Software Requirements Engineering (SWEG3104)

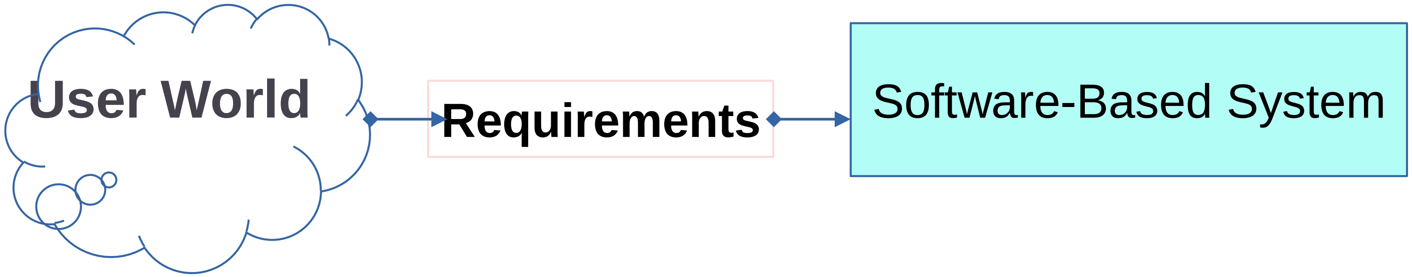
Instructor: Behailu Getachew (Asst. Prof.)

Chapter 1- Introduction to Requirement Engineering

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ Definitions:

* Requirements (Ian Sommerville, 2016): A statement identifying a capability, physical characteristic, or quality factor that meet the stakeholders' need between the user world and software-based system.



* What are "Requirements"?

- A requirement is:
  - capturing the purpose of a system.

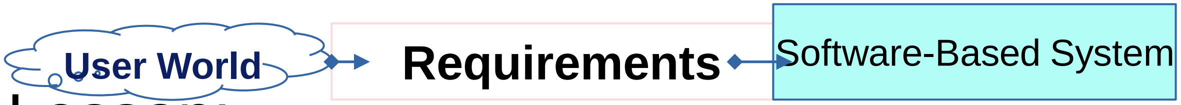
User World
Requirements
Software-Based System

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ Definitions:

* What are “Requirements” ...?

- An expression of the ideas to be embodied in the system or application under development.
- A statement about the proposed system that all stakeholders agreement must be made true in order for the customer’s problem to be adequately solved.
  - Short and concise piece of information.
  - Says something about the system.
  - All the stakeholders have agreed that it is valid.
  - It helps solve the customer’s problem.



User World

Requirements

Software-Based System

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ Definitions:

* What are “Requirements” ...?

According to IEEE 830-1993:

- A **requirement** is defined as:
  - A condition or capability needed by a user to solve a problem or achieve an objective.
  - A condition or a capability that must be met or possessed by a system … to satisfy a contract, standard, specification, or other formally imposed document …

* What is a requirement Engineering (RE)?

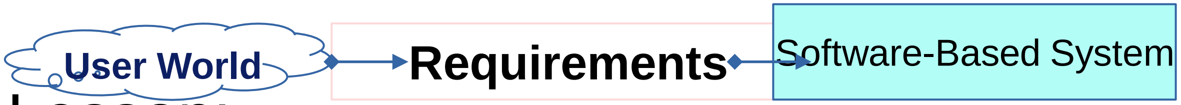
- covers all of the activities involved in discovering, documenting, and maintaining a set of requirements for a computed-based system.

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ Definitions:

* What is a requirement Engineering (RE) ...?

- RE is concerned with identifying the purpose of a software system… and the contexts in which it will be used.
  - How/where the system will be used.
  - Big picture is important.
- captures real world needs of stakeholders affected by a software system and expresses them as artifacts that can be implemented by a computing system.
  - Bridge to design and construction.
  - How to communicate and negotiate?
  - Is anything lost in the translation between different worlds?

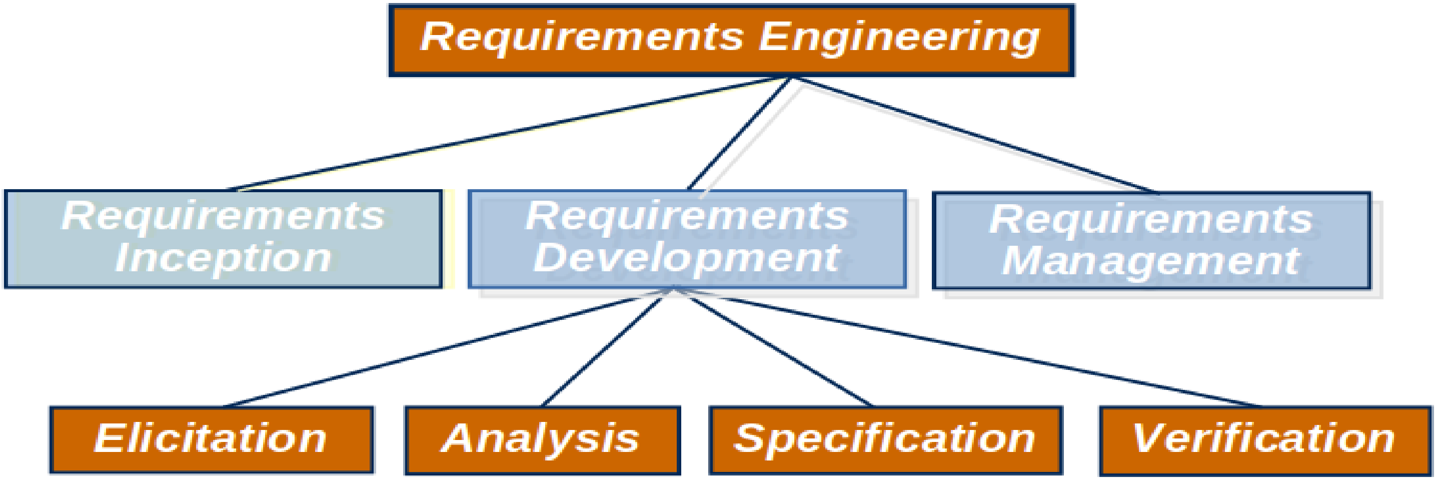


Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ Definitions:

* ‘engineering’ implies that systematic and repeatable techniques should be used to ensure that system requirements are complete, consistent, relevant, etc.

* Requirements Engineering Activities



Source: Larry Boldt, Trends in Requirements Engineering People-Process-Technology, Technology Builders, Inc., 2001

User World

Requirements

Software-Based System

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ Definitions:

* RE Activities …

- **Inception**
  - Start the process (business need, market opportunity, great idea, ...), business case, feasibility study, system scope, risks, etc.
- **Requirements elicitation**
  - Requirements discovered through consultation with stakeholders.
- **Requirements analysis and negotiation**
  - Requirements are analyzed and conflicts resolved through negotiation.
- **Requirements specification**
  - A precise requirements document is produced.

User World
Requirements
Software-Based System

Week 1-2 Lesson:
Introduction to Requirements Engineering

* RE Activities …

- **Requirements validation**

  ○ The requirements document is checked for consistency and completeness.

- **Requirements management**

- ○ Needs and contexts evolve, and so do requirements!

* Are the requirements important?

- The principle problem areas in software development and production are the requirement specification and the management of customer requirement.

- Difficulties with requirements are the key root-cause of the safety-related software errors that have persisted until integration and system testing.

User World
Requirements
Software-Based System

Week 1-2 Lesson:
Introduction to Requirements Engineering

* General Problems with the Requirements Process:

- Lack of the right expertise (software engineers, domain experts, etc.).

- Initial ideas are often **incomplete**, wildly **optimistic**, and firmly entrenched in the minds of the people leading the **acquisition process.**

- **Difficulty** of using **complex tools** and **diverse methods** associated with **requirements gathering** may **negate** the anticipated benefits of a **complete** and **detailed approach.**

* Example: Data Evidences for General Problems

- NIST (National Institute of Standards and Technology, 2002):
  - has published a comprehensive (309 pages) and very interesting report on project statistics and experiences based on data from a large number of software projects:
    - 70% of the defects are introduced in the specification phase.
    - 30% are introduced later in the technical solution process.
    - Only 5% of the specification inadequacies are corrected in the specification phase.
    - 95% are detected later in the project or after delivery where the cost for correction on average is 22 times higher compared to a correction directly during the specification effort.

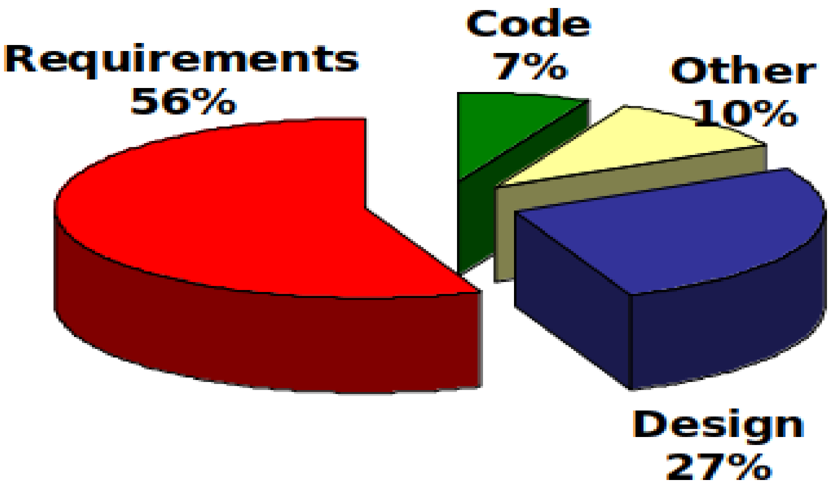
Week 1-2 Lesson:
Introduction to Requirements Engineering

* General Problems ...

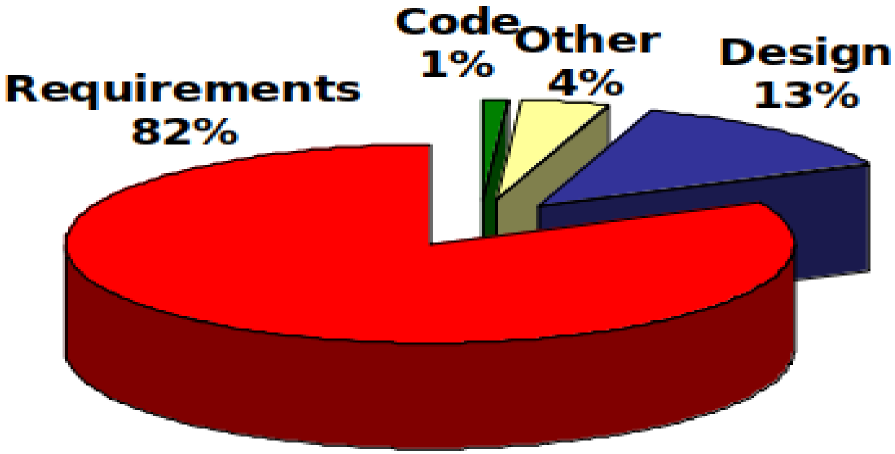
* Why focus on requirements ?

- The NIST report concludes that extensive testing is essential, however testing detects dominating specification errors late in the process.

- Distribution of Defects



- Distribution of Effort to Fix Defects



Week 1-2 Lesson:
Introduction to Requirements Engineering

* General Problems ...

* If the requirements are **wrong**,

- the system may be delivered **late** and cost **more than** originally expected.
- the customer and end-users may **not** satisfied with the system.
- they may **not** use its facilities or may even decide to scrap it altogether.

* If a system **continues in use**, the costs of maintaining and evolving the system are **very high**.

Week 1-2 Lesson:
Introduction to Requirements Engineering

* General Problems ...

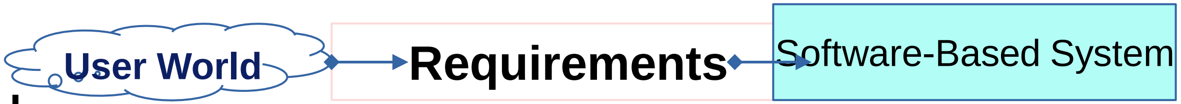
✧ * Difficulties with requirements:

- Stakeholders **do not** know what they want from a new system.
- It is very **difficult to imagine** how future systems might work.
- Businesses operate in a **rapidly** changing environment so their requirements for system support are **constantly changing.**
- **Multiple stakeholders** with **different goals** and **priorities** are involved in the **requirements engineering process.**

* General Problems ...

✧ * Difficulties with requirements ...

- System stakeholders do **not** have **clear ideas** about what they need.
- They can **only** describe their requirements in a **vague** and **ambiguous** way.
- Requirements are often **influenced** by **political** and **organization** factors that stakeholders will **not** admit to publicly.



User World

Requirements

Software-Based System

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ *Role in SDLC:

● A software process model (or SDLC) is a simplified representation of a software process.

- SDLC is a systematic and step by step approach to develop a software.
- Describes **different phases** involved in the development process.
- Detailed **roles** involved in each phase.

✧ *Role in SDLC ...

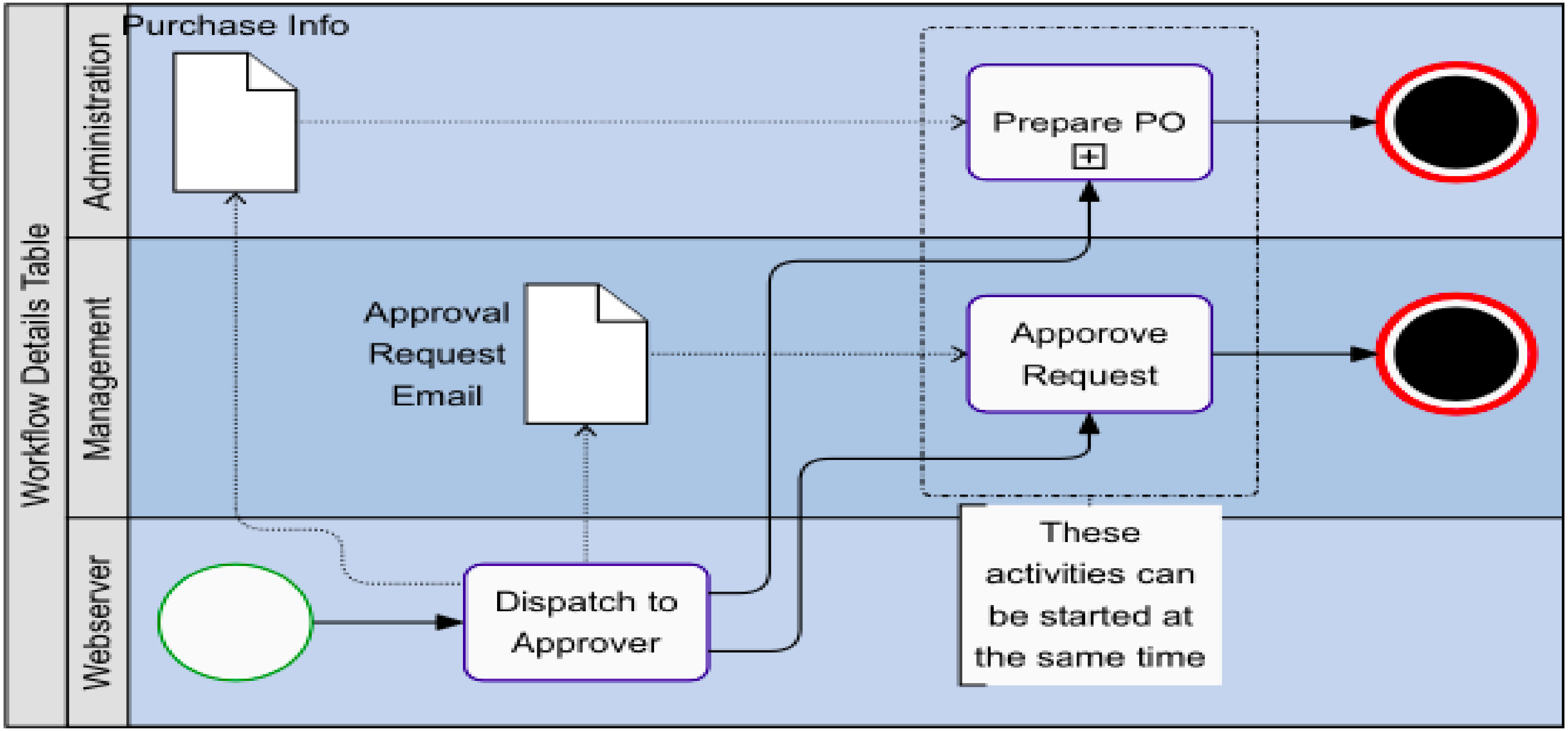
- Each phase/process model represents a process from a **particular perspective** and thus **only** provides **partial** information about that process.

- “For example, a **process activity model** shows the **activities** and their **sequence** but may **not show** the **roles** of the **people involved** in these activities.”

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ *Role in SDLC ...

- Example: Partial view of purchase order (PO) activity.



User World

Requirements

Software-Based System

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ *Role in SDLC ...

- Four fundamental activities that are part of all software development processes:
  - Specification ● Design & Implementation
  - Validation ●Evolution

- **Specification**: defining what the software should do.
- **Design** and **Implementation**: defining the software and data organization and implementing the system.
- **Validation:** testing the system for bugs and to check it meets its requirements.
- **Evolution:** changing the system **after** it has gone into use.

Week 1-2 Lesson: Introduction to Requirements Engineering

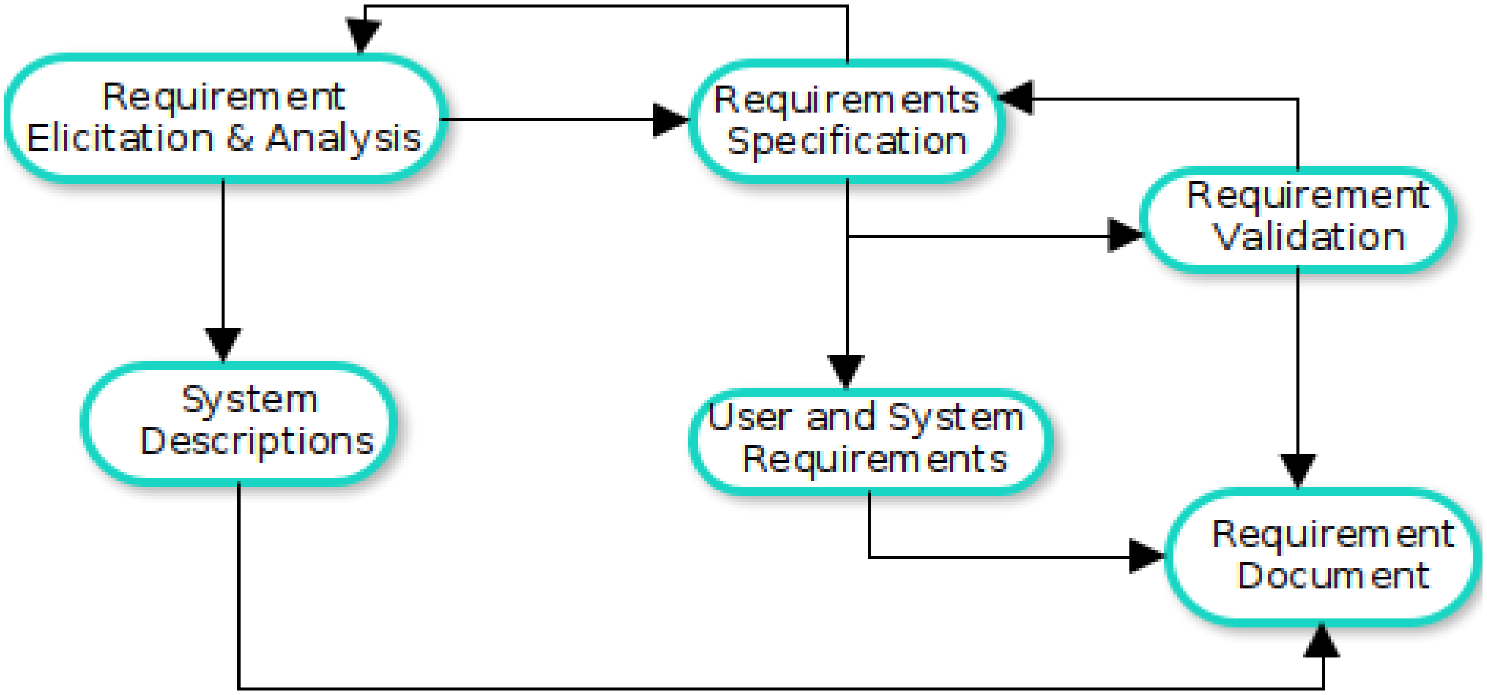
✧ *Role in SDLC ...

- The four basic processes activities of specification, development, validation, and evolution are **organized differently in the development processes.**

- For example, in the **waterfall model**, they are **organized in sequence**, whereas in the **incremental development** they are **inter-leaved**.

Week 1-2 Lesson:
Introduction to Requirements Engineering

* Components of the requirements engineering process.



Requirement Elicitation & Analysis

Requirements Specification

Requirement Validation

System Descriptions

User and System Requirements

Requirement Document

User World

Requirements

Software-Based System

Week 1-2 Lesson:
Introduction to Requirements Engineering

* There are three main activities in the requirements engineering process:

- **Requirements elicitation and analysis:**
  - is the process of **deriving** the system requirements through **observation** of existing systems, **discussions** with potential users and procurers, task analysis, and so on.
- **Requirements specification:**
  - is the activity of **translating** the information gathered during **requirements analysis** into a document that **defines a set of requirements.**

Week 1-2 Lesson:
Introduction to Requirements Engineering

* There are three main activities …

- **Requirements validation:**
  - This activity checks the requirements for realism, consistency, and completeness.
  - During this process, errors in the requirements document are inevitably discovered.
  - It must then be modified to correct these problems.

Week 1-2 Lesson:
Introduction to Requirements Engineering

* There are three main activities …

- Requirements analysis continues during definition and specification, and **new requirements come to light** throughout the process.

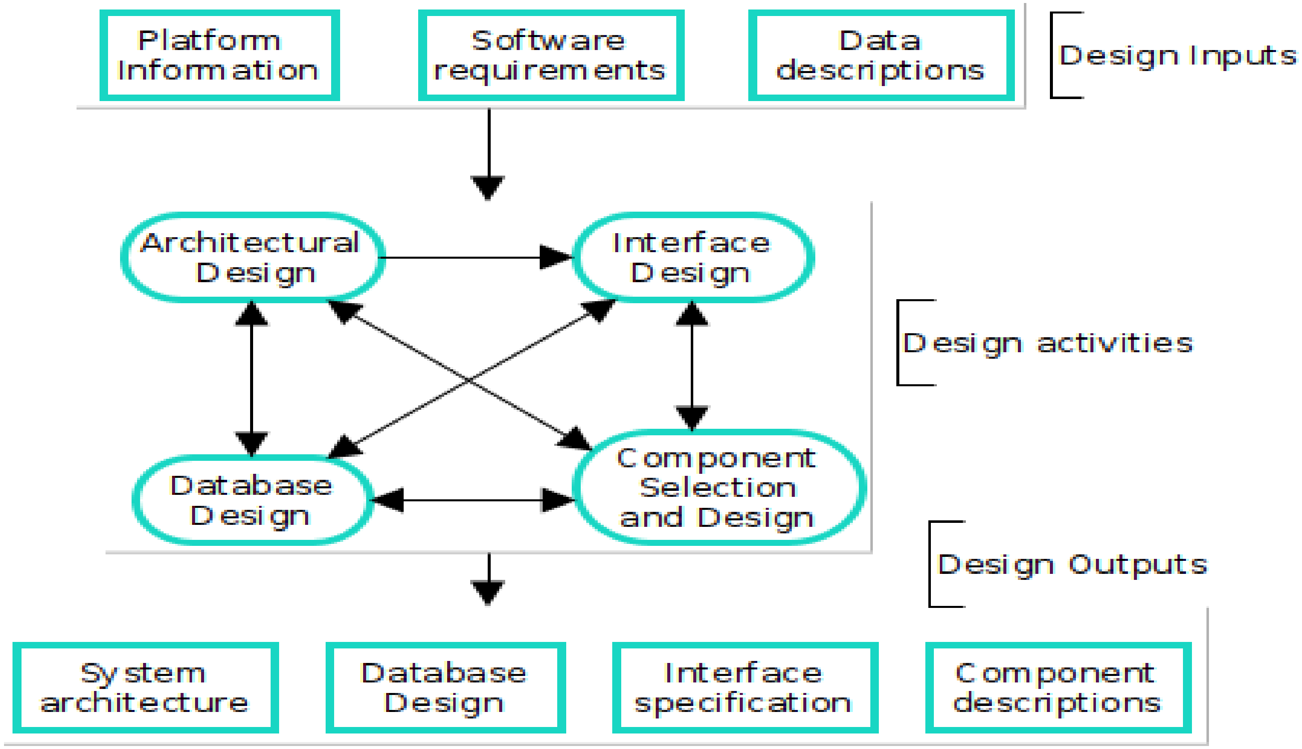
- Therefore, the activities of analysis, definition, and specification are interleaved.

- Implementation involves adding detail to the design and programming the system.

- Design and implementation are closely related and are normally inter-leaved activities.

Week 1-2 Lesson:
Introduction to Requirements Engineering

* A **general model** of the design process.



Week 1-2 Lesson:
Introduction to Requirements Engineering

* A **general model** of the design process …

- The activities in the **design process vary**, depending on the type of system being developed. For example, **real-time systems** require an additional stage of timing design.

  - **Architectural design:**
    - capability to identify the overall structure of the system, the principal components or subsystems or modules, their relationships, and how they are distributed.

Week 1-2 Lesson:
Introduction to Requirements Engineering

* A **general model** of the design process …

- **Database design:**
  - describes the system data structures and how these are to be represented in a database.
  - the work here depends on whether an existing database is to be reused or a new database is to be created.

- **Interface design:**
  - defines the interfaces between system components.
  - This interface specification must be unambiguous.
  - With a precise interface, a component may be used by other components without them having to know how it is implemented.

Week 1-2 Lesson:
Introduction to Requirements Engineering

* A **general model** of the design process …

- **Component selection and design:**
  - give to search for reusable components and, if no suitable components are available, design new software components.
  - The design at this stage may be a simple component description with the implementation details left to the programmer.
  - These activities lead to the design outputs, for **critical systems,** the outputs of the design process are **detailed design documents** setting out precise and accurate descriptions of the system.

Week 1-2 Lesson:
Introduction to Requirements Engineering

* A **general model** of the design process …

● If a **model-driven** approach is used, the design outputs are **design diagrams.**

● Where **agile methods** of development are used, the outputs of the design process **may not be separate specification documents** but may be represented in the **code of the program.**

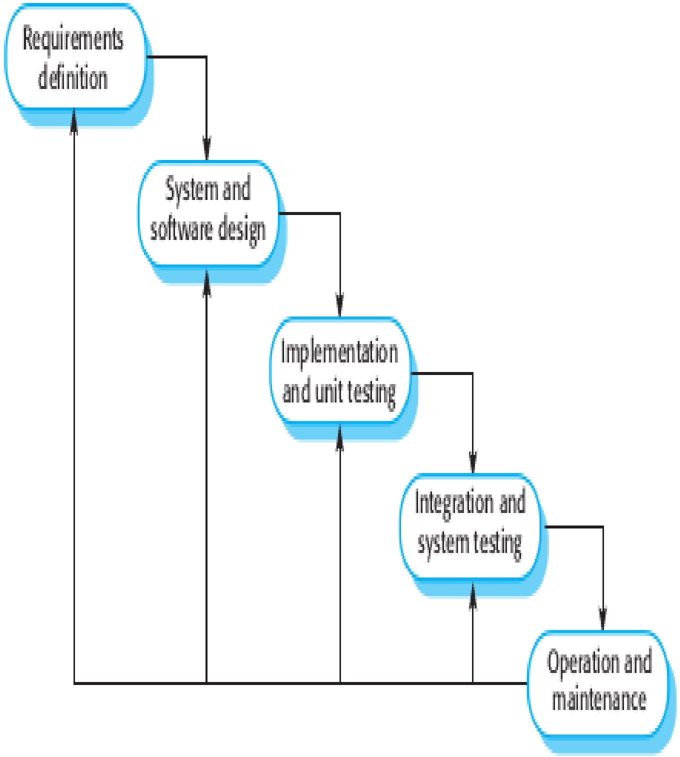
Week 1-2 Lesson:
Introduction to Requirements Engineering

- * Layered Model is a process framework that associates with three generic layers that are capable to extend into applicable desired model.
- **1. Waterfall model:**
  - takes the fundamental process activities of specification, development, validation, and evolution and represents them as **separate process phases** such as requirements specification, software design, implementation, and testing.

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ * Layered Model …

1. Waterfall model ---: In short, this model covers,


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

●**Requirements:** defines needed information, function behavior, performance and interfaces.

●**Design:** data structures, software architecture interface representations, algorithmic details.

●**Implementation:** source code, database, use documentation, testing.

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ *Layered Model …:

* The stages of the waterfall model directly reflect the fundamental software development activities:

- **Requirements analysis and definition**
  - the system's services, constraints, and goals are established by consultation with system users.
- **System and software design**
  - the systems design process allocates the requirements to either hardware or software systems.
  - It establishes an overall system architecture.
  - Software design involves identifying and describing the fundamental software system abstractions and their relationships.

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ *Layered Model …: The stages of the waterfall model …:

- **Implementation and unit testing:**
  - During this stage, the software design is realized as a set of programs or program units.
  - Unit testing involves verifying that each unit meets its specification.

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ *Layered Model …: The stages of the waterfall model …:

- **Integration and system testing:**
  - The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met.
  - After testing, the software system is delivered to the customer.

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ *Layered Model …: The stages of the waterfall model …:

- **Operation and maintenance:**
  - The system is installed and put into practical use.
  - Maintenance involves correcting errors that were not discovered in earlier stages of the life cycle,
    - improving the implementation of system units, and
    - enhancing the system's services as new requirements are discovered.

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ *Layered Model …

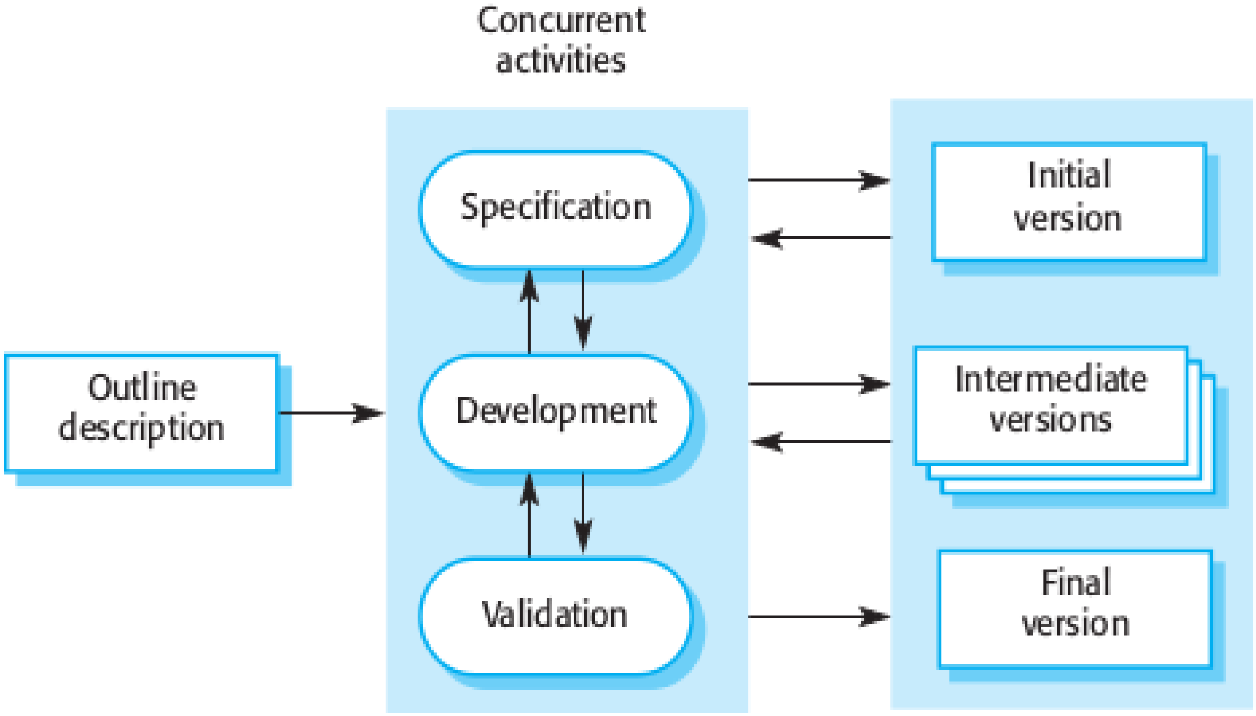
2. **Incremental development:**

- interleaves the activities of specification, development, and validation.
- The system is developed as a series of version (increments), with each version adding functionality to the previous version.

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ *Layered Model …

2. **Incremental development …**



Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ *Layered Model …

2. **Incremental development …**

- Incremental development has major advantages over the waterfall model:

  - The cost of implementing requirements changes is reduced.

  - The amount of analysis and documentation that has to be redone is significantly less than is required with the waterfall model.

✧ *Layered Model …

2. **Incremental development …**

- Incremental development has major advantages over the waterfall model …:

- It is easier to get customer feedback on the development work that has been done.

- Customers can comment on demonstrations of the software and see how much has been implemented.

- Customers find it difficult to judge progress from software design documents.

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ *Layered Model …

2. **Incremental development …**

- Incremental development has major advantages over the waterfall model …:

- Early delivery and deployment of useful software to the customer is possible,
- even if all of the functionality has not been included.

- Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Week 1-2 Lesson:
Introduction to Requirements Engineering

✧ *Layered Model …

3. **Integration and configuration:**

- relies on the availability of reusable components or systems.
- The system development process focuses on configuring these components for use in a new setting and integrating them into a system.

User World → Requirements → Software-Based System

Week 1-2 Lesson:
Introduction to Requirements Engineering

- ***Group Mock-up Practice:*** *Develop the requirement engineering activity models based on the previous and the current courses seen in the lessons.*