SOEN 341
Software Process

Lecture 03:
Process Models

Emad Shihab, PhD

Process 0

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

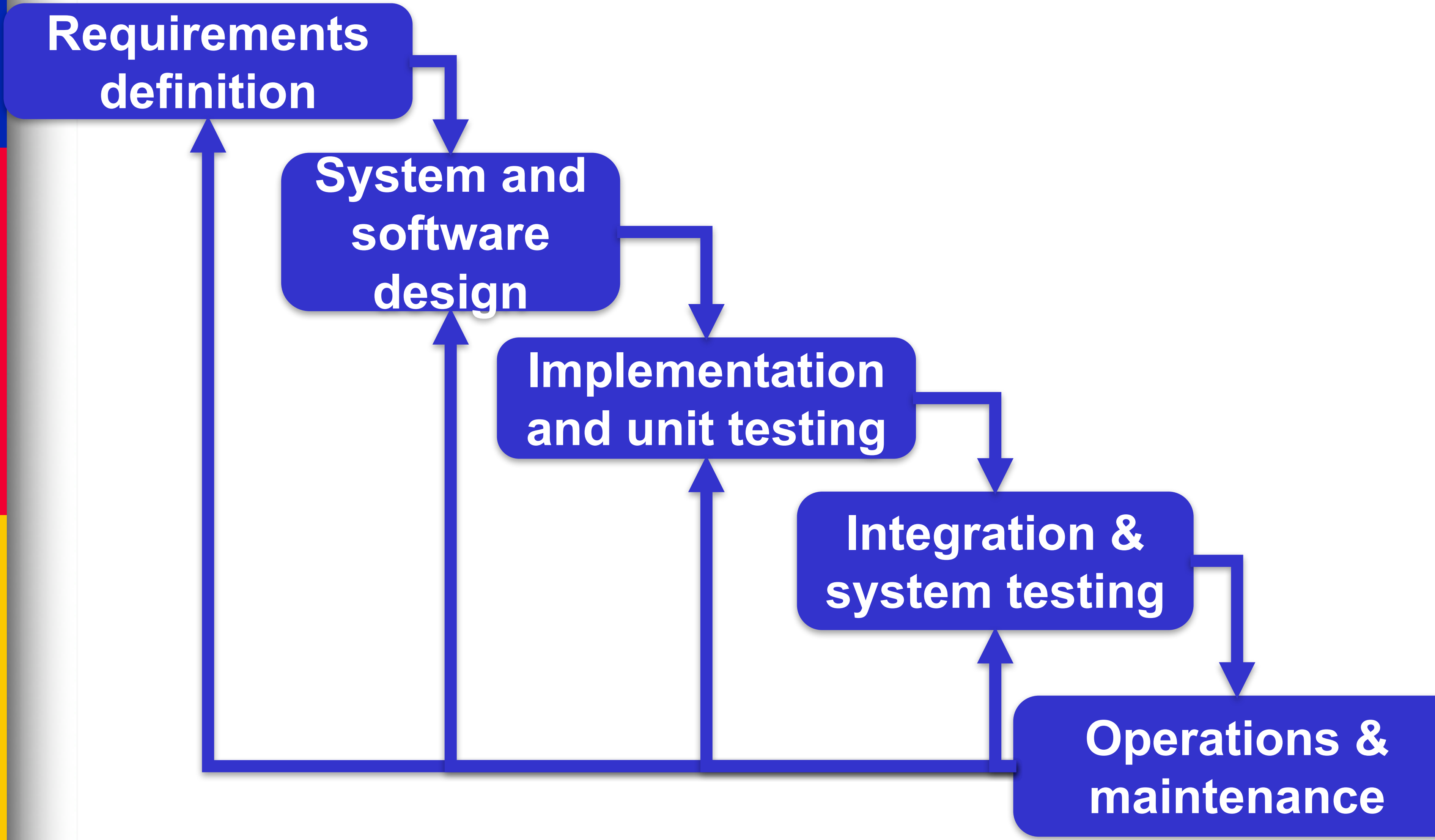
1. Write some code.
2. Fix the problems in the code.

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



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

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 easily **progress is 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

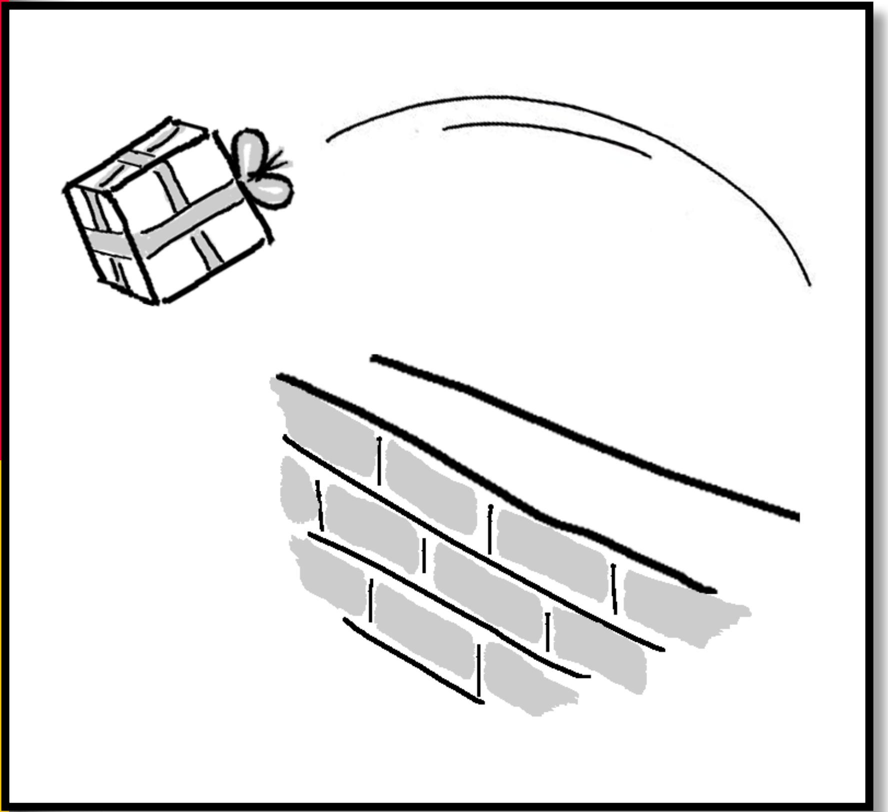
- The reasons that it does not work for programming :
  - the software development process is not well-understood &
  - **software requirements change. RAPIDLY.**

Pros of the Waterfall Model

Rigid and formal process, fits well for:

- Safety-critical systems
- Embedded systems
- Etc…

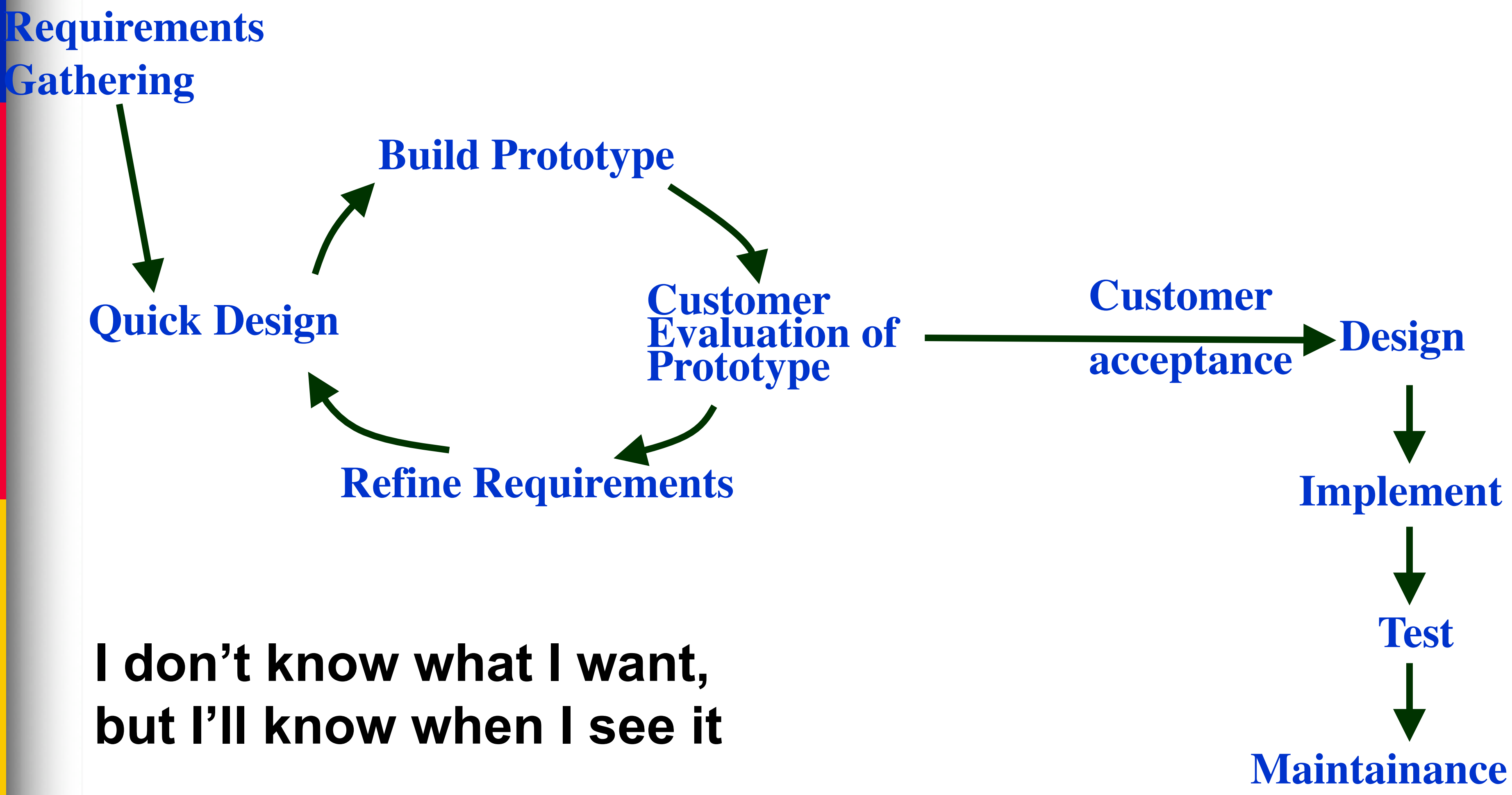
Cons of the Waterfall Model



Activities are isolated:

- Late-changing requirements require a lot of rework!

Process 1.5: Prototype/evolutionary model

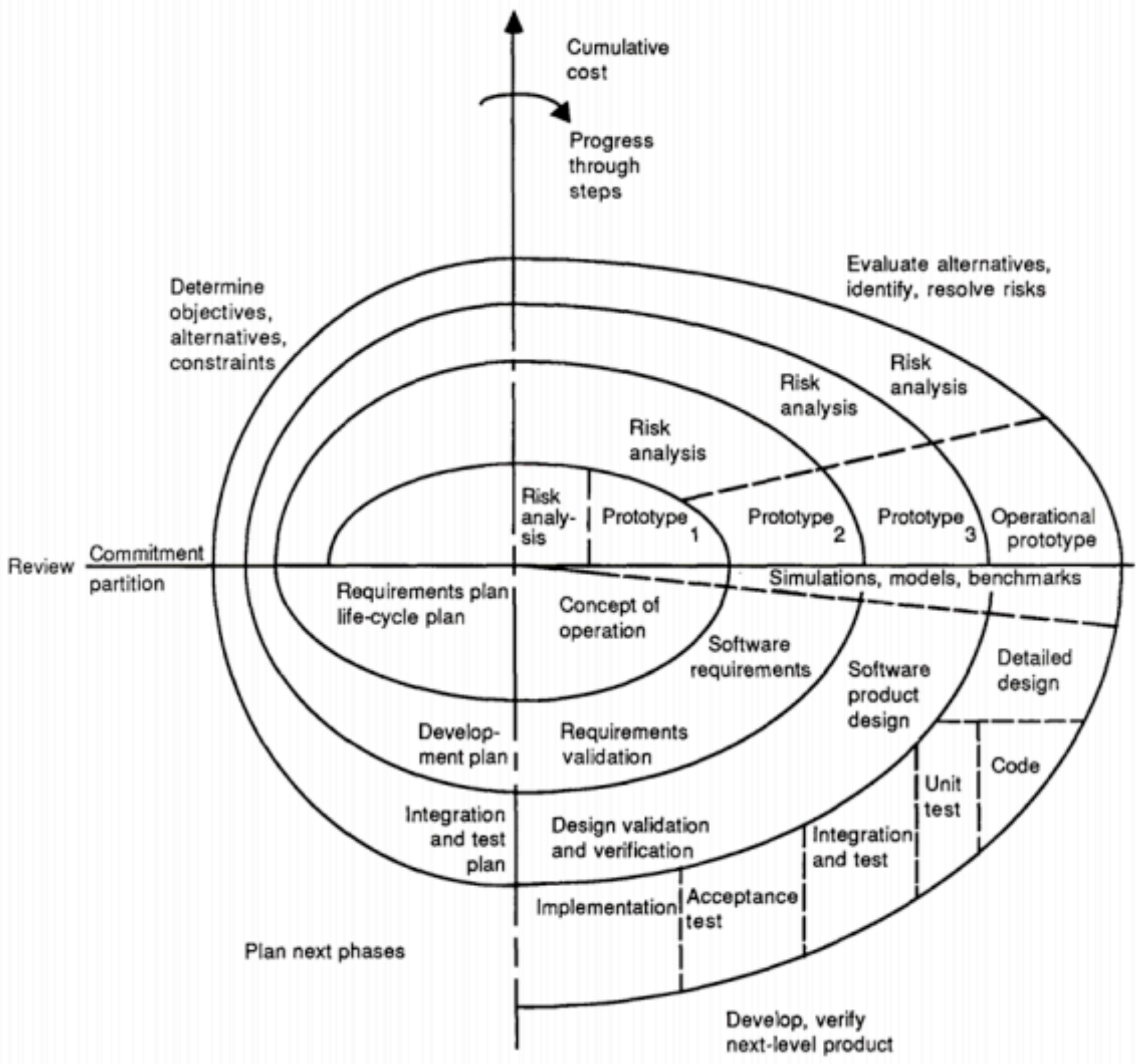


I don’t know what I want,
but I’ll know when I see it

Issues with the Prototype model

- It can be difficult to apply if you have **multiple, evolving applications** that you want to **integrate**
    - E.x., temporary **workarounds increasingly solidify** into unchangeable constraints

Process 2.0: Spiral Model



http://csse.usc.edu/TECHRPTS/1988/usccse88-500/usccse88-500.pdf

The Spiral Model-1

- Each phase **starts with a design goal** and ends with the **client reviewing progress**.
- The spiral model **combines features** of the **prototyping** model and the **waterfall** model.
- It is intended for large, expensive, and complicated projects.

The Spiral Model-2

- The **preceding steps are iterated until the customer is satisfied** that the refined prototype represents the (semi) final desired product.

- A **first prototype** of the new system is **constructed from the preliminary design** (a scaled-down system, an approximation of the of the final product)

The Spiral Model-3

The following prototypes are evolved by a fourfold procedure:

(1) **evaluating** the preceding prototype (e.x., document) in terms of its **strengths, weaknesses, and risks**;

(2) **defining the requirements** of the next prototype (e.x., document);

(3) **planning and designing** the next prototype (e.x., document);

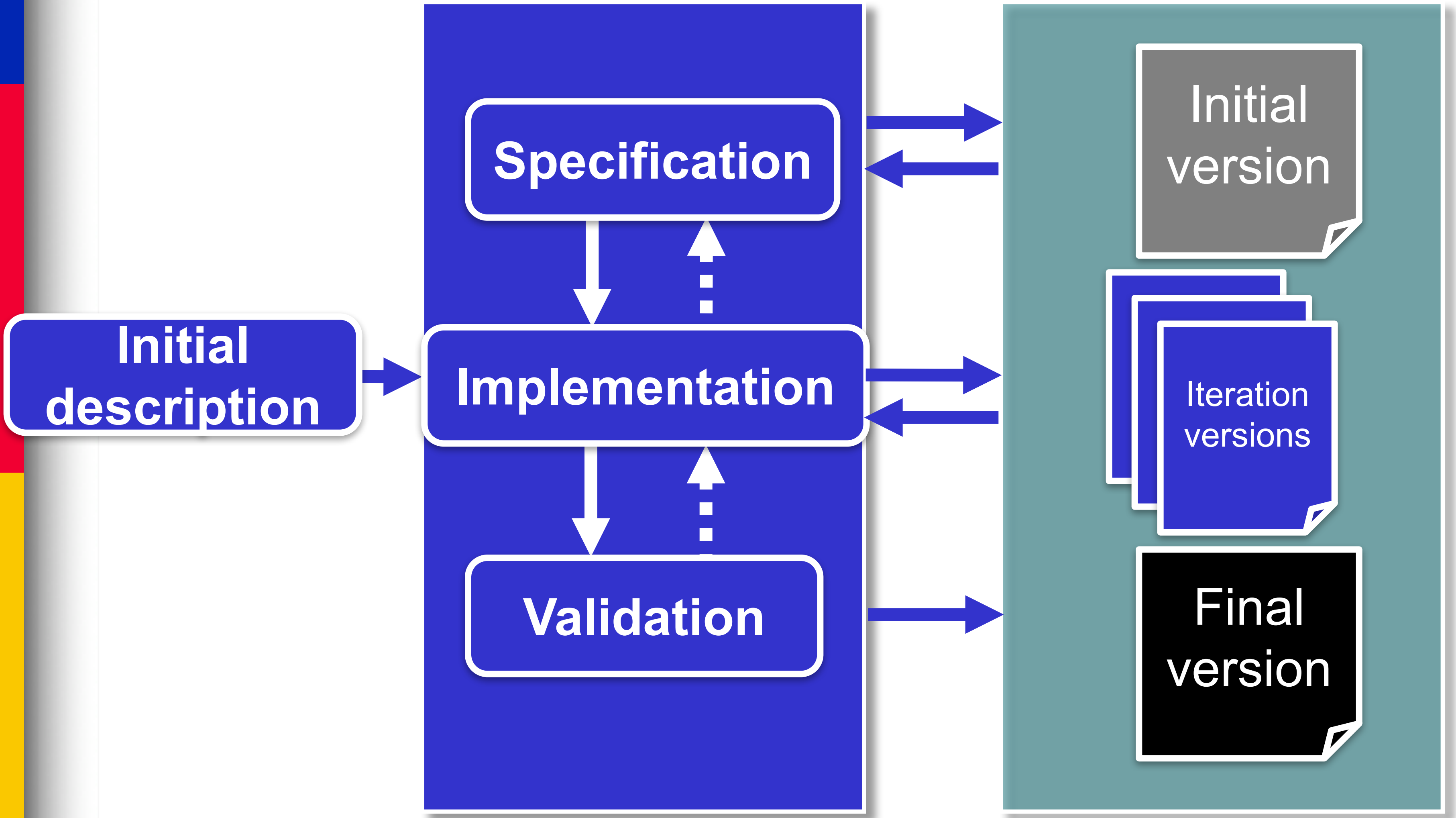
(4) **constructing and testing** the next prototype (e.x., document).

The Spiral Model-4 -Risk

- The project can be **aborted by the customer**, if the project is deemed **too risky**

- Risk factors might involve:
  - development cost overruns,
  - operating-cost miscalculation
  - any other factor that result in an unsatisfactory final product

Process 3: Incremental Development



Incremental Development/Growing Software

Advantages of incremental development:

- There is a **working system at all times;**
- **Clients** can see the system and **provide feedback**
- **Progress is visible**, rather than being buried in documents
- **Less error prone**

Errors

- **Errors made early** in development tend to be **more serious** (and more expensive to fix) than errors made later
  - E.x., consider an **error in the requirements**. With the **waterfall** model, the **error may not be detected** until **acceptance testing**, when it is probably too late to correct it.
  - (Note that the client probably does not see the software running until the acceptance tests)

Error Avoidance

- Even in a **large project** incremental development and prototyping **can help avoid extreme situations due to errors**

  - e.x., there is a good chance that a requirements **error will be recognized as soon as the corresponding software is built**. It is then not a big deal to correct it.
  - On the other hand, the **waterfall model relies on careful review of documents** to avoid errors. Once a phase has been completed, there is no stepping back.

Iterative Process Advantages

- **Can reach the design goals** of customers who do not know how to define what they want
- The **cost** of accommodating changing **customer requirements is reduced**
- It is **easier** to get customer **feedback**
- Customers are **able to use and gain value** from the software **earlier**

Applicability of the Iterative Process

Flexible and informal process, fits well for:

- Consumer software
- Web-based systems
- Mobile app systems

Iterative Process Challenges

- Software **architects** are still faced with the **challenge of creating a reliable foundation** upon which to develop.

- **Architectures** that is in flux tends to **degrade quickly**

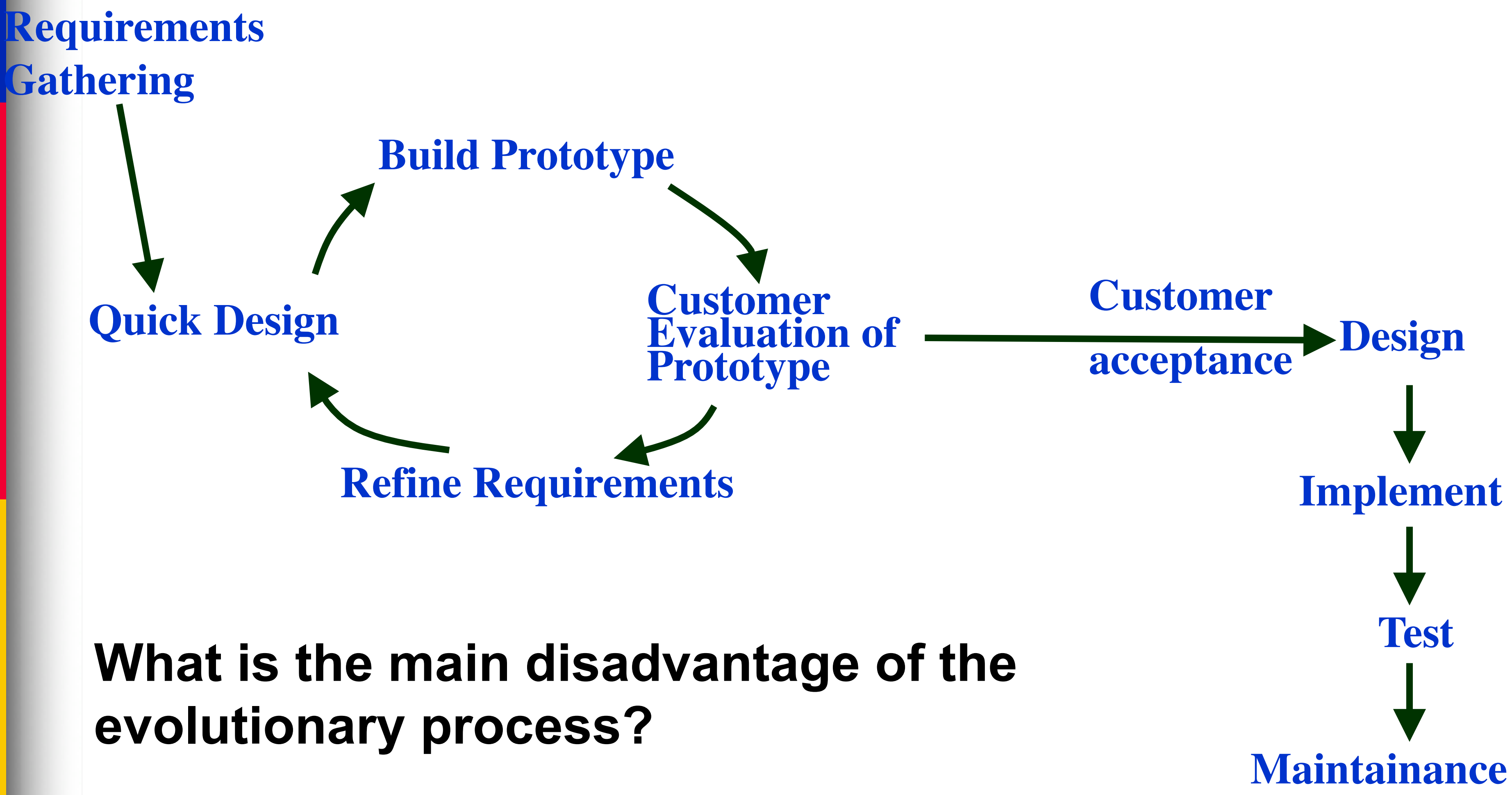
- **Large organizations** with many teams struggle to adopt

- Clients **see the possibility for change** (of req.) and want/demand it

Next class

Agile Methodology

Quiz



What is the main disadvantage of the evolutionary process?