SOEN 341
Software Process

Lecture 02:
Process Models

Emad Shihab, PhD

Software Process Activities

软件规范

- **Software specification:** customers & engineers **define the software** that is to be produced and the **constraints on its operation**

软件规范:客户和工程师定义要生产的软件及其操作的约束

- **Software development:** the software is **designed** and **programmed**

软件开发:软件是设计和编程

Software Process Activities

- **Software validation:** the software is **checked to ensure** that it is what the **customer requires**

  软件验证:检查软件以确保它是客户所需要的

- **Software evolution:** modifications done to meet **changing customer and market needs**.

  软件演进:为满足不断变化的客户和市场需求而进行的修改

Phases and Models of Software Process

- There are many **different software process models**, but they all **share the same basic elements**

  有许多不同的软件
  过程模型，但它们都共享
  相同的基本元素

- The difference is in how these elements are organized.

Fundamental SE Activities

Specification

Development

Validation

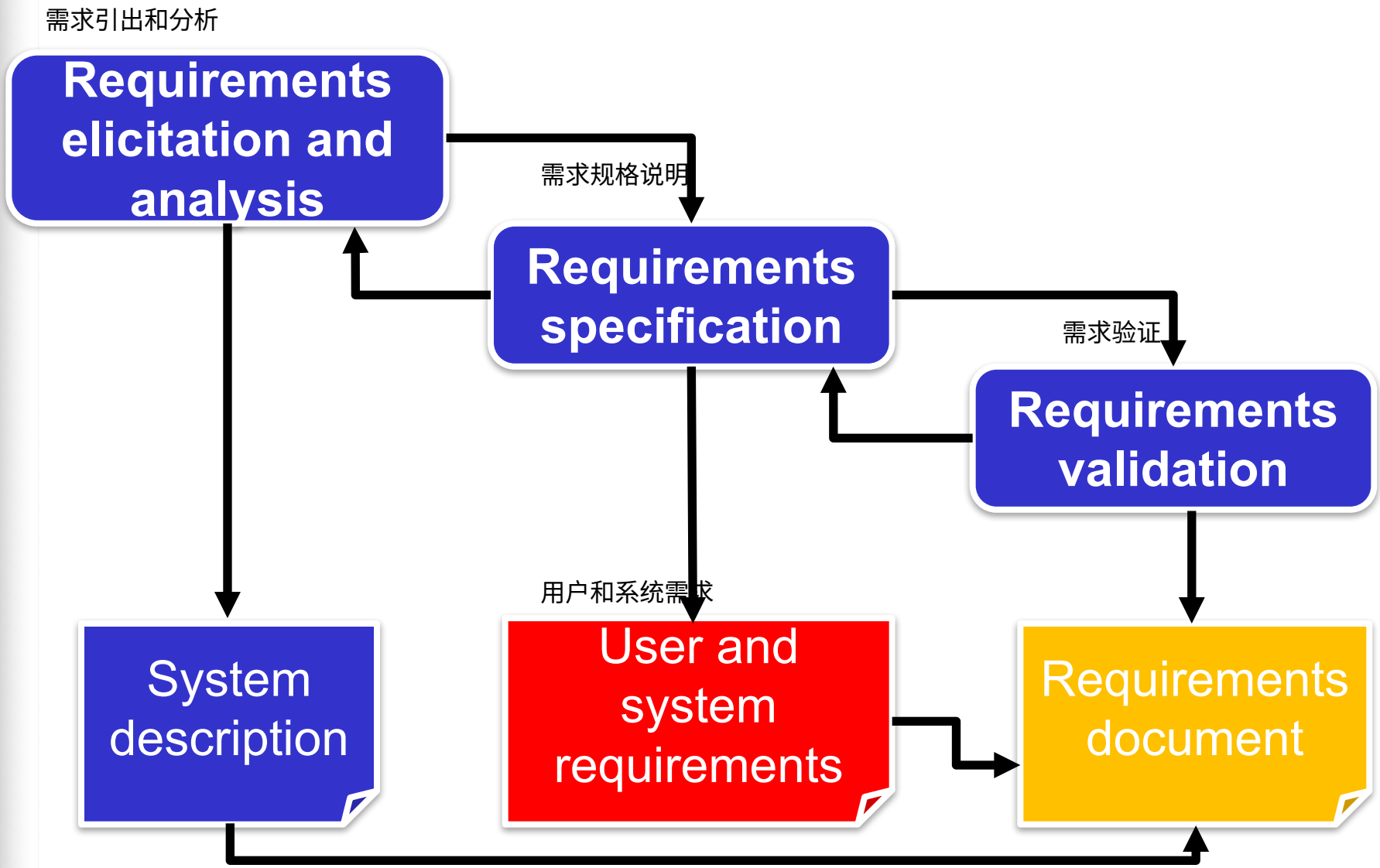
Evolution

Requirements/Specification

- **Specification is the task of precisely describing the software to be written**

规范是准确描述要编写的软件的任务

Requirements/Specification



Requirements/Specification

- **Element Analysis:** Customers know what they want, but **not what software should do**.
  - Demonstrating **live code helps reduce the risk** that the requirements are incorrect.

元素分析:客户知道他们想要什么，但不知道软件应该做什么。
• 演示实时代码有助于降低错误需求的风险

- **Scope Analysis: scope of the development should be determined** and **clearly stated.**
  - Certain functionality may be out of scope of the development project as a function of cost

范围分析:确定和明确开发范围。
•由于成本的关系，某些功能可能超出了开发项目的范围

Requirements/Specification

- Specifications are **most important for external interfaces** that must remain stable.

对于必须保持稳定的外部接口来说，规范是最重要的。

Fundamental SE Activities

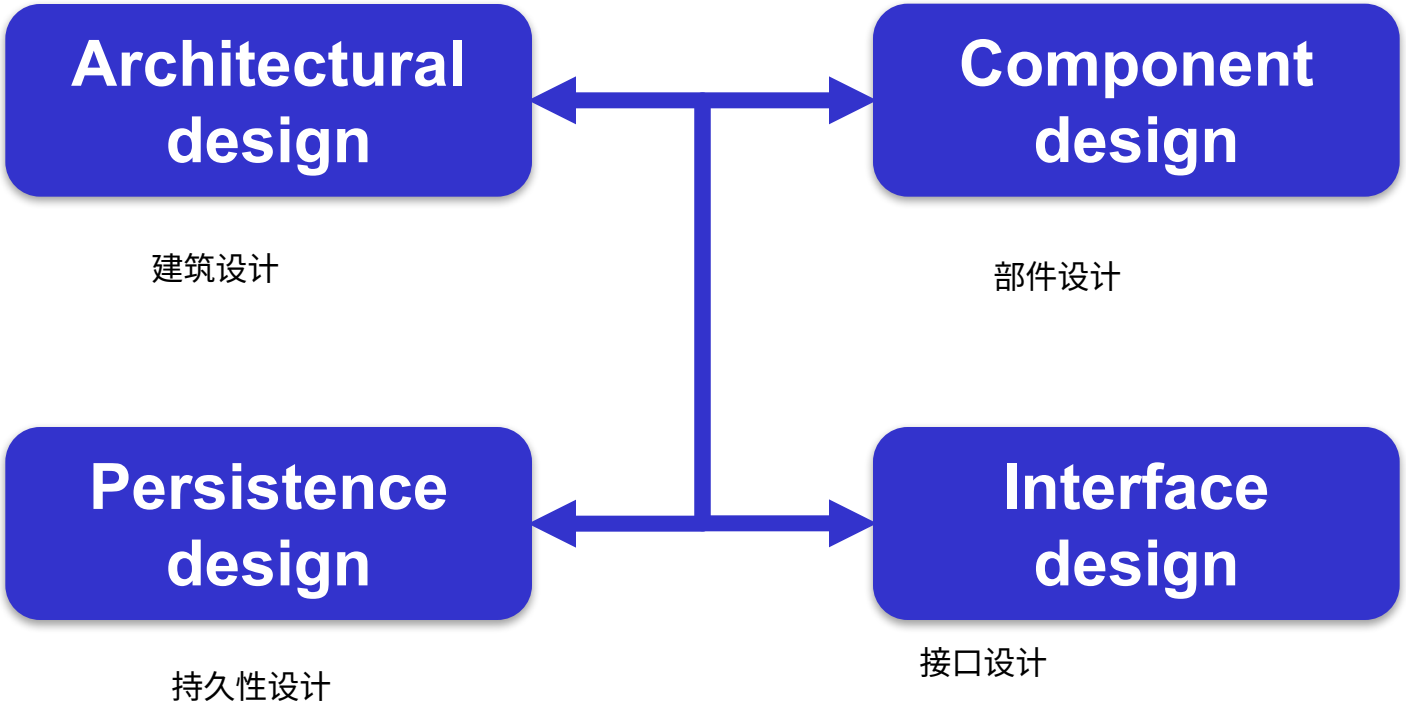
Specification

Development

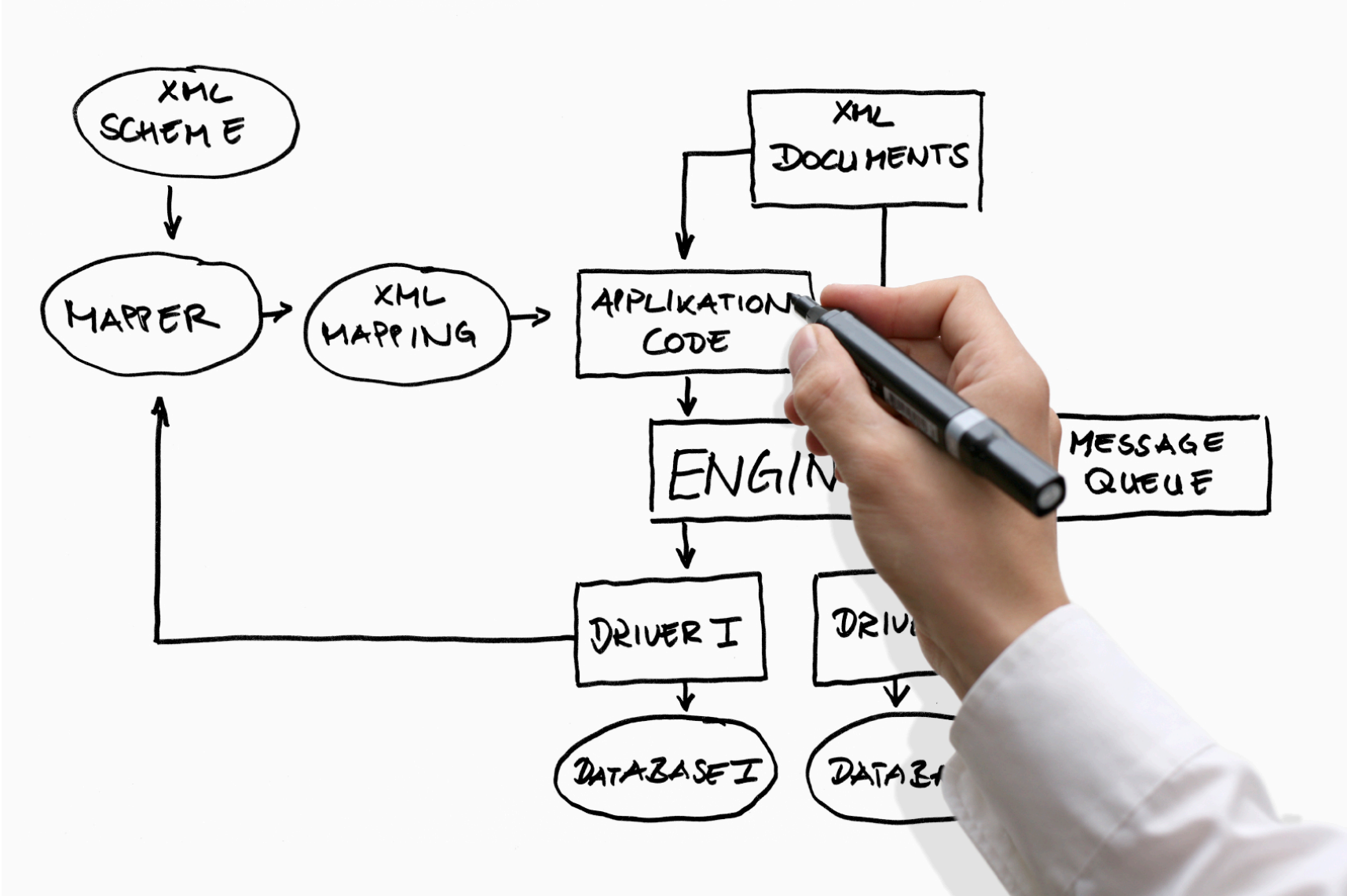
Validation

Evolution

Design


Architectural design
建筑设计
Component design
部件设计
Persistence design
持久性设计
Interface design
接口设计

Design


XML SCHEME
XML DOCUMENTS
MAPPER
XML MAPPING
APPLIKATION CODE
ENGIN
MESSAGE QUEUE
DRIVER I
DRIV
DATABASE I
DATAB

Fundamental SE Activities

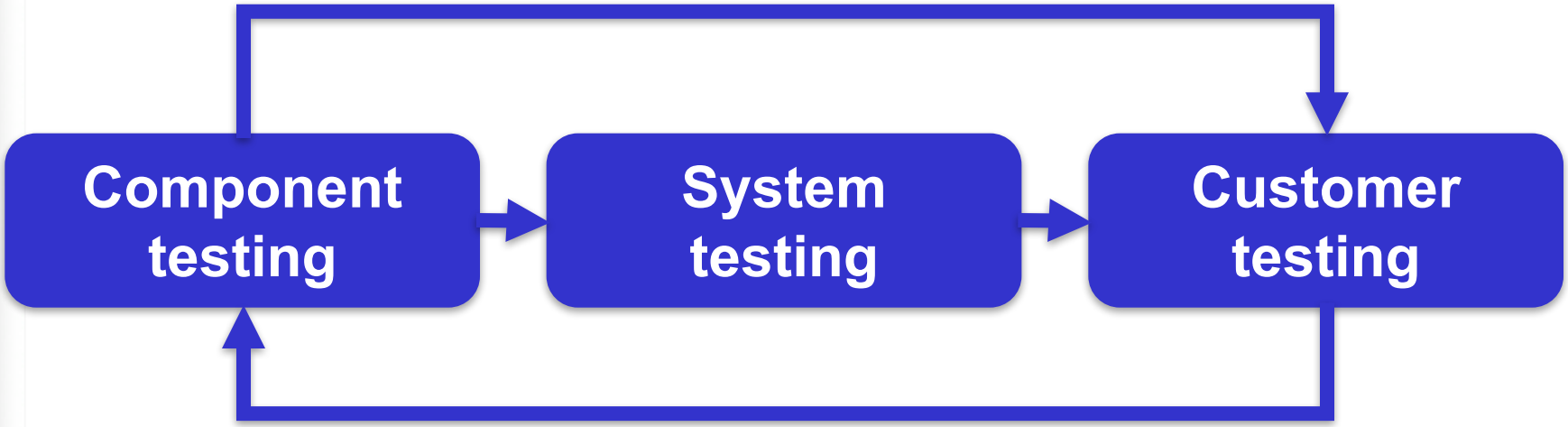
Specification

Development

Validation

Evolution

Validation/Testing



Validation/Testing

单一元件测试

Component testing

写代码的人测试

- Done by the person who writes the code
- Often considered as part of coding

被认作是coding的一部分

System testing

- Feature testing and performance testing

特性测试和性能测试

- Regression testing 回归测试
- Different levels of system testing

Validation/Testing

Customer testing

- Acceptance testing] 验收测试
- Field testing 实地测试

Fundamental SE Activities

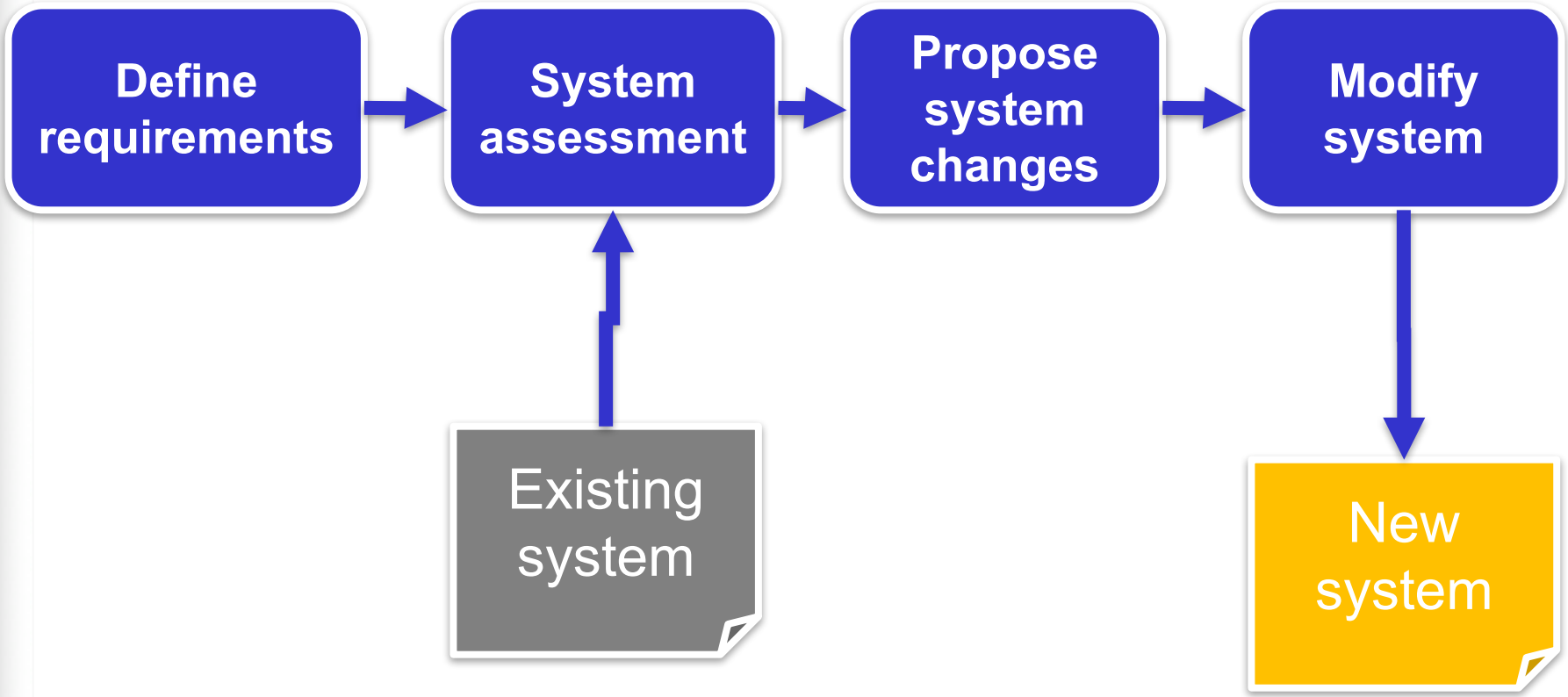
Specification

Development

Validation

Evolution

Evolution


Define requirements
System assessment
Propose system changes
Modify system
Existing system
New system

Evolution/Maintenance

- **Maintaining and enhancing** software to cope with **newly discovered problems** or new requirements can take far **more time than the initial development** of the software.

- A **small part** of that is **fixing bugs**. **Most maintenance extends systems** to do new things, which in many ways can be considered new work.

Other Activities

- **Deployment:** Moving code into production environment i.e. is made available for business use.

- **Documentation:** documenting the internal design of software for the purpose of future maintenance and enhancement.
  - Most important for external interfaces.

- **Software Training and Support**


How the customer explained it

Process Models

Building vs. Growing

Building software

The “building” metaphor: planning; specification as blueprint; components; assembly; scaffolding; etc.

Idea: **planning preceded construction**

Growing software rather than build it.

Start with a very simple system that runs but has minimal functionality and then add to it and let it grow.

Process 0

The basic model used in the earliest days of software development contained the following steps:

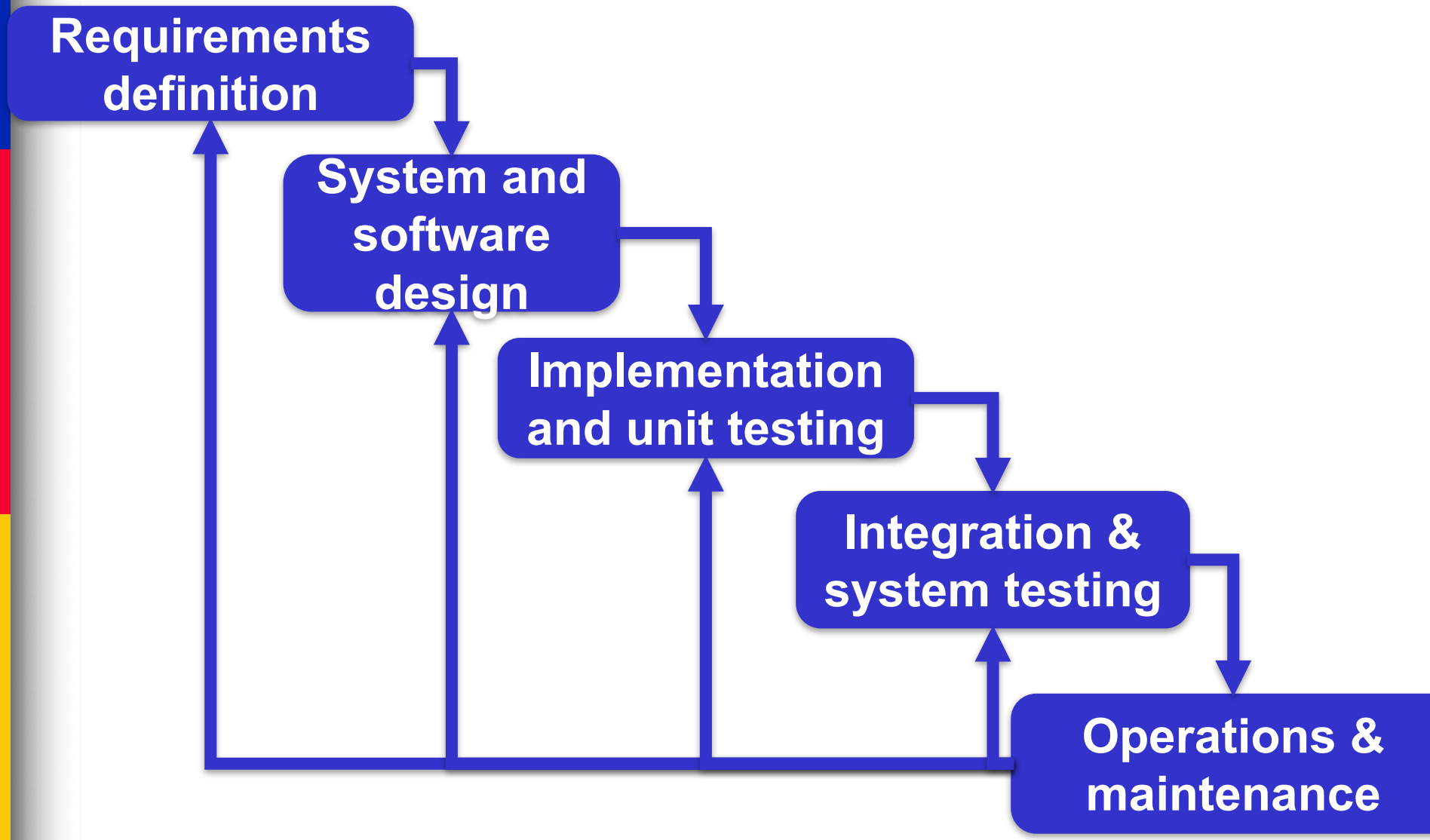
1. Write some code.
2. Fix the problems in the code.

Process 0: The code-and-fix model

- After a number of fixes, the code can become so **poorly structured** that **subsequent fixes were very expensive**.
  - **Need to design and evolve/test**

- Even **well-designed** software can be a **poor match** for **users' needs**.
  - **Need for requirements**

Process 1: The Waterfall Model



Process 1: The Waterfall Model

- First complete the "**requirements** specification”.
- Then **design a "blueprint"** for implementers (coders) to follow.
- This design is a plan for the requirements given.
- When the **design is complete**, **implementation begins**.

Process 1: The Waterfall Model

- **Components** produced by different teams **are integrated**.
- Software is **tested and debugged**; any **faults** introduced in earlier phases are **removed**.
- Software **product is installed**, and later **maintained** to introduce new functionality and remove bugs.

The Waterfall Model is Document Driven

- Each step of the process yields **documents**.

- For example, when **Requirements Analysis** has been completed, there is a **Requirements Document**. **Before coding** starts, there must be a set of **Design Documents**.

The Waterfall Model is Document Driven

- Documents produced during **one step are needed for the next step** and possibly for **later steps**.
  - For example, the **Requirements Document is needed for design**, the next step.
  - Later, the **Requirements Document** is needed to ensure that the developed product **meets the requirements during Acceptance Testing**.

The Waterfall Model and Management

- Managers ~~like~~ love the waterfall model because **progress is easily observable and measurable**.
- The transitions between **steps become project "milestones"** that indicate progress made.
- **Documents** are tangible **evidence of progress**.

The Waterfall Model and Cost Estimation

- We can estimate **cost by adding** the estimated **costs of each phase** and then adding a **safety factor**.

- A problem is that we may not have enough information during the early phases to make **accurate predictions about the effort needed**, and hence the cost, of **later phases**.

Waterfall Model: The Original Theory

The common understanding of the classical waterfall model maintains that **one should move to a phase only when its preceding phase is completed** and perfected.

Classical vs. Software Engineering

- A classical view compares **building a bridge, to constructing a software product**. The waterfall model works for bridges because bridge-building is well-understood

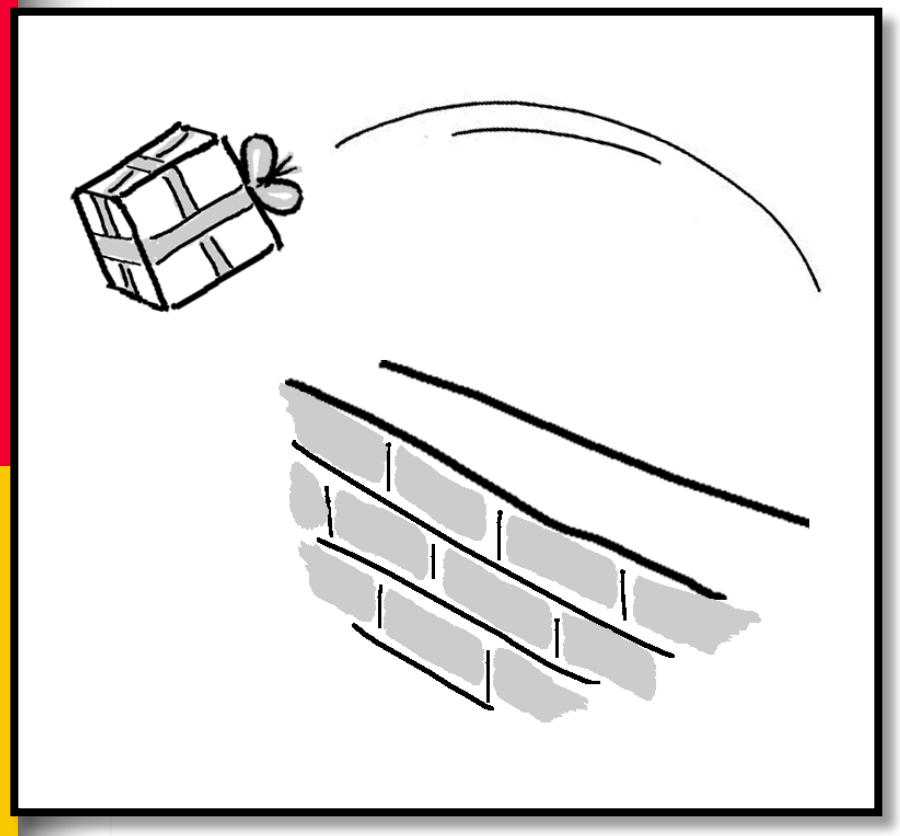
- The reasons that it does not work for programming :
  - the software development process is not well-understood &
  - **software requirements change. RAPIDLY.**

Pros of the Waterfall Model

Rigid and formal process, fits well for:

- Safety-critical systems
- Embedded systems
- Etc…

Cons of the Waterfall Model



Activities are isolated:

- Late-changing requirements require a lot of rework!

Next class

More Process models

Quiz

There are two metaphors for how software systems are created, they are building vs. growing? What is the difference between the two?