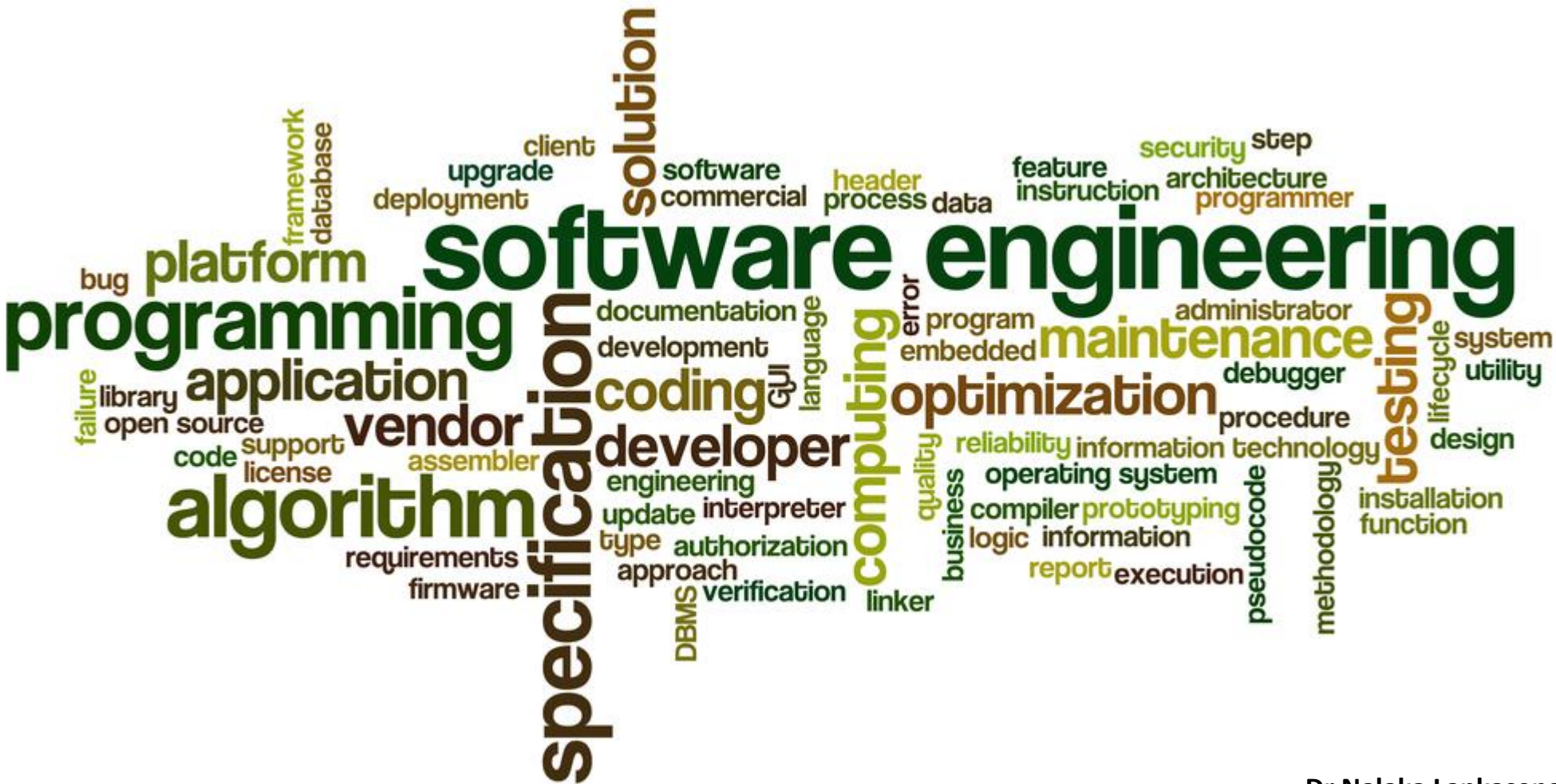


Dr Nalaka Lankasena
Faculty of Technology
University of Sri Jayewardenepura

Software Process Models

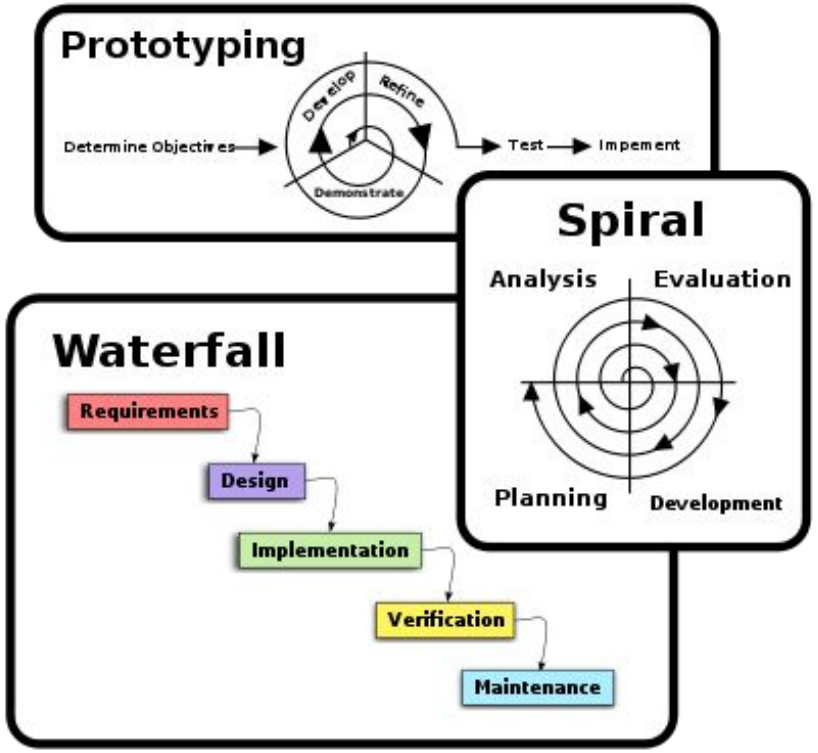
- Reference: Sommerville, I., *Software Engineering.* 10th ed., Addison Wesley.

Content

- Software process models
- Plan-driven and agile processes
- Software process models
  - Waterfall model
  - Incremental development
  - Reuse-oriented software engineering

The software process

- A software process (also known as software methodology) is **a set of related activities that leads to the production of the software**. These activities may involve the **development of the software from the scratch, or, modifying an existing system**.



..contd…

..contd.. The software process

Fundamental software engineering processes/methodologies?
- **Software specification**
- **Software design and Implementation**
- **Software validation**
- **Software evolution/maintenance**

- There are many different software processes/methodologies

but all must include:

- **Specification** – The functionality of the software and constraints on its operations must be defined (defining what the system should do);
- **Design and implementation** – defining the organization of the system and implementing the system;
- **Verification and validation** – The software must conforms to it's specification and meets the customer needs (checking that it does what the customer wants)
- **Evolution/maintenance** – changing the system in response to changing customer needs.

- A software **process model** is an **abstract representation** of a process. It presents a **description of a process from some particular perspective**.

Software process descriptions

- When we describe and discuss processes, we usually talk about the **activities in these processes** such as **specifying a data model, designing a user interface**, etc. and the ordering of these activities.
- Process descriptions may also include:
  - **Products**, which are the **outcomes** of a process activity;
    - Ex. Activity -architectural design -> outcome - model of the software architecture
  - **Roles**, which reflect the **responsibilities of the people** involved in the process;
    - Ex. Project manager, configuration manager, programmer etc.
  - **Pre- and post-conditions**, which are **statements** that are true before and after a process activity has been enacted or a product produced.
    - Ex. Before architectural design begins, a precondition may be that all requirements have been approved by the customer; after this activity is finished post condition might be that the UML models describing the architecture have been reviewed

..contd..

Example: Process of testing a single module in a large system

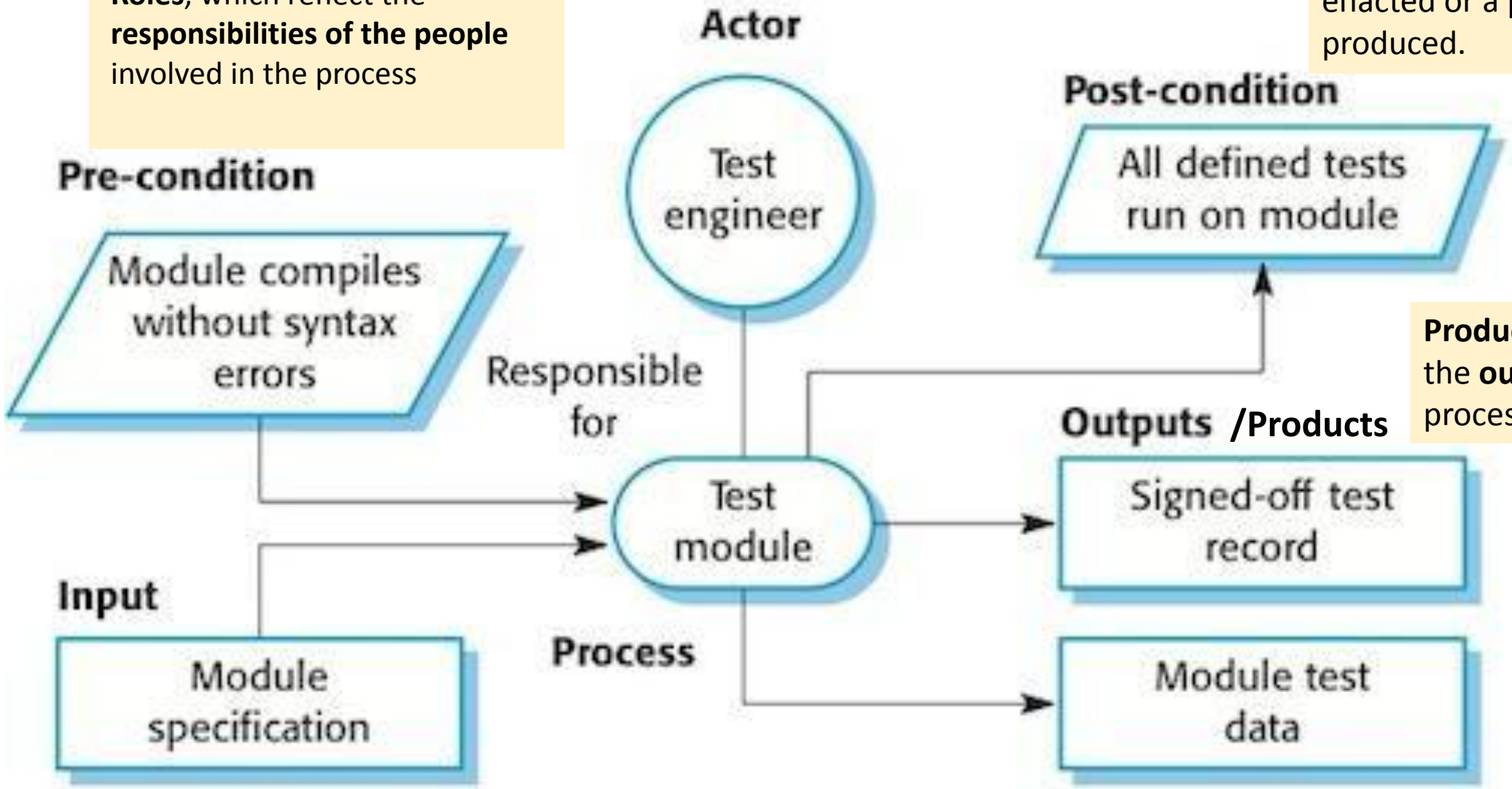
- Process?
- Actors?
- Roles?
- Pre-condition?
- Post-condition?
- Input?
- Output/products?

..contd.. Software process descriptions

Example: Process of testing a single module in a large system

Roles, which reflect the **responsibilities of the people** involved in the process

Pre- and post-conditions, which are **statements** that are true before and after a process activity has been enacted or a product produced.



Products, which are the **outcomes** of a process activity;

Plan-driven and agile processes

- Software processes/methdologies can be categorized as either **plan-driven or agile processes**

- Processes/methdologies have evolved to take the **advantage of the capabilities of the people** in an organization and the **specific characteristics of the systems that are being developed**.

Examples

- **Critical systems** – a very structured development process is required
- **Business systems** with rapidly changing requirements, a less formal, flexible process is like to be more effective

..contd..

..contd.. Plan-driven and agile processes

Plan-driven processes

- All of the **process activities are planned in advance** and progress is measured against this plan
- Examples
  - **Safety-critical systems** - Failure results in loss of life, injury or damage to the environment
    - Chemical plant protection system
  - **Mission-critical systems** - Failure results in failure of some goal-directed activity
    - Spacecraft navigation system
  - **Business-critical systems** - Failure results in high economic losses
    - Customer accounting system in a bank

..contd.. Plan-driven and agile processes

Agile = able to move quickly and easily

Agile processes

- **Planning is incremental** and it is easier to change the process to reflect changing customer requirements.
- Business systems with rapidly changing requirements, a less formal, flexible process is likely to be more effective

Each approach (i.e. plan driven or agile) is suitable for different type of software. In practice, most practical processes include **elements of both plan-driven and agile approaches**. There are no right or wrong software processes.

Software process models

- A software process model is a **simplified representation** of a software process.
- Each **process model represents** a process from a **particular perspective,** and thus **provides only partial** information about the process.
  - Example. Process activity model shows the activities and their sequence but may not show the roles of the people involved in these activities

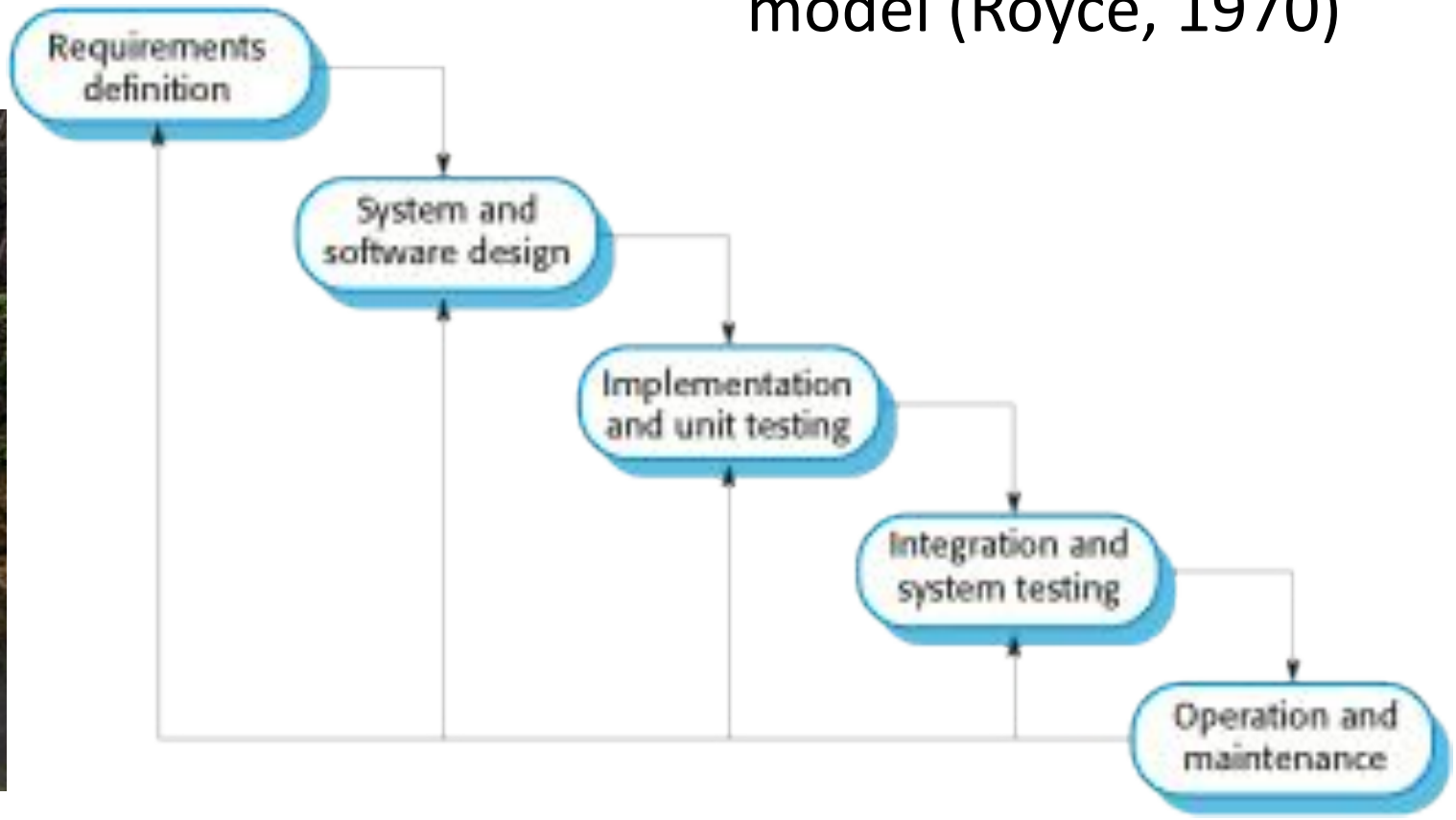
.. Contd..

..contd… Software process models

- **Waterfall model**
  - A **Plan-driven** model.
  - **Separate and distinct phases** of specification and development.
- **Incremental development**
  - Specification, development and validation are alternating between them.
  - Plan-driven or agile.
- **Reuse-oriented software engineering**
  - The system is assembled from existing components.
  - Plan-driven or agile.
- In practice, most large systems are developed using a process that incorporates elements from all of these models.

Waterfall model

The earliest software development model (Royce, 1970)



Waterfall model phases

- Developments occur in a **linear fashion**. Because of the cascade from one phase to another, this model is known as the 'waterfall model' or software lifecycle.
- **Plan and driven** process model – One must plan and schedule all of the process activities before starting work on them.
- **Assumptions** behind the model:
  - a phase takes place **in sequence** to another.
  - **each activity is completed** before the next starts.

.. Contd..

..contd.. Waterfall model phases

Separate identified phases in the waterfall model:

- Requirements gathering, analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance

..contd..

..contd.. Waterfall model phases

Requirements gathering, analysis and definition

- The **system's services, constraints, and goals** are established by consultation **with system users**
- Define **system specification**.

System and software design

- This involves identifying and describing the **fundamental software system abstractions and their relationships**
- The system design process **allocates the requirements** to either **hardware or software systems by establishing an overall system architecture**.

..contd..

..contd.. Waterfall model phases

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**.

Integration and system testing

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

..contd..

..contd.. Waterfall model phases

Operation and maintenance (/software evolution)

- Normally this is the **longest** life cycle phase.
- The system is **installed and put into practical** use.
- Maintenance involves **correcting errors** which 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.

..contd..

Features of waterfall model

- According to the concept:
  - **Tangible products** (various documents) at the end of every phases → good to measure progress.
  - Each phase produces (result of each phase) documents that are:
    - Verified and validated.
    - Assumed to be complete.
- In principle the following face should not start until the previous phase has finished.
- In practice these **stages overlap and feed information to each other**
  - During design, problems with requirements are identified
  - During coding design problems are found

..contd..

Waterfall model problems

- The main drawback of the waterfall model is the difficulty of **accommodating changes after the process is done**. In principle, a phase has to be complete before moving onto the next phase.
  - In real software process is not linear but involves feedback from one phase to another.
- **Due to cost of producing and approving documents**, after a small number of iterations it is normal to freeze parts of the development and continue with the later development stages. The **premature freezing** may mean that the system won't do what the user wants. It may also lead to **badly structured systems**.

..contd..

Waterfall model problems ..contd…

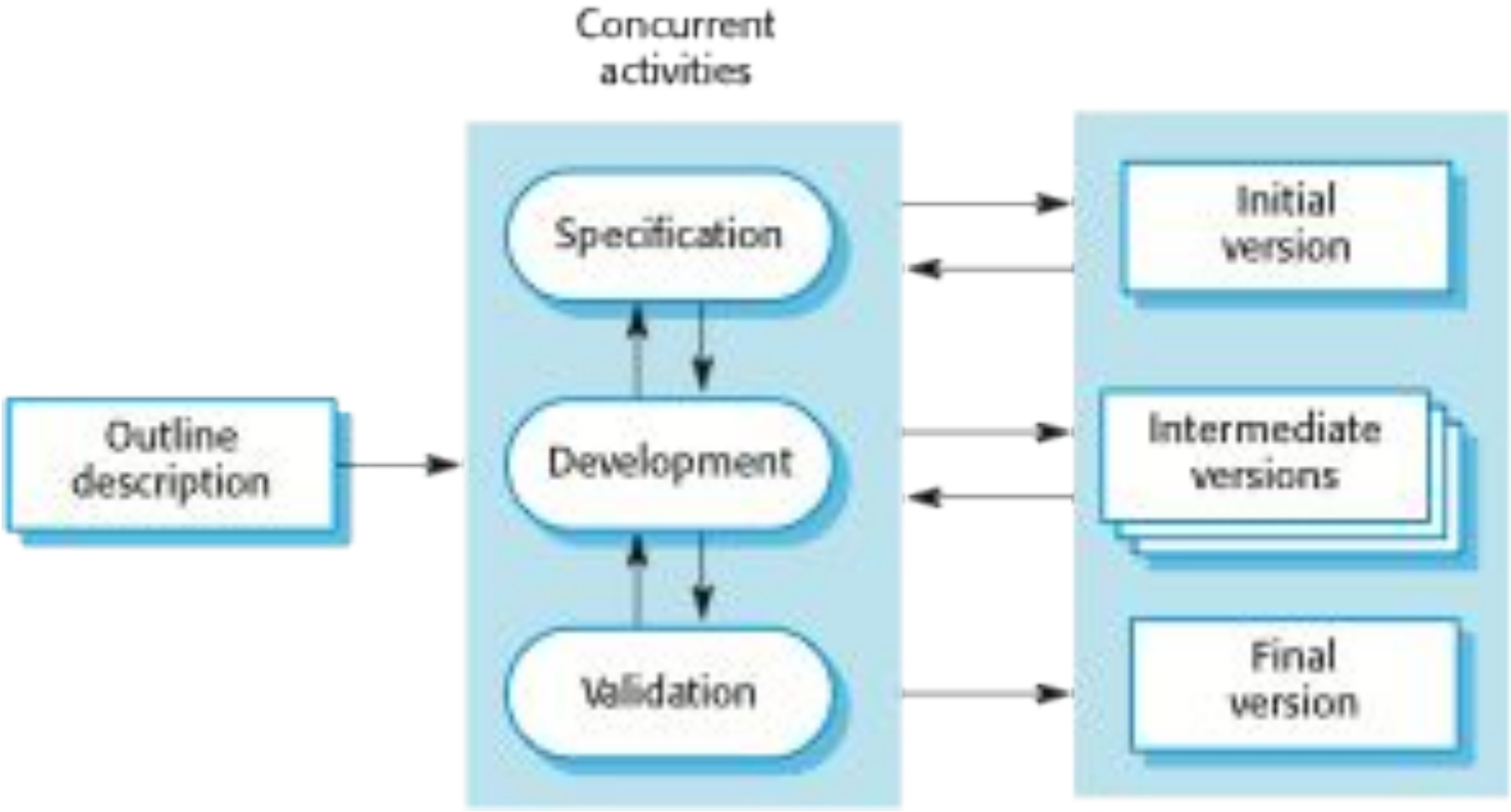
- **Inflexible partitioning** of the project into distinct stages makes it **difficult to respond to changing customer requirements**. Therefore, this model is only appropriate when;
  - **Requirements are well-understood and changes will be fairly limited** during the design process.
  - Business systems have stable requirements.

..contd…

When to use waterfall model?

- In principle, the waterfall model should only be used when the **requirements are well understood** and **unlikely to change radically** during system development
- The waterfall model is **mostly used for large systems** engineering projects where a system is developed at several sites.
  - In circumstances, the plan-driven nature of the waterfall model helps to coordinate the work.

Incremental development



…contd..

… contd.. Incremental Development

- Incremental development is based on the idea of;
  - **developing an initial implementation**,
  - provide the **version to the user for comments** and
  - **evolving it through several versions** until an adequate system has been developed.
- **Specification, development and validation activities are concurrent/mixed** rather than separate with rapid feedback across activities.

…
contd..

… contd.. Incremental Development

- Each increment or version of the system **incorporates some of the functionality needed by the customer**.
- **Incremental delivery and deployment means** that the **software is used in real, operational process**.
- **Early increments** of the system include the most important/**most urgently required functionality.**

… contd..

Incremental development benefits

- The **cost of accommodating changing customer requirements** is reduced.
  - The amount of analysis and documentation that has to be done is much less than is required with 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.
- **More rapid delivery and deployment of useful software** to the customer is possible.
  - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Incremental development problems

- The process is not visible.
  - Some organizations **need regular deliverables to measure the progress**. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- **System structure tends to degrade** as new increments are added.
  - Unless time and money is spent on refactoring to improve the software, **regular change tends to corrupt its structure**. **Incorporating further software changes becomes increasingly difficult and costly**.

Appropriate use of incremental development model

- This model is usually **not suitable for large, complex, long-lifetime systems** where **different teams develop different parts of the system**.
  - **Large systems** need a **stable framework** or architecture and the responsibilities of the different teams working on parts of the system need to be clearly defined with respect to that architecture. This has to be planned in advance rather than incrementally.
- Incremental software development is suitable for most business, e-commerce, and personal systems.

Reuse-oriented software engineering

- Based on **systematic reuse** where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Process stages
  - Component analysis;
  - Requirements modification;
  - System design with reuse;
  - Development and integration.
- Reuse is now the standard approach for building many types of business system

.. Contd..

..contd..Reuse-oriented software engineering

Discuss

.contd..

..contd..Reuse-oriented software engineering

Component Analysis

- A search is made to see the **possibilities of implementing components** which matches user specification
- Usually, there's **no exact match** and the components that may be used

.contd..

..contd..Reuse-oriented software engineering

Requirements Modifications

- Requirements are analyzed using information about the components that have been discovered.
- These components are then modified to reflect the available components
- If modifications are impossible, component analysis may be re-entered to search for alternative solutions.

..contd.. Reuse-oriented software engineering

System Design with Reuse

- During this phase, the framework of the system is designed or an existing framework is reused.
- Some new software may have to be designed if reusable components are not available.

Development and integration

- Software that cannot be externally procured is developed and the components and COTS (Commercial-off-the-shelf) systems are integrated to create the new system.

.contd..

Reuse-oriented software engineering

Advantages

- **Reduce the amount of time and cost to be developed** and this has an impact in **reducing the cost and risks**.
- Faster delivery of the software

Disadvantages

- May not be able to deliver the exact user requirements
- The evolution of the system (i.e. when releasing new versions by the supplier) is not under control and may lead to compatibility and various issues


Q&A
Questions
Answers


THANK
YOU