Software Process Models

Momna Zaneb

Week 2

- Software Process
- Umbrella Activities
- Capability Maturity Model (CMM)
- The Linear Sequential Model
- Prototyping
- Unified Model

Software Process

When you build a product or system, it's important to go through a series of predictable steps—a road map that helps you create a timely, high-quality result. The road map that you follow is called a '*software process*.'

It provides stability, control, and organization to an activity that can, if left uncontrolled, become quite chaotic.

Software Process

Software engineers and their managers adapt the process to their needs and then follow it. In addition, the people who have requested the software play a role in the software process.

Generic Software Process Framework

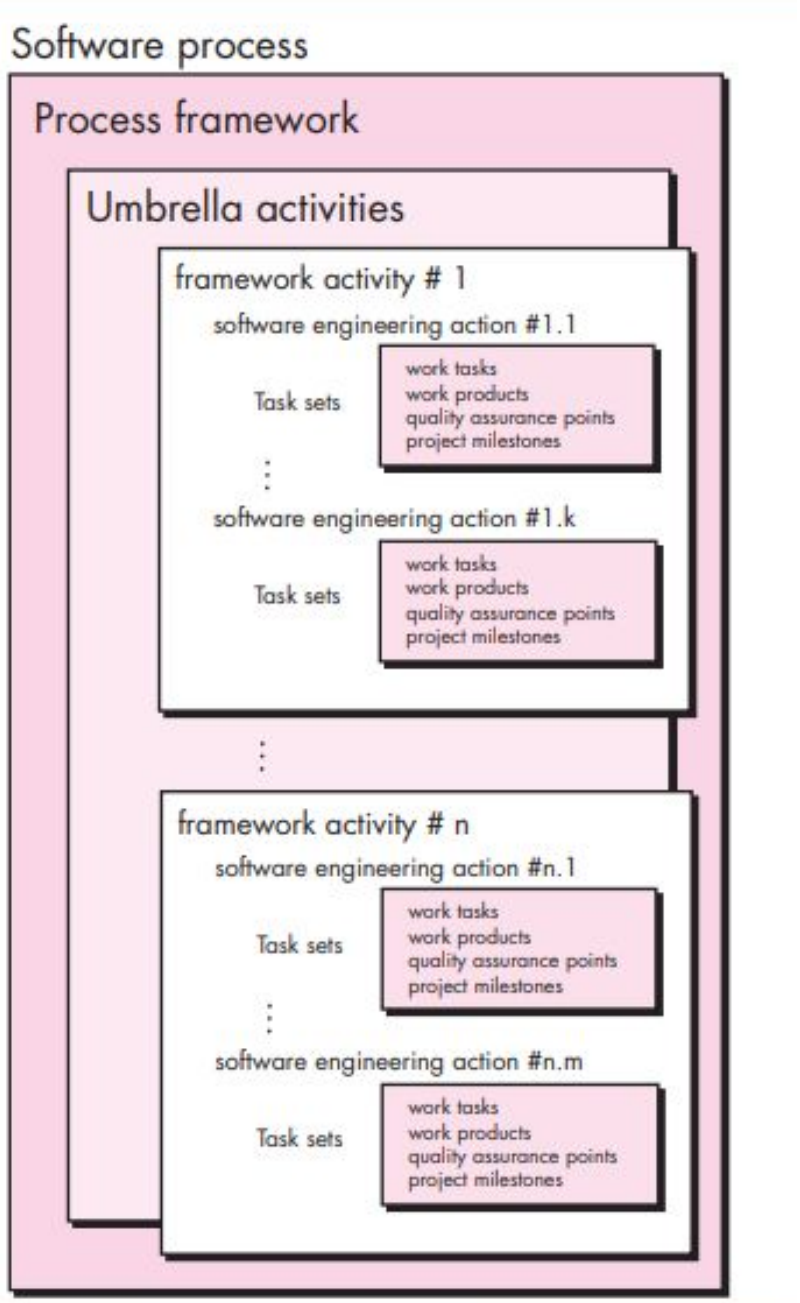
A process framework establishes the foundation for a complete software engineering process by identifying a small number of framework activities that are applicable to all software projects, regardless of their size or complexity.

- Communication
- Planning
- Modeling
- Construction
- Deployment

Umbrella Activities

Umbrella activities are applied throughout the software process. Typical umbrella activities are the following.

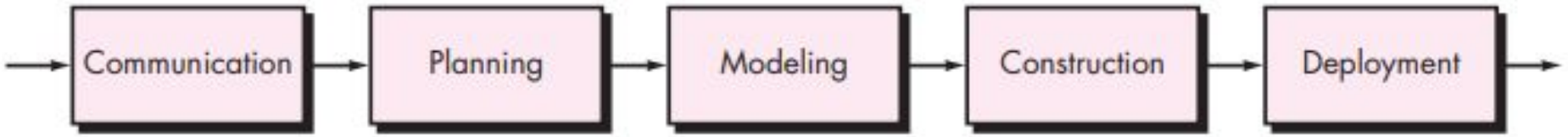
- Software project tracking and control
- Formal technical reviews
- Software quality assurance
- Software configuration management
- Document preparation and production
- Reusability management
- Measurement
- Risk management



- Process Framework Activities are supported by Umbrella activities.
- Each Framework Activity contains Software Engineering Actions.
- Each Action can contain one or more tasks. (can also be called task sets)

Process Flow

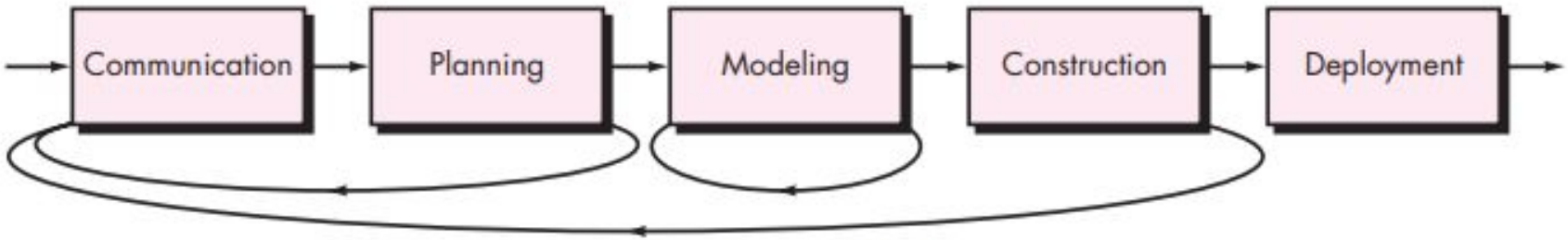
- A linear process flow executes each of the five framework activities in sequence, beginning with communication and culminating with deployment (Figure 2.2a).



(a) Linear process flow

Process Flow

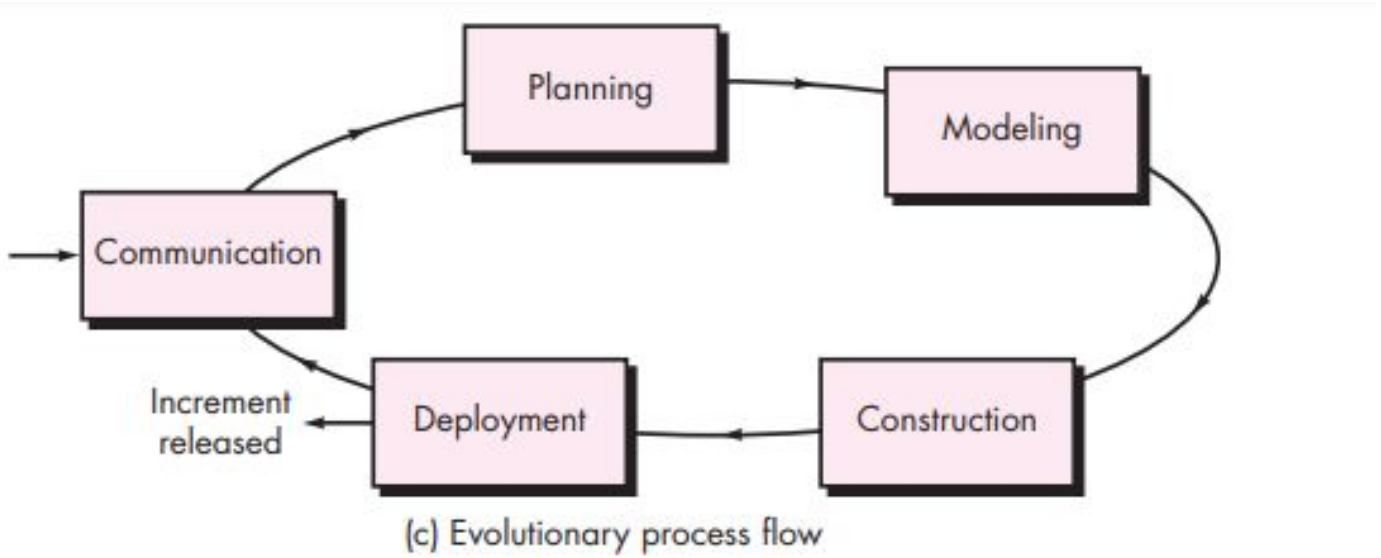
- An iterative process flow repeats one or more of the activities before proceeding to the next (Figure 2.2b).



(b) Iterative process flow

Process Flow

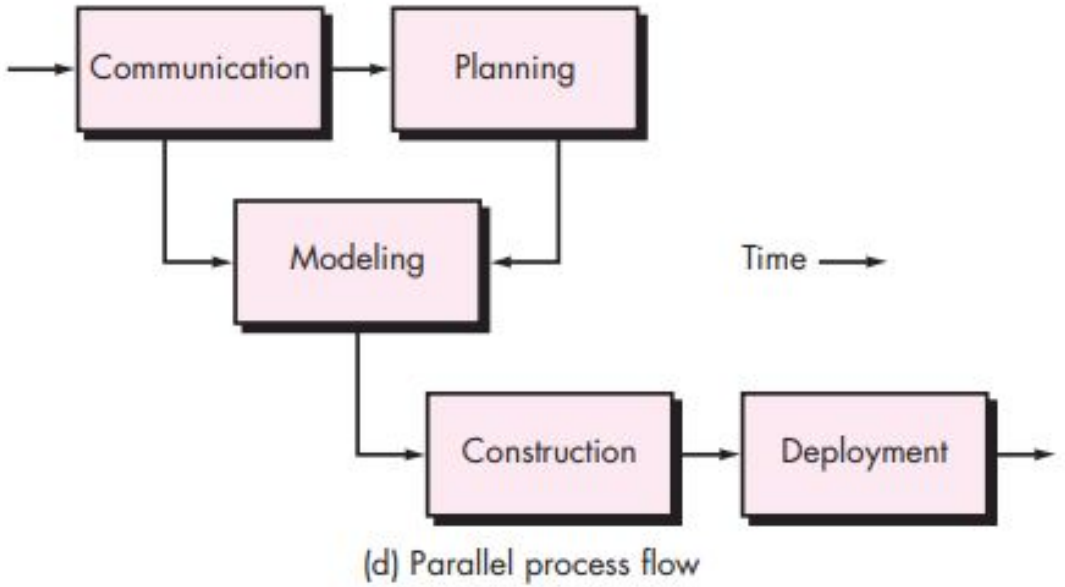
- An evolutionary process flow executes the activities in a "circular" manner. Each circuit through the five activities leads to a more complete version of the software (Figure 2.2c).



(c) Evolutionary process flow

Process Flow

- A parallel process flow (Figure 2.2d) executes one or more activities in parallel with other activities (e.g., modeling for one aspect of the software might be executed in parallel with construction of another aspect of the software).



(d) Parallel process flow

What is a Process Model?

- Description of a process, evolved overtime, in a certain format
  - May use text, pictures, or a combination
  - Contains key process features

Why is a Process Model Needed?

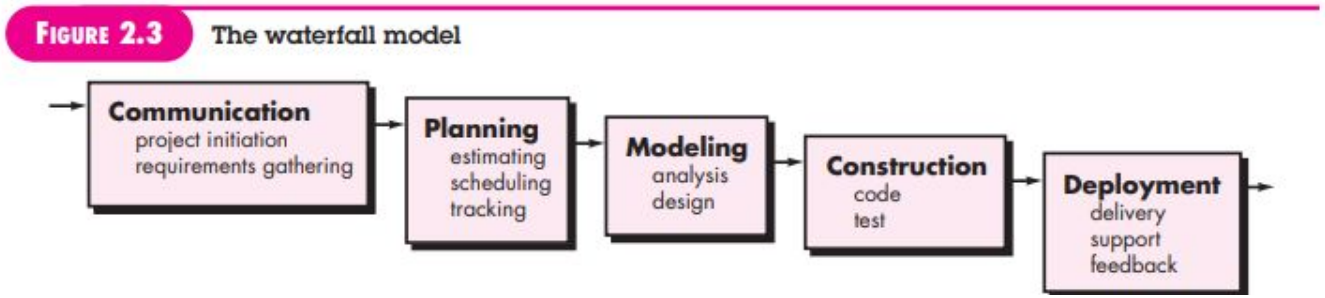
- To form a common understanding
- To find inconsistencies, redundancies, omissions
- To find and evaluate appropriate activities for reaching process goals
- To tailor a general process for a particular situation in which it will be used

Software Development Process Models

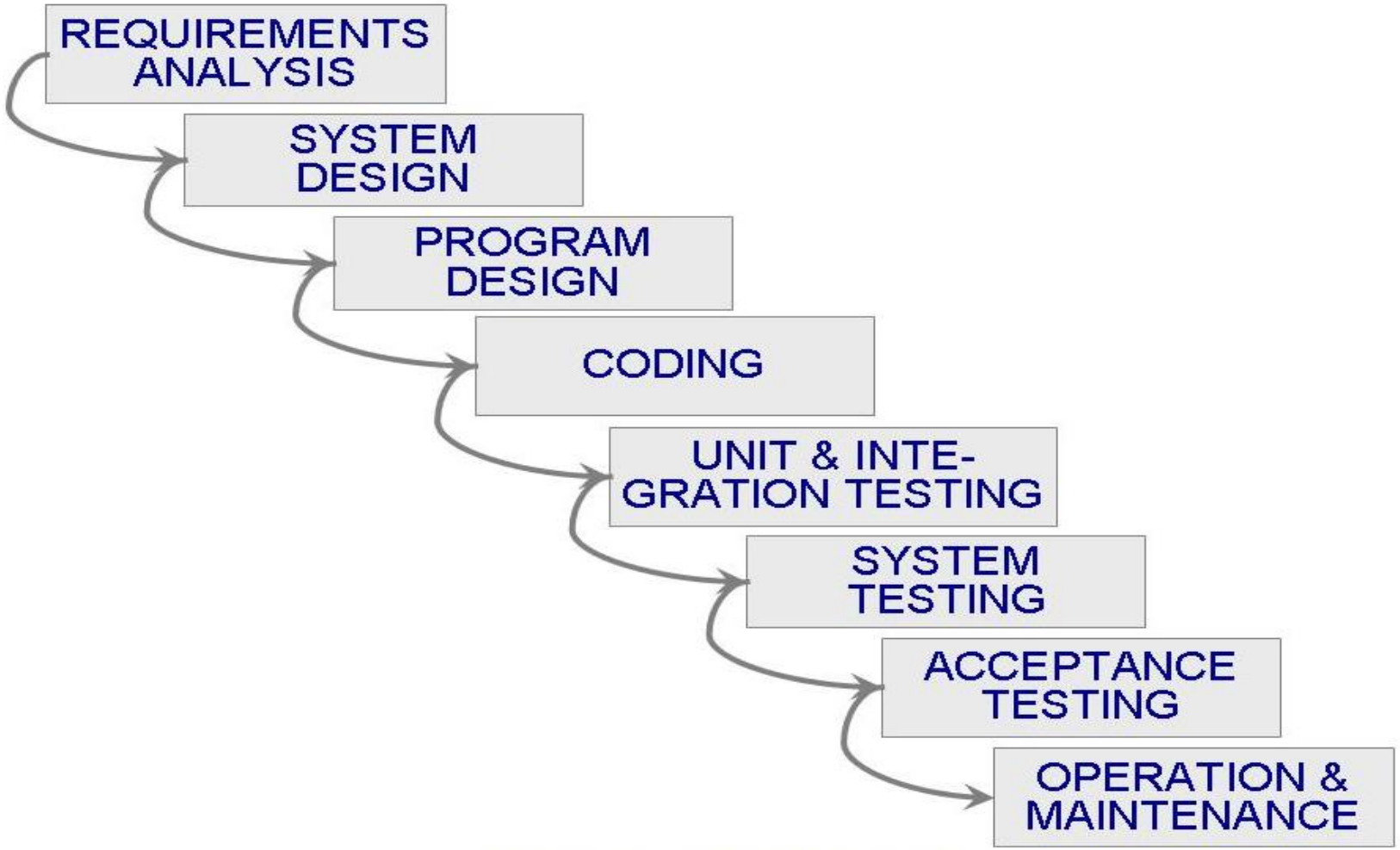
- Waterfall model
  - Classical
  - With prototyping
- V model
- Prototyping model
- Phased development: increments and iterations
- Spiral model
- Unified process
- Rapid Application Development (RAD)
- Agile methods
  - XP
  - Scrum
  - Kanban

Waterfall Model

- One of the first process development models proposed
- Works for well understood problems with minimal or no changes in the requirements
- Simple and easy to explain to customers
- It presents
  - a very high-level view of the development process
  - sequence of process activities
- Each major phase is marked by milestones and deliverables (artefacts)



Waterfall Model



Waterfall Model

- Provides no guidance how to handle changes to products and activities during development (assumes requirements can be frozen)
- Views software development as manufacturing process rather than as building process
- There are no iterative activities that lead to building a final product
- Long wait before a final product
- Generates lots of documentation
- Not suitable for large projects

Problems Waterfall Model

1. Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds.
2. It is often difficult for the customer to state all requirements explicitly. The waterfall model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.
3. 3. The customer must have patience. A working version of the program(s) will not be available until late in the project time span. A major blunder, if undetected until the working program is reviewed, can be disastrous.

Loop: Problem Definition, Technical Design and Development, Integration, Operations and Maintenance

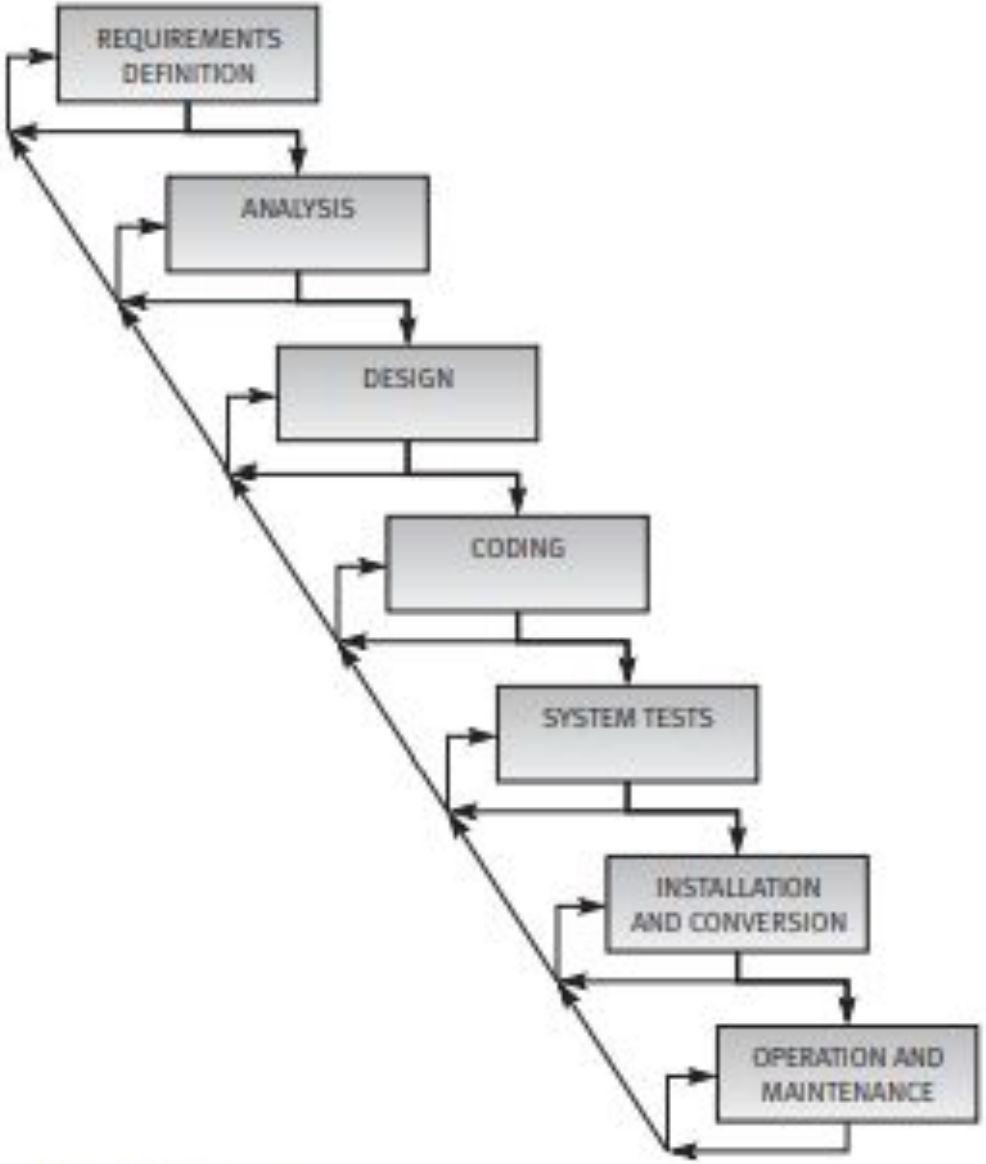


Figure 7.1: The waterfall model

Source: After Boehm (1981) and Royce (1970) (© 1970 IEEE)

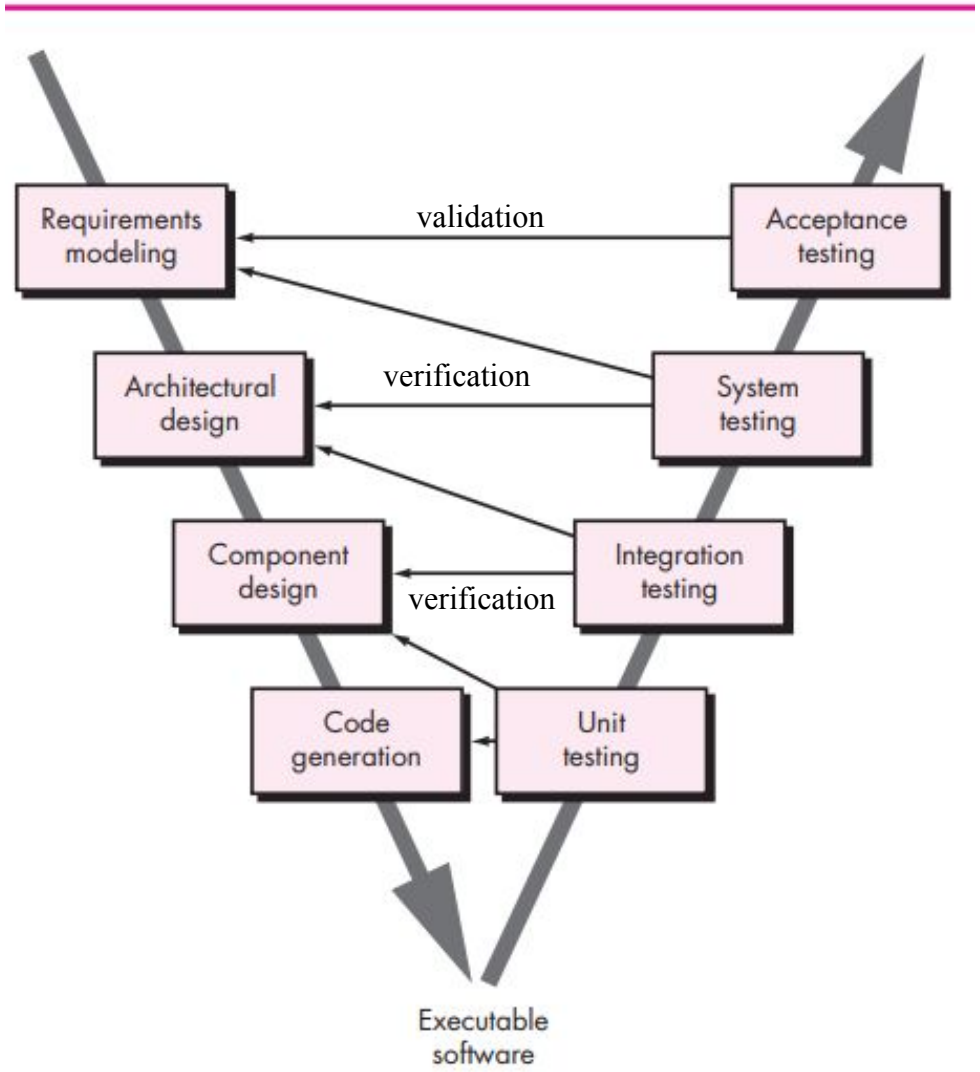
V Model

- A variant of the waterfall model
- Depicts relationship among quality assurance actions to the actions associated with process framework activities.
- Uses unit testing to verify procedural design
- Uses integration testing to verify component design
- Uses system testing to verify architectural design
- Uses acceptance testing to validate the requirements
- If problems are found during verification and validation, the left side of the V can be re-executed before testing on the right side is re-enacted

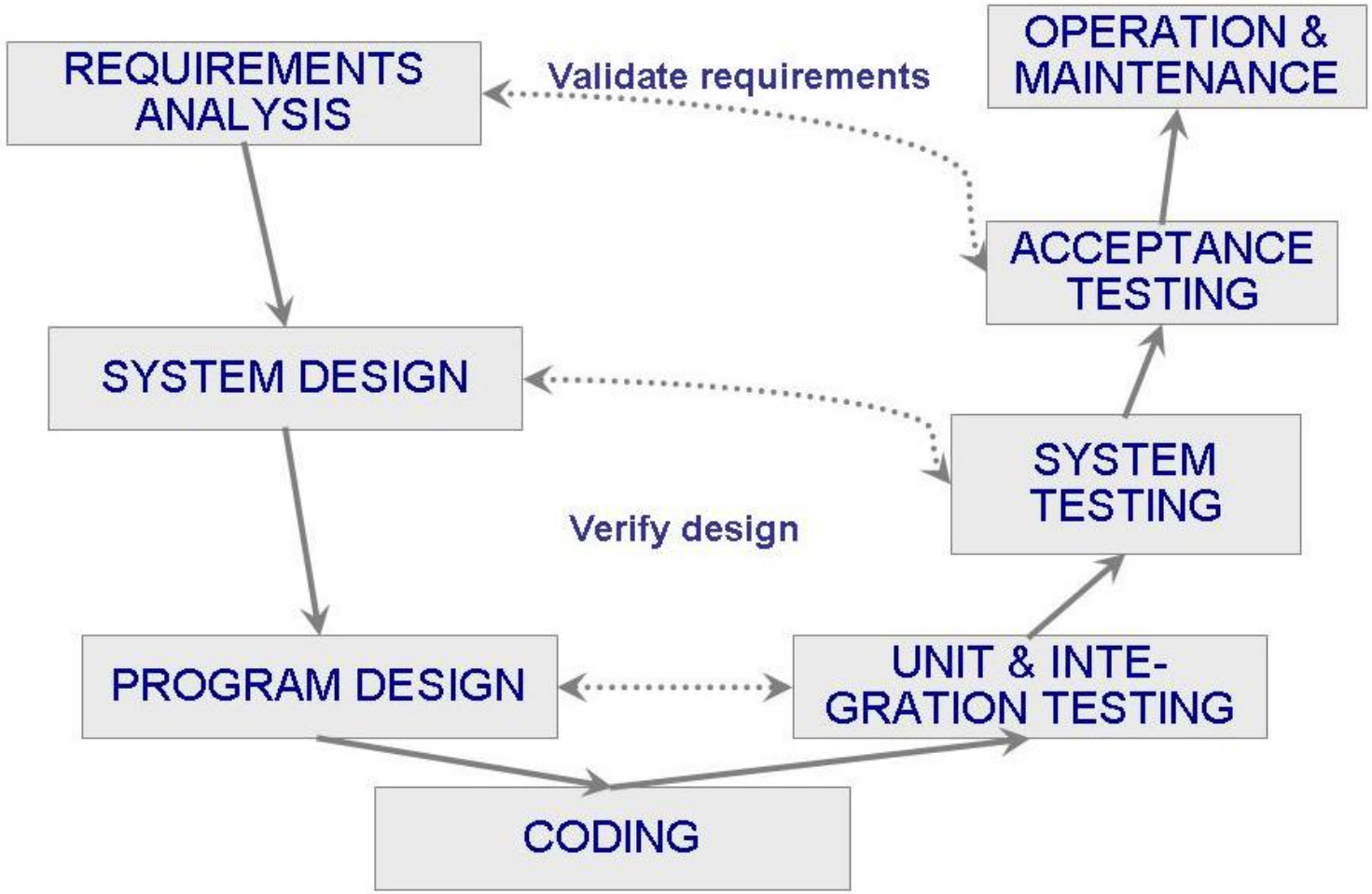
Verification: Each function works correctly
Validation: All requirements have been implemented and each functionality can be traced back to a particular requirement

V Model

As a software team moves down the left side of the V, basic problem requirements are refined into progressively more detailed and technical representations of the problem and its solution. Once code has been generated, the team moves up the right side of the V, essentially performing a series of tests (quality assurance actions) that validate each of the models created as the team moved down the left side.7

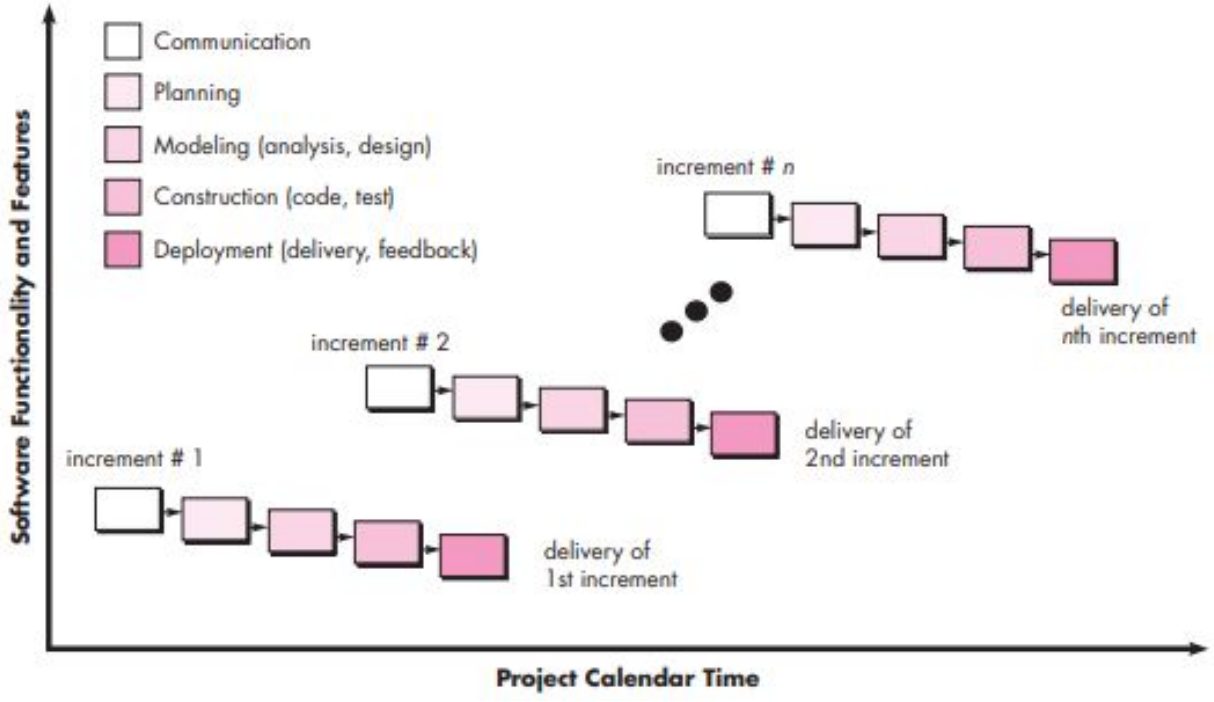


V Model (Contd.)



Incremental Process Models

- The incremental model combines elements of linear and parallel process flows
- The incremental model applies linear sequences in a staggered fashion as calendar time progresses
- Each linear sequence produces deliverable “increments” of the software
- The incremental process model focuses on the delivery of an operational product with each increment



Incremental Process Models

When to use incremental model:

- When Initial software requirements are reasonably well defined but project structure does not allow linear process.
- Project Scope is clear
- What comes next is unknown or undecided
- A need exists to provide a limited set of software functionality to users quickly
- Core functionality is urgently needed

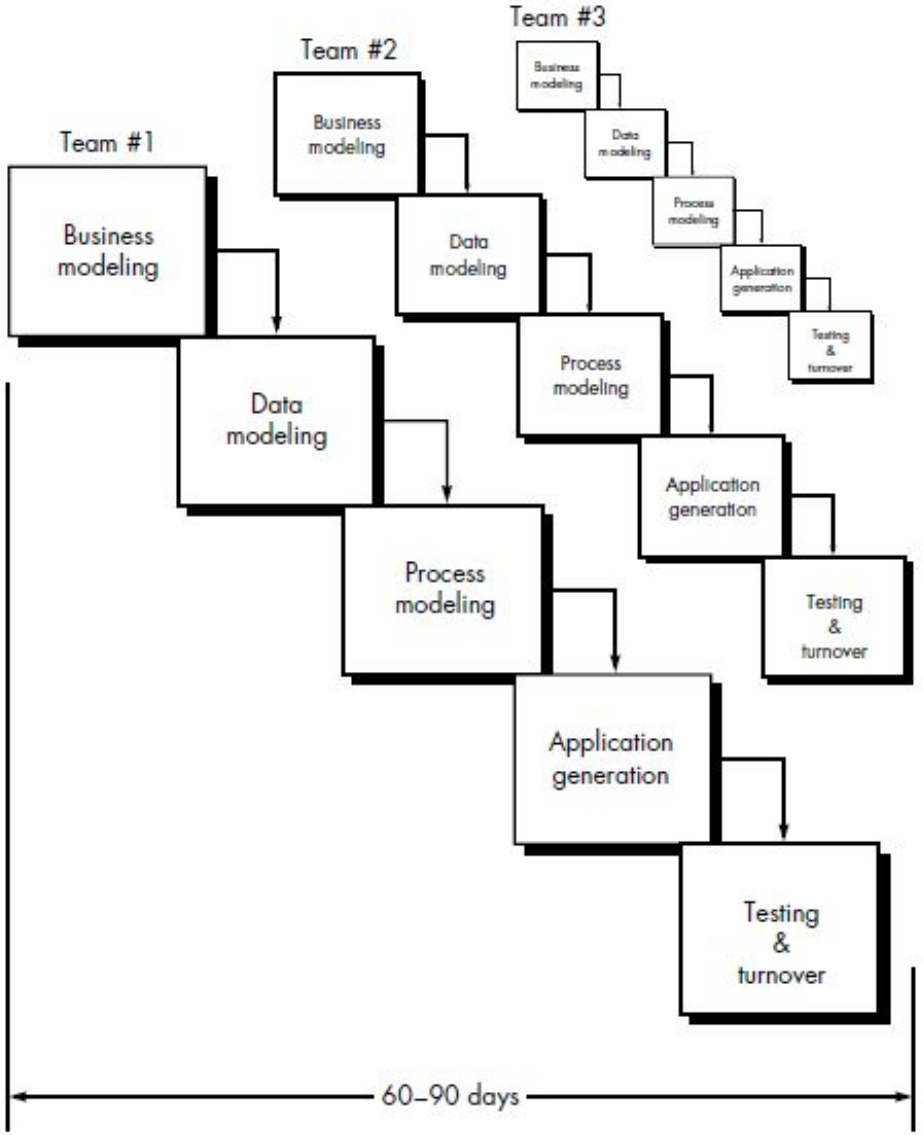
Incremental Process Models (Example)

Word-Processing Software :

- Basic file management, editing, and document production functions in the first increment
- More sophisticated editing and document production capabilities in the second increment
- Spelling and grammar checking in the third increment
- Advanced page layout capability in the fourth increment.

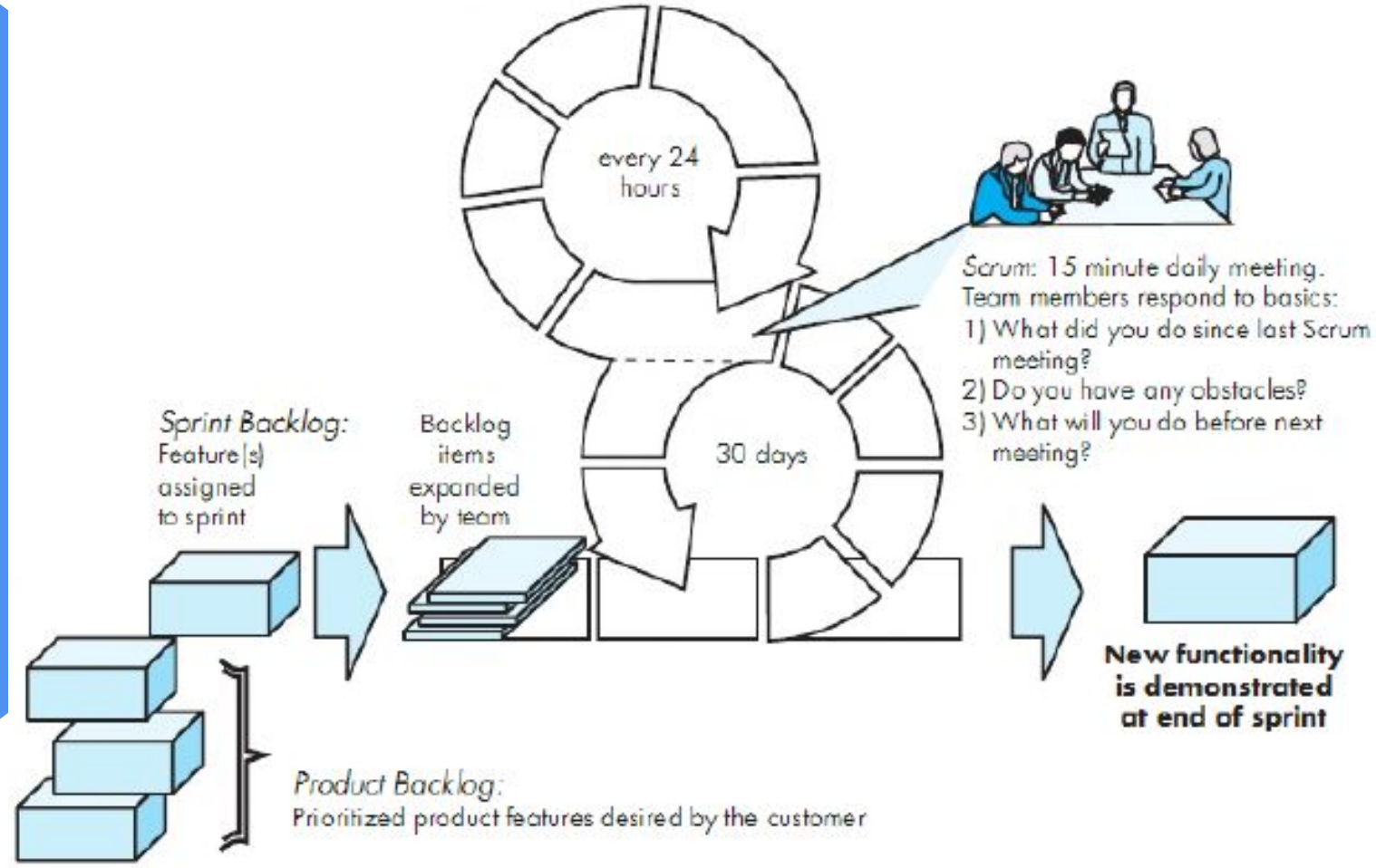
Models of Incremental Nature

Rapid Application Development (RAD)



Models of Incremental Nature

Scrum



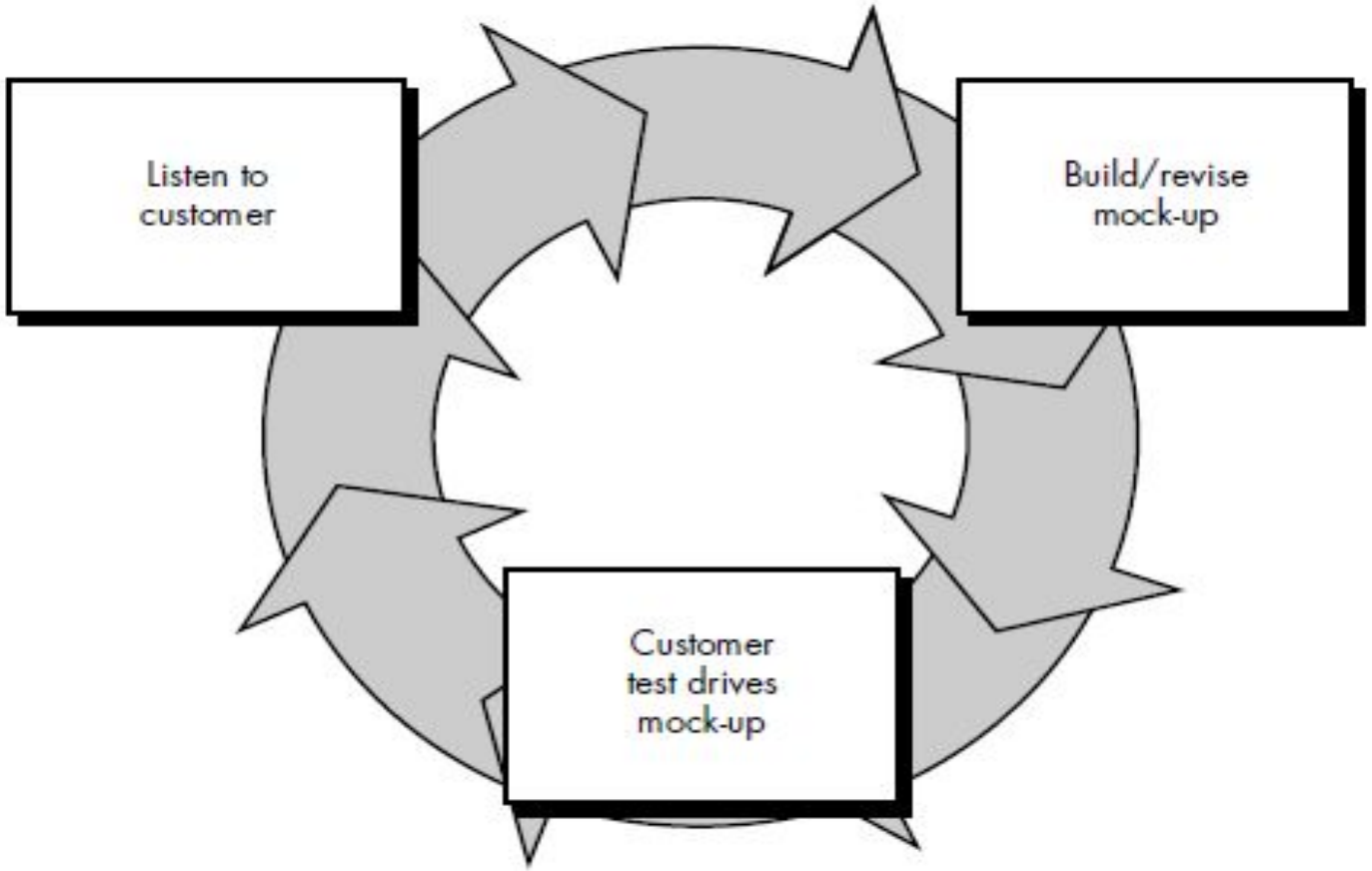
Evolutionary Process Models

- For complex projects that tend to evolve over a period of time.
- Scope is not clearly defined
- For projects whose business and product requirements tend to change frequently
- Details of product or system extensions have yet to be defined
- Evolutionary models are iterative

Prototyping

- In prototyping, the customer is presented at a very early stage with a working version of the system. (It may not be a complete system, but it is at least part of the system and it works.)
- Although prototyping can be used as a stand-alone process model, it is more commonly used as a technique that can be implemented within the context of any one of the process models that we have discussed so far.
- Tool for defining requirements?

Prototyping



Prototyping

These prototypes are tested intensively before a production line is set up. It is possible to follow a similar approach with software development.

Types of Prototyping

Throwaway – the various versions of the system are constructed and then thrown away. (The final system is implemented in some different way.)

For example, a throwaway prototype might be written very quickly in Visual Basic to demonstrate the essential functions that a system will carry out. But then the software might be rewritten using careful and systematic development methods.

Throwaway prototype is also known as **Rapid Prototyping**

Types of Prototyping

- **Evolutionary**– An initial implementation evolves towards the final version. (The prototype becomes the final system.)

- An evolutionary prototype might be implemented in C# to demonstrate to the user the main features of the system. Having checked that the system does what is required, new features and facilities are added to the prototype, gradually transforming it into its complete form.

Problems in Prototyping Model?

1. Stakeholders see what appears to be a working version of the software, unaware that the prototype is held together haphazardly, unaware that in the rush to get it working you haven't considered overall software quality or long-term maintainability. When informed that the product must be rebuilt so that high levels of quality can be maintained, stakeholders cry foul and demand that "a few fixes" be applied to make the prototype a working product. Too often, software development management relents.

2. As a software engineer, you often make implementation compromises in order to get a prototype working quickly. An inappropriate operating system or programming language may be used simply because it is available and known; an inefficient algorithm may be implemented simply to demonstrate capability. After a time, you may become comfortable with these choices and forget all the reasons why they were inappropriate. The less-than-ideal choice has now become an integral part of the system.

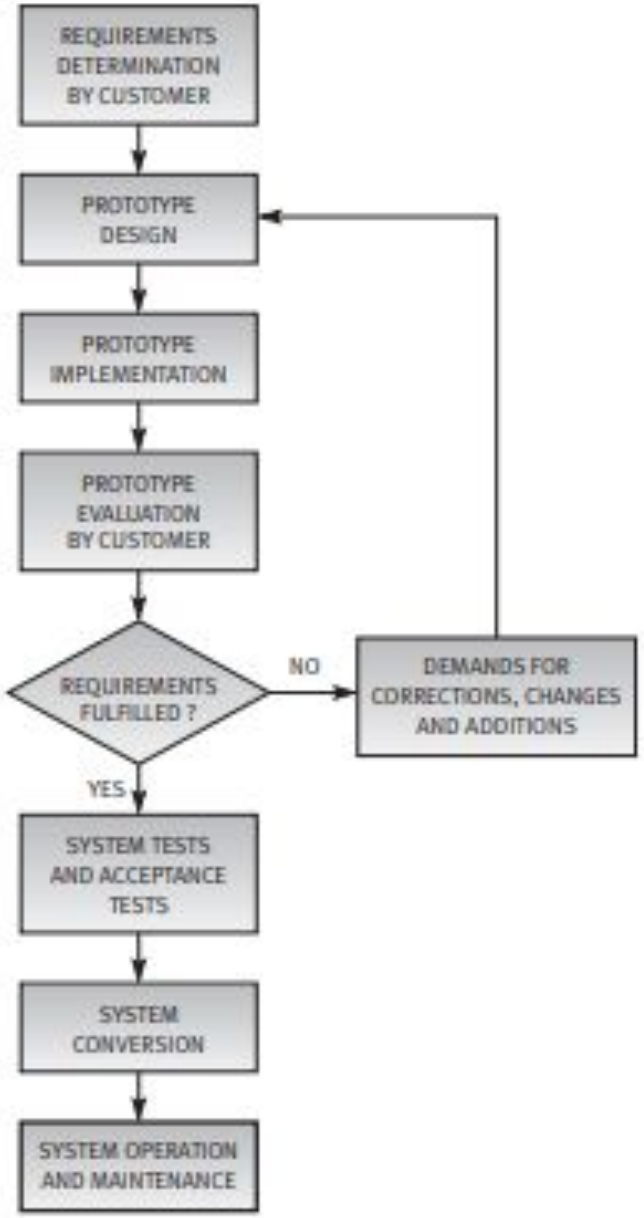
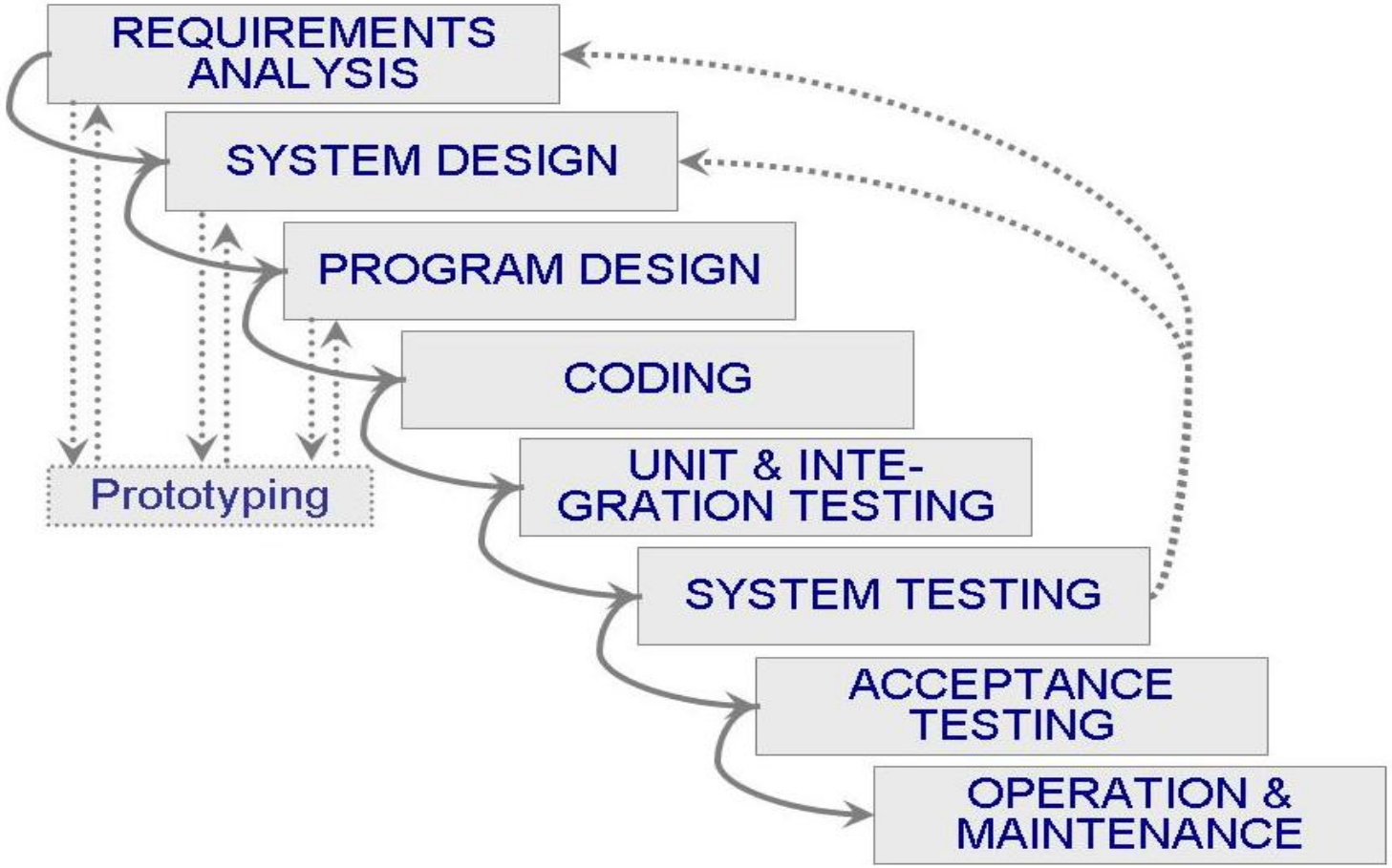


Figure 7.2: The prototyping model

Waterfall Model with Prototyping

- A prototype is a partially developed product
- Prototyping helps
  - developers assess alternative design strategies (design prototype)
  - users understand what the system will be like (user interface prototype)

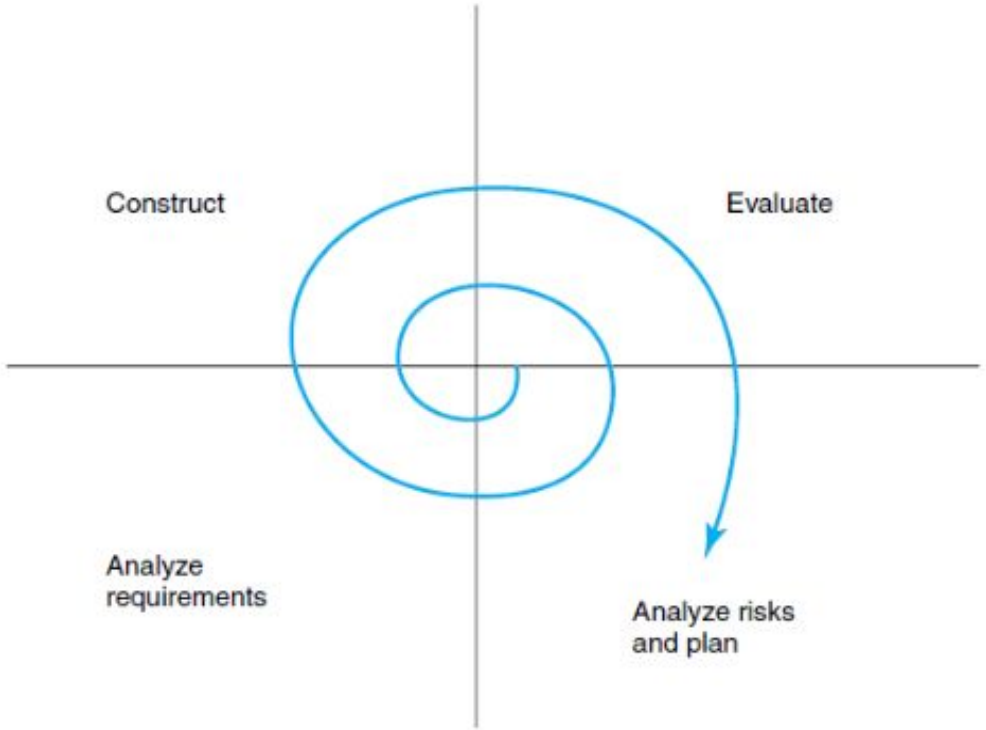
Waterfall Model with Prototyping



The Spiral Model of Software Development

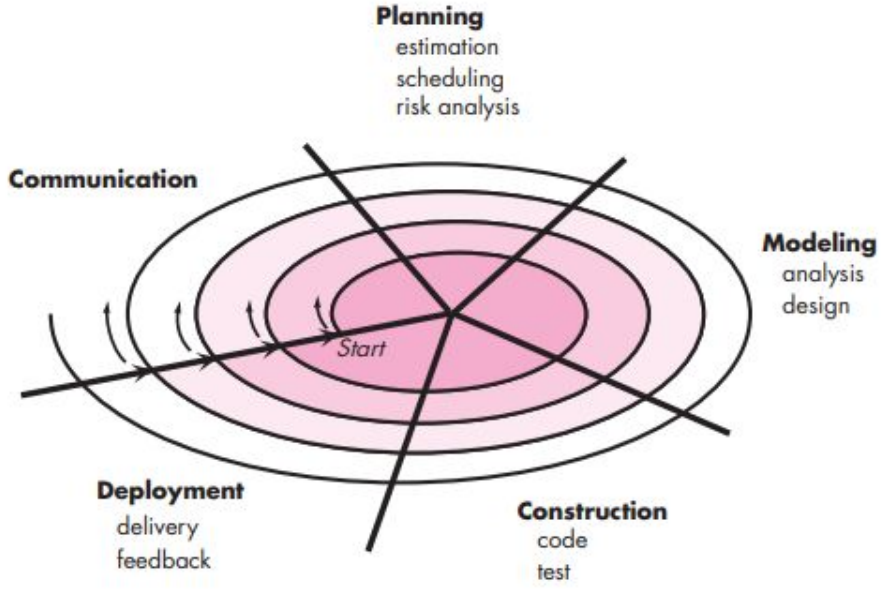
- What is a risk in Software Development?
- Software risk encompasses the probability of occurrence for uncertain events and their potential for loss within an organization.
  - Schedule
  - Budget
  - Operational Risk
- The main feature of the spiral model is the recognition that there is often big uncertainty at many stages during a software development project.
- It therefore incorporates periodic risk assessment.
- These assessments are followed by identifying alternative actions, selection of the best action and re-planning.

The Spiral Model



The spiral model

FIGURE 2.7
A typical spiral model



Spiral Model

- Suggested by Boehm (1988)
- Combines development activities with risk management to minimize and control risks
- The model is presented as a spiral in which each iteration is represented by a circuit around four major activities(Danial Glan)
  - Planning
  - Risk analysis and resolution
  - Engineering activities according to the stage of the project: design, coding, testing, installation and release
  - Customer evaluation, including comments, changes and additional requirements, etc.

Spiral Model

When to use it?

- A Spiral model in software engineering is used when project is large
- When releases are required to be frequent, spiral methodology is used
- When creation of a prototype is applicable
- When risk and costs evaluation is important
- Spiral methodology is useful for medium to high-risk projects
- When requirements are unclear and complex, Spiral model in SDLC is useful
- When changes may require at any time
- When long term project commitment is not feasible due to changes in economic priorities

Spiral Model

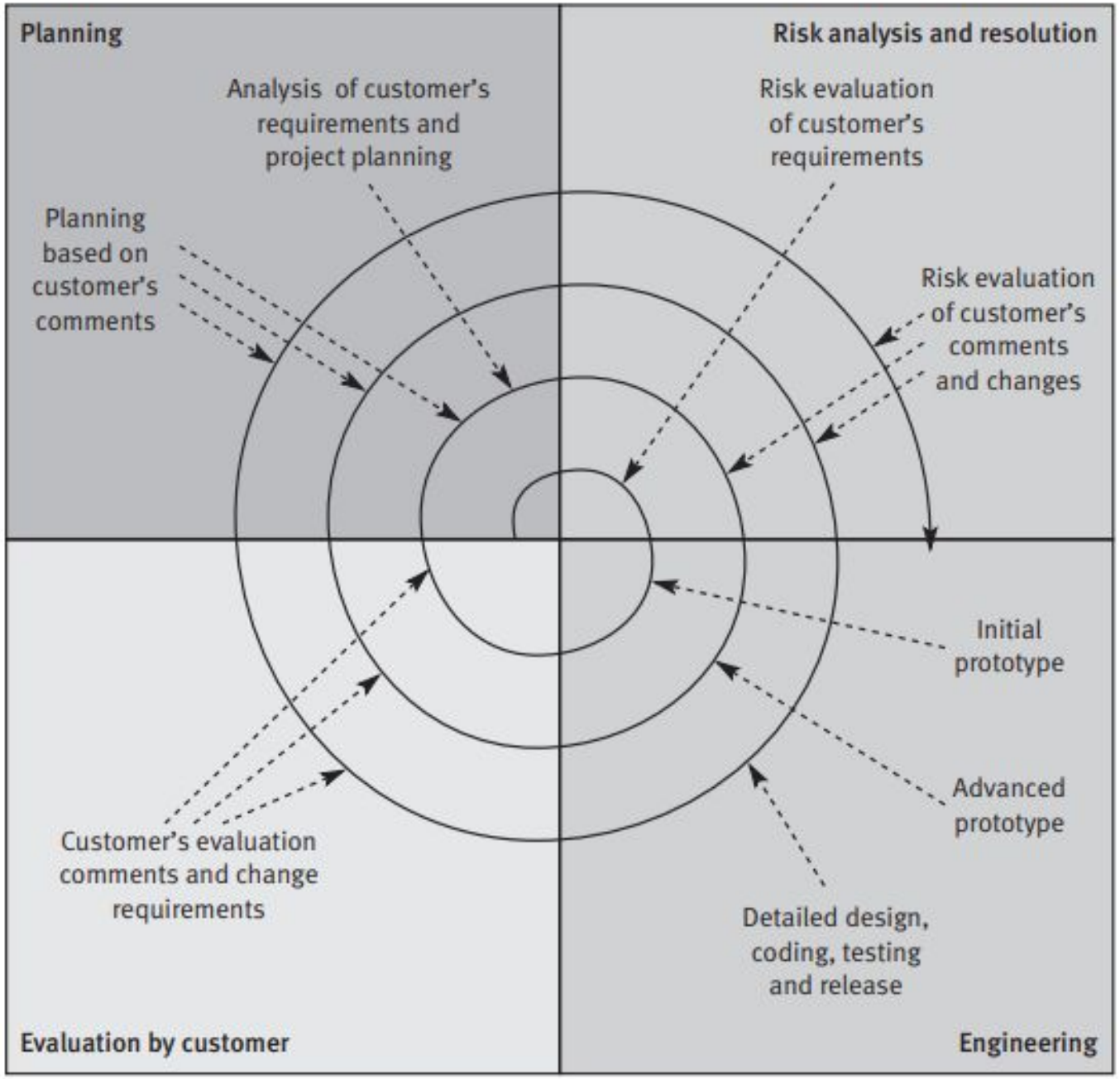


Figure 7.3: The spiral model (Boehm, 1988)

The Spiral Model

The distinctive feature of the spiral model is that it makes explicit the idea of risk.

During software development there can be difficult problems to be overcome.

The spiral model explicitly recognizes that there are uncertainties associated with software development and that they should be dealt with as carefully as possible.

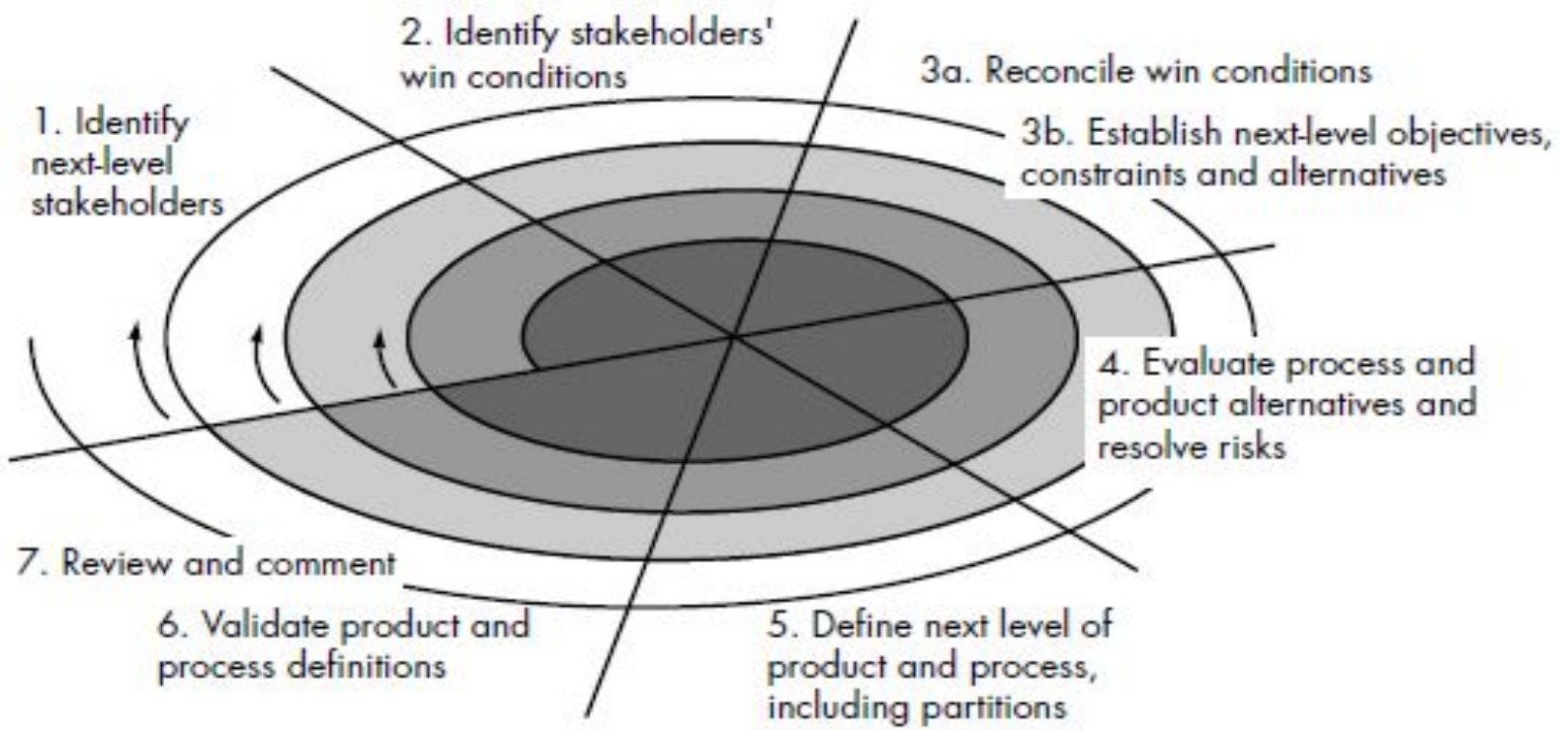
The WINWIN Spiral Model

The spiral model suggests a framework activity that addresses customer communication. The objective of this activity is to elicit project requirements from the customer. In an ideal context, the developer simply asks the customer what is required and the customer provides sufficient detail to proceed.

Unfortunately, this rarely happens. In reality, the customer and the developer enter into a process of negotiation, where the customer may be asked to balance functionality, performance, and other product or system characteristics against cost and time to market.

The best negotiations strive for a “win-win” result. That is, the customer wins by getting the system or product that satisfies the majority of the customer’s needs and the developer wins by working to realistic and achievable budgets and deadlines.

The WINWIN Spiral Model



Boehm's WINWIN spiral model [BOE98] defines a set of negotiation activities at the beginning of each pass around the spiral. Rather than a single customer communication activity.

Win-Win Spiral Model

An advanced spiral model, the Win–Win Spiral model (Boehm, 1998), enhances the Spiral model (Boehm, 1988) still further.

The advanced model places extra emphasis on **communication and negotiation between the customer and the developer**.

The model's name refers to the fact that by using this process, the customer "wins" in the form of improved chances to receive the system most satisfying to his needs, and the developer "wins" in the form of improved chances to stay within the budget and complete the project by the agreed date.

This is achieved by increasing emphasis on customer participation and on engineering activities.

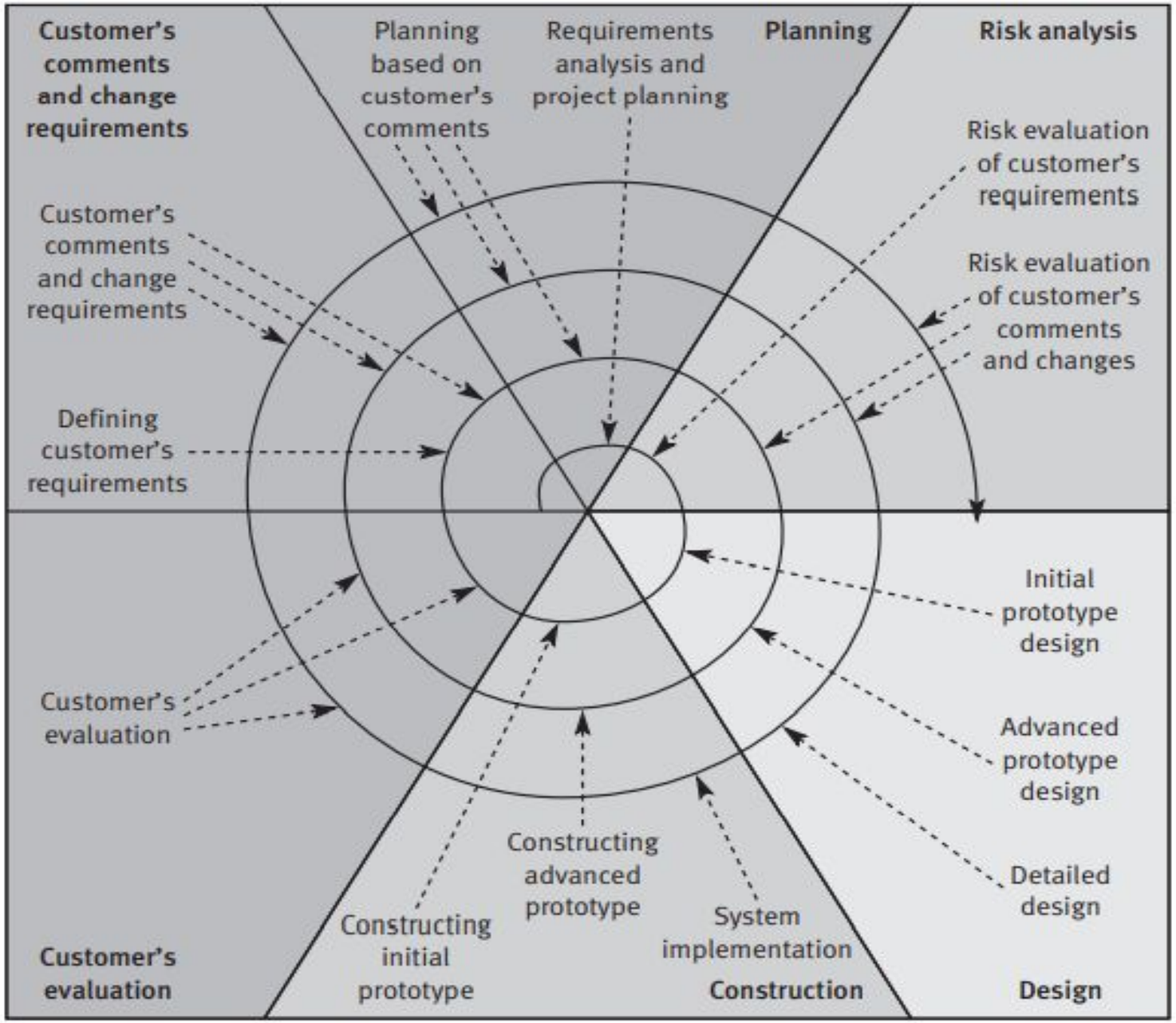
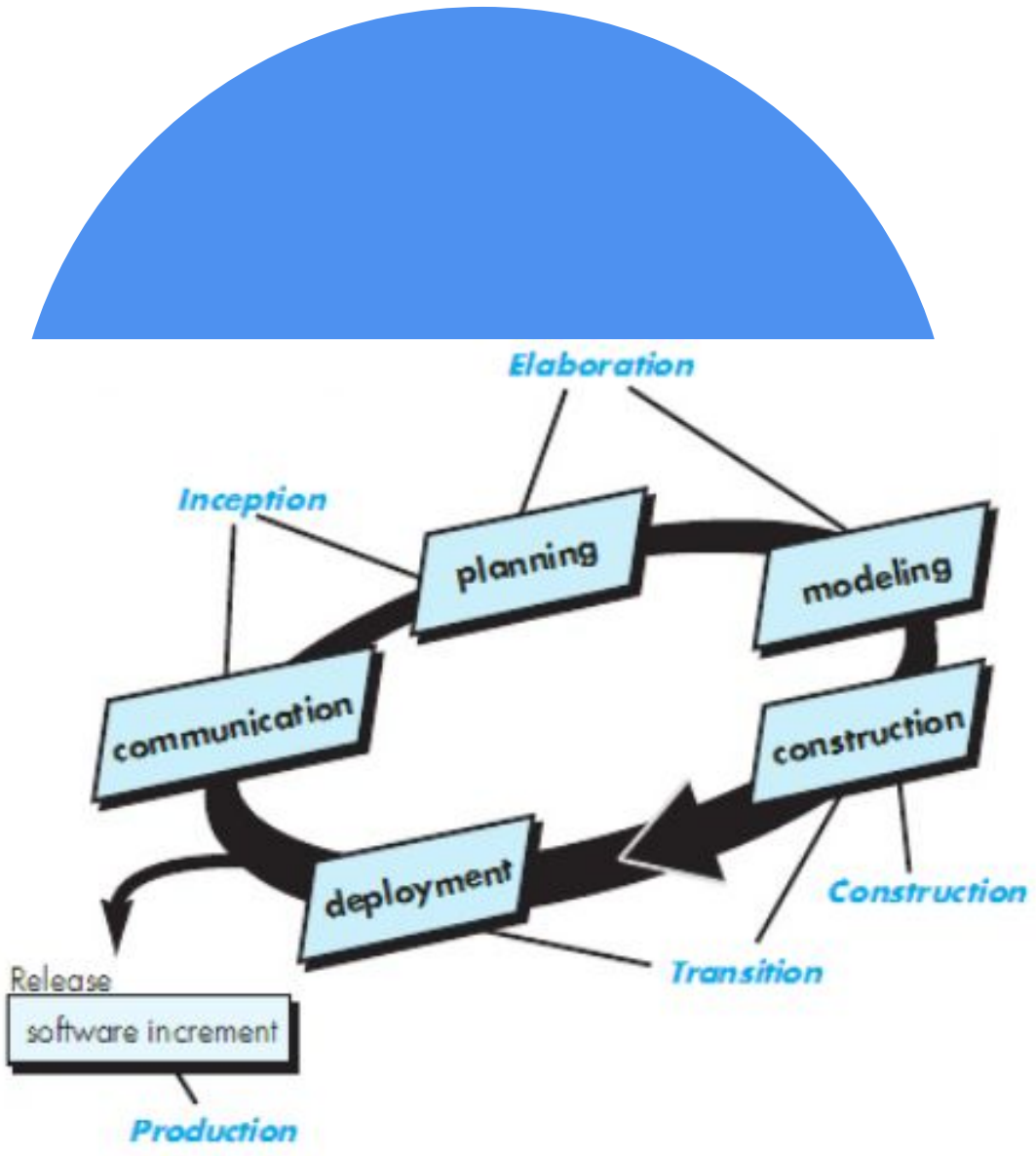
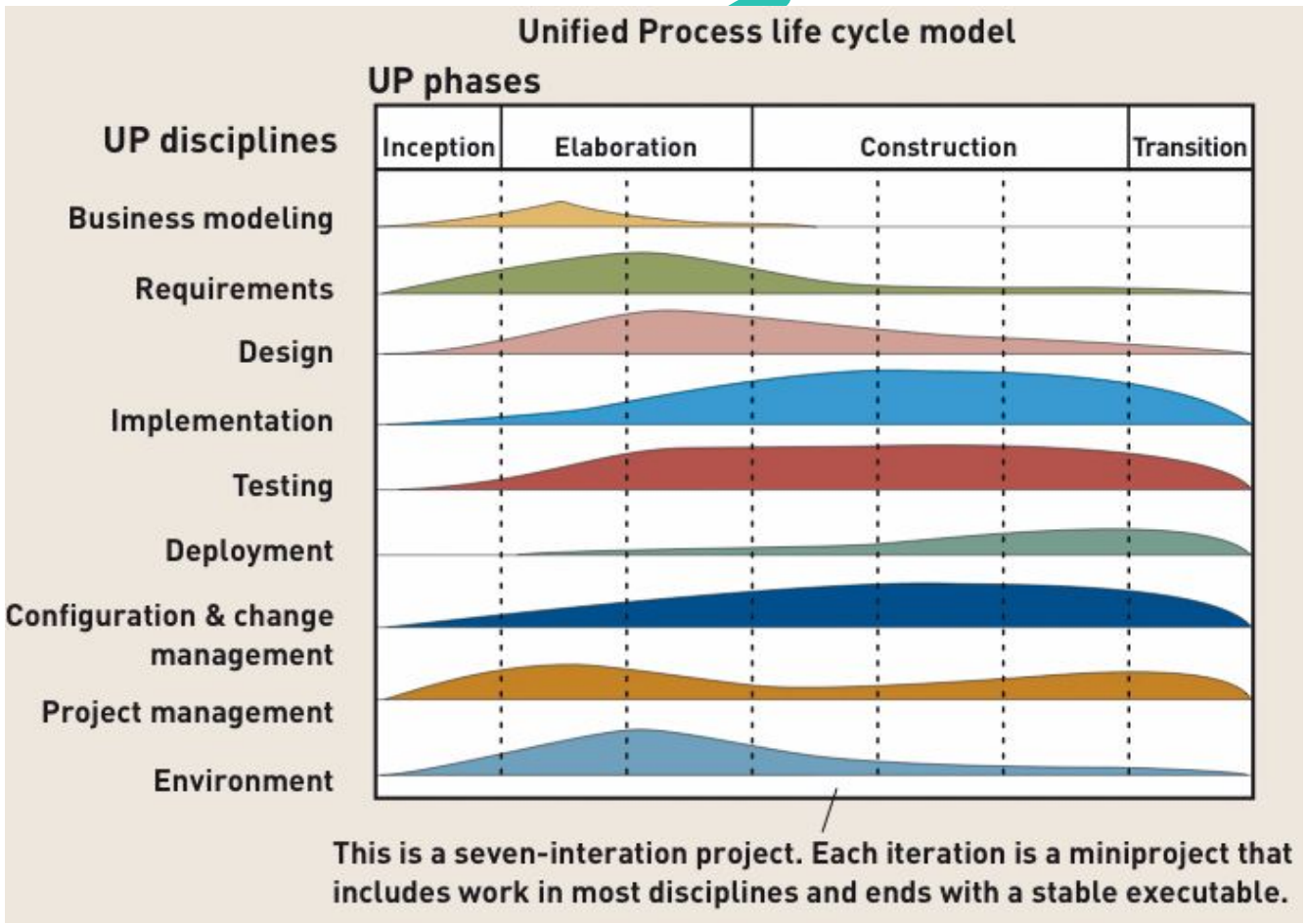


Figure 7.4: The advanced spiral model (Boehm, 1998)

Source: After Boehm (1988) (© 1988 IEEE)


Elaboration
Inception
planning
modeling
communication
construction
deployment
Construction
Transition
Release
software increment
Production


Unified Process life cycle model
UP phases
UP disciplines
Inception
Elaboration
Construction
Transition
Business modeling
Requirements
Design
Implementation
Testing
Deployment
Configuration & change management
Project management
Environment
This is a seven-interation project. Each iteration is a miniproject that includes work in most disciplines and ends with a stable executable.


Thank you