SWE3002-42: Introduction to Software Engineering

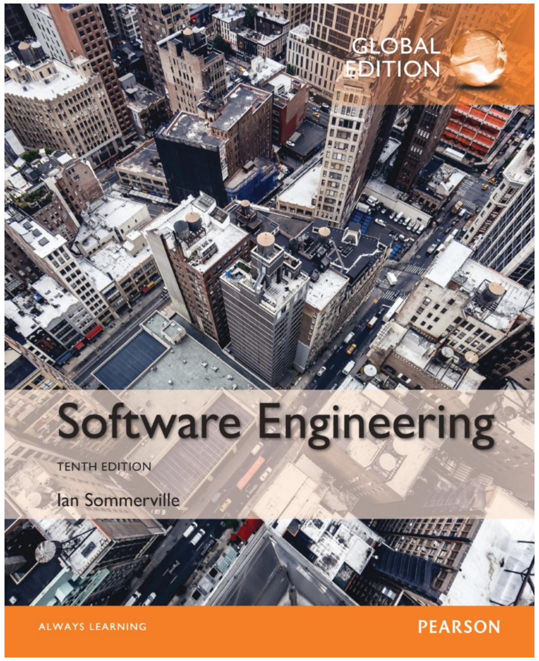
Lecture 2 – Software Processes

Sooyoung Cha

Department of Computer Science and Engineering

Chapter2. Software Processes

Topics covered



Chapter 2. (p.42 ~ p.71)

The Software process

[What is a software process?]

A set of related activities that leads to the production of a software

[4 fundamental SE activities in many different SW processes]

SW Specification	Defining the functionality of the software and constraints on its operation
SW Development	Producing the software to meet the specification
SW Validation	Checking that the software does what the customer wants
SW Evolution	Evolving the software to meet changing customer needs.

The Software process

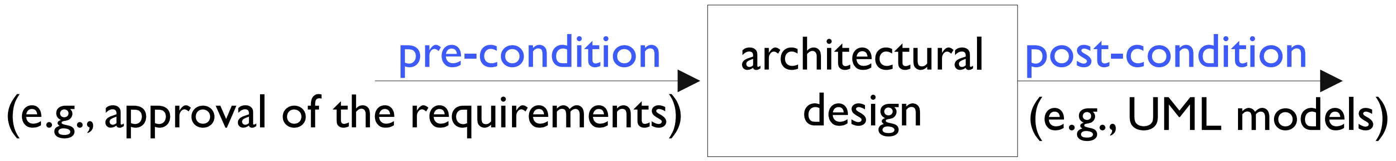
[What is a software process?]

A set of related activities that leads to the production of a software

[When describing the SW process, it is also critical to describe:]

Product & Deliverables	The outcomes of a process activity.
Roles	The responsibilities of the people involved in the process.
Pre- and post-conditions	Conditions that must hold before and after a process activity.

A Process Activity



Plan-driven and agile processes

[In plan-driven processes]

- All of the process activities are planned in advance.
- ex) Safety-critical systems

[In agile processes]

- Planning is incremental and continual during SW development.
- ex) Business systems with rapidly changing requirements

Generally, for large systems, we need to find a balance between plan-driven and agile processes.

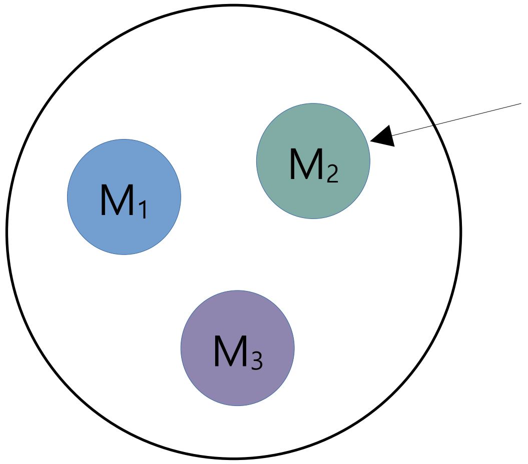
The Software process

[What is a software process model?]

- A simplified representation of a software process

(= High-level, abstract descriptions of software processes)

A Software Process Model = **A SW Development Life Cycle (SDLC model)**



A process from a particular perspective

(providing partial information about the process)

"A set of software process models"

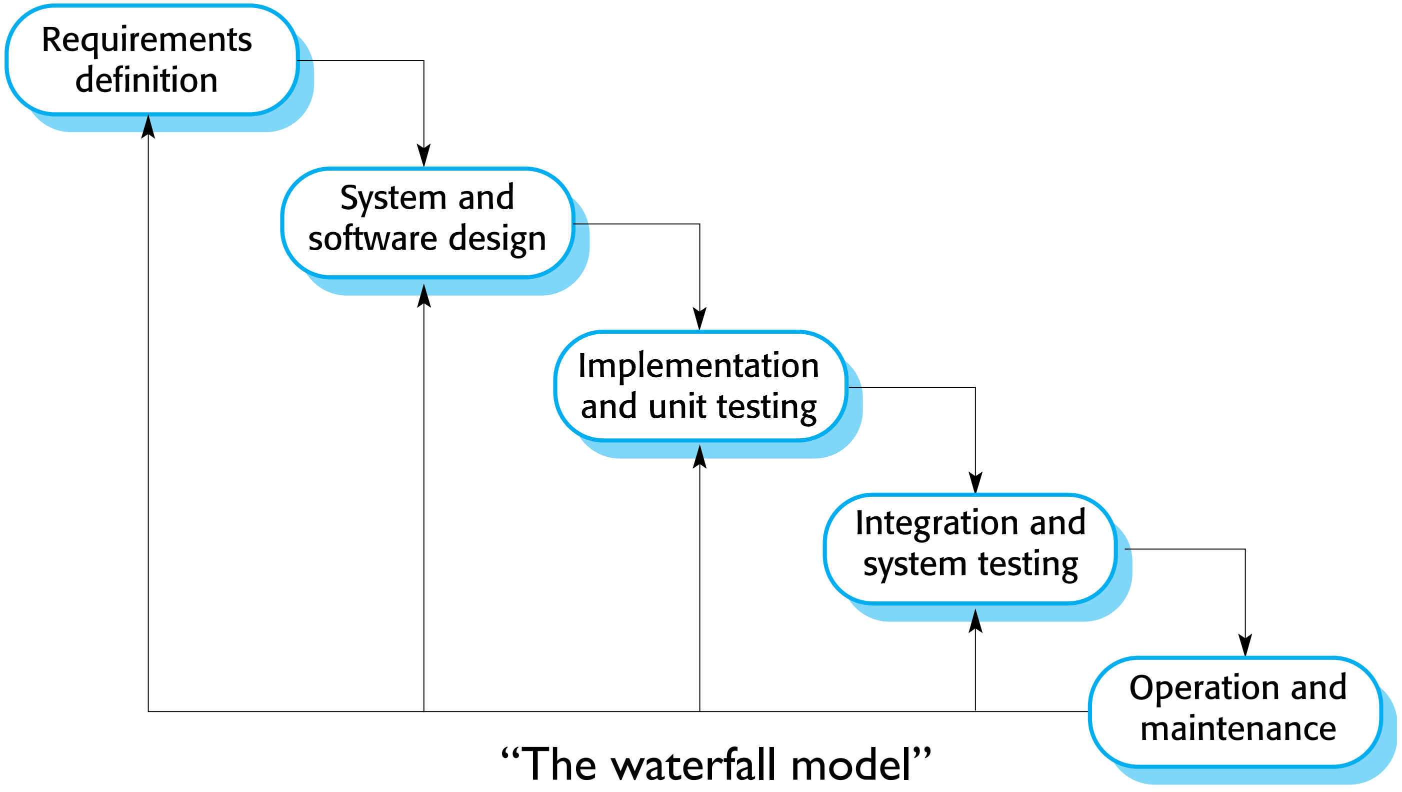
Software process models

The waterfall model	Representing the fundamental process activities as separate process phases. (e.g., a plan-driven process)
Incremental development	Interleaving the fundamental process activities. Developing as a series of versions (increments).
Integration and configuration	Relying on reusable software components and an integrating framework for the composition of the components.

In practice, most large systems are developed using a process that incorporates elements from all of these models.

The waterfall model

- **Planing all of the process activities before starting SW development.**
- **Reflecting the fundamental SW development activities directly.**

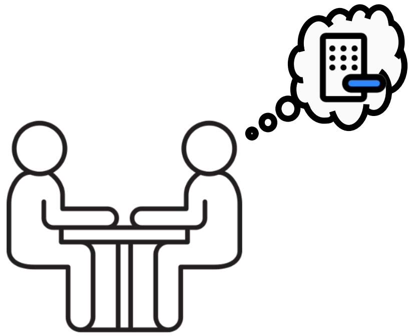

Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

“The waterfall model”

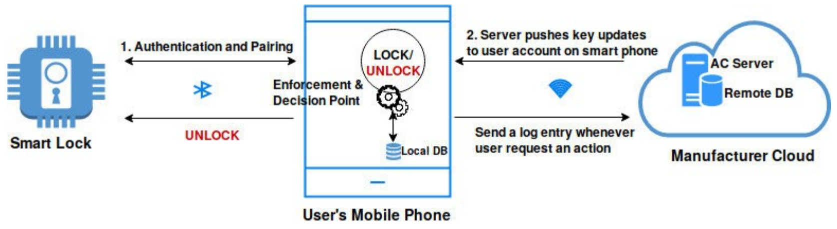
Chapter2-2. Software process models

The waterfall model

1. Requirements analysis and definition



2. System and software design



3. Implementation and unit testing

4. Integration and system testing

5. Operation and maintenance

The waterfall model

The next phase should not start until the previous phase has finished.

$\downarrow$

The model is not appropriate when software requirements change quickly.

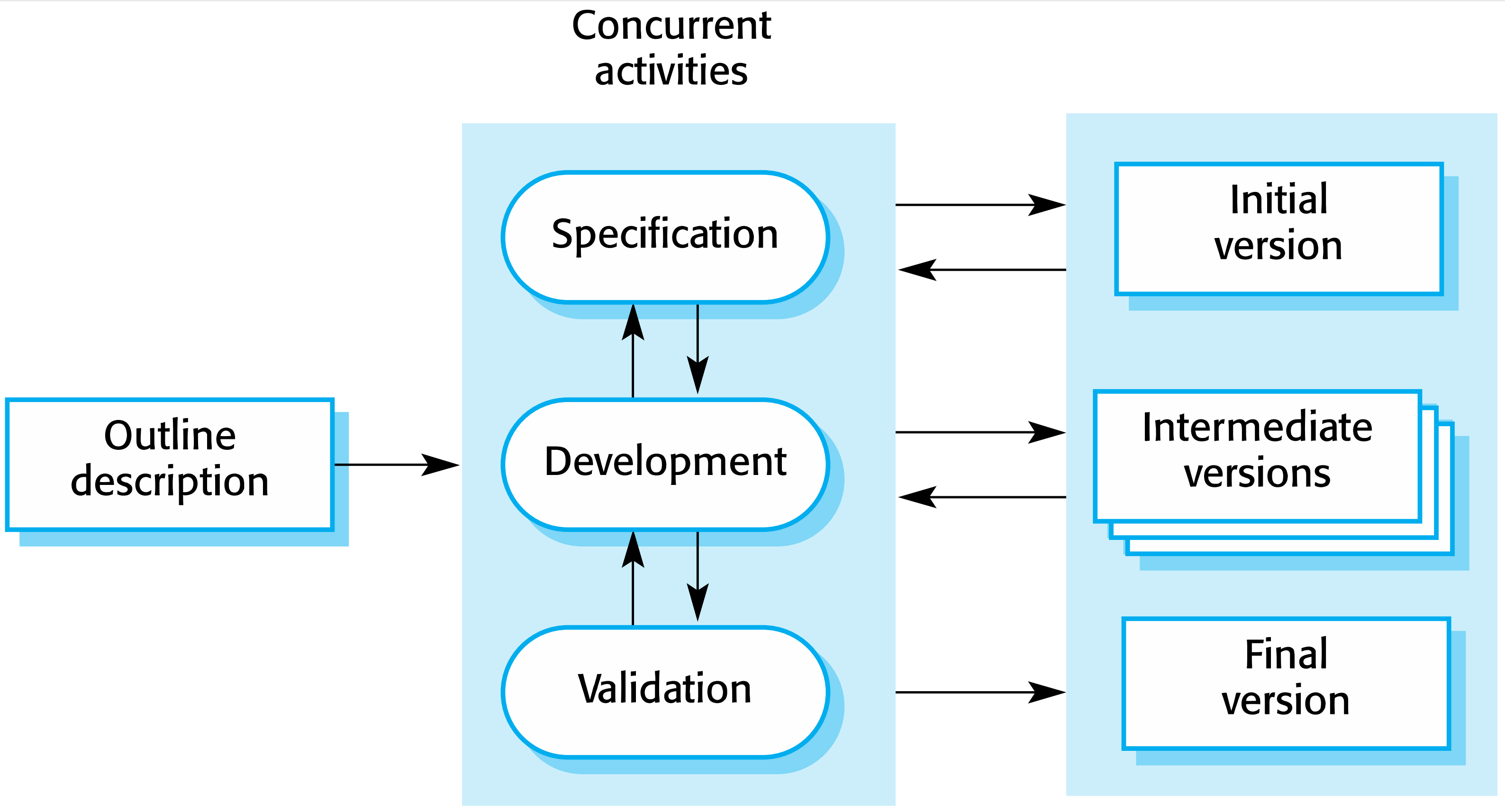
$\downarrow$

The model is only appropriate for some types of system.

Embedded systems	Critical systems

Incremental development

- **Three activities are interleaved rather than separate.**
- **Developing an initial implementation, getting feedback from users, and evolving the software through several versions.**



Incremental development benefits

Waterfall Model < Incremental Development

- **The cost of implementing requirement changes is reduced.**
  - The amount of documentation: Incremental development < waterfall model
- **It is easier to get customer feedback on the development work.**
  - Customers can comment on demonstrations of the software.
- **Early deployment of useful software to the customer is possible.**
  - Customers are able to gain value from the software earlier (vs a waterfall process).

Incremental development problems

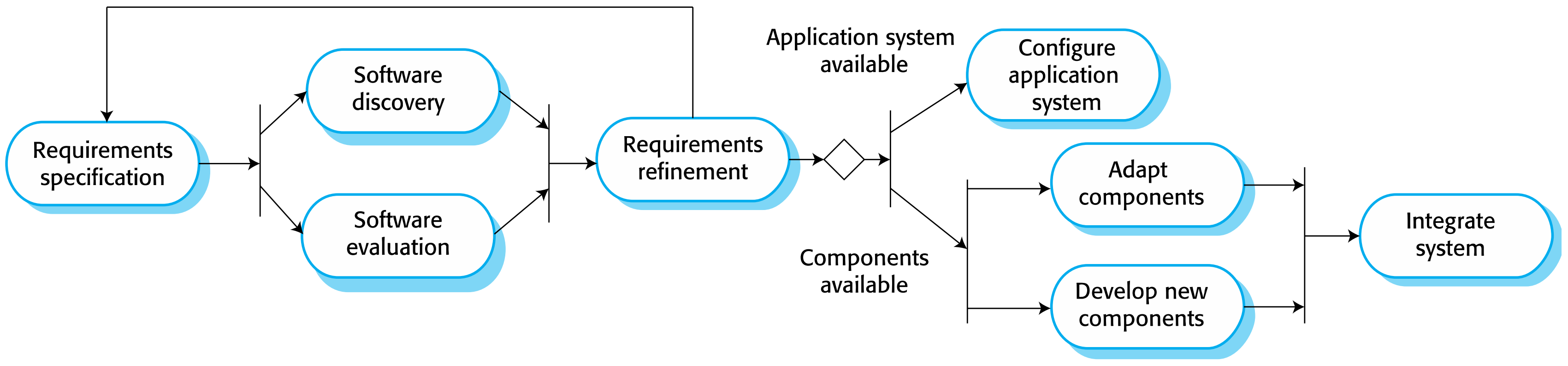
- **The process is not visible.**
  - Need regular deliverables to measure progress.
  - If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.

- **System structure tends to degrade as new increments are added.**
  - Regular change leads to messy code.
  - It becomes increasingly difficult and costly to add new features to a system.
  - Solution: should regularly refactor (improve and restructure) the software.

Chapter2-2. Software process models

Integration and configuration

✧ **Integrating existing reusable components into a software rather than developing it from scratch.**

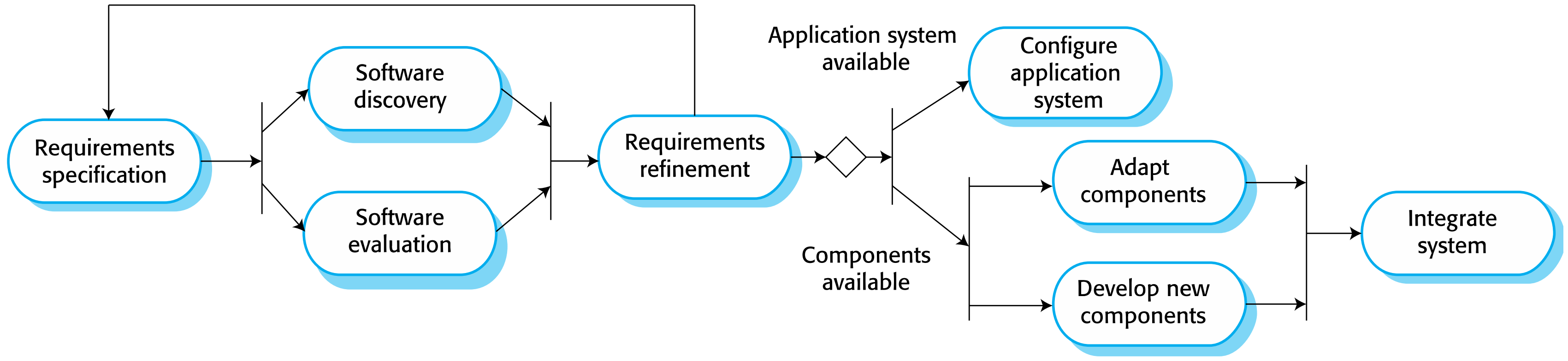


"A general process model for reuse-oriented development based on integration and configuration"

Chapter2-2. Software process models

Integration and configuration

[Key process of reuse-oriented SE]



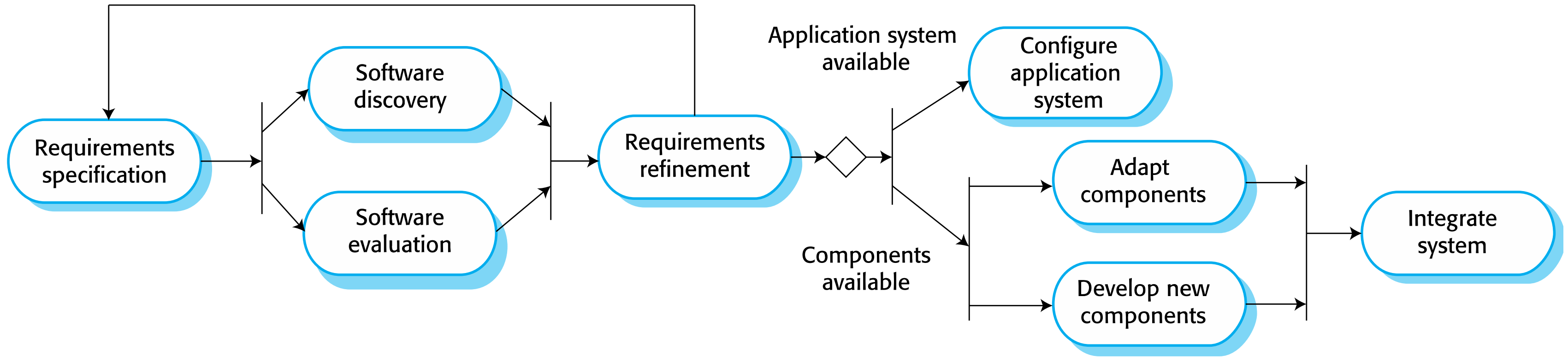
Requirements specification: Don't have to be elaborated in detail but include brief descriptions of essential requirements.

Software discovery and evaluation: Search and evaluate the components and systems that provide the required functionality.

Requirements refinement: Refine the requirements using information about the reusable components that have been discovered.

Integration and configuration

[Key process of reuse-oriented SE]



Application system configuration	COTS application system is configured for use to create the new system. (COTS is available)
Component adaptation and integration	Reusable components and newly developed components are integrated to create the system. (COTS is not available)

Reuse-oriented software engineering

[Advantages]

- **Reducing the amount of software** to be developed (cost ↓ and risk ↓)
- **Faster delivery** of the system

[Disadvantages]

- A system may **not meet real needs of users** because requirement compromises are **inevitable.**
- Some control over the system evolution **is lost.**

Chapter2-3. Process activities

Process activities

- The four basic process activities.
  - Specification, development, validation and evolution.

- These are organized differently in each development process.
  - Waterfall model
    - The activities are organized in sequence.
  - Incremental development
    - The activities are interleaved.

Chapter2-3. Process activities

Software specification

[Software specification or requirements engineering]

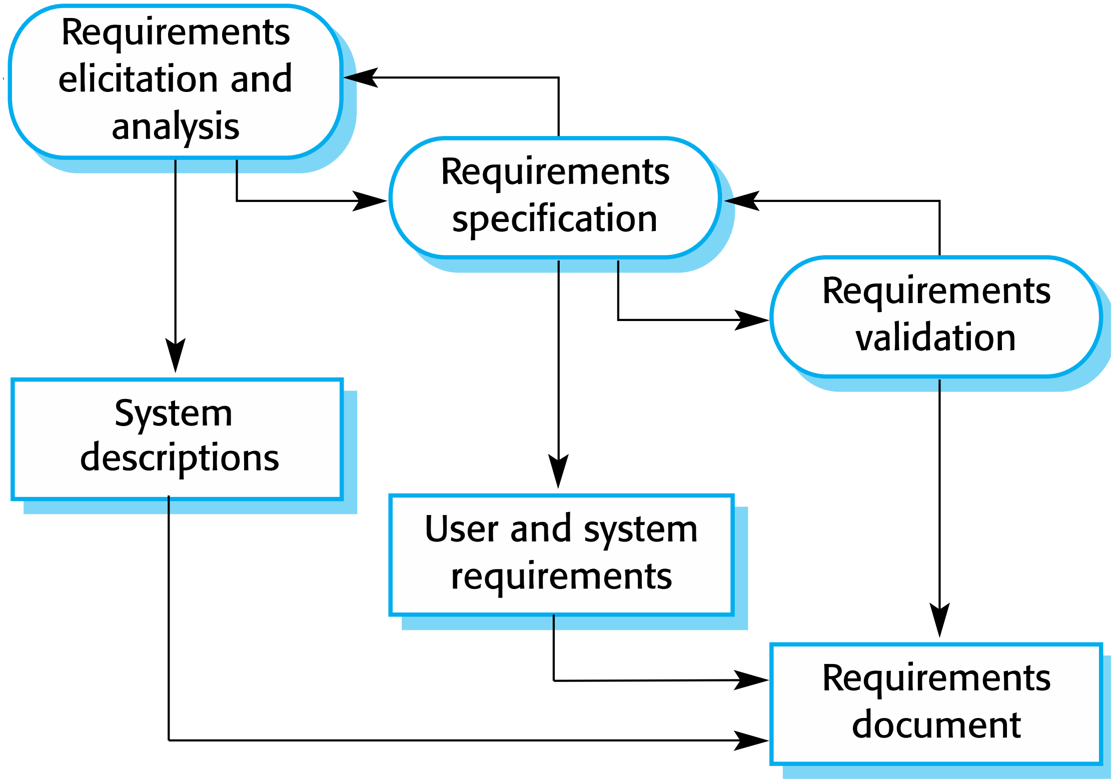
- Understanding and defining what services are required from the system.
- Identifying the constraints on the system's operation and development.
- The mistakes will lead to later problems in the system design and implementation.

[Requirements engineering process]

- Requirements **elicitation** and **analysis**
  - Deriving the system requirements through observation of existing systems and discussions with potential users.
- Requirements **specification**
  - Translating the information gathered into a document that defines a set of requirements
- Requirements **validation**
  - Checking the requirements for realism, consistency, and completeness.

Software specification

[The requirements engineering process]


Requirements elicitation and analysis
Requirements specification
Requirements validation
System descriptions
User and system requirements
Requirements document

“The requirements engineering process”

Software design and implementation

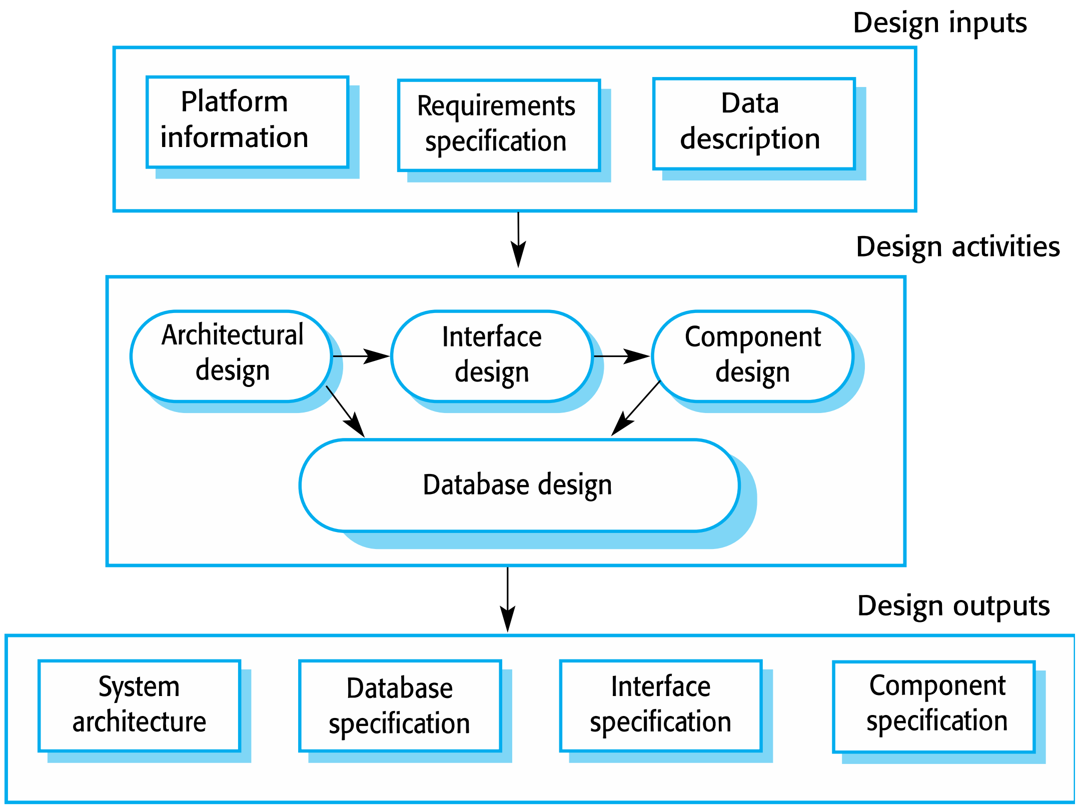
[Software design activities]

Architectural design	Identifying the **overall structure** of the system, the **principal components**, and their **relationships**.
Database design	Designing the system **data structures** and how these are to be **represented in a database.**
Interface design	Defining **the interfaces** between system components.
Component selection and design	Searching for **reusable** components and designing **new software** components.

Chapter2-3. Process activities

Software design and implementation

[A general model of the design process]



Software design and implementation

[Implementation]

- **Programming** is an individual activity with **no standard process**.
- **Testing** aims to **reveal program bugs** that must be removed from the program.
- **Debugging** is concerned with locating and correcting **these bugs**.

Software validation

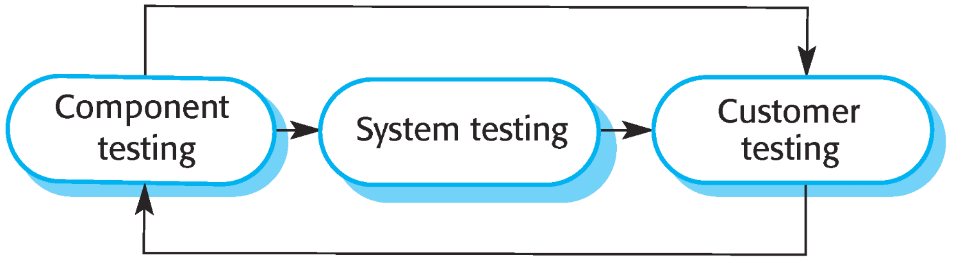
[Software validation] or [Verification and validation (V & V)]

- Checking whether a system **conforms to its specification** and **meets the requirements** of the system customer.

- Checking at each stage of the software process from user requirements definition to **program development**.
  - Program testing > user requirements

Software validation

[States of testing]



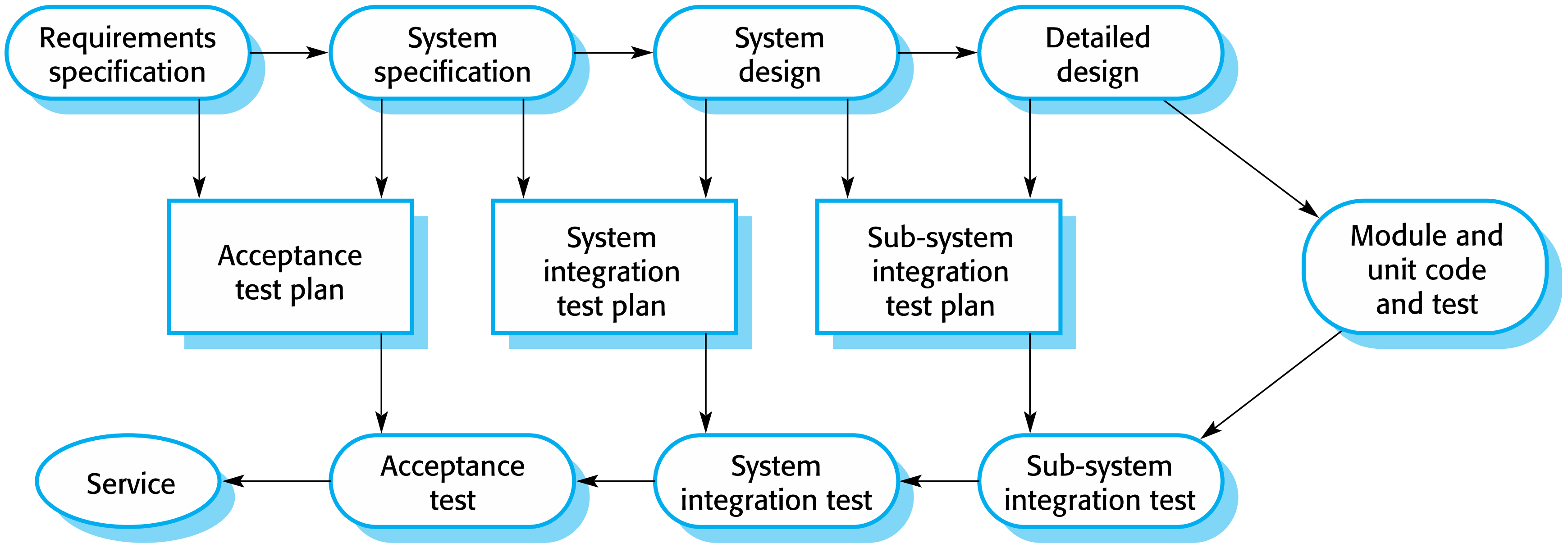
Component testing	: **Testing Individual components** (e.g., functions) independently. (The programmer is the best person to generate test cases.)
System testing	: **Testing the entire system** in which the components are integrated.
Customer testing	: **Tested by the system customer** rather than with simulated test data.

Chapter2-3. Process activities

Software validation

[Testing phases in a plan-driven software process (V-model)]

- illustrating how test plans are the link between testing and development activities.

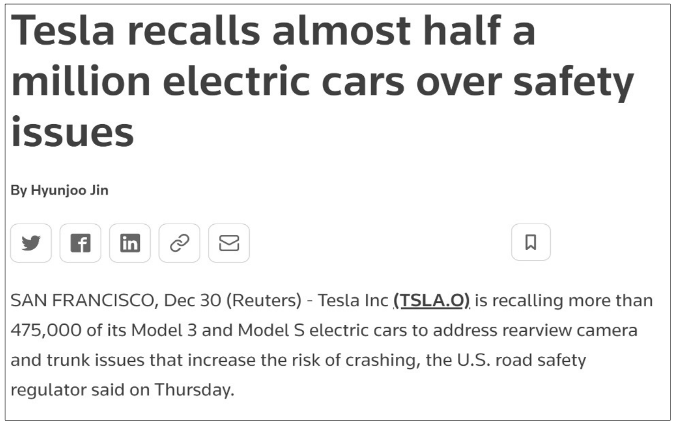

Requirements specification
System specification
System design
Detailed design
Acceptance test plan
System integration test plan
Sub-system integration test plan
Module and unit code and test
Service
Acceptance test
System integration test
Sub-system integration test

Chapter2-3. Process activities

Software evolution

[Software evolution]

- Software is inherently **flexible and can change**.
- Even extensive changes of **SW are still much cheaper** than the changes of **HW**.



Tesla recalls almost half a million electric cars over safety issues

By Hyunjoo Jin

SAN FRANCISCO, Dec 30 (Reuters) - Tesla Inc (TSLA.O) is recalling more than 475,000 of its Model 3 and Model S electric cars to address rearview camera and trunk issues that increase the risk of crashing, the U.S. road safety regulator said on Thursday.

- **This distinction** between development and maintenance is **increasingly irrelevant**.
  - Seeing development and maintenance as a continuum.

https://www.reuters.com/business/autos-transportation/tesla-recalls-over-475000-electric-vehicles-2021-12-30/

Coping with change

[Change is inevitable in all large software projects.]

$$\begin{tikzcd} \boxed{\text{Change in SW}} \arrow[r] & \boxed{\text{The completed tasks have to be redone.}} \arrow[r] & \boxed{\text{The costs of SW development} \uparrow} \end{tikzcd}$$

[Two approaches for reducing the costs of rework.]

Change anticipation	Including activities that can anticipate **possible changes before** significant rework is required. (e.g., prototype system)
Change tolerance	the process is designed so that **changes can be easily made to the system**. (e.g., incremental development, refactoring)

[Two ways of coping with change.]

Prototyping

Incremental delivery

Chapter2-4. Coping with change

Software prototyping

[Benefits of prototyping]

- Allowing potential users to see how well the system supports their work.
- Revealing errors and omissions in the system requirements.
- Reflecting the changed understanding of the requirements.
- The only sensible way to develop user interfaces.

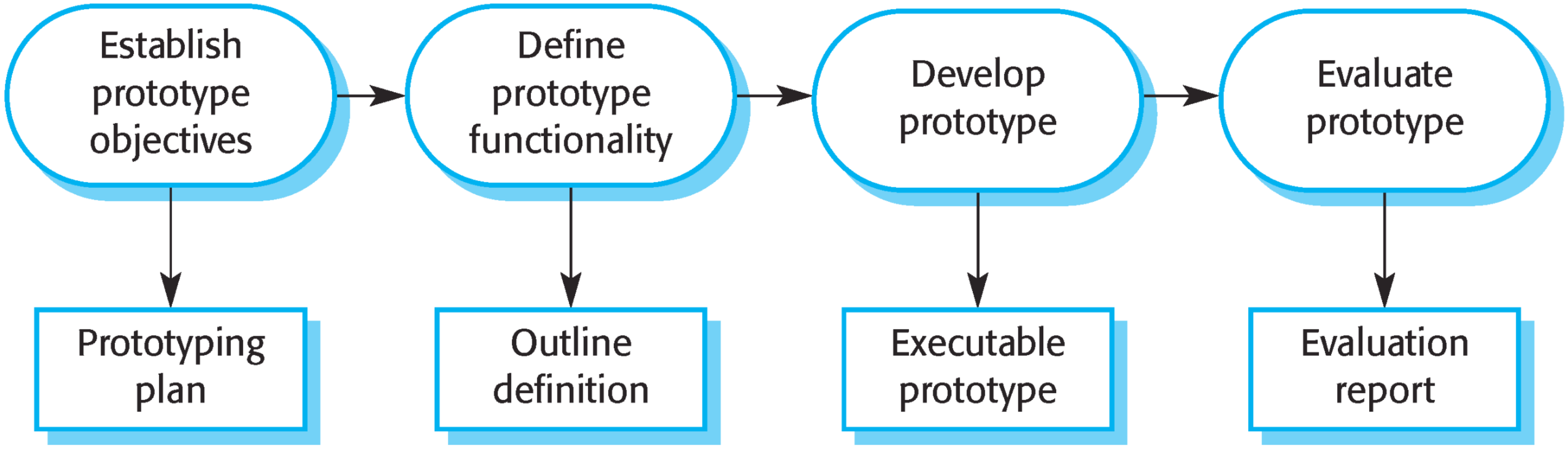
[Problems of prototyping]

- Using the prototype system ≠ Using the final system.
- Prototype testers ≠ typical of system users.
- Avoid those system features that have slow response times.

Software prototyping

[Prototype development]

- Key point: Deciding what to leave out of the prototype system.
  - Leaving out some functionality of the prototype.
  - Relaxing non-functional requirements (e.g., response time and memory usage).
  - Ignoring error handling and management.

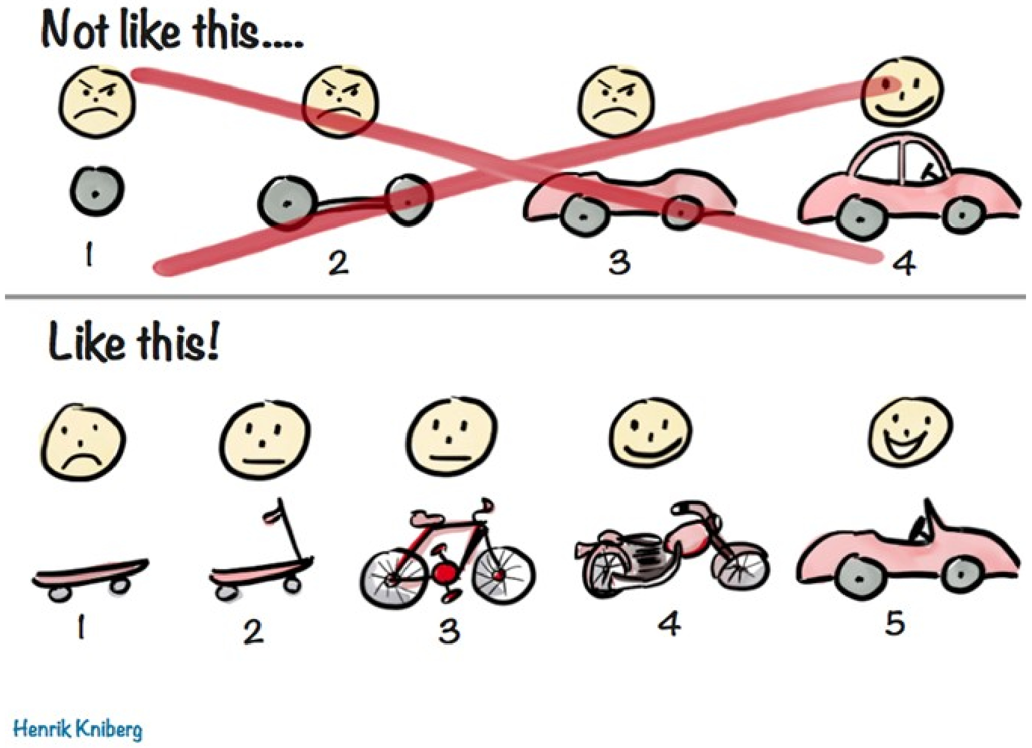


Chapter2-4. Coping with change

Incremental delivery

[Incremental delivery]

- Deliver **some of the developed increments** to the customers.
- Define **which of the services are most important** to the customers.



Incremental development and delivery

[Incremental delivery advantages]

- Customers can use the early increments as prototypes and gain experience. or (Customers do not have to wait until the entire system is delivered)
- Relatively easy to incorporate changes into the system.
- The most important system services receive the most testing.

[Incremental delivery disadvantages]

- When replacing an existing system with the new system, users need all of the functionality of the old system.
- As requirements are not defined in detail, it can be hard to identify common facilities that are needed by all increments.

Process improvement

- **Understanding** existing processes and **changing** these processes to increase product quality and/or reduce costs and development time.

The process maturity approach

- Improving process and project management.
- Introducing good software engineering practice into an organization.

The agile approach

- Iterative development and the reduction of overheads in the software process.

Summary

SW Process Model
- The waterfall model
- Incremental development
- Integration and configuration

SW Activities
- SW specification
- SW design and implementation
- SW validation
- Software evolution

Coping with Change
- SW prototyping
- Incremental delivery

Thank You