Software Engineering

HUANG Jie

School of Software Engineering

Tongji University, 2023

Lesson 3
Generic Process Model

In lesson 2, we have learned

- ✓ Why Software has Process Framework?
- ✓ Who is involved in a Software Engineering Project?
- ✓ What are Software Engineering Activities & Elements?
- ✓ General Principles of Software Engineering Practice.
- ✓ Examples of software development myths.

Lesson 3
Generic Process Model

In this lesson, we will discuss

- ✓ The meaning of Software Process.
- ✓ The generic concept of software engineering process models.
- ✓ The technical issues in development & support of a system.
- ✓ The nontechnical issues of developing and supporting a system.
- ✓ The coordination efforts needed for process.
- ✓ Some traditional process models.

Software Process vs Software Engineering

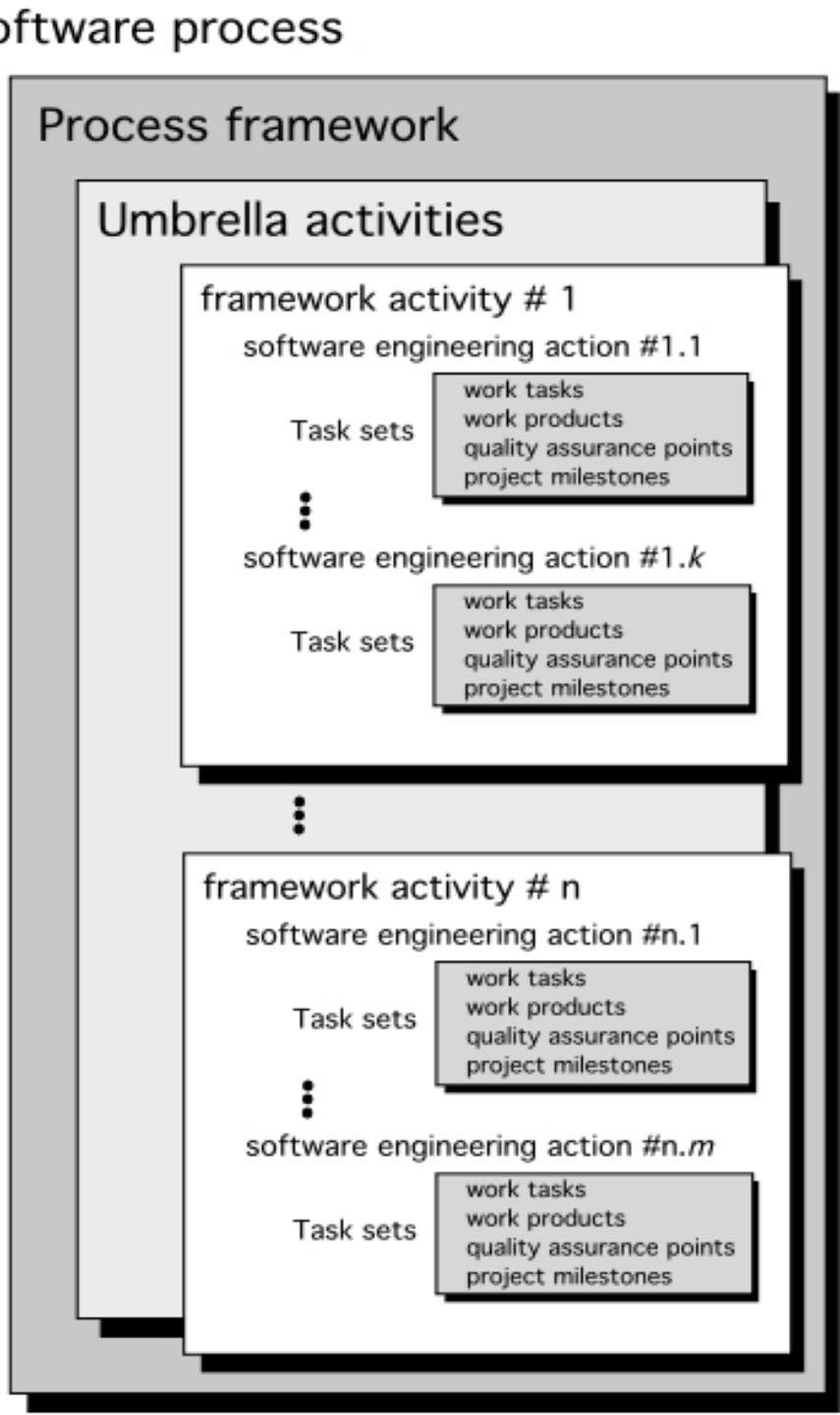
- Howard Baetjer Jr. 1998
  - Because software, like all capital, is embodied knowledge, and because that knowledge is initially dispersed(分散的), tacit(默示的), latent(隐藏的), and incomplete in large measure, software development is a social learning process.
  - The process is a dialogue in which the knowledge that must become the software is brought together and embodied in the software. The process provides interaction between users and designers, between users and evolving tools, and between designers and evolving tools [technology].
  - It is an iterative process in which the evolving tool itself serves as the medium for communication, with each new round of the dialogue eliciting more useful knowledge from the people involved.

The Meaning of Software Process

- Within the context of this book, we define a software process as a framework for the activities, actions, and tasks that are required to build high-quality software.
- Is process synonymous with “software engineering”?
  - The answer is Yes and No.
  - A software process defines the approach that is taken as software is engineered.
  - But software engineering also encompasses technologies that populate the process - technical methods and automated tools.
- More important, software engineering is performed by creative, knowledgeable people who should adopt a mature software process so that it is appropriate for the products that they build and demands of their marketplace.

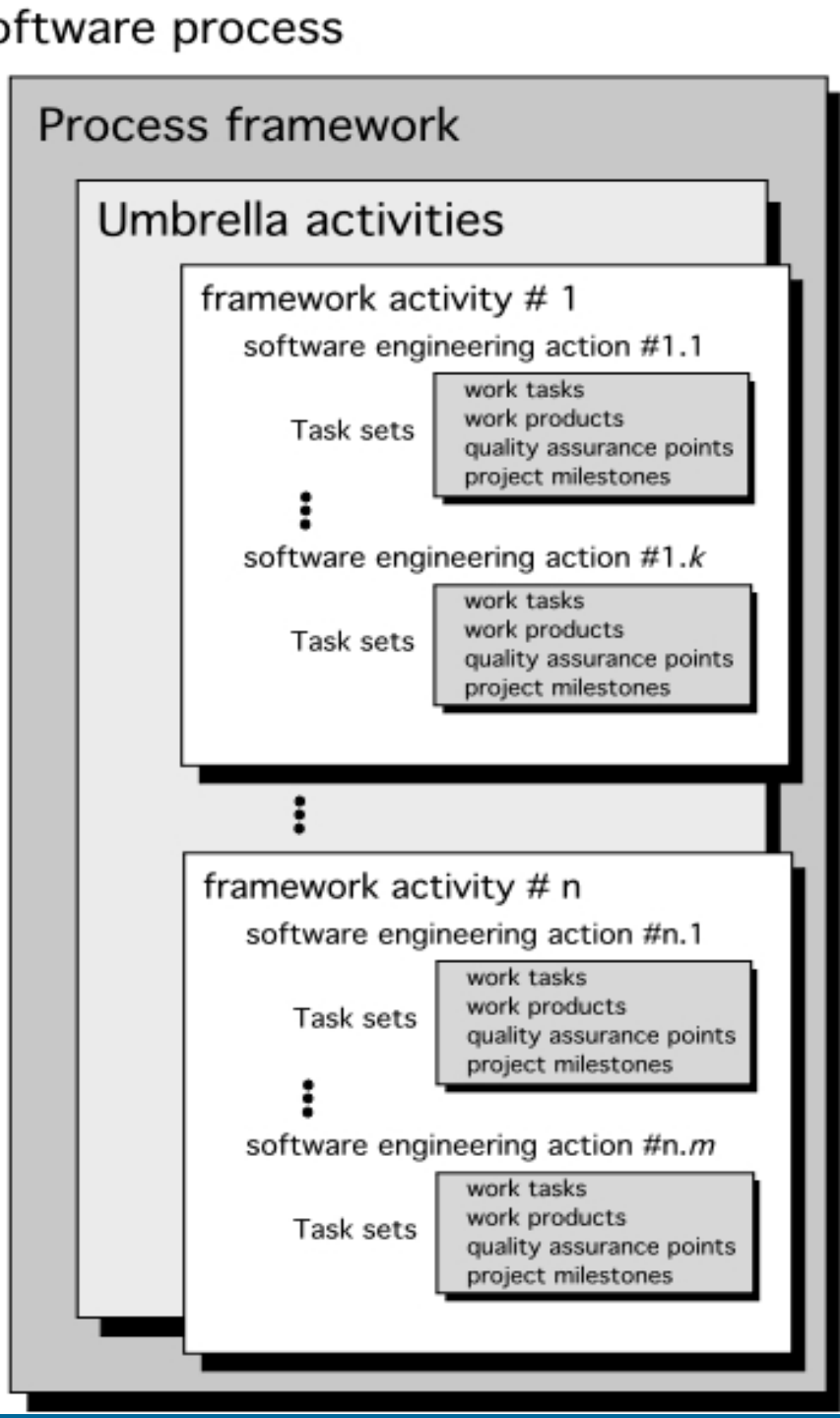
A Generic Process Model

- Framework activities
  - Communication.
  - Planning.
  - Modeling.
    - Analysis of requirements.
    - Design.
  - Construction.
    - Code generation.
    - Testing.
  - Deployment.

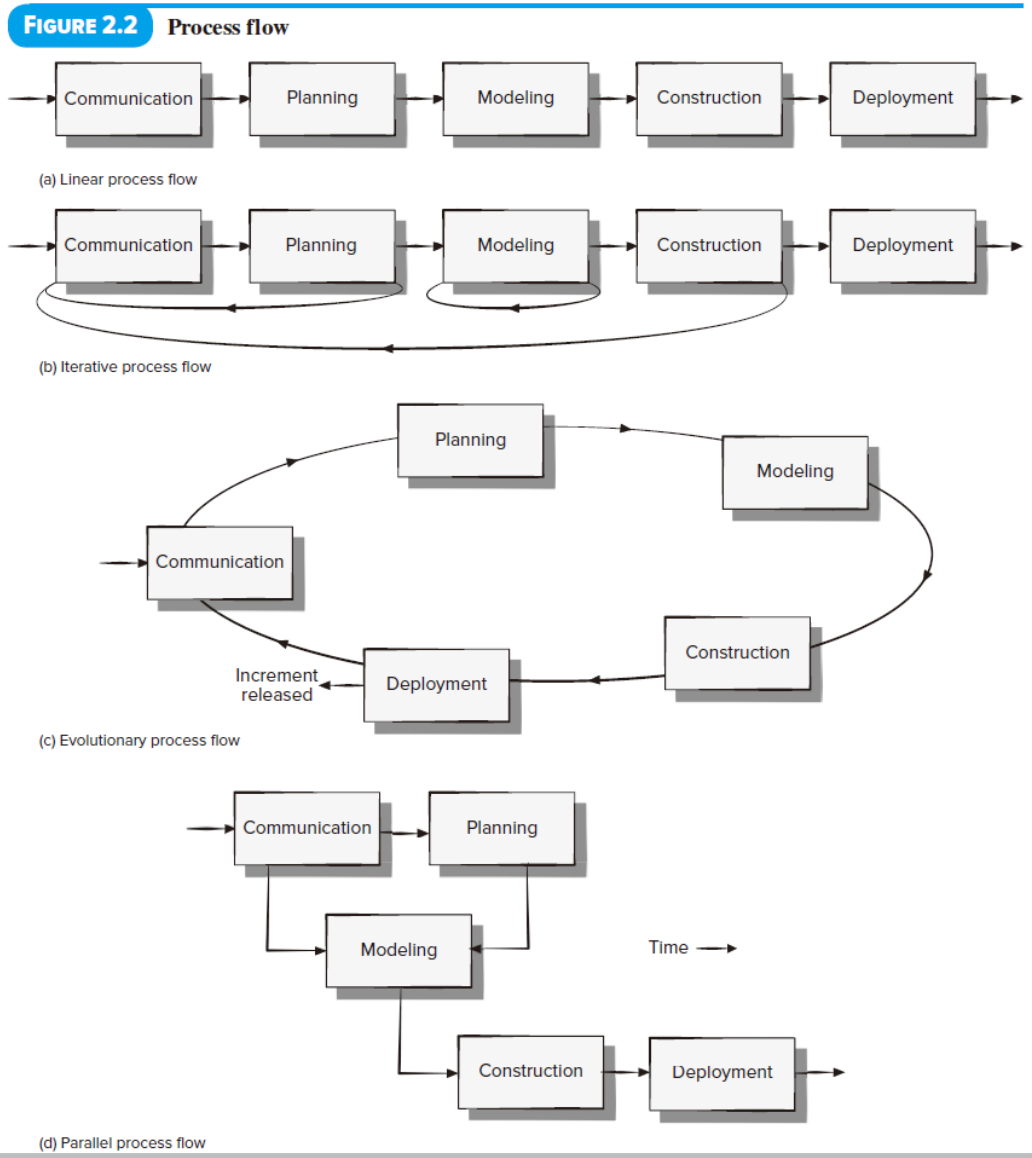


A Generic Process Model

- Umbrella activities
  - Software project tracking and control.
  - Risk management.
  - Software quality assurance.
  - Technical reviews.
  - Measurement.
  - Software configuration management.
  - Reusability management.
  - Work product preparation and production.



Types of process flow



Defining a framework activity

- Different projects demand different task sets. The software team chooses the task set based on problem and project characteristics.
- **E**xample
  - Communication activity in small project.
  - What are the contents of this activity?
    - Step 1 ?
    - Step 2 ?
    - Step 3 ?
    - Step 4 ?

    From Textbook《Software Engineering》, P23～P24.

Identifying a Task Set

- A task set defines the actual work to be done to accomplish the objectives of a software engineering action.
  - A list of the task to be accomplished.
  - A list of the work products to be produced.
  - A list of the quality assurance filters to be applied.
- Example
  - Identifying a Task Set
  - Compare the difference between small and large software projects.

From Textbook《Software Engineering》, P19.

Process Assessment and improvement

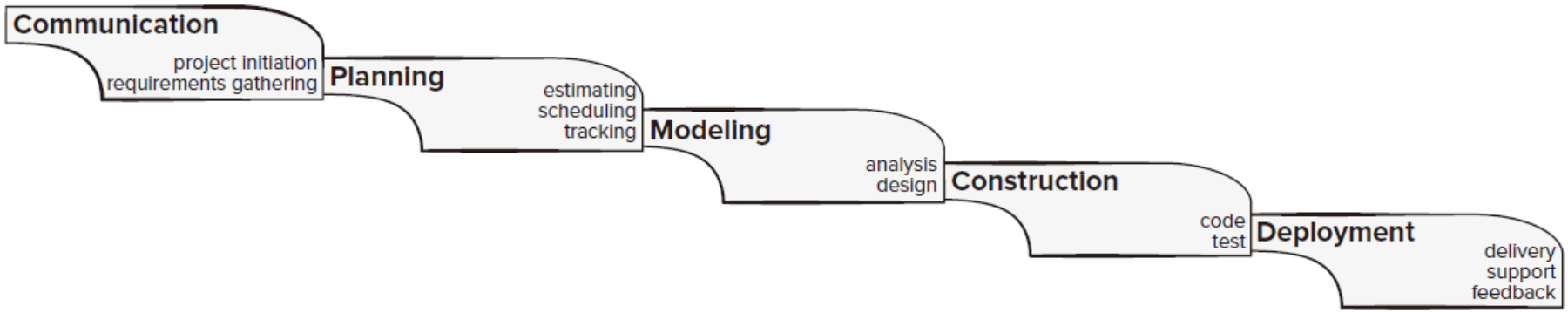
- A software process is no guarantee that software will be delivered on time, that it will meet the customer's needs, or that it will exhibit the technical characteristics that will lead to long-term quality characteristics.
- Process patterns must be coupled with solid software engineering practice.
- The current thinking among most engineers is that software processes and activities should be assessed using numeric measures or software analytics (metrics).
- The progress you have made in a journey toward an effective software process will define the degree to which you can measure improvement in a meaningful way.

Prescriptive Models (惯用模型)

- Prescriptive process models advocate(提倡) an orderly & structural approach to software engineering.
- That leads to a few **Questions**
  - If prescriptive process models strive for structure and order, are they inappropriate for a software world that thrives on change?
  - Yet, if we reject traditional process models (and the order they imply) and replace them with something less structured, do we make it impossible to achieve coordination(协调) and coherence(一致) in software work?
- The **Reason**, why we use these traditional process models
  - A set of process elements being prescribed.
  - Each model defines a process flow (work flow), the manner in which the process elements are interrelated to one another.

The Waterfall Model

FIGURE 2.3 The waterfall model



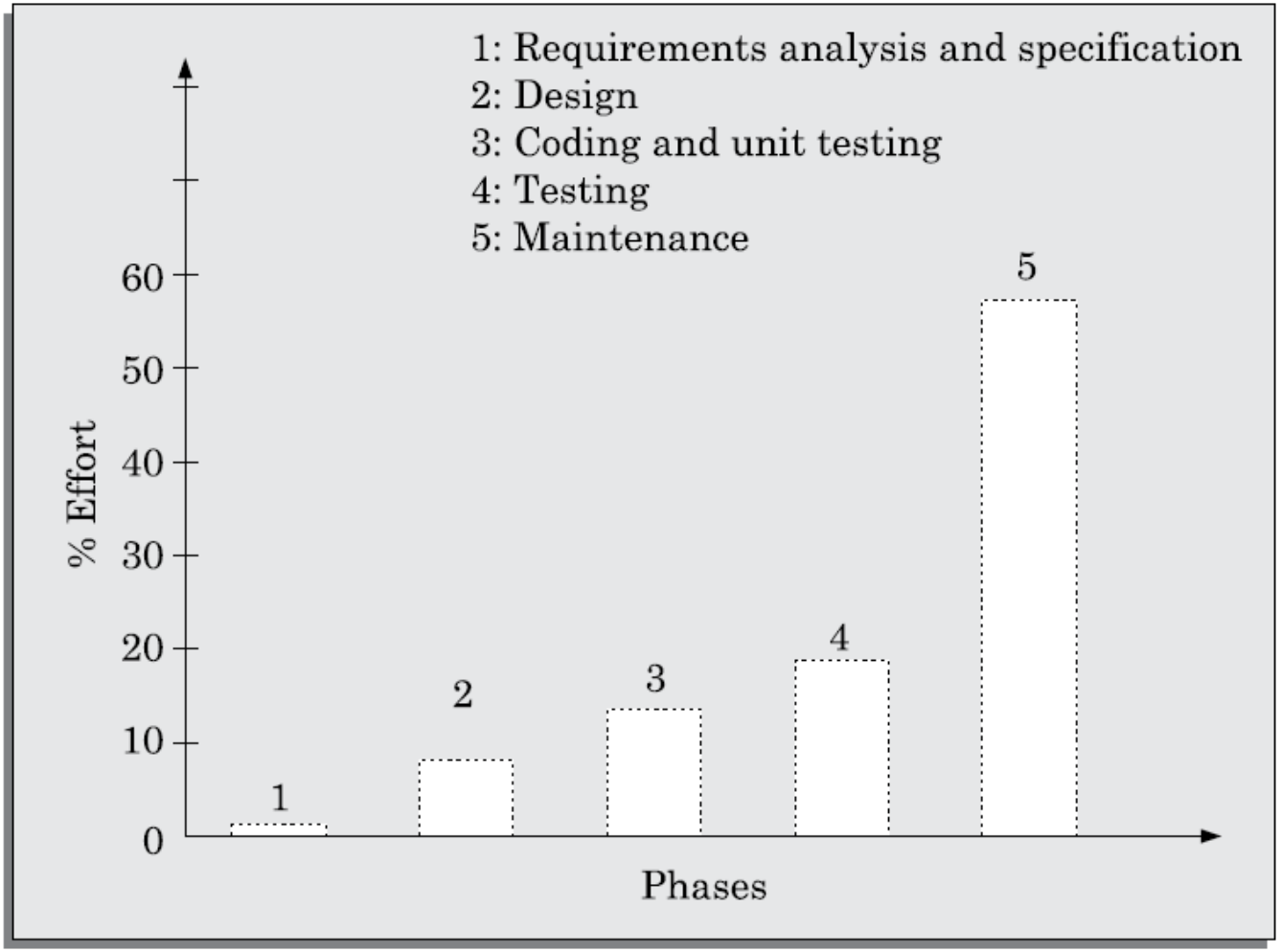
The Waterfall Model

- The waterfall model, the oldest commonly used software life cycle, is a linear approach to software development that was first suggested by Royce in 1970.
- This development and maintenance paradigm, also known as the classic life cycle, lays out a set of development phases which are to be completed in order.
- A new phase begins only when the previous phase has been fully completed, which includes the finalization of all documentation from that phase, as well the approval of completion by the software quality assurance (SQA) group.

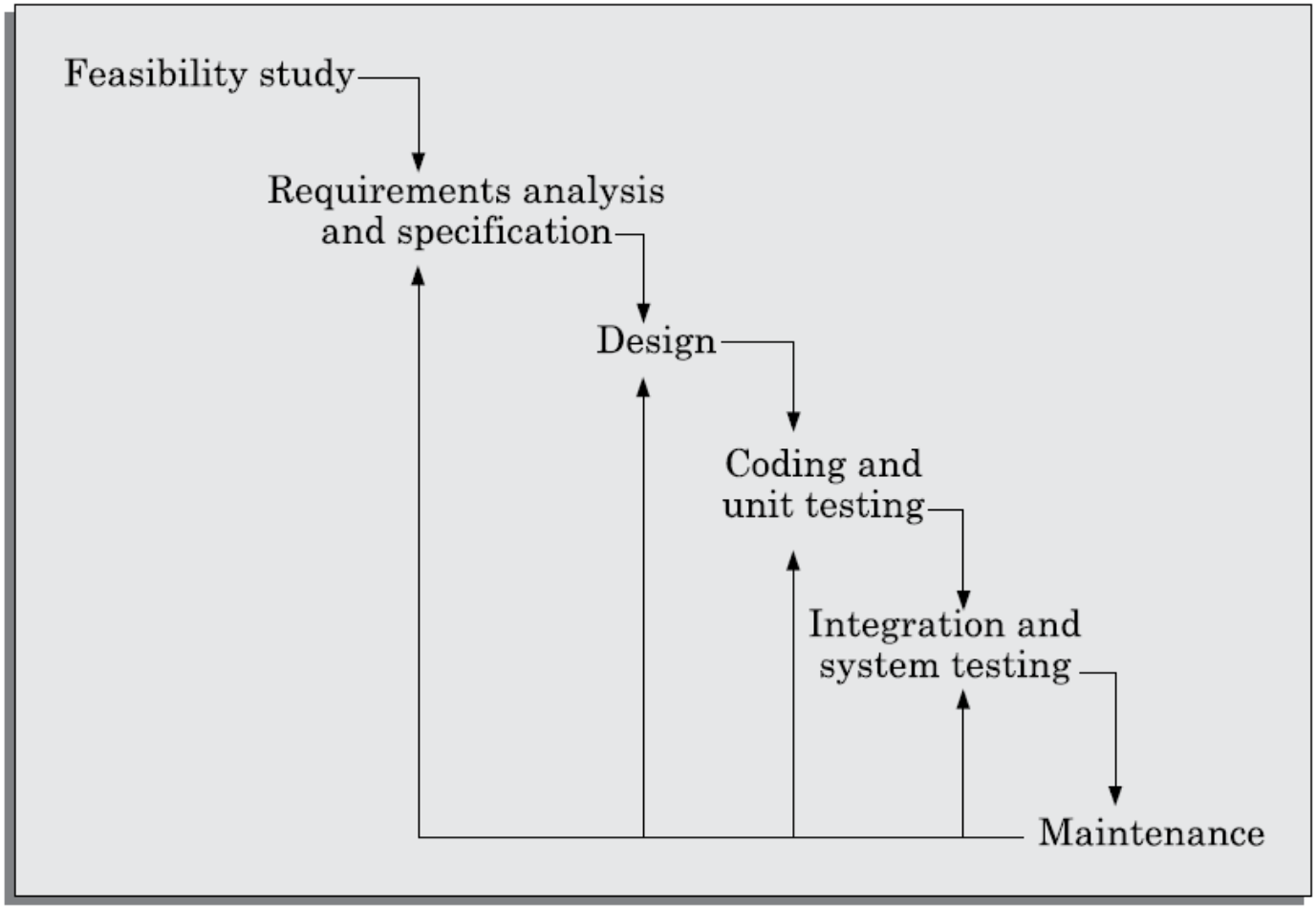
The Waterfall Model

- The waterfall model represents an idealized version of the software life cycle.
- If a problem that was created in an early development phase is not discovered until a later stage in the development process, the problem must either be ignored or the entire development process must be altered.
- A step-by-step development means that no tangible product is available for assessment by the client until late in the development process.
- The waterfall approach is generally only taken when the requirements of a project are well defined and not likely to change.

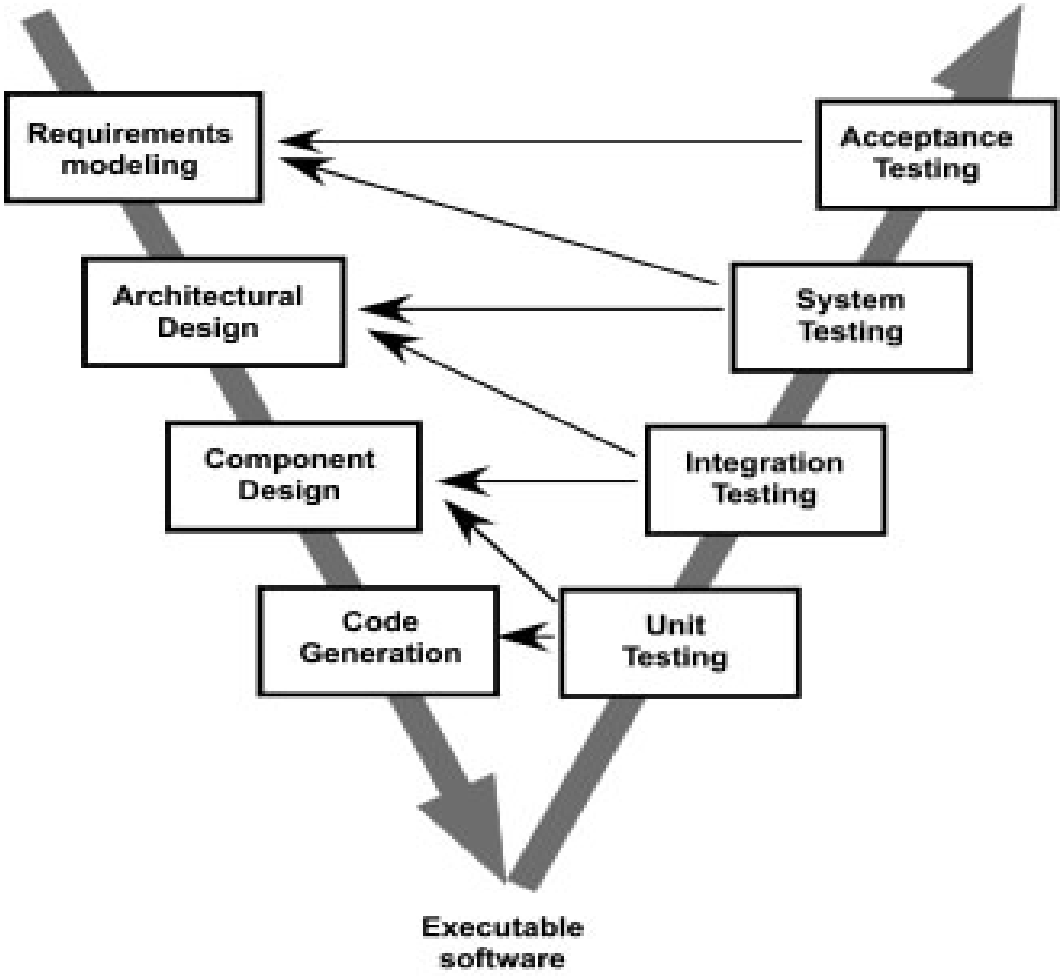
The Waterfall Model



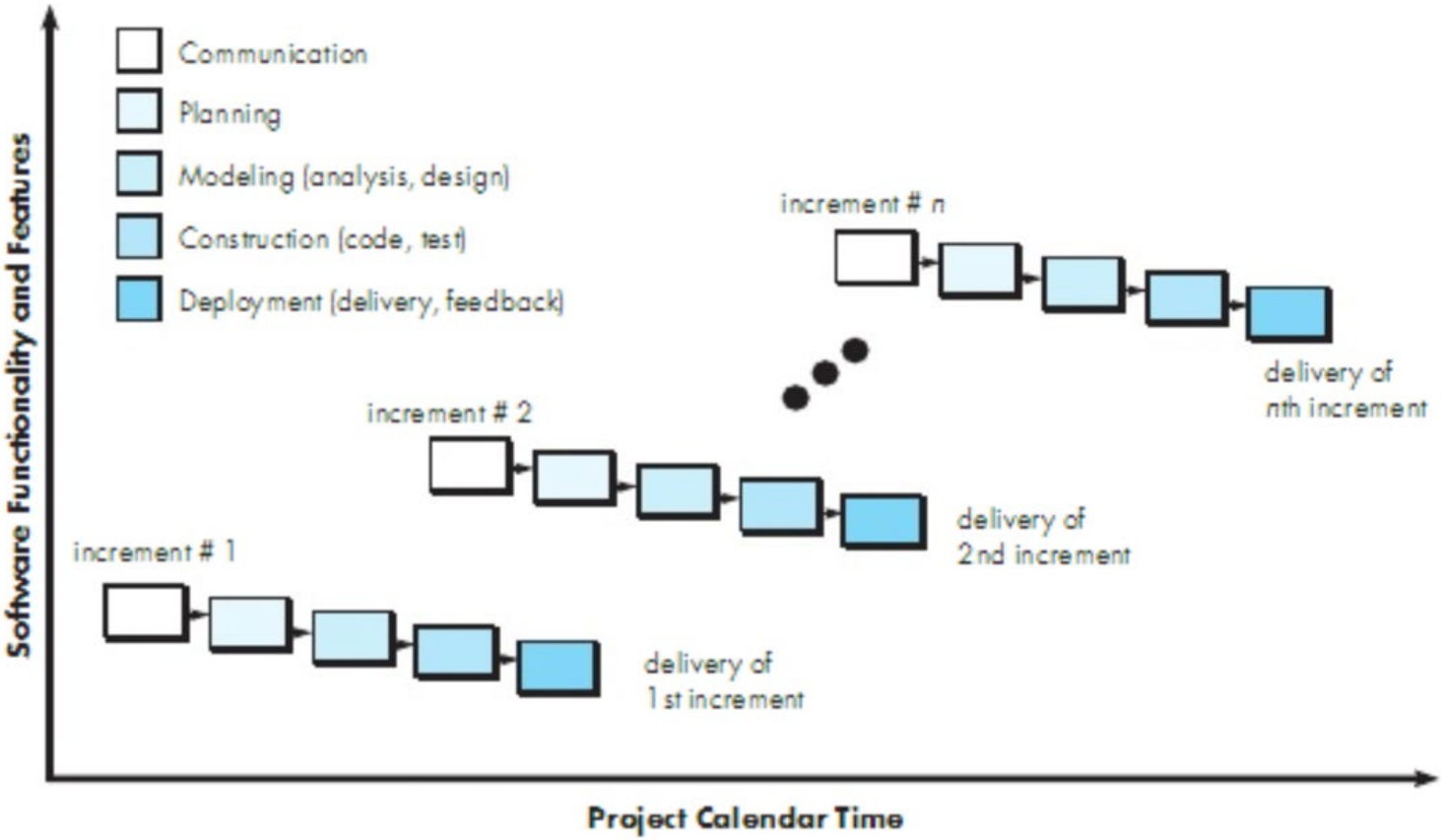
Iterative waterfall model



The V-Model



The Incremental Model



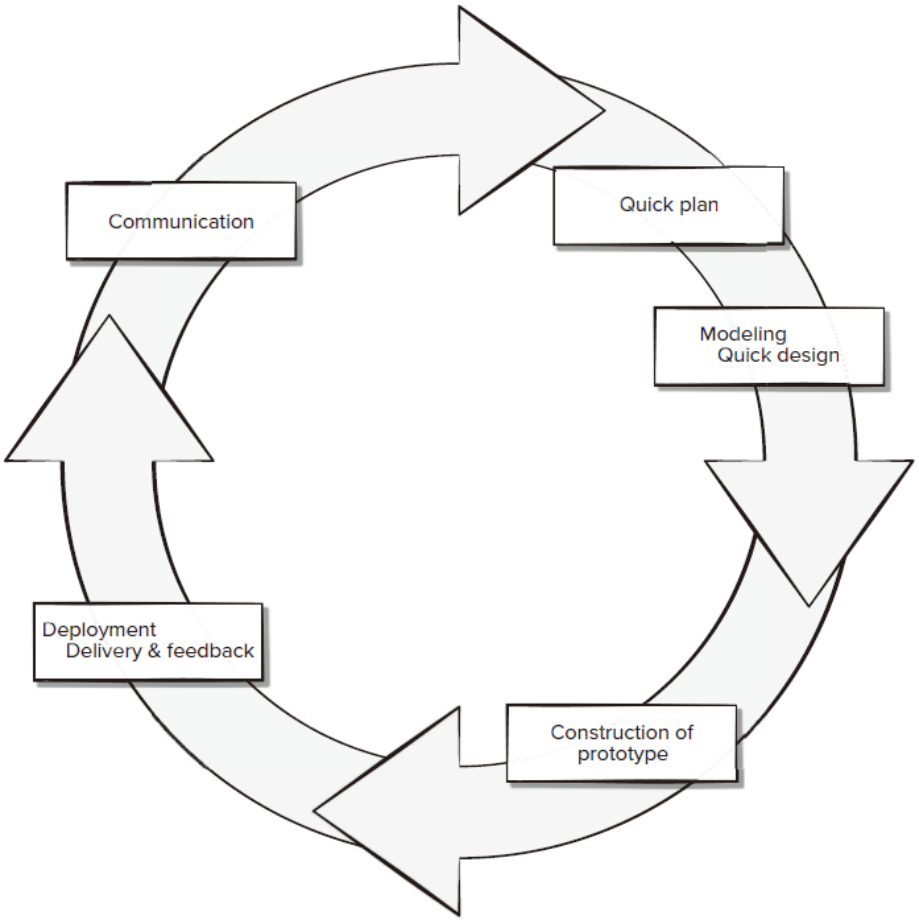
The Incremental Model

- The incremental model combines elements of the linear sequential model (applied repetitively) with the iterative philosophy of prototyping.
- Figure 4.3, the incremental model applies linear sequences in a staggered fashion as calendar time progresses. Each linear sequence produces a deliverable “increment” of the software.
- Example: Word-processing software developed using the incremental paradigm might deliver 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; and advanced page layout capability in the fourth increment… and so on.

Prototyping Model

FIGURE 2.4

The prototyping paradigm



Prototyping Model

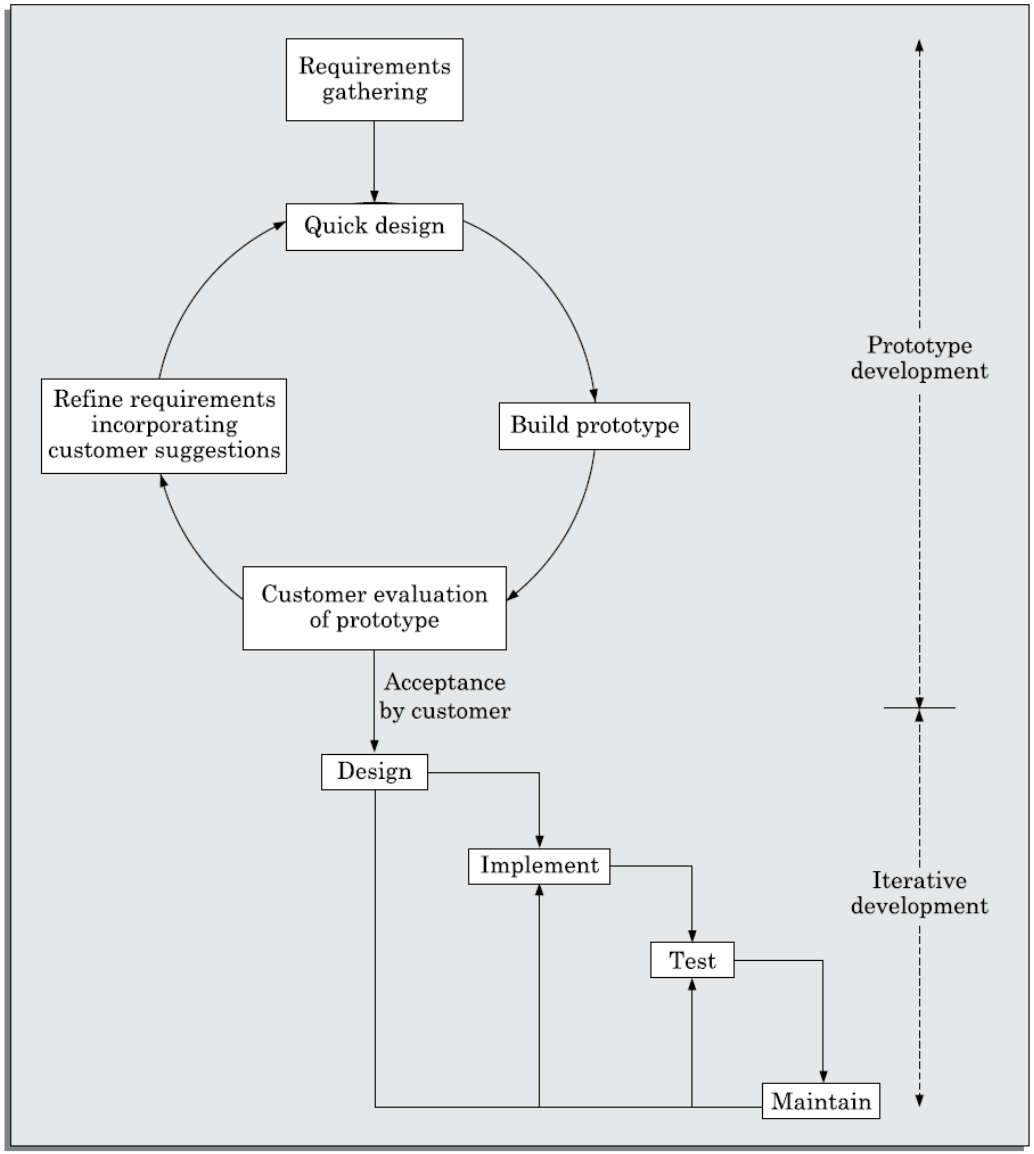
- Prototype model can be used for two types of project
  - The prototype model turns out to be especially useful in developing the GUI. For the user, it becomes much easier to form an opinion regarding what would be more suitable by experimenting with a working user interface, rather than trying to imagine the working of a hypothetical user interface.
  - The prototype model is especially useful when the exact technical solutions are unclear to the development team. A prototype can help them to critically examine the technical issues associated with product development. A prototype is often the best way to resolve the technical issues.

Prototyping Model

- Prototyping can also be problematic for the following reasons:
  - What are questions of Prototyping model?

    See Textbook, page 28.
- Although problems can occur, prototyping can be an effective paradigm for software engineering.
- The key is to define the rules of the game at the beginning. that is, the customer and developer must both agree that the prototype is built to serve as a mechanism for defining requirements. It is then discarded (at least in part) and the actual software is engineered with an eye toward quality and maintainability.

Case: Prototyping Model Usage



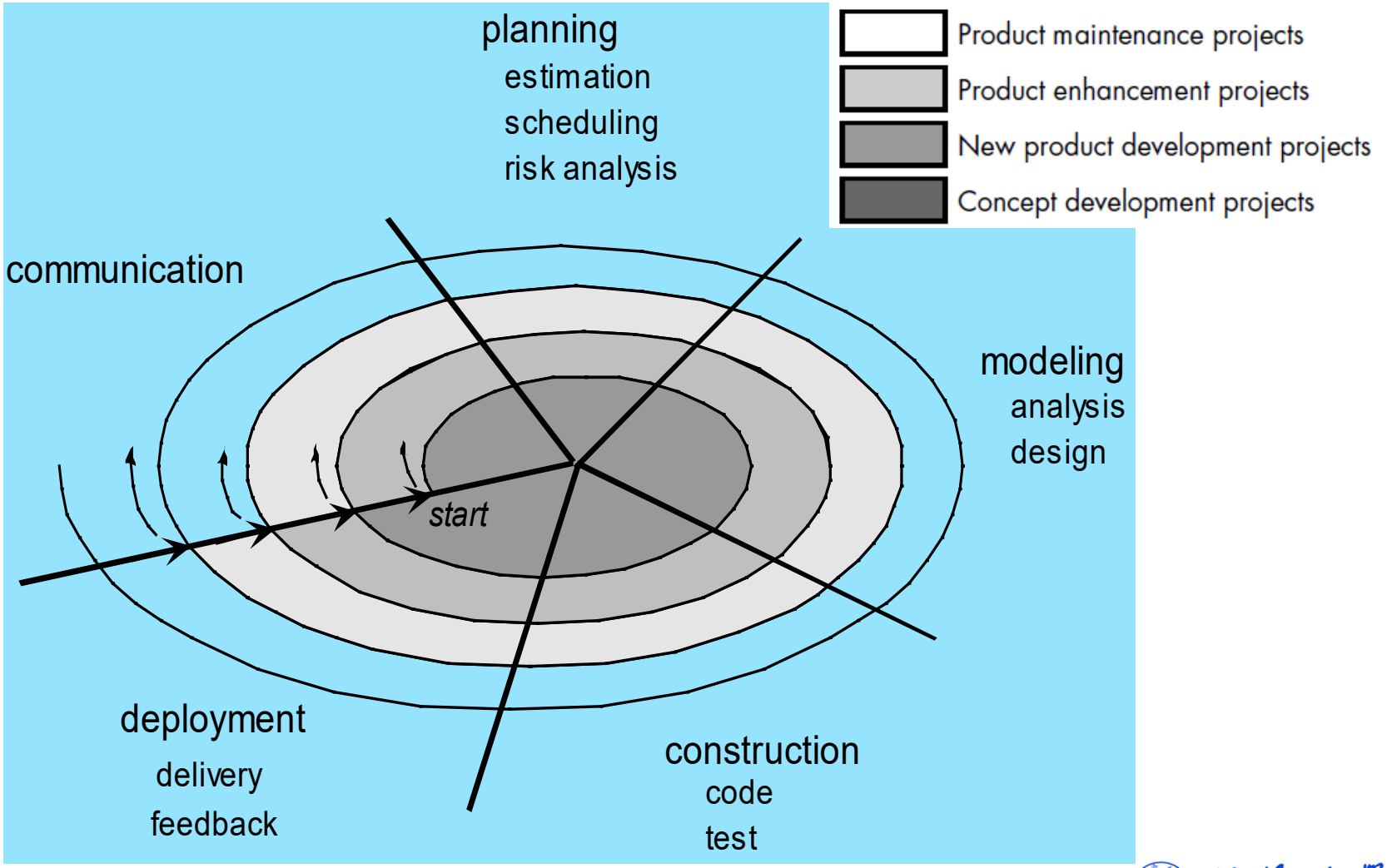
The Incremental Model and Prototype Model

- The prototype is evaluated by the customer/user and used to refine requirements for the software to be developed. Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done.
- Brooks [BRO75] *In most projects, the first system built is barely usable. It may be too slow, too big, awkward in use or all three. There is no alternative but to start again, smarting but smarter, and build a redesigned version in which these problems are solved . . . The management question, therefore, is not whether to build a pilot system and throw it away. You will do that. The only question is whether to plan in advance to build a throwaway, or to promise to deliver the throwaway to customers . . .*

The Incremental Model and Prototype Model

- **S**cenario (**E**xample from p37 of the textbook 8th Edition)
  - A customer often defines a set of general objectives for software but does not identify detailed input, processing, or output requirements. In other cases, the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human/machine interaction should take. In these, and many other situations, a prototyping paradigm may offer the best approach.
- The prototyping paradigm (page 24) begins with requirements gathering. Developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory(必须的).
  - A "quick design" leads to the construction of a prototype.

Evolutionary Model: The Spiral Model



Evolutionary Model: The Spiral Model

- The Spiral model, originally proposed by Boehm [BOE88], is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model.
- It provides the potential for rapid development of incremental versions of the software.
- Using the spiral model, software is developed in a series of incremental releases. During early iterations, the incremental release might be a paper model or prototype. During later iterations, increasingly more complete versions of the engineered system are produced.

Evolutionary Model: The Spiral Model

- The spiral model is a realistic approach to the development of large-scale systems and software. Because software evolves as the process progresses, the developer and customer better understand and react to risks at each evolutionary level.
- The spiral model uses prototyping as a risk reduction mechanism but, more important, enables the developer to apply the prototyping approach at any stage in the evolution of the product.
- It maintains the systematic stepwise approach suggested by the classic life cycle but incorporates it into an iterative framework that more realistically reflects the real world.
- The spiral model demands a direct consideration of technical risks at all stages of the project and, if properly applied, should reduce risks before they become problematic.

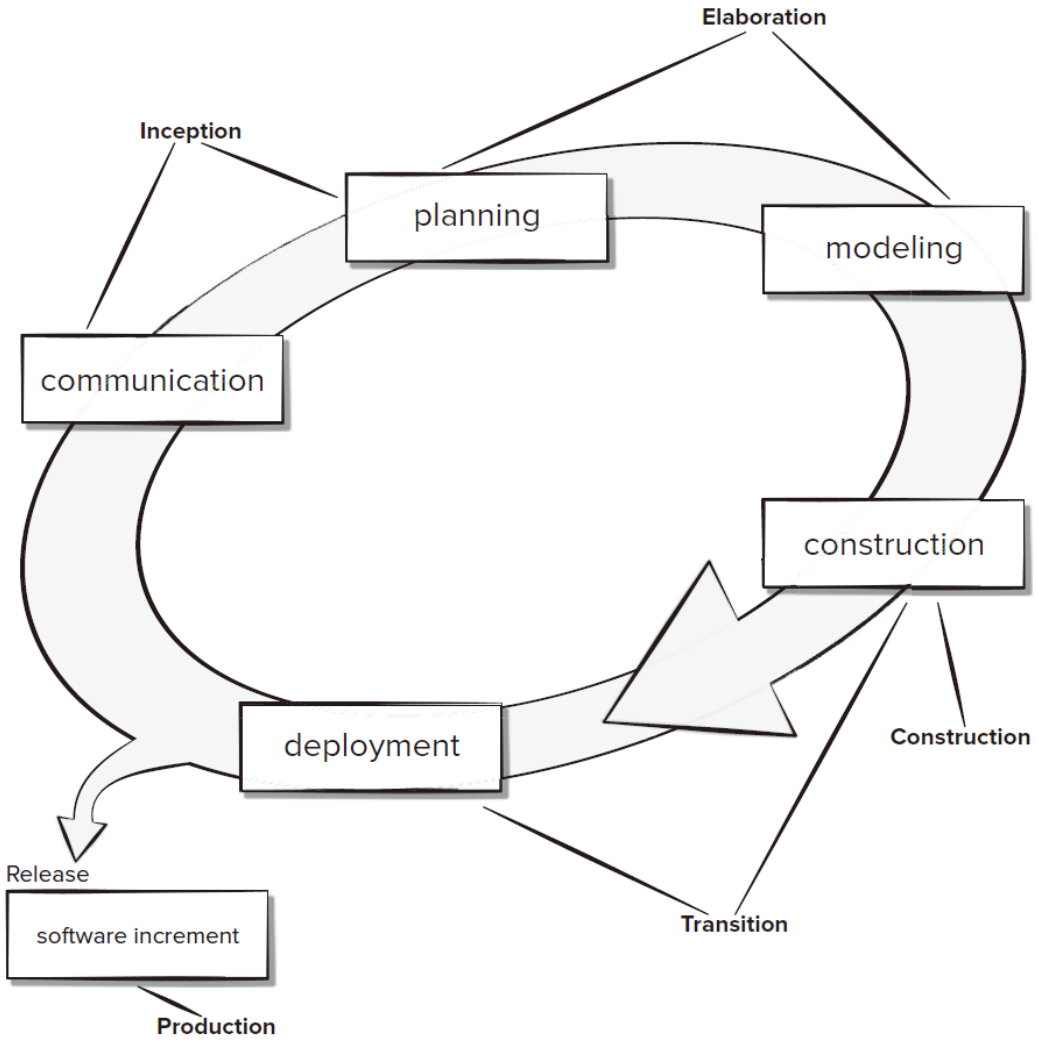
Evolutionary Model: The Spiral Model

- Spiral model as a meta model
  - As compared to the previously discussed models, the spiral model can be viewed as a meta model.
  - Since this model subsumes all the discussed models.
  - Example
    - A single loop spiral actually represents the waterfall model. The spiral model incorporates the systematic step-wise approach of the waterfall model.
    - The spiral model uses the approach of the prototyping model by first building a prototype in each phase before the actual development starts. This prototype is used as a risk reduction mechanism.
    - The spiral model can be considered as supporting the evolutionary model – the iterations along the spiral can be considered as evolutionary levels through which the complete system is built. This enables the developer to understand and resolve the risks at each level (i.e. iteration along the spiral).

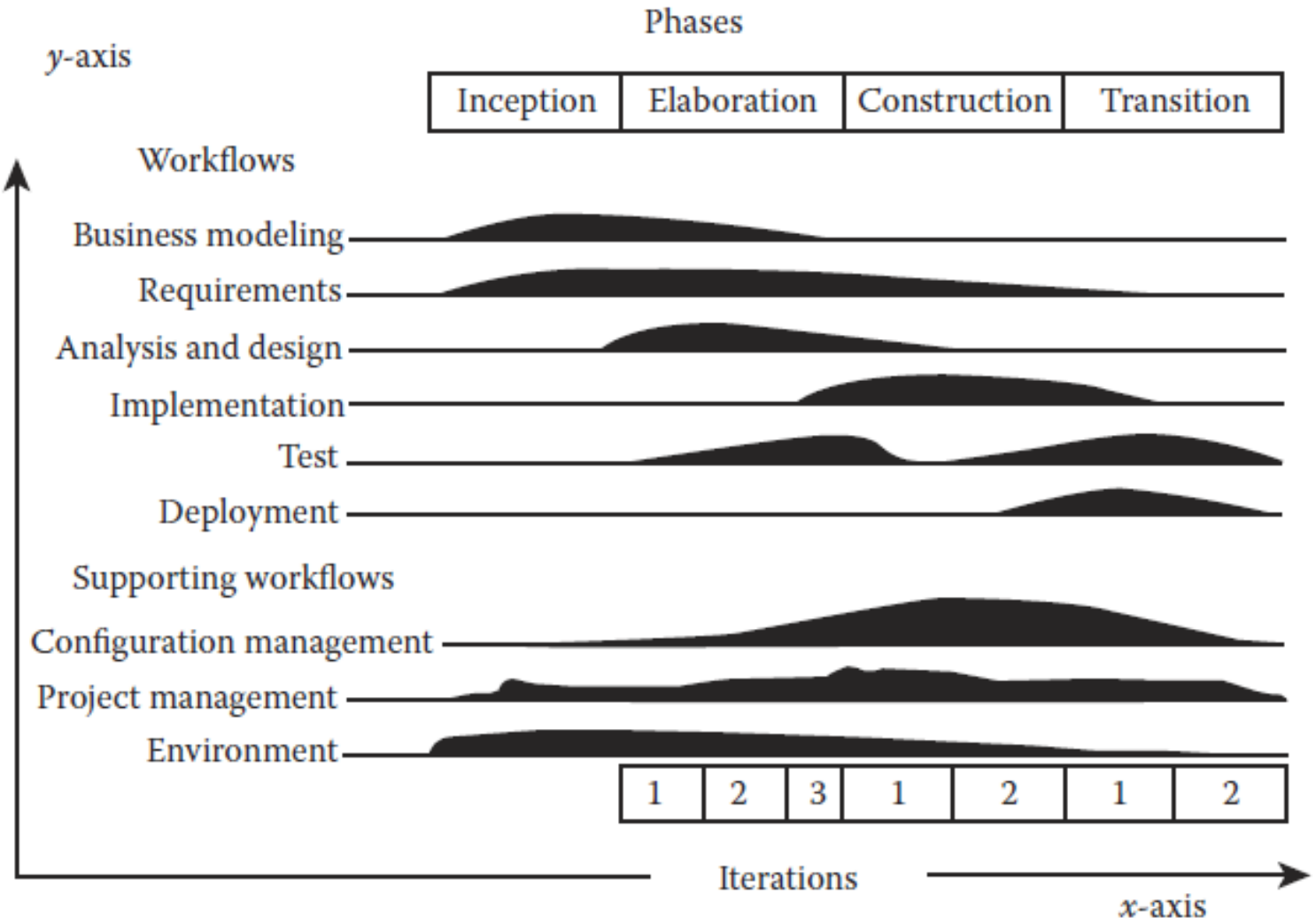
Evolutionary Model: the final word

- The intent of evolutionary models is to develop high-quality software in an iterative or incremental manner. However, it is possible to use an evolutionary process to emphasize flexibility, extensibility, and speed of development. The challenge for software teams and their managers is to establish a proper balance between these critical project and product parameters and customer satisfaction (the ultimate arbiter仲裁人 of software quality).

The Unified Process (UP)



UP Phases



UP Work Products

Inception phase

- Vision document
- Initial use-case model
- Initial project glossary
- Initial business case
- Initial risk assessment.
- Project plan,
  - phases and iterations.
- Business model,
  - if necessary.
- One or more prototypes

Elaboration phase

- Use-case model
- Supplementary requirements
  - including non-functional
- Analysis model
- Software architecture
  - Description.
- Executable architectural
  - prototype.
- Preliminary design model
- Revised risk list
- Project plan including
  - iteration plan
  - adapted workflows
  - milestones
  - technical work products
- Preliminary user manual

Construction phase

- Design model
- Software components
- Integrated software
  - increment
- Test plan and procedure
- Test cases
- Support documentation
  - user manuals
  - installation manuals
  - description of current
    - increment

Transition phase

- Delivered software increment
- Beta test reports
- General user feedback

Software product & process

TABLE 2.1

Comparing process models

Waterfall pros	It is easy to understand and plan. It works for well-understood small projects. Analysis and testing are straightfoward.
Waterfall cons	It does not accommodate change well. Testing occurs late in the process. Customer approval is at the end.
Prototyping pros	There is a reduced impact of requirement changes. The customer is involved early and often. It works well for small projects. There is reduced likelihood of product rejection.
Prototyping cons	Customer involvement may cause delays. There may be a temptation to "ship" a prototype. Work is lost in a throwaway prototype. It is hard to plan and manage.
Spiral pros	There is continuous customer involvement. Development risks are managed. It is suitable for large, complex projects. It works well for extensible products.
Spiral cons	Risk analysis failures can doom the project. The project may be hard to manage. It requires an expert development team.
Unified Process pros	Quality documentation is emphasized. There is continuous customer involvement. It accommodates requirements changes. It works well for maintenance projects.
Unified Process cons	Use cases are not always precise. It has tricky software increment integration. Overlapping phases can cause problems. It requires an expert development team.

Software product & process

- Margaret Davis [Dav95a] makes timeless comments on the duality(二元性、对偶性) of product and process:
  - . . . You can never derive or understand the full artifact, its context, use, meaning, and worth if you view it as only a process or only a product.
  - All of human activity may be a process, but each of us derives a sense of self-worth from those activities that result in a representation or instance that can be used or appreciated either by more than one person, used over and over, or used in some other context not considered. That is, we derive feelings of satisfaction from reuse of our products by ourselves or others.

Summary

- A generic process model for software engineering encompasses a set of Framework and Umbrella activities, actions, and work tasks.
- Each of a variety of process Models can be described by a different process flow – a description of how the framework activities, actions, and tasks are organized sequentially and chronologically(时间顺序).
- Software engineering is a discipline that integrates process, methods, and tools for the development of computer software.
- A number of different process models for software engineering have been proposed, each exhibiting strengths and weaknesses, but all having a series of generic phases in common.
- The principles, concepts, and methods that enable us to perform the process that we call software engineering are considered throughout the remainder of this course

Assignment

- Chapter 2: Problems to ponder
  - 2.1. Baetjer [Bae98] notes: “The process provides interaction between users and designers, between users and evolving tools, and between designers and evolving tools [technology].” List five questions that：

    (1) designers should ask users；

    (2) users should ask designers；

    (3) users should ask themselves about the software product that is to be built；

    (4) designers should ask themselves about the software product that is to be built and the process that will be used to build it.

huangjie@tongji.edu.cn

Assignment

- Chapter 2: Problems to ponder
  - 2.4 A common problem during communication occurs when you encounter two stakeholders who have conflicting ideas about what the software should be. That is, they have mutually conflicting requirements. Develop a process pattern that addresses this problem and suggest an effective approach to it.

huangjie@tongji.edu.cn

Assignments

- Design Thinking

  Based on the course selected and the characteristics of the course project, select an appropriate software process for the project design and development.

- Preview

  《Software Engineering》

  Chapter 3 Agile and Agile Process

  written by R.S.Pressman

huangjie@tongji.edu.cn