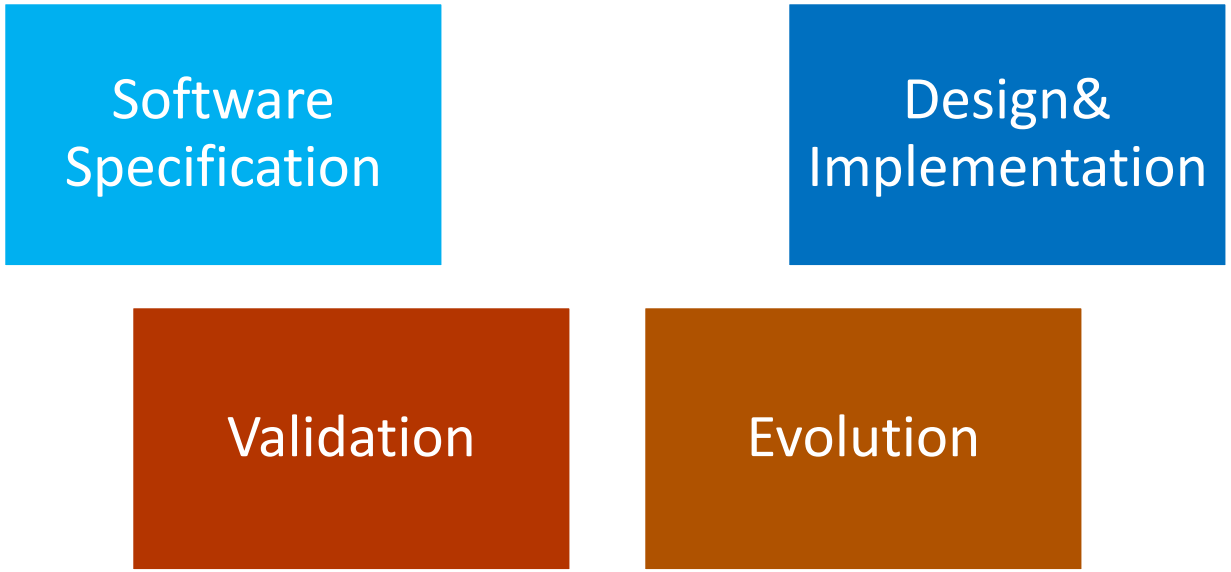
Software Development Processes

Chapter 2

Software Processes: Intro

- A software process is a set of related activities that leads to the production of a software product.
- There are many different software processes but all must include four activities that are fundamental to software engineering are:



Software Processes: intro..2

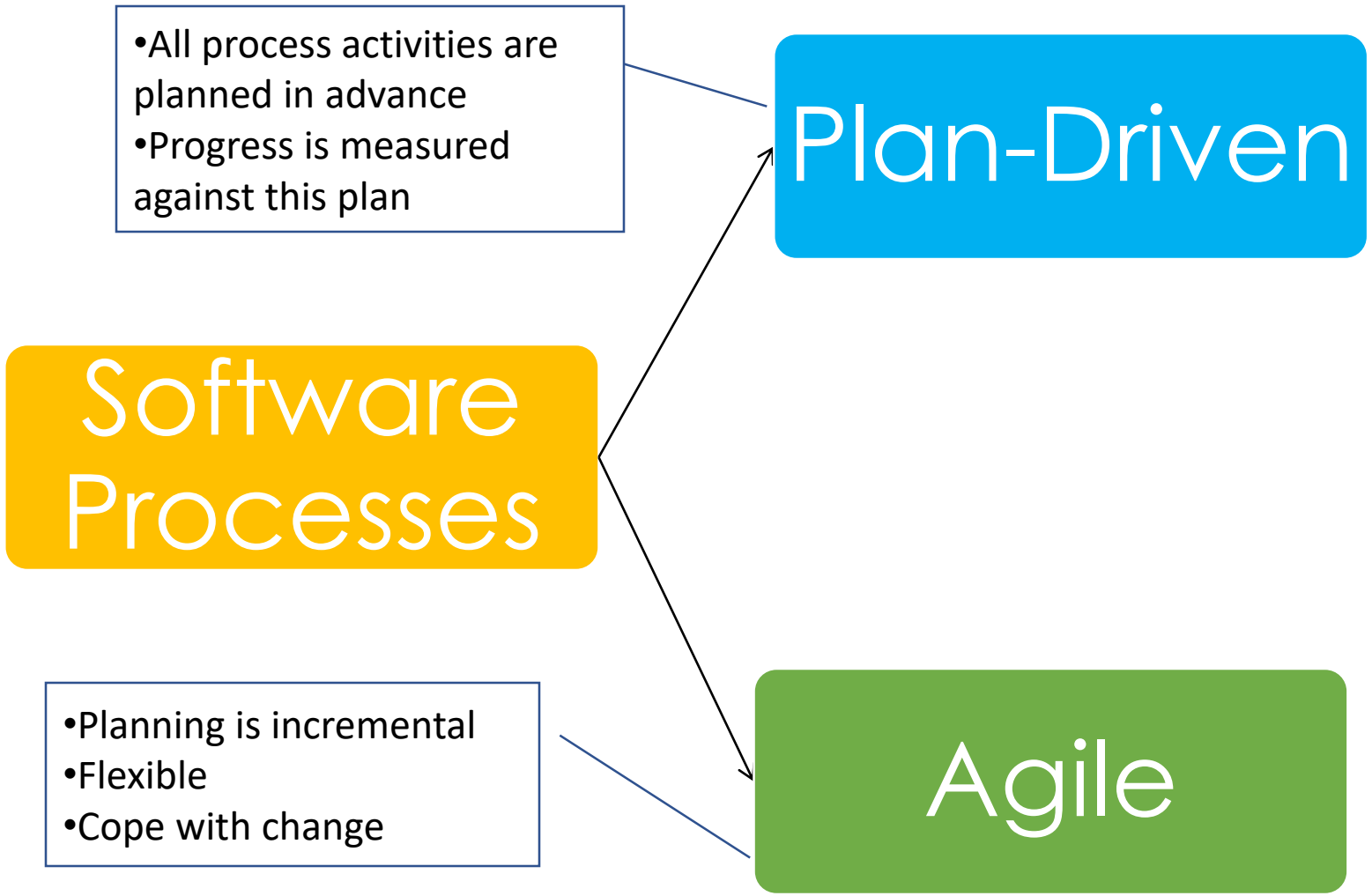
- When we talk about processes, it is not only about the activities and their order.
- It is also about:
  - **Products: outcomes of each activity, example Software Architecture Document**
  - **Roles: programmer, designer, analyst, tester, team leader, etc.**
  - **Pre and Post Conditions: example**
    - **Software requirements must be ready before building software architecture.**
    - **UML models must be designed and reviewed**.

Software Process: Intro..2

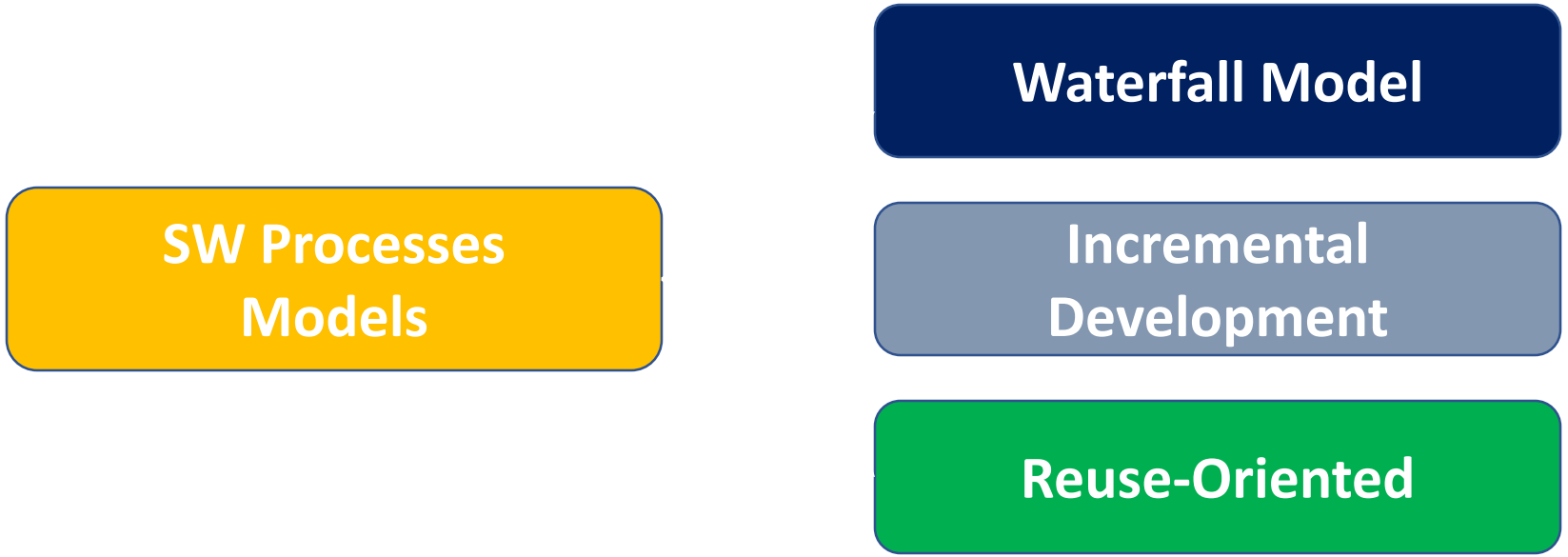
- Software processes **are complex** and, like all intellectual and creative processes, **rely on people making decisions and judgments**.

- There is **no ideal process** and most organizations have developed their **own**.
- Processes have evolved to take advantage of the specific characteristics of the systems that are being developed.
  - **Critical Systems: a very structured development process is required**
  - **A Business System with rapidly changing requirements, a less formal, flexible process is likely to be more effective**

Two Categories of Software Processes



Software Process Models

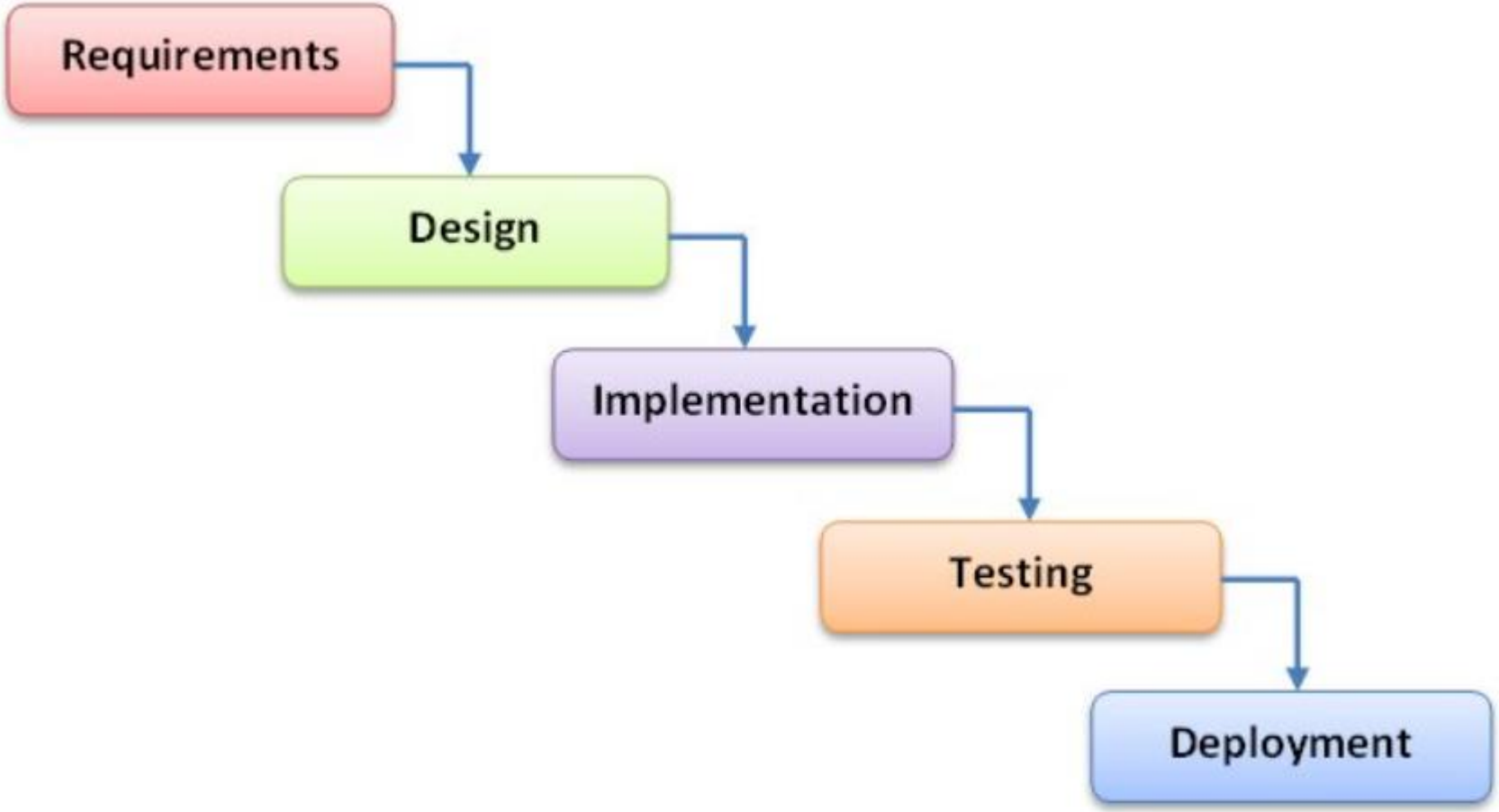


- a software process model is a simplified representation of a software process.
- **These models are not mutually exclusive and are often used together, especially for large systems development.**

Waterfall Model

- Derived from more general system engineering processes
- Plan- driven process
- you must plan and schedule all of the process activities
- In principle, the result of each phase is one or more documents that are approved before starting work on them
- It is best used for critical systems, where requirements are well understood and will not change (eg. Aviation and medical systems)
- It can also be used with small systems (mobile apps) with the help of prototypes.

Waterfall Model



Advantages of Waterfall Model

- Developers and customers agree on what will be delivered early in the development lifecycle. This makes **planning and designing more straightforward**.
- Progress is more **easily measured**, as the full scope of the work is known in advance.
- Throughout the development effort, it's possible for various members of the team to be involved or to continue **with other work,** depending on the active phase of the project
- customer presence is **not strictly required** after the requirements phase.
- The software can be designed **completely and more carefully**, based upon a more complete understanding of **all** software deliverables

References: http://www.seguetech.com/waterfall-vs-agile-methodology/

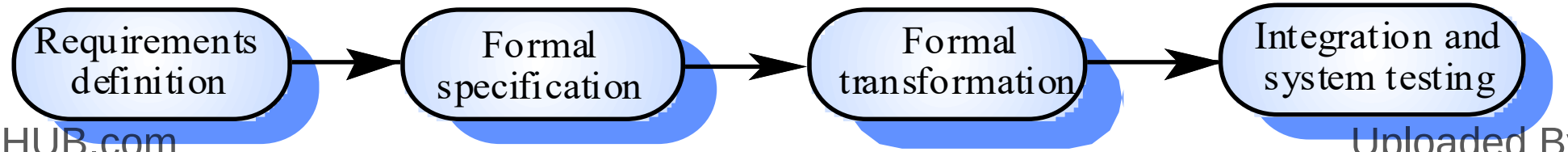
Disadvantages of Waterfall Model

- One area which almost always falls short is the effectiveness of **requirements**.
- Gathering and documenting requirements in a way that is meaningful to a customer is often **the most difficult part** of software development.
- Another potential drawback of pure Waterfall development is the possibility that the customer will be **dissatisfied** with their delivered software product
- And **Testing** of whole system that only happens at end of project

Formal System Development

- An important variant of the waterfall model is formal system development, where a **mathematical model** of a system specification is created.
- This model is then refined, using mathematical transformations that preserve its consistency, into executable code
- Is particularly suited to the development of systems that have stringent **safety, reliability, or security** requirements.
- The formal approach simplifies the production of a safety or security case.

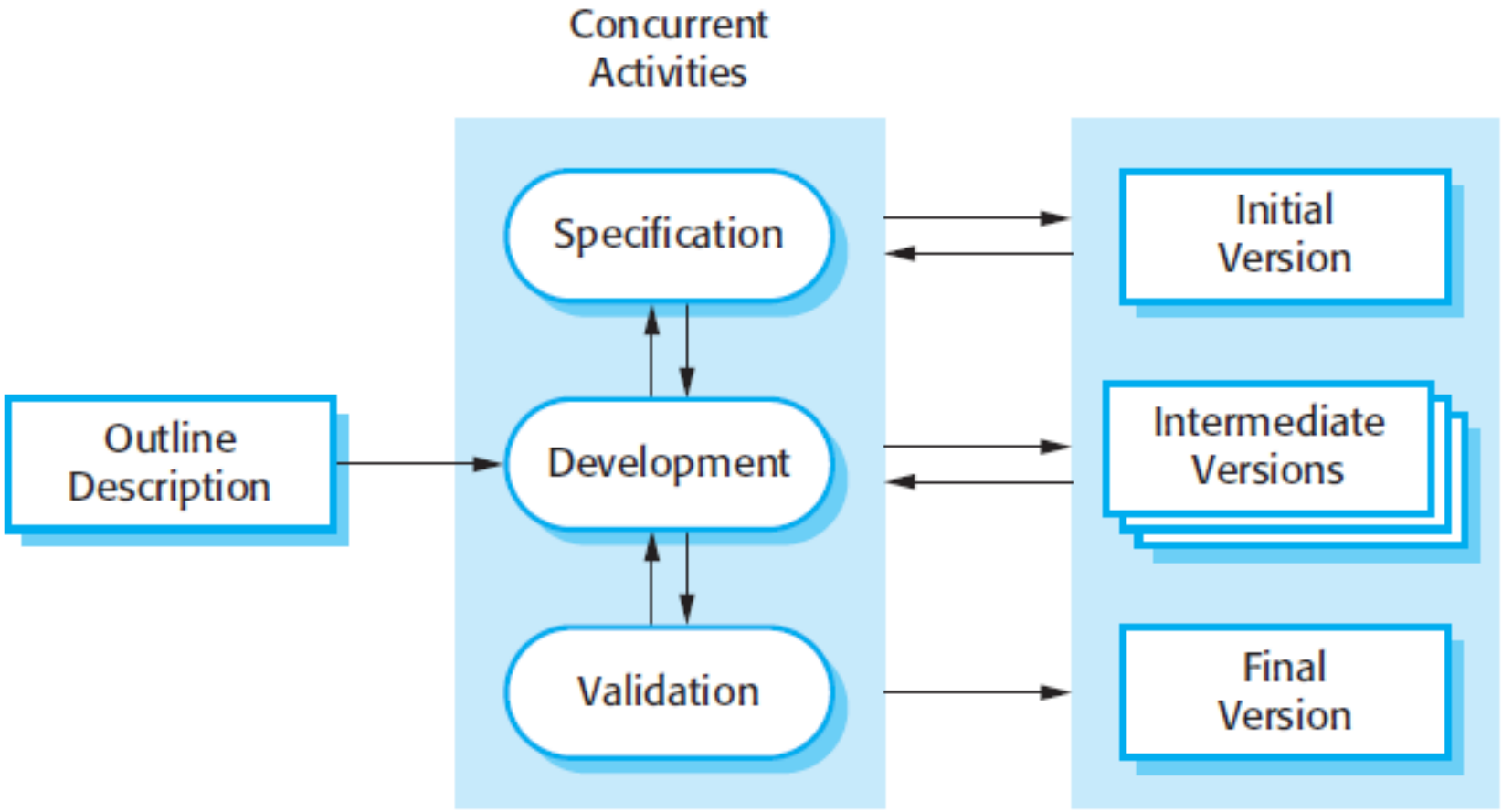
Birzeit University,
CS Dept, Samer
Zein (Ph.D)



Incremental Development

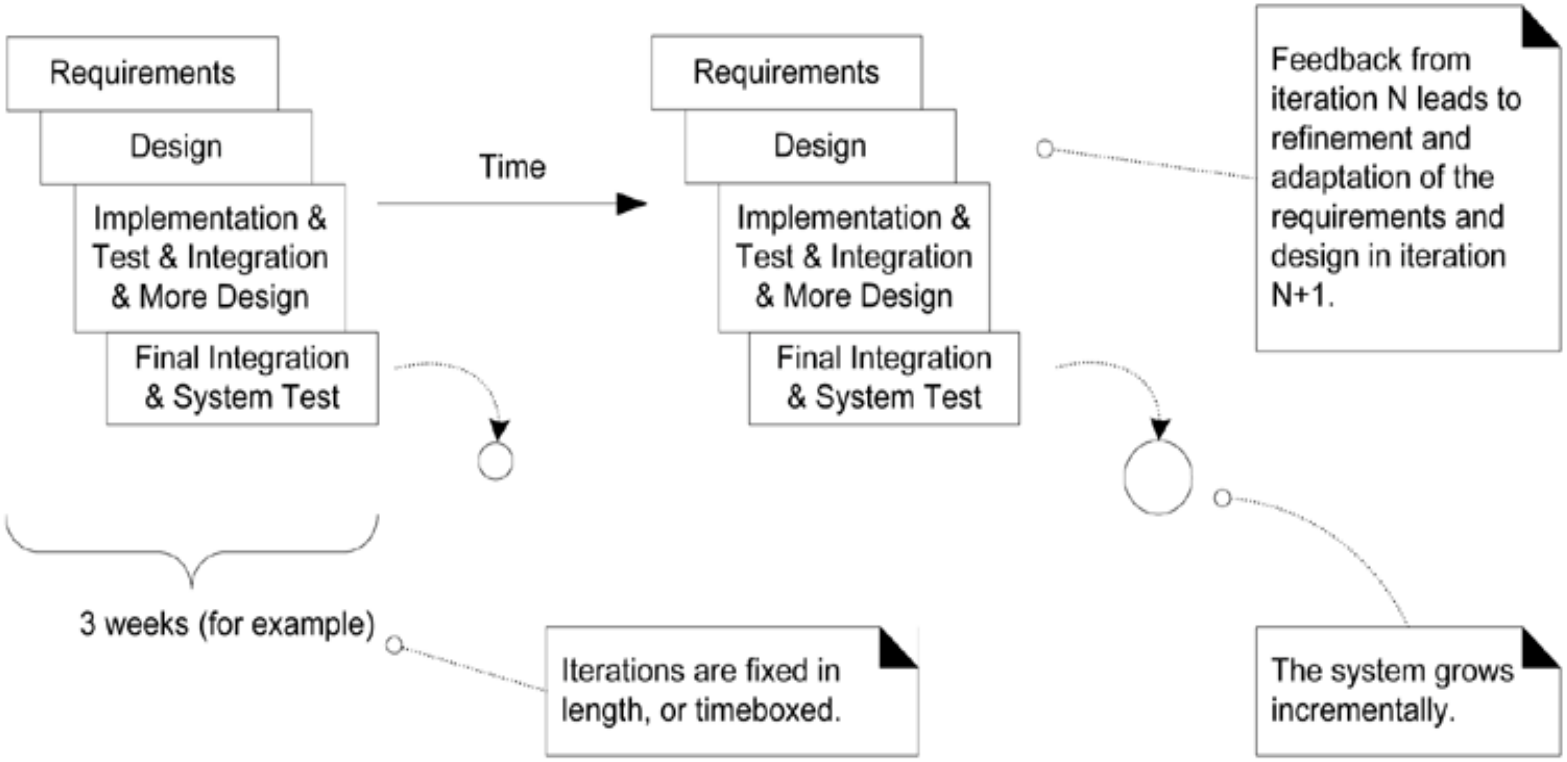
- Incremental development is based on the idea of
  - A) developing an initial implementation,
  - B) exposing this to user comment and
  - C) evolving it through several versions until an adequate system has been developed
- fundamental part of **agile** approaches
- Better than waterfall approach for most e-business, e-commerce, and personal systems.
- **Can be plan-driven, agile, or a mix of both!**

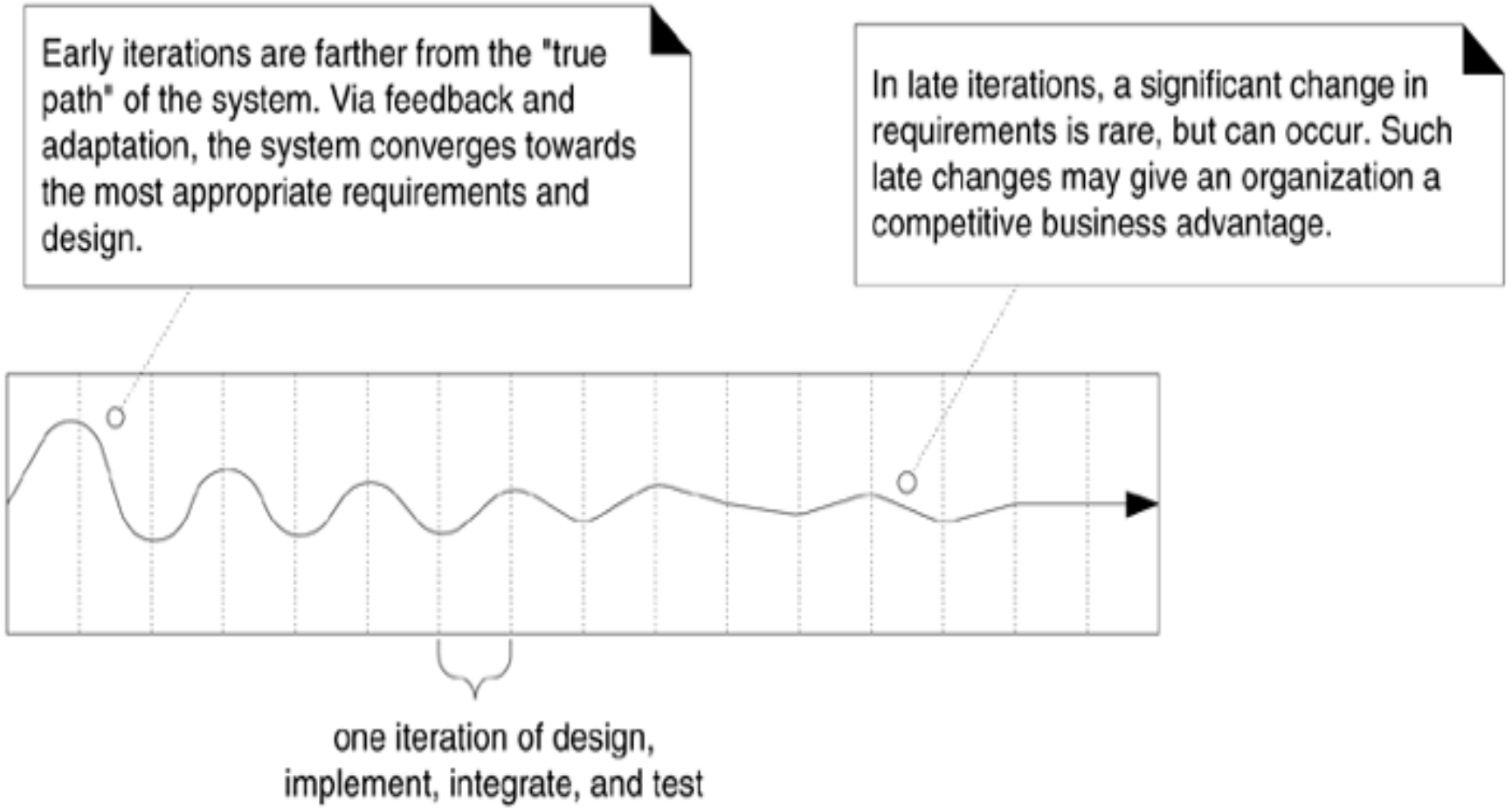
Incremental Development Model



Core Practices - Iterations

– ITERATIVE, EVOLUTIONARY, AND AGILE





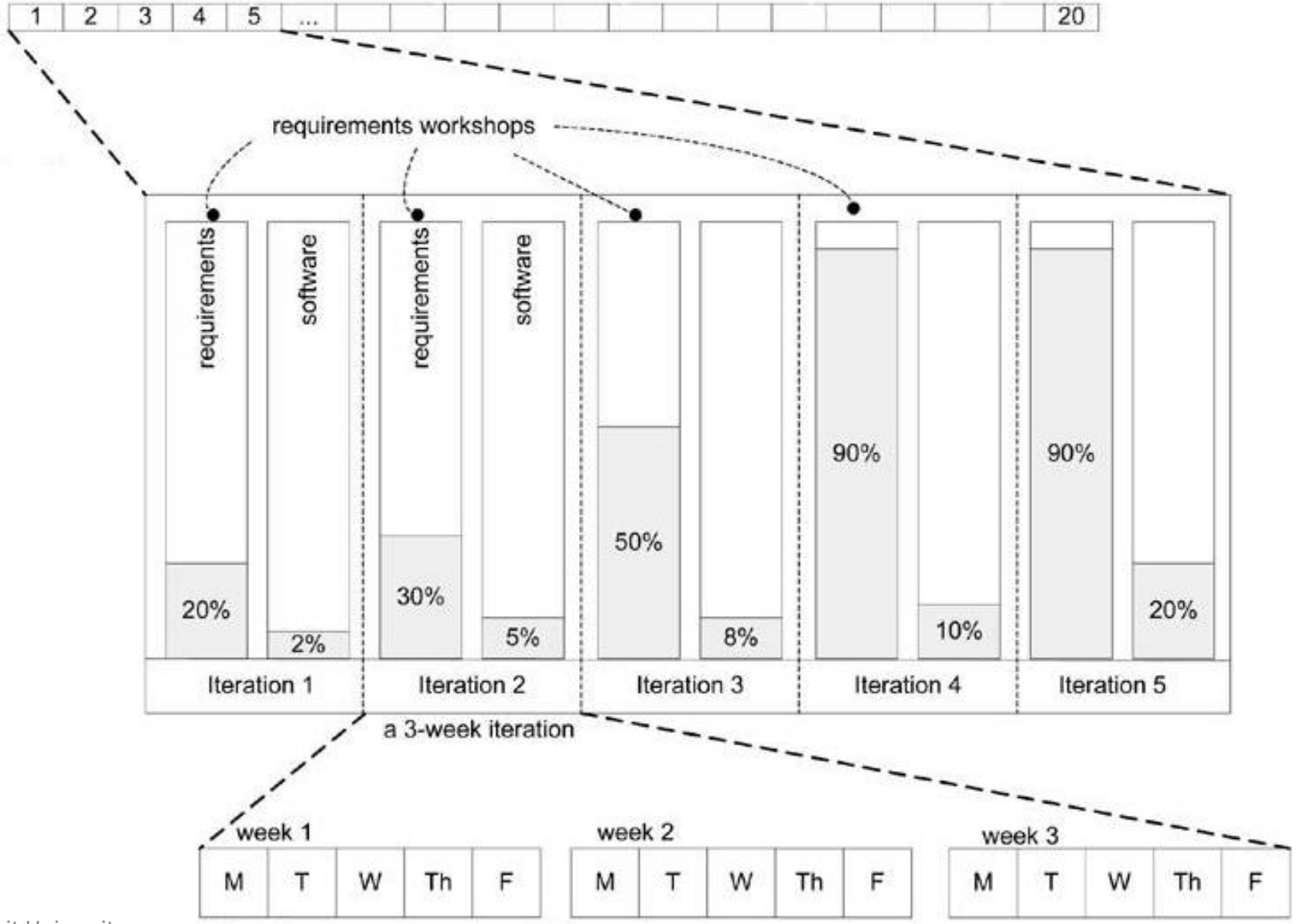
Incremental Development..3

- Benefits of incremental development
  - The **cost** of accommodating **changing customer requirements** is reduced.
  - It is easier to get customer **feedback** on the development work that has been done.
  - More **rapid delivery** and deployment of useful software to the customer is possible
  - Better fit for short time-to-market
- Problems with incremental development
  - The process is **not visible**. Managers need regular deliverables to measure progress.
  - System **structure** tends to degrade as new increments are added.
  - **Additional unplanned** iterations may be needed.
  - Customer may **not** have the required **free time** to be involved.
  - Team may **not** be located in **same place** (distributed teams)

How to do Iterative and Evolutionary Analysis and Design?

- Suppose that iteration period = 4 weeks

Week 1	Week 2	Week 3	Week 4



How to plan an iteration?

- **Step #1** is to decide the length of the iteration; 2 - 6 weeks is the common range
- **Step #2** is to convene an iteration planning meeting
- **Step # 3** A list of potential goals (new features or use cases, defects, …) for the iteration is presented, ranked by some priority scheme

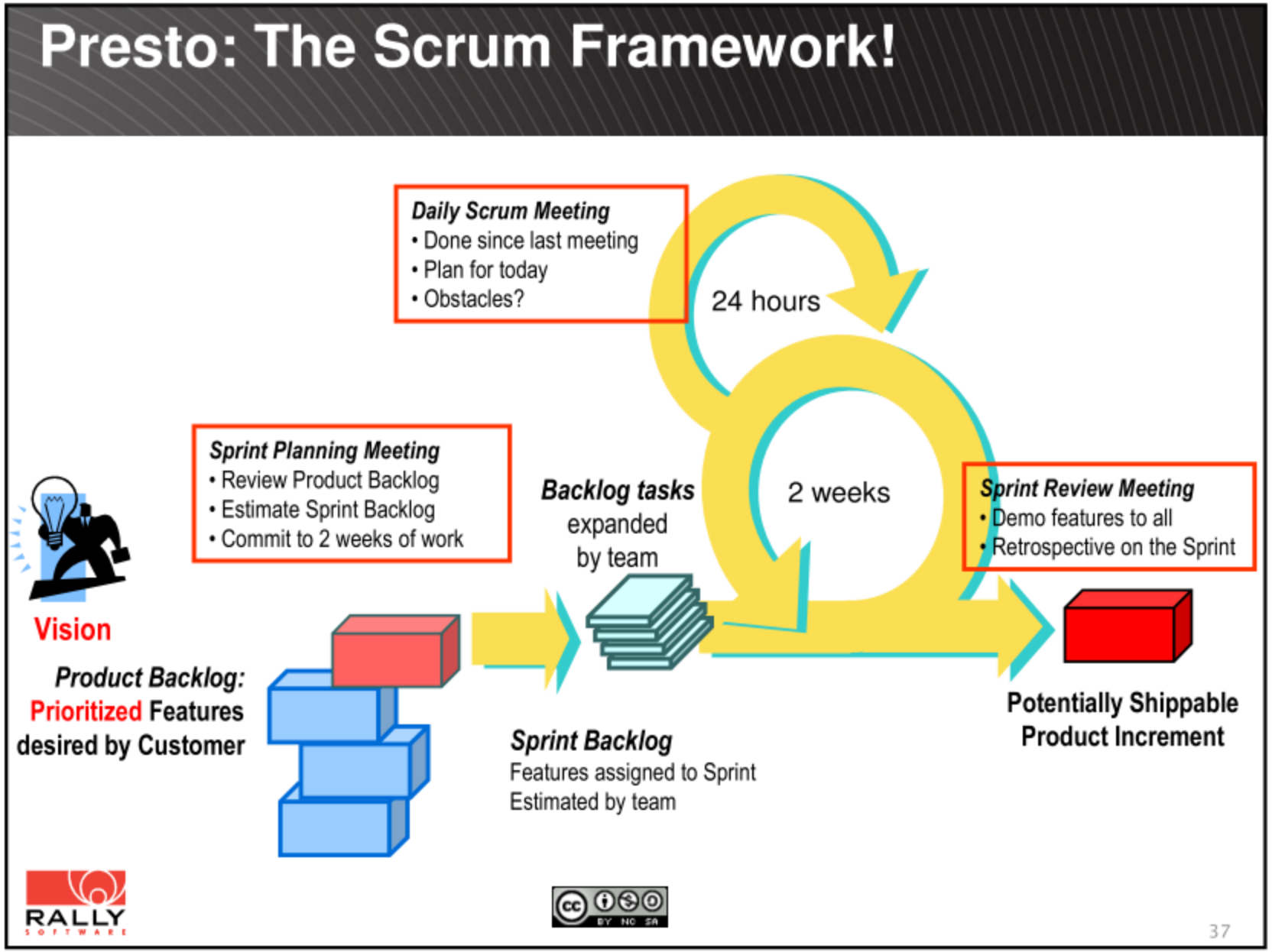
- **Step # 4** Each member of the team is asked for their individual resource budget

- **Step # 5** For one goal (such as a use case), it is described in some detail. For example, UI tasks, database tasks, domain layer OO development tasks, external systems integration tasks, and so forth.

- All the task estimates are summed into a running total

- **Step #6** repeats until enough work has been chosen: The iteration task total is divided by the resource budget total

Presto: The Scrum Framework!



Reuse-Oriented Software Engineering

- In the majority of software projects, there is some software reuse.

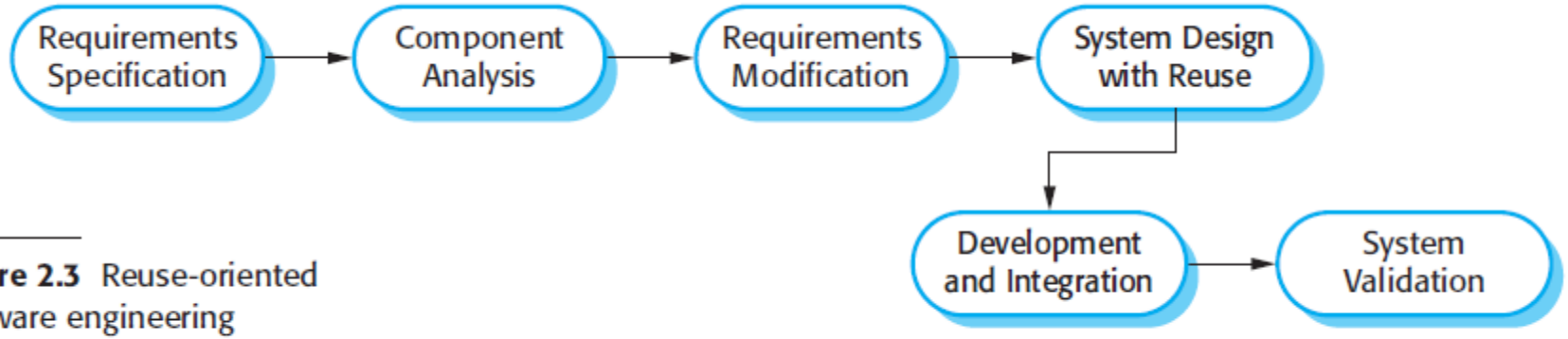


Figure 2.3 Reuse-oriented software engineering

Types of Reuse-Oriented Software Engineering

There are three types of software component that may be used in a reuse-oriented process:

1. Web services that are developed according to service standards and which are available for remote invocation.
2. Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.
3. Stand-alone software systems that are configured for use in a particular environment.

Pros & Cons for Reuse Oriented Model

- **Problems**

• Need for specialized (component) analysis and integration skills to ensure appropriate selection of components, for both functionality and quality aspects.

• Some aspects (or parts) of the system may not be easily reused, such as the user interface

• Concerns over maintainability and support of reused components

• Concerns over system evolution that development is controlled by reused component suppliers.

Applicability

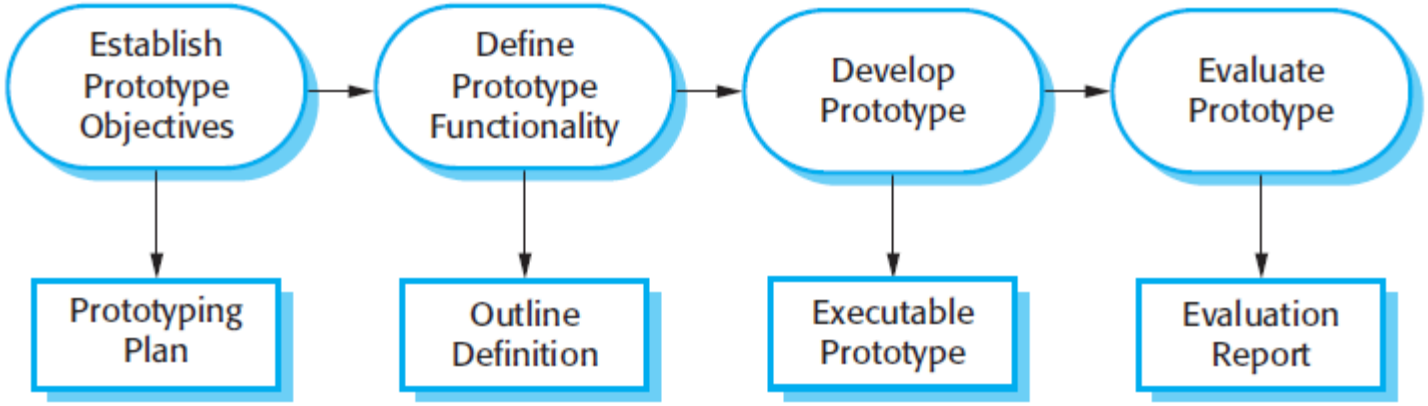
Prototyping

- A prototype is an initial version of a software system that is used to demonstrate:
  - **concepts, try out design options, and find out more about the problem and its possible solutions**
- A software prototype can be used in a software development process to help anticipate changes that may be required:
  - **Requirements engineering**
  - **System design, UI design**

Prototyping 2

- System prototypes allow users to see how well the system supports their work.
- They may get new ideas for requirements, find **strength** and **weakness** in the software.
- They may then propose new system requirements.
- It may also reveal **errors** and **omissions** in the requirements.
- A function described in a specification may seem useful and well defined.
- But, when that function is combined with other functions, **users often find that their initial view was incorrect or incomplete**.
- The system specification may then be modified to reflect their changed understanding of the requirements.

Prototyping 3



- A general problem with prototyping is that the prototype may not necessarily be used in the same way as the final system.

Prototyping 3

- A system prototype may be used while the system **is being designed** to carry out design **experiments** to check the **feasibility** of a proposed design.

- For example, a **database design** may be prototyped and tested to check that it supports **efficient data access** for the most common user queries.

Incremental Delivery

- Is an approach to software development where **some of the developed increments** are delivered to the customer and deployed for use in an operational environment.
- In an incremental delivery process, customers identify, in outline, the services to be provided by the system.
- They identify which of the **services** are **most important** and which are least important to them.
- A number of delivery increments are then defined, with each increment providing a sub-set of the system functionality.
- With the **highest-priority** services implemented and delivered first.

Advantages of Incremental Delivery

1. Customers can use the early increments as prototypes and gain experience that informs their requirements for later system increments. Unlike prototypes, these are part of the real system so there is no re-learning when the complete system is available.
2. Customers do not have to wait until the entire system is delivered before they can gain value from it. The first increment satisfies their most critical requirements so they can use the software immediately.
3. The process maintains the benefits of incremental development in that it should be relatively easy to incorporate changes into the system.
4. As the highest-priority services are delivered first and increments then integrated, the most important system services receive the most testing. This means

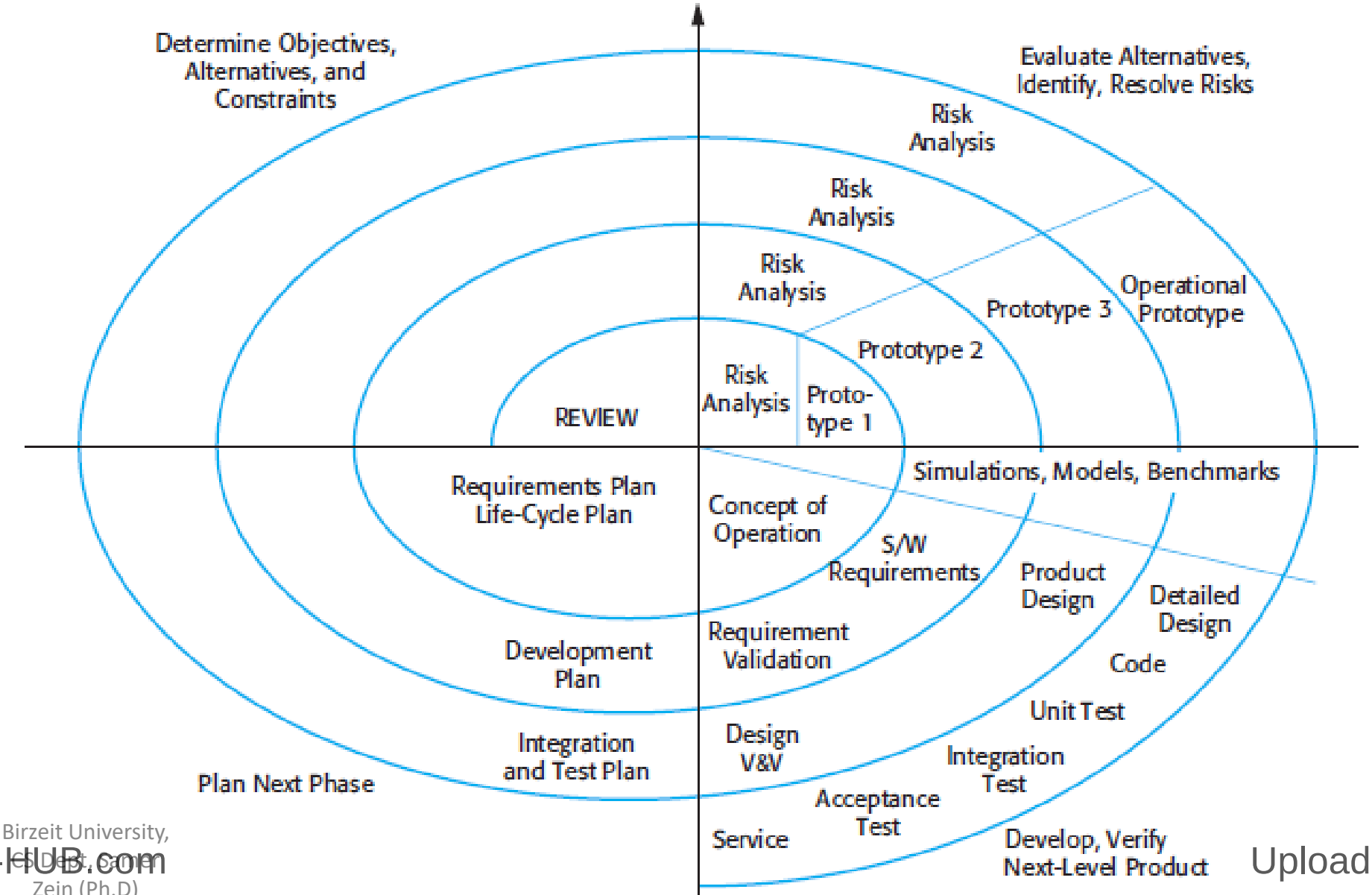
However, there are problems with incremental delivery:

1. Most systems require a set of basic facilities that are used by different parts of the system. As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.
2. Iterative development can also be difficult when a replacement system is being developed. Users want all of the functionality of the old system and are often unwilling to experiment with an incomplete new system. Therefore, getting useful customer feedback is difficult.
3. The essence of iterative processes is that the specification is developed in conjunction with the software. However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract. In the incremental approach, there is no complete system specification until the final increment is specified. This requires a new form of contract, which large customers such as government agencies may find difficult to accommodate.

Spiral Model of Bohem

- A **risk-driven** software process framework
- Each loop in the spiral represents a phase of the software process.
- Thus, the innermost loop might be concerned with system **feasibility**, the next loop with **requirements** definition, the next loop with system **design**, and so on.
- includes explicit **risk management** activities
- to reduce these risks.

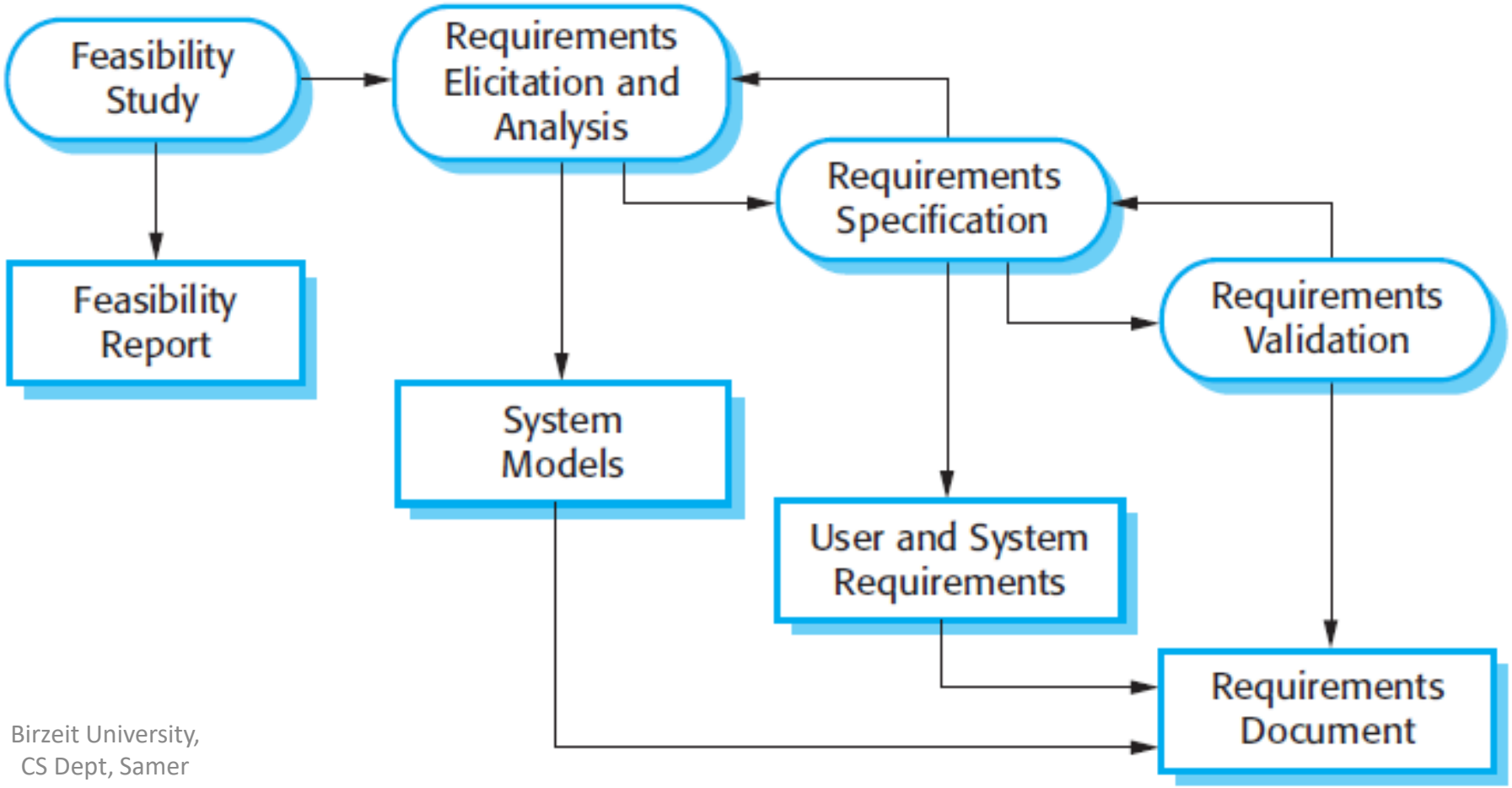
Spiral Model of Bohem



Process Activities: Software Specifications

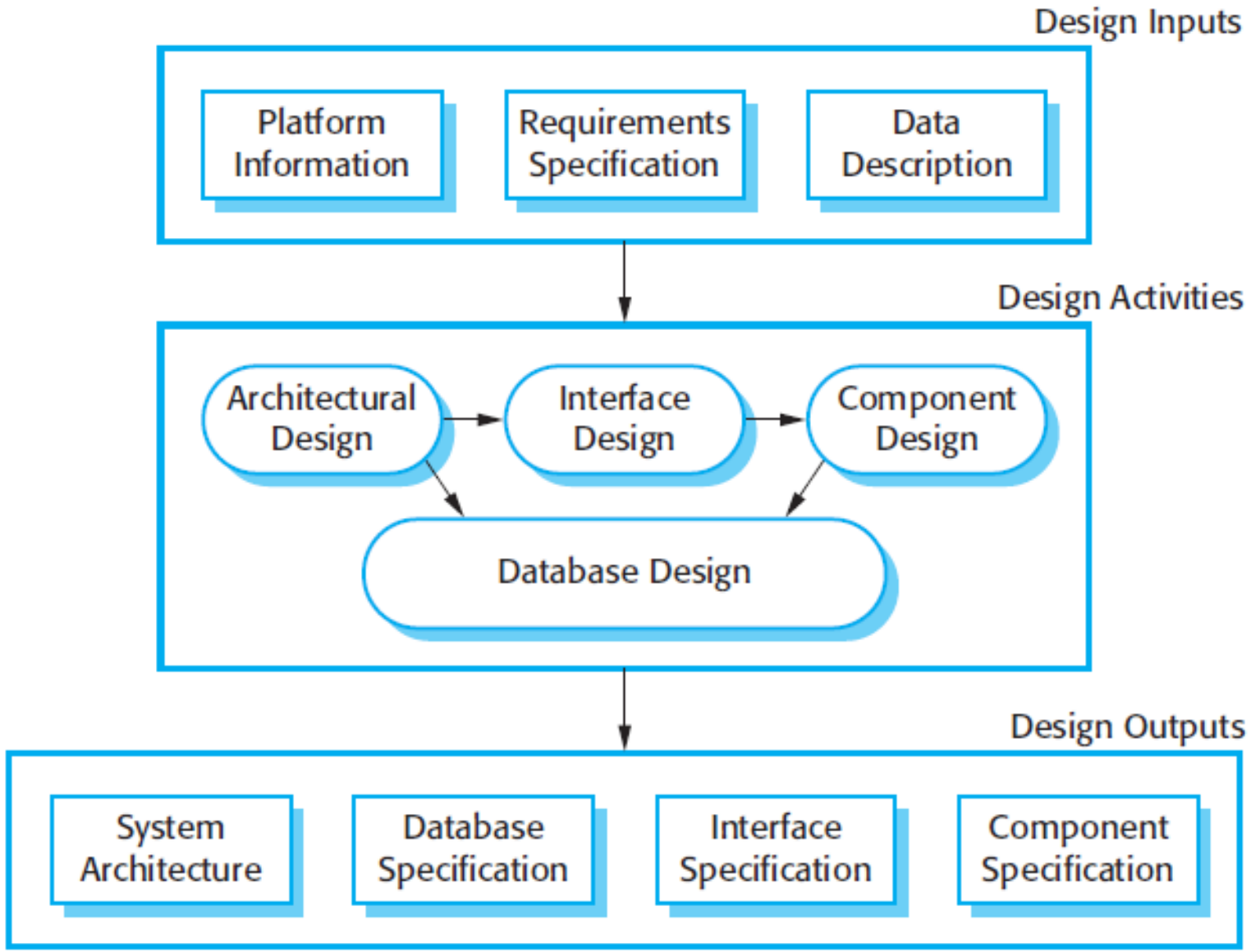
- Software specification or requirements engineering is the process of understanding and defining:
  - what services are required from the system
  - and identifying the constraints on the system's operation and development
- Requirements are usually presented at two levels of detail.
  - End-users and customers need a high-level statement of the requirements;
  - system developers need a more detailed system specification.

Process Activities: Software Specifications, 2



Process Activities: Software Design & Implementation

- A software design is a description of:
  - the structure of the software to be implemented,
  - the data models and structures used by the system,
  - the interfaces between system components
  - and, sometimes, the algorithms used.
- Creating detailed designs for critical systems.
- Generating code from designs and diagrams



Process Activities: Validation

- intended to show that a system both conforms to its **specification** and that it meets the **expectations** of the system customer.
- Program testing, where the system is executed using simulated test data, is the principal validation technique.
- **Alpha** Testing VS **Beta** Testing