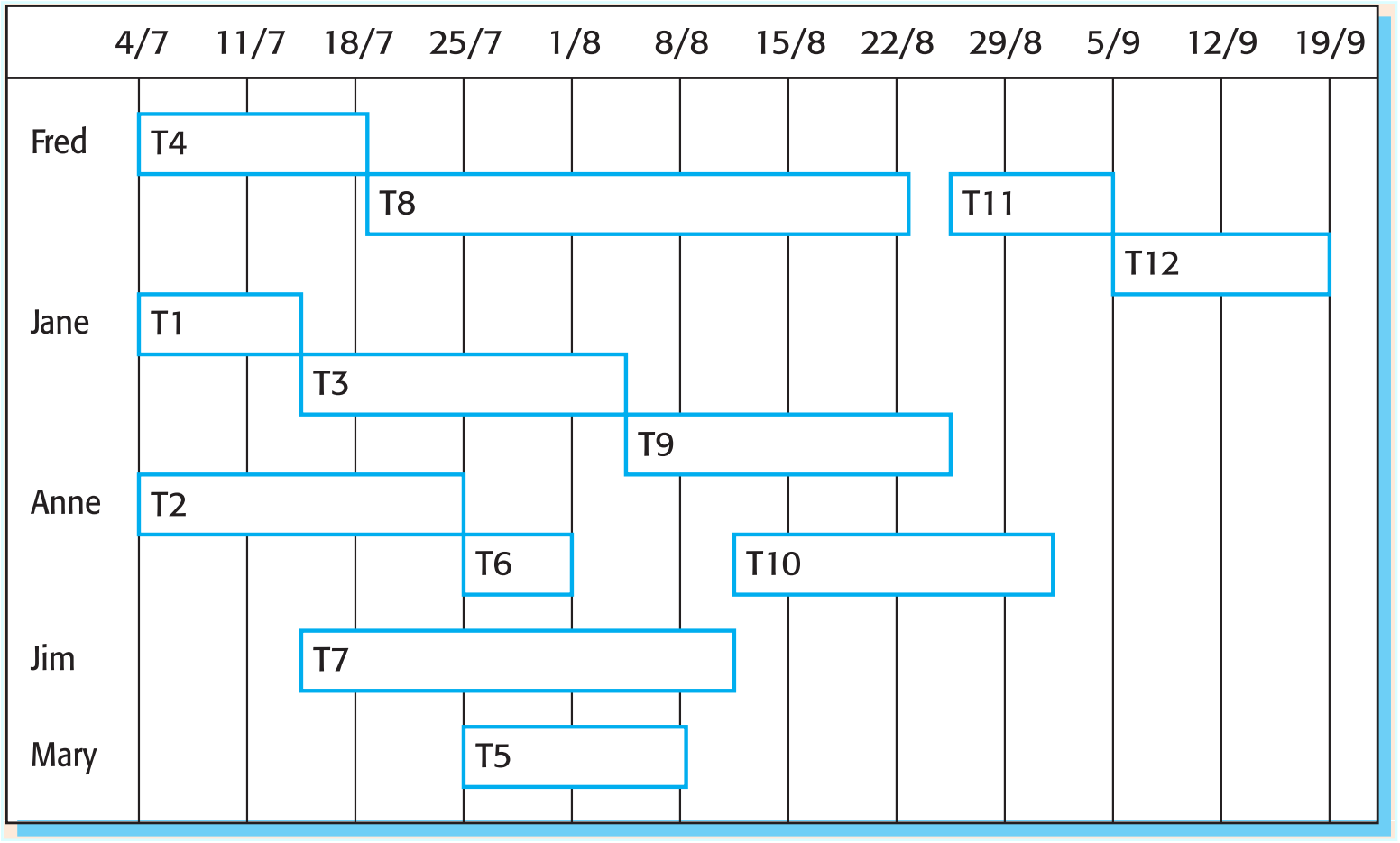
Activity Network



Activity Timeline



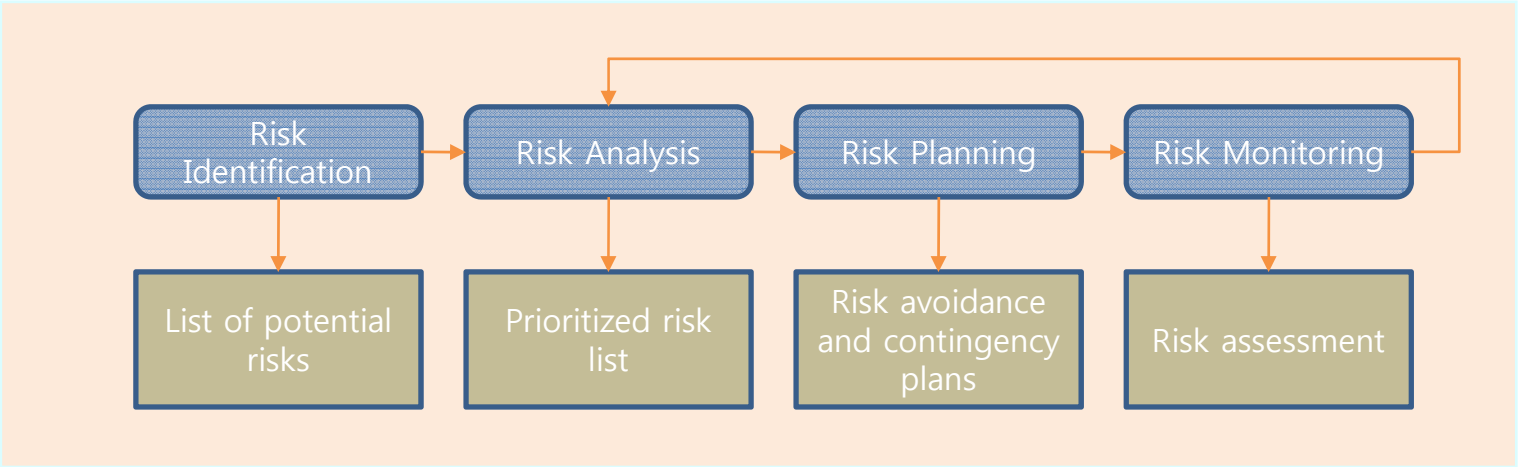
Staff Allocation



Risk Management

- Concerned with identifying risks and drawing up plans to minimize their effect on a project.
- A risk is a probability that some adverse circumstance will occur
  - Project risk affects schedule or resources.
  - Product risk affects quality or performance of the software being developed.
  - Business risk affects the organization developing or procuring the software.

- Risk management process



Summary

- Good project management is essential for project success.
- Managers have diverse roles but their most significant activities are planning, estimating and scheduling.
- Project scheduling involves preparing various graphical representations showing project activities, their durations and staffing.
- Risk management is concerned with identifying risks which may affect the project, and planning to ensure that these risks do not develop into major threats.