WDMM 3314 Web and Multimedia Engineering
SDEV 2301 Principles of Software Engineering

Chapter 2 Software processes

Part 1 Process activities

Dr. Nour Almadhoun Alserr

Week 4

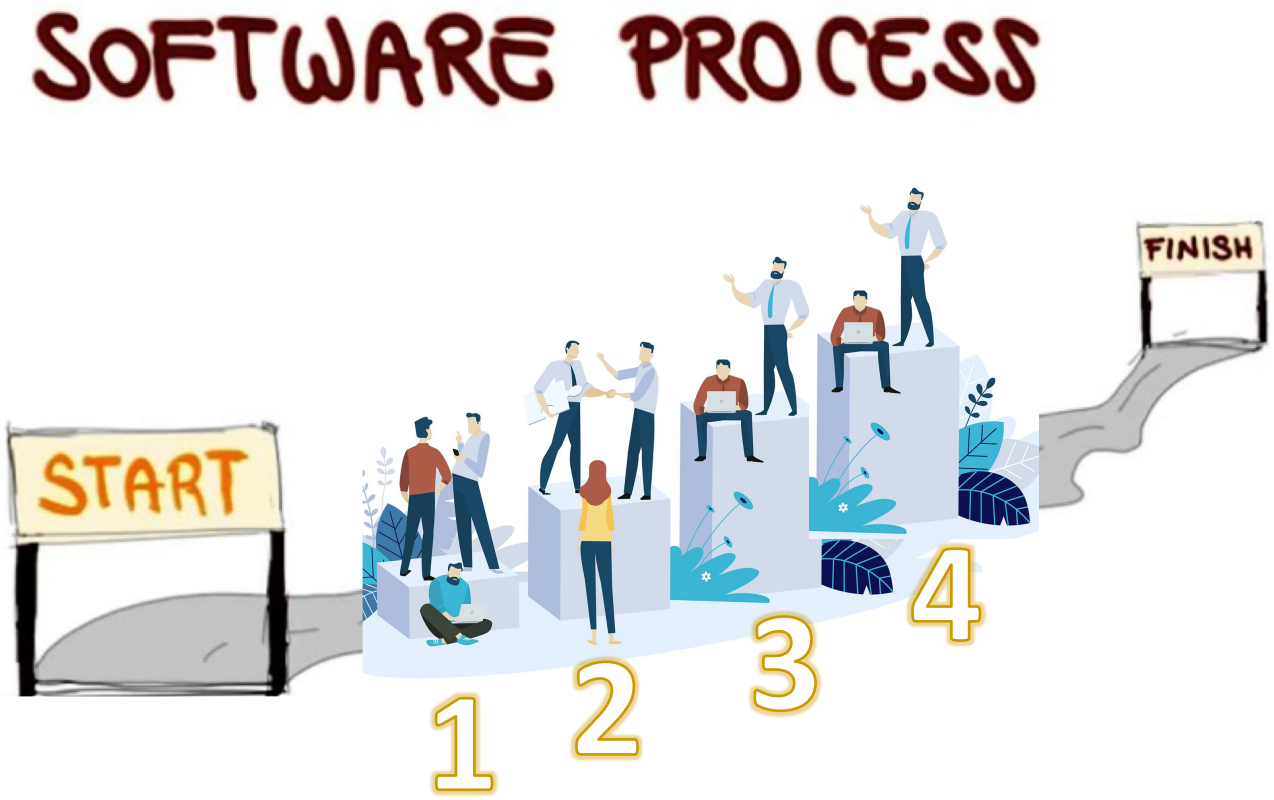
30 Sep 2023

Objectives

- Understand the concepts of **software processes**.
- Know about the fundamental **process activities** of software **requirements** engineering, software **development**, **testing**, and **evolution**.

Software Process

- A set of **related activities** that leads to the production of a **software system**.



https://www.majerrecruitment.com.au/6-awesome-team-building-activities/

RECALL: FAQs

9 What are the best software engineering techniques and methods?

- You can **NOT** say that one method is better than another.

No universally applicable software process

RECALL: FAQs

4 What are the fundamental software engineering activities?

- All software processes include the following four fundamental software engineering activities.

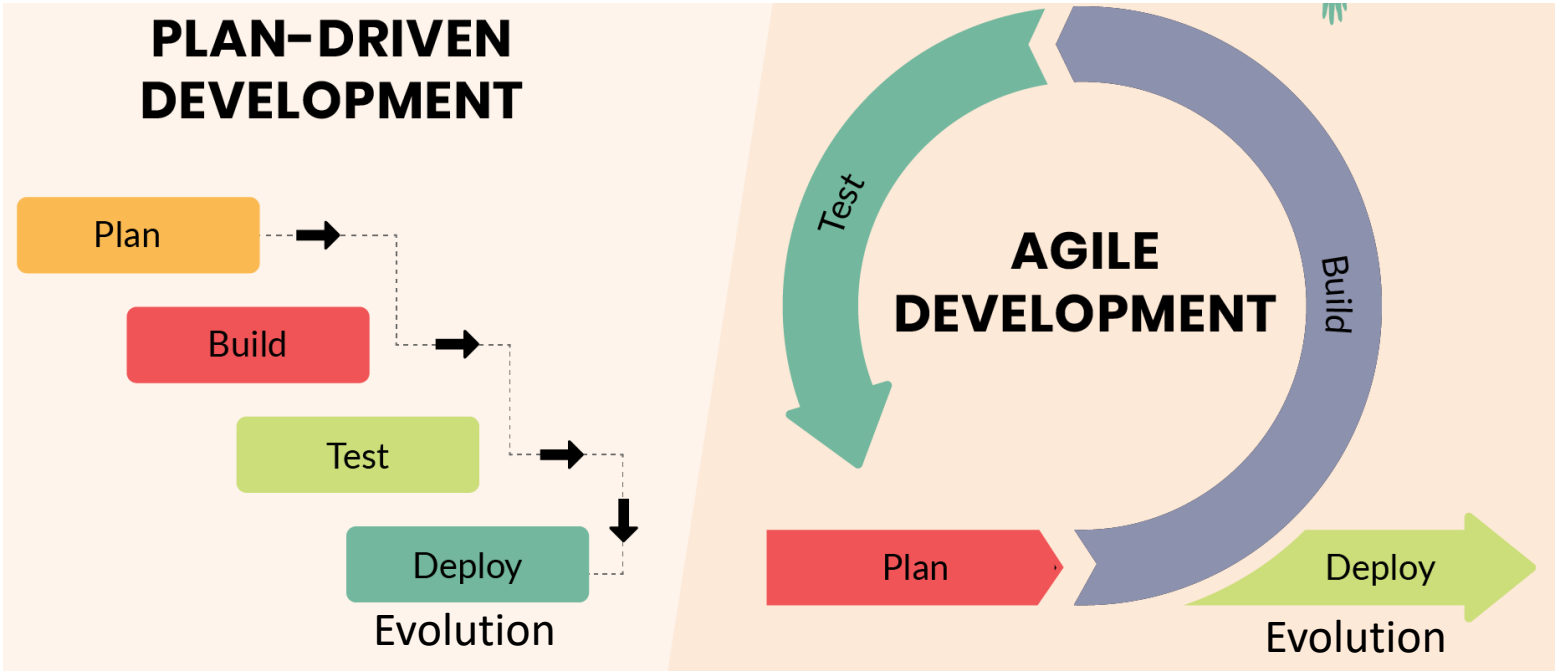
Software specification	Defining the software and specifying constraints on its use.
Software development	Designing and building the software.
Software validation	Testing the software to ensure it meets requirements, is **efficient, usable, maintainable, dependable and secure.**
Software evolution	Modifying the software to meet changes in the business or organisational context.

Software Process

- Describe process ⇨ describe **activities**.
- Describe process ⇨ describe **1.** what is produced, **2.** who is involved, and **3.** conditions that influence the sequence of activities.
  1. **Products:** **outcomes** of a process activity.
  2. **Roles:** the **responsibilities** of the people involved in the process.
  3. **Pre- and post-conditions**: **statements** that are true before and after a process activity.

Software Process

- **Plan-driven processes:** all process activities are planned in advance and progress is measured against this plan.
- **Agile processes:** planning is incremental, so it is easier to change the process to reflect **changing requirements**.



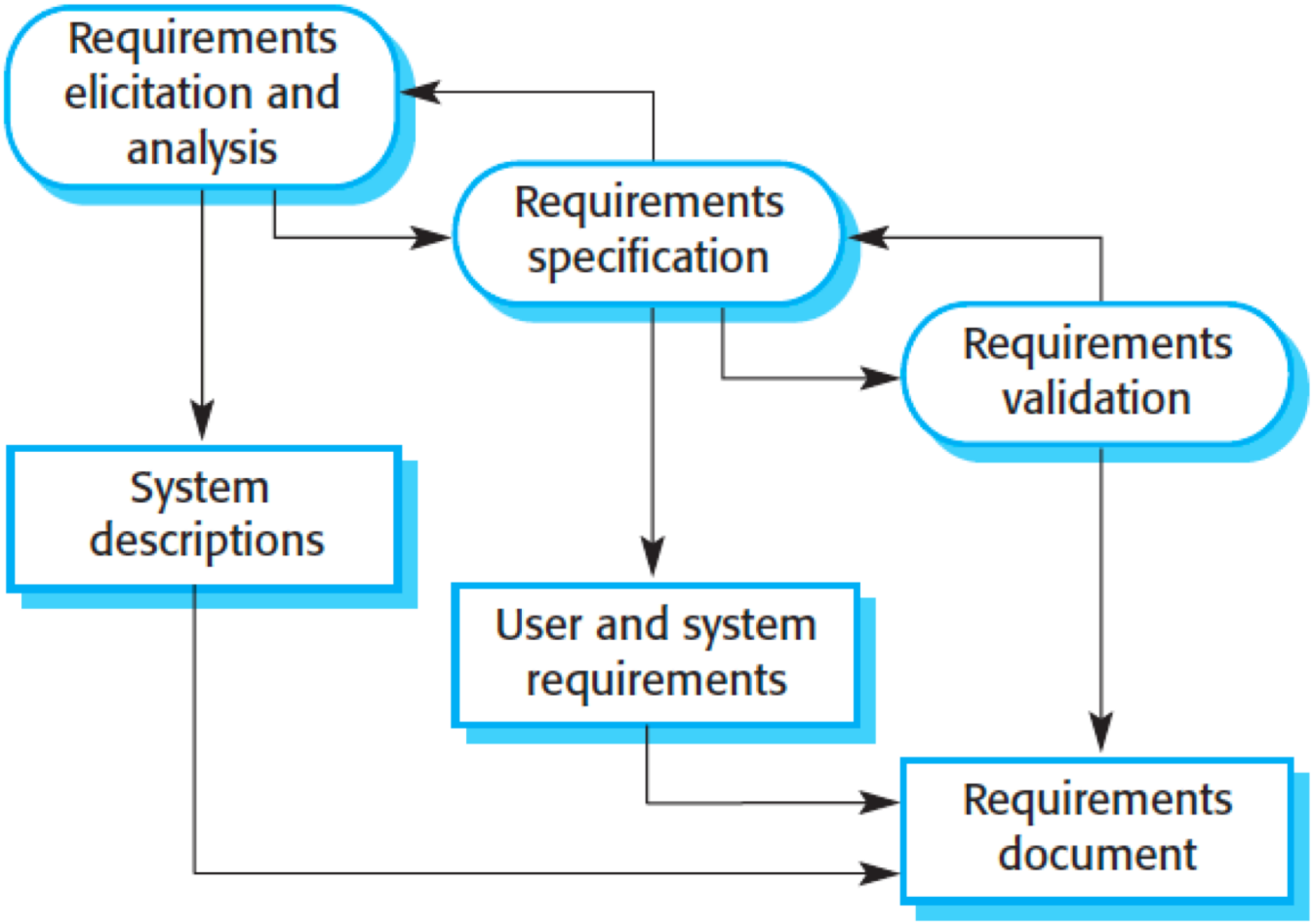
In sequence **Interleaved**

https://www.tryscrum.com/blogs/

(1/4) Software Specification

- Software Specification or **Requirements Engineering** (RE):
  - Understanding and **defining** what **services are required.**
  - Identifying the **constraints** on the system's operation and development.
- Critical stage: mistakes lead to later problems.
- Presented at **two levels** of detail:
  - End-users and customers > high-level statement (abstract).
  - System developers > a more detailed system specification.

(1/4) Requirements Engineering



(1/4) Requirements Engineering

I. **Requirements elicitation and analysis**
   - Deriving the system requirements (understand the system).

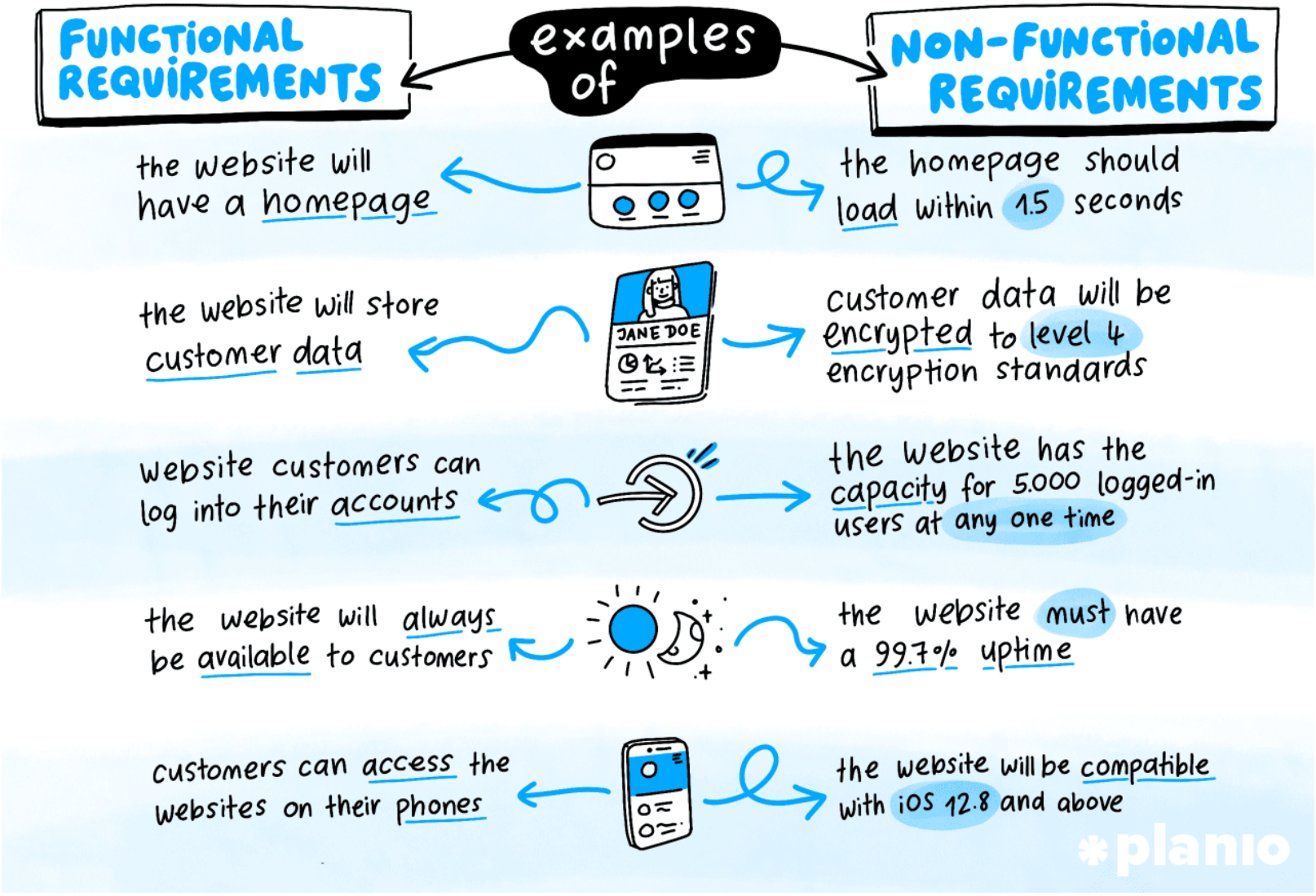
II. **Requirements specification**
   - Translating the information gathered into a document (user and system requirements).

III. **Requirements validation**
   - Checking the requirements

(realism, consistency, and completeness).

https://www.modernanalyst.com/Community/CommunityBlog/tabid/182/ID/5833/Top-10-Mistakes-in-Requirements-Elicitation.aspx

(1/4) Requirements Engineering



https://plan.io/blog/non-functional-requirements-nfrs/

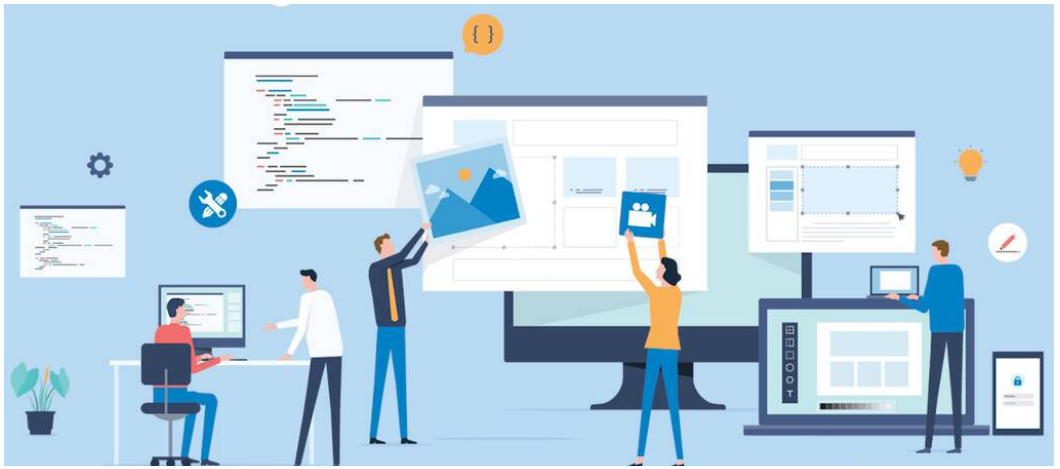
(2/4) SW Design and Implementation

- The process of **converting** the system specification into an executable system.

❑ Software design

Description of:

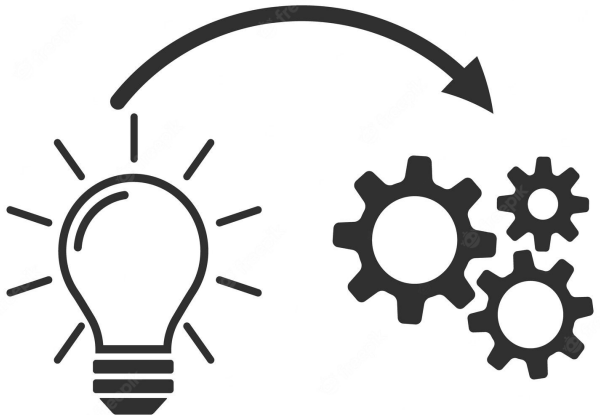
- Software structure.
- Data models and structures.
- Interfaces between system components.
- Algorithms used (sometimes).



https://study.com/learn/lesson/software-engineering-definition-processes.html

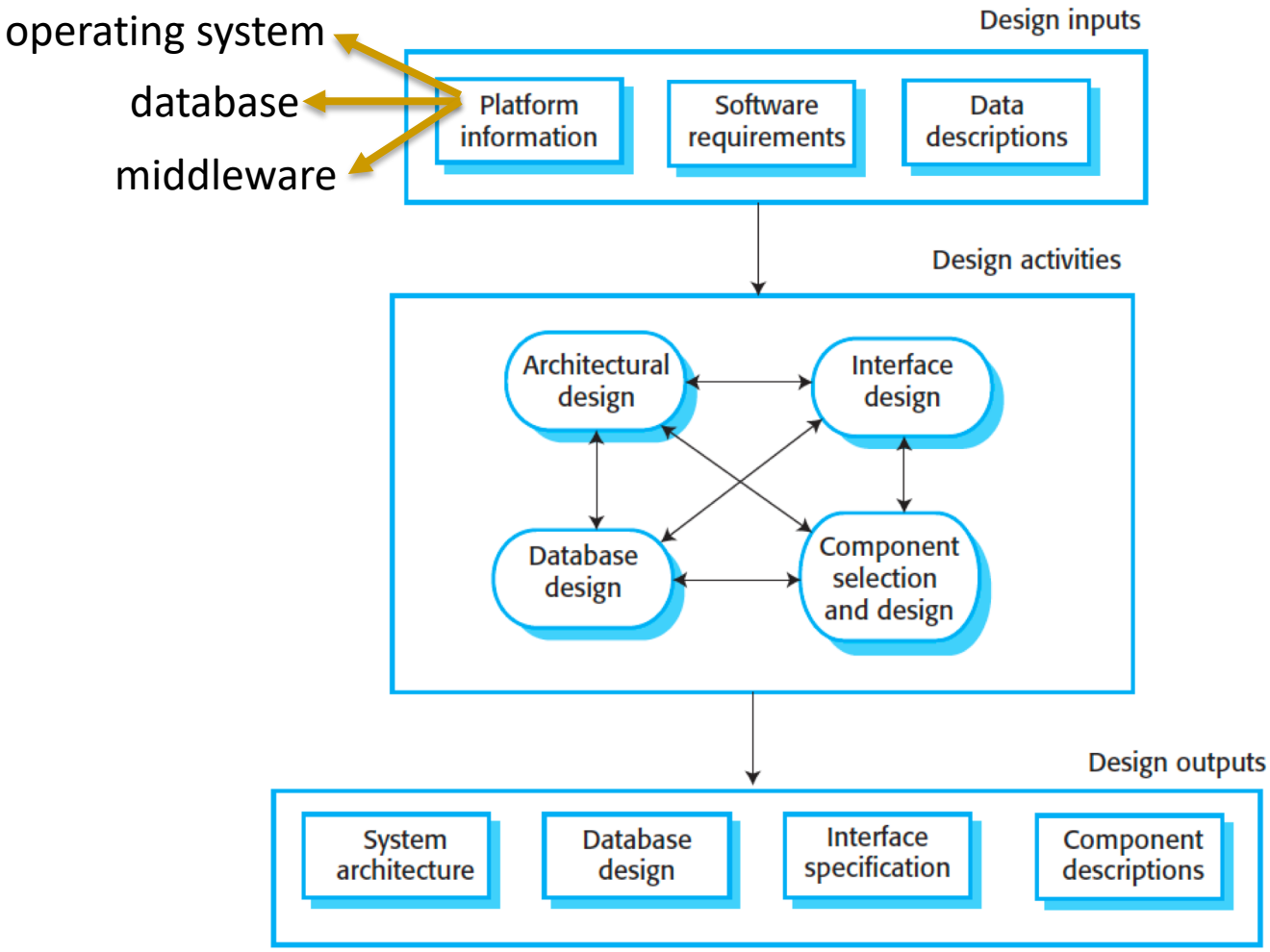
(2/4) SW Design and Implementation

- **Implementation**
  - Translate these structures into an **executable** program.
- The activities of design and implementation are closely **related** and may be **interleaved**.



https://study.com/learn/lesson/software-engineering-definition-processes.html

(2/4) SW Design and Implementation

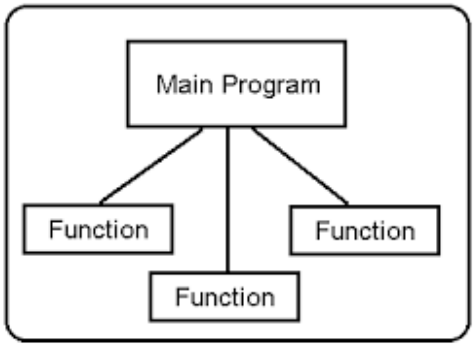


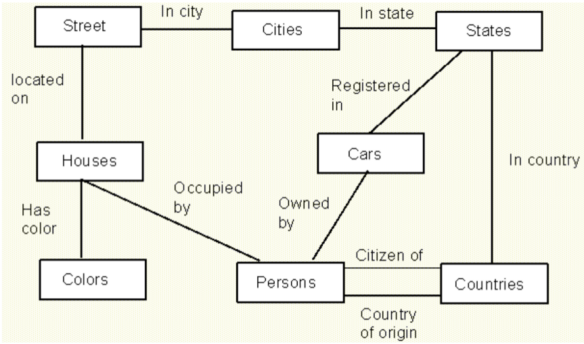
A General Model of The Design Process

(Information systems)

(2/4) Design Activities

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





- Database design, where you design the system **data structures** and how these are to be **represented** in a database.

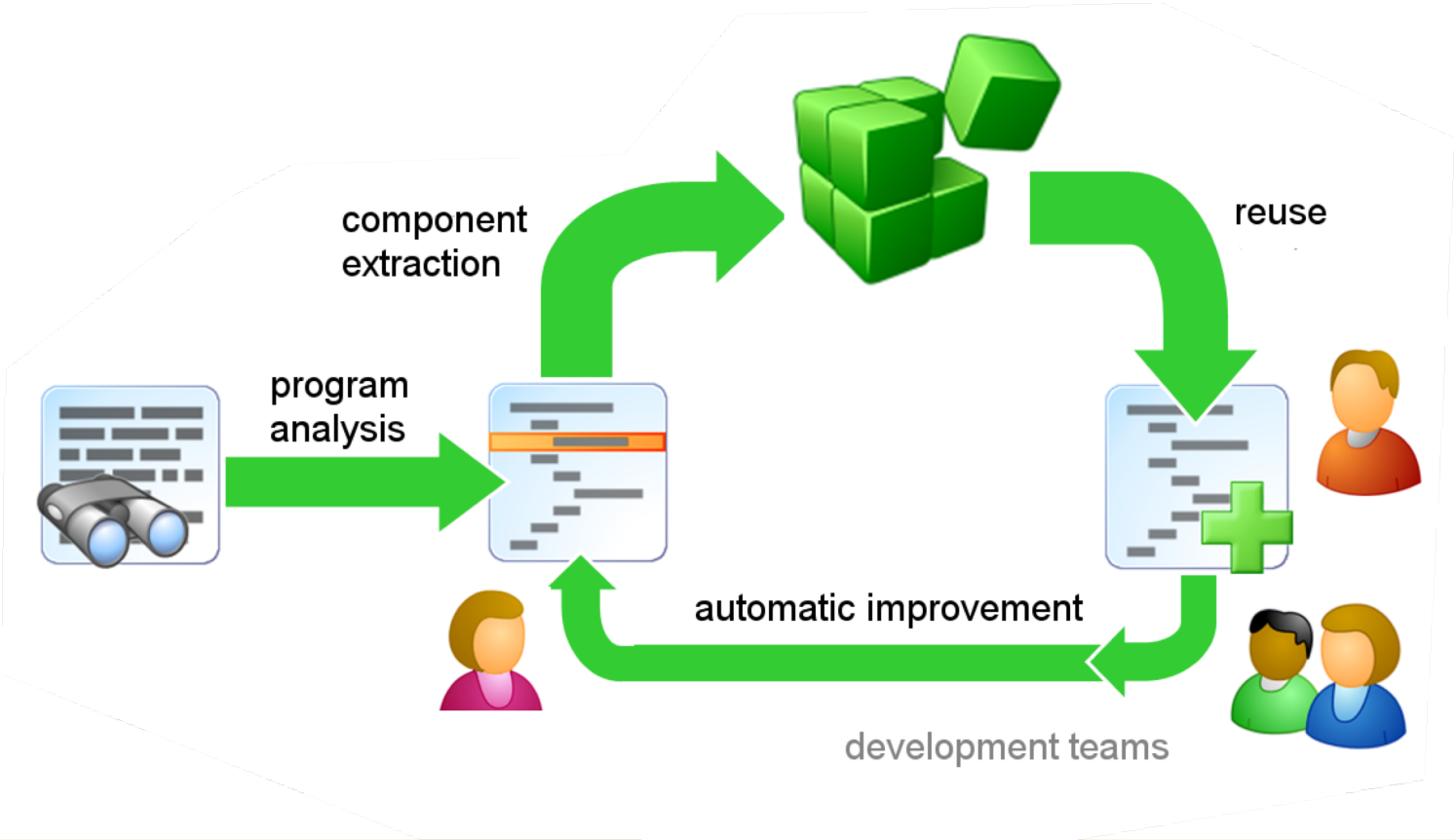
(2/4) Design Activities

- Interface design: define the **interfaces** between system components.

https://www.foroes.net/tips-for-creating-the-best-user-interface-design-for-your-app/

(2/4) Design Activities

- Component selection and design: search for **reusable components.** If not available, you design **new software components** (simple component description).



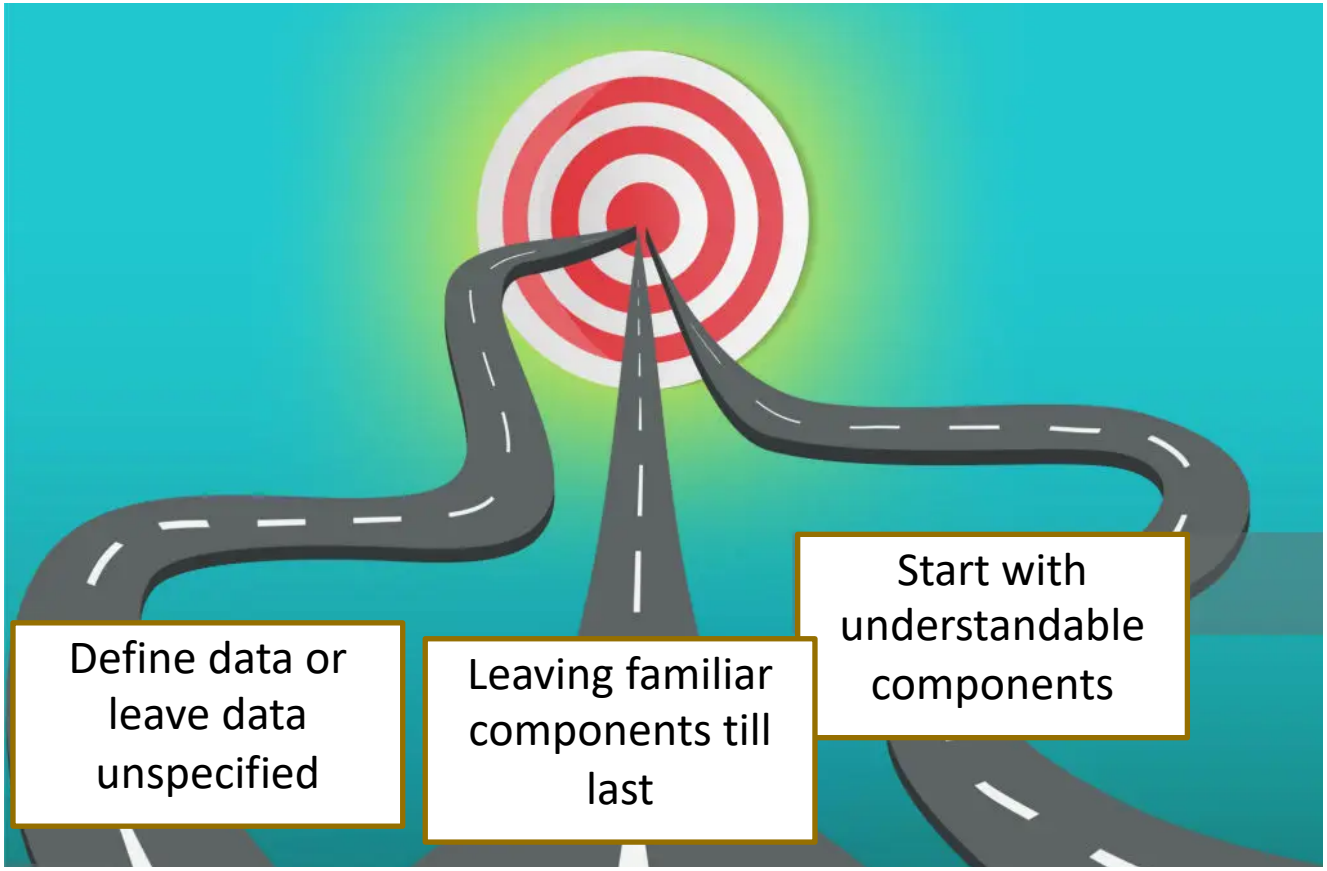
https://www.loginworks.com/blogs/reuse-code-object-oriented-programming/

(2/4) Design Activities

- For **critical systems**, the outputs of the design process are **detailed** design documents (accurate descriptions).
- If a **model-driven** approach is used, the design outputs are design **diagrams**.
- In **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**.

(2/4) Implementation

- **Programming** is an individual activity (no general process).



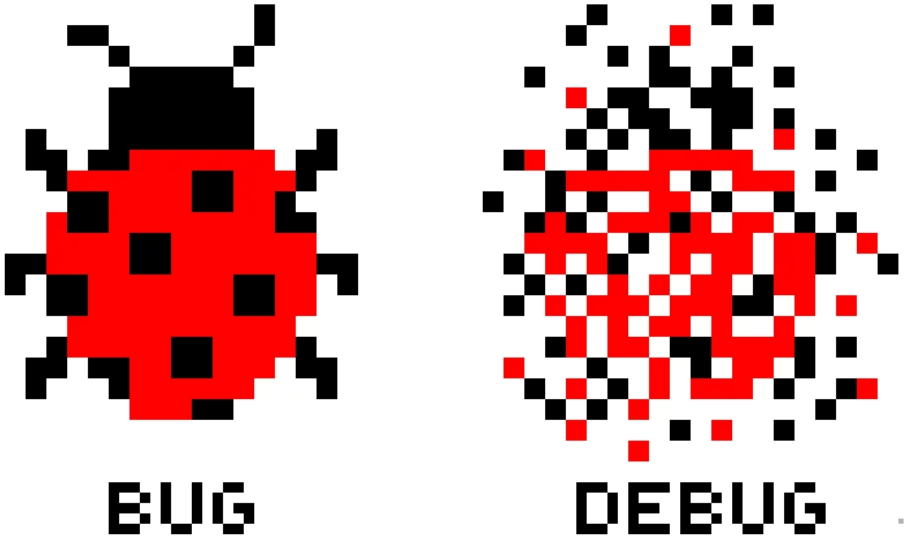
كل الطرق تؤدي الى روما

https://www.istockphoto.com/vector/many-way-to-reach-a-target-or-goal-gm1257411900-368493609

Code Testing

- An error in a program is called a **bug**.
- Eliminating errors is called **debugging** (generate hypotheses about the observable behavior of the program "**code tracing**").

Defect Testing



Debugging

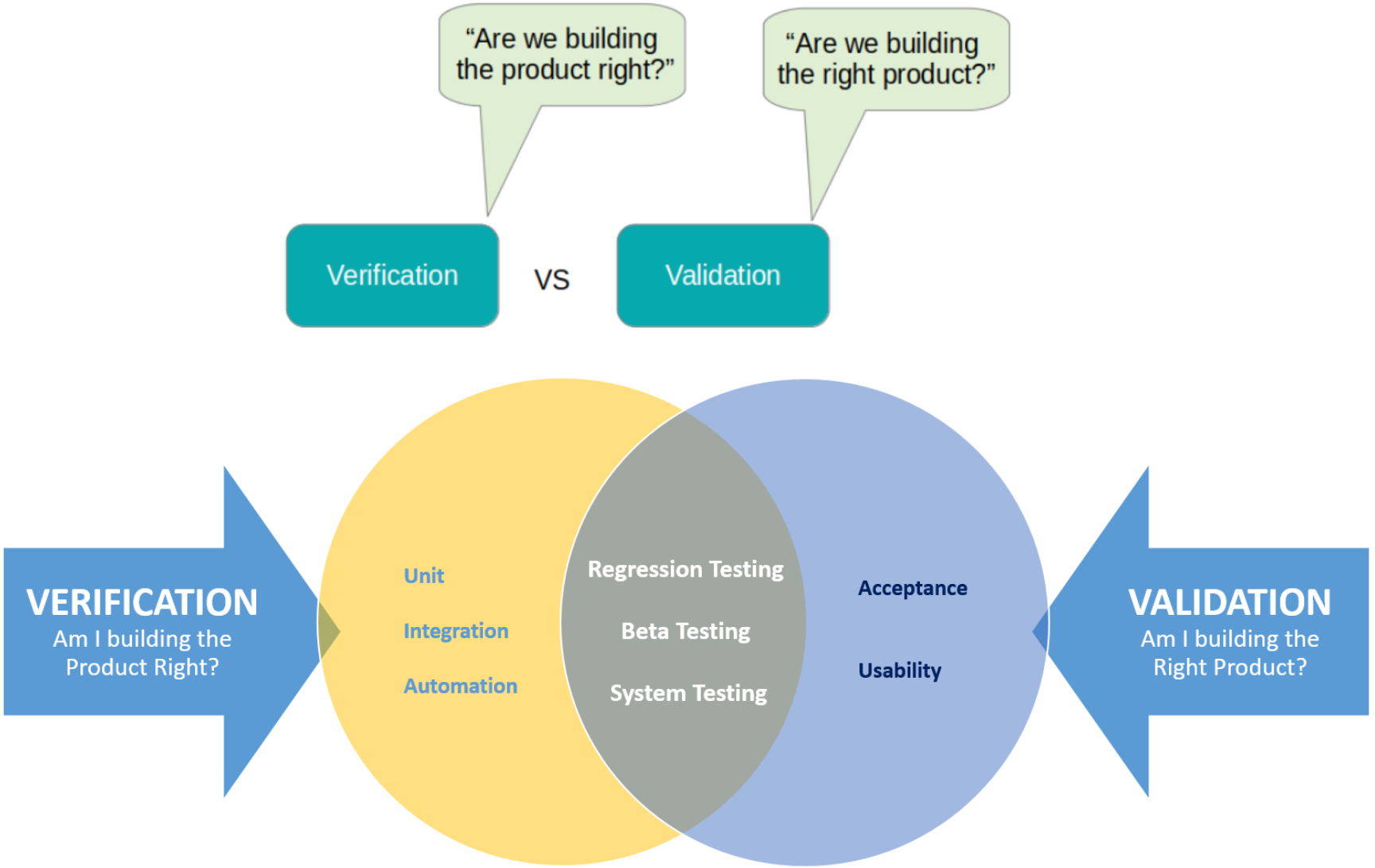
https://www.beningo.com/embedded-basics-classifying-software-bugs/
https://medium.com/@imAkash25/bug-debug-3445497d4daf

(3/4) Software Validation

- **Verification and Validation (V & V)** is intended to show that a system **conforms** to its **specification** and **meets** the **requirements** of the system customer.
- Involves:
  - **Program testing:** the system is executed using **simulated test data** (principal validation technique, most time and efforts).
  - Checking processes, such as **inspections** and **reviews**.

https://www.optimizely.com/insights/blog/what-is-derived-data-and-how-can-you-use-it-for-your-business/

(3/4) Software V & V



https://programmerbay.com/difference-between-verification-and-validation-testing/
https://www.pcloudy.com/blogs/verification-and-validation-in-testing/

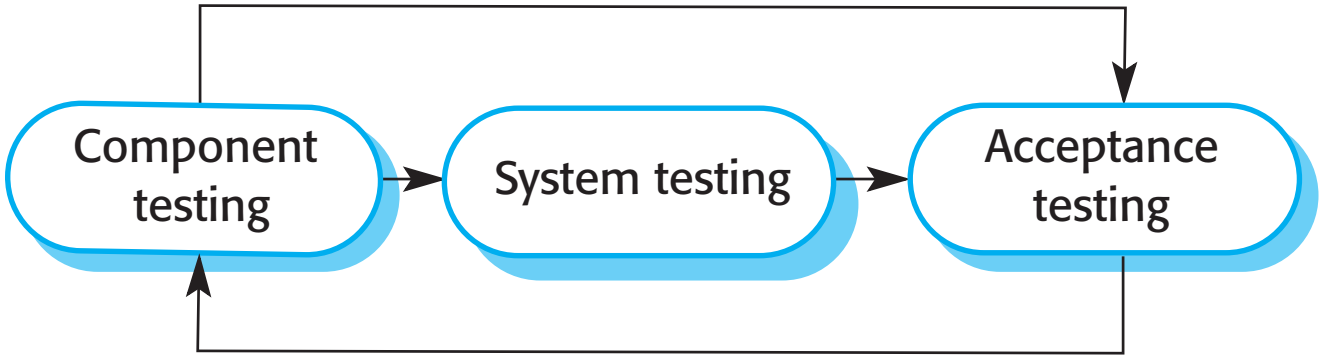
(3/4) Testing

- **Component (Unit) testing**
  - **Individual** components are tested **independently**.
  - Components: functions, objects or coherent groupings of these entities.

- **System testing** **(system: integrated components)**
  - If System meets its functional and non-functional requirements.
  - Testing of emergent properties.
  - Large systems: multistage process (subsystems).

(3/4) Testing

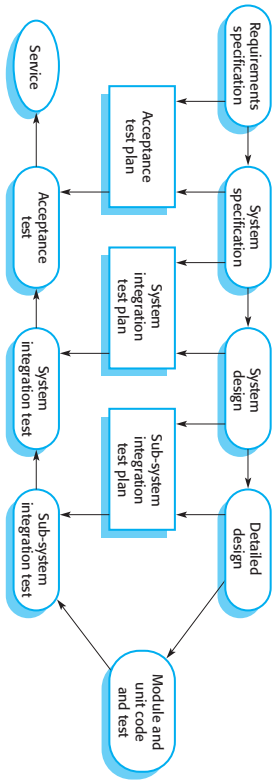
- **Customer (Acceptance) testing**
  - The system is tested by the **system customer** "real customer data" (or "**beta testing**" for selected potential customer) rather than with simulated test data.
  - Customer testing shows **how well** the software product meets the **customer's needs**.



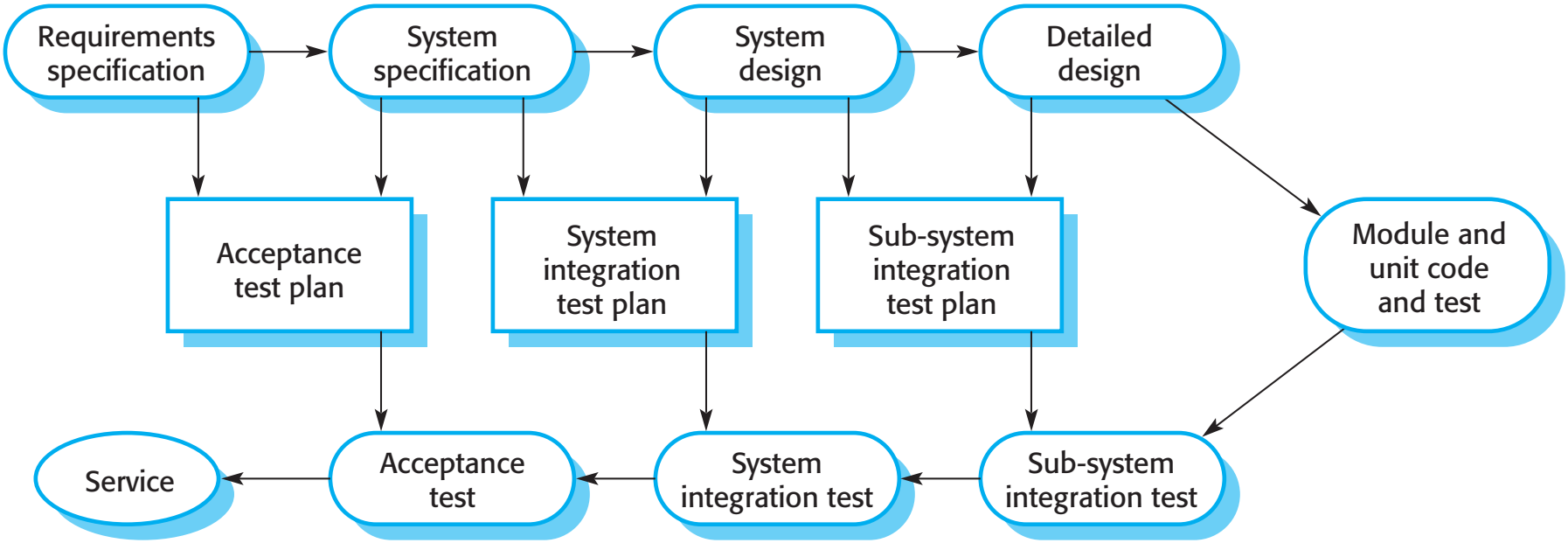
(3/4) Testing

- **Incremental and agile process** (**test-driven**): tests are developed along with the requirements before development starts (no delays).

- **Plan-driven software process** (**test plans**) : V-model



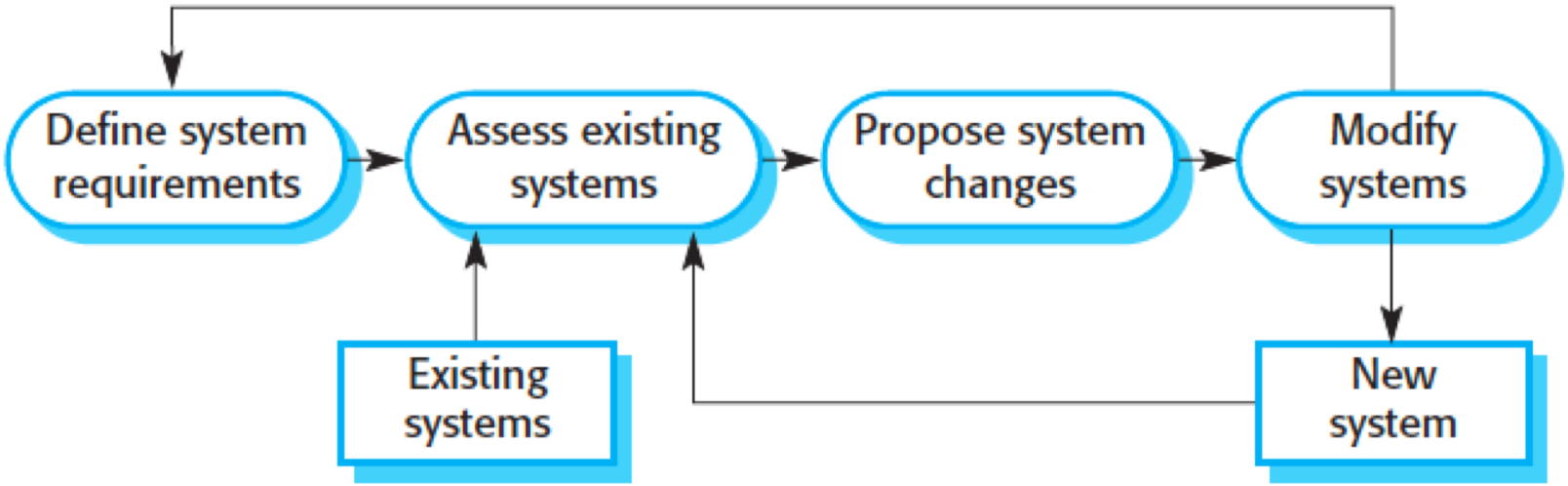
(3/4) Testing



Testing phases in a plan-driven software process

(4/4) Software Evolution

- Software must be **flexible** and can **change** (evolve).
- **Development** and **Evolution** **"maintenance"** (a continuum, very few software systems are completely new systems).



Software system evolution

(4/4) Software Evolution

- It is more realistic to think of software engineering as where software is **continually changed** over its lifetime in response to:

1

changing requirements

2

customer needs

https://mobidev.biz/blog/managing_evolution_of_a_software_product