Chapter 3
Process Models

Software Engineering: A Practitioner's Approach, 6th edition
by Roger S. Pressman

本章要点

- 惯例模型
  - 瀑布模型
  - V模型
  - 增量模型
  - 演化模型
  - UP模型

线性模型

非线性模型

Prescriptive Models[惯例模型]

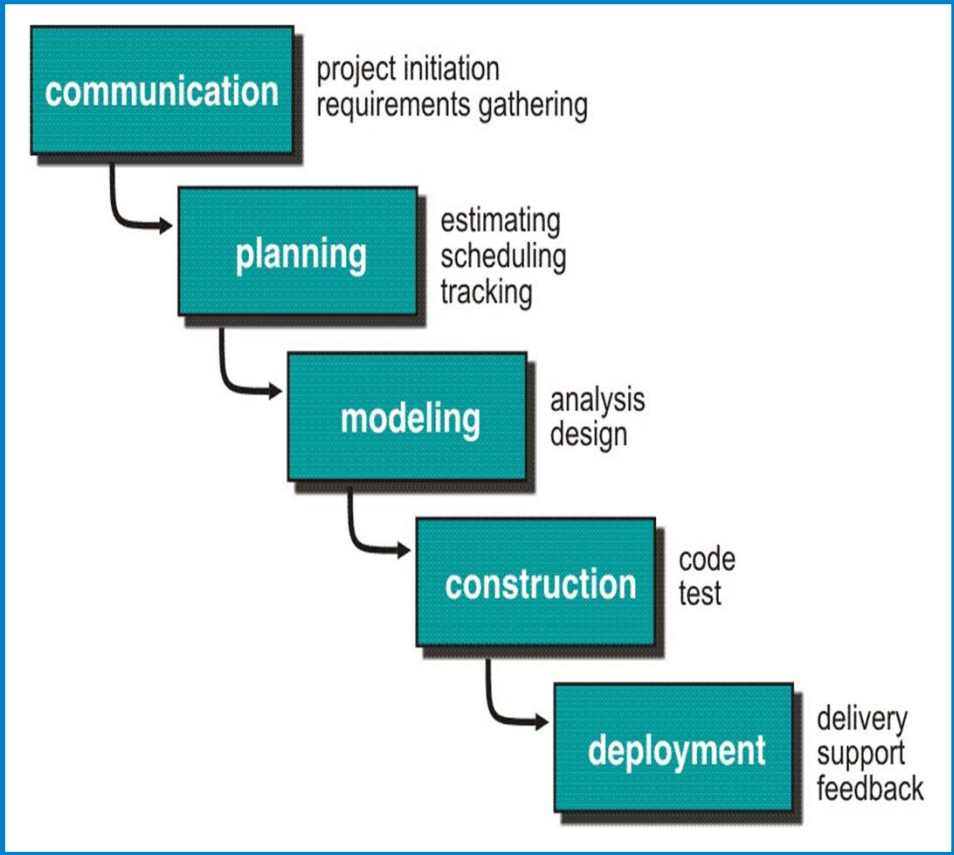
提倡一种有序的方法

Prescriptive process models **advocate an orderly 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 and coherence in software work?

The Waterfall Model



- **Bradac** found that the linear nature of the waterfall model leads to "**blocking states**".
  - Where some members must wait for other members of the team to complete dependent tasks.
  - Especially, at the beginning of the project.
- Still, however, the waterfall model serve as a useful process model where requirements are fixed and work is to proceed to completion in a linear manner.

Waterfall Model Assumptions

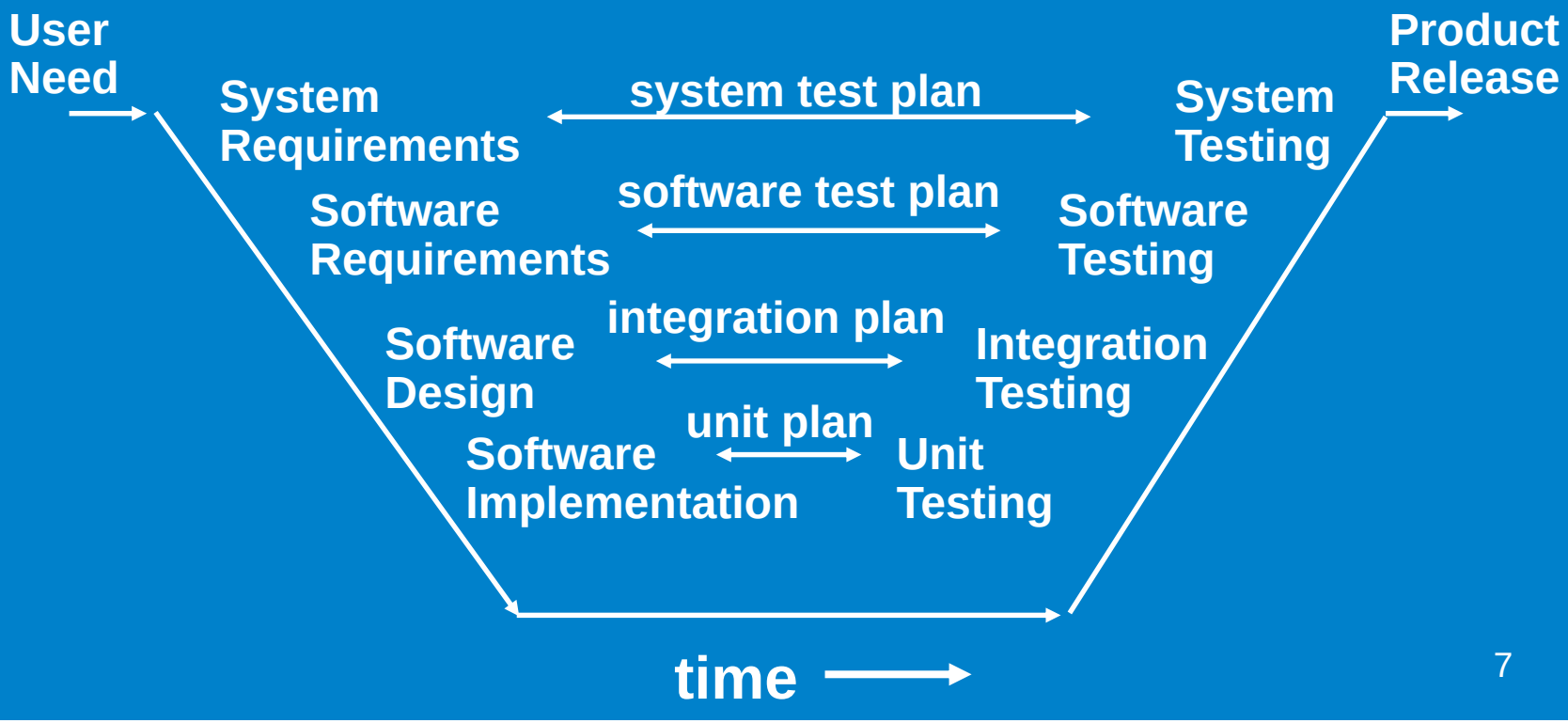
1. The requirements are knowable in advance of implementation.
2. The requirements have no unresolved, high-risk implications.
   - e.g., risks due to COTS choices, cost, schedule, performance, safety, security, user interfaces, organizational impacts.
3. The nature of the requirements will not change very much.
   - During development; during evolution.

Waterfall Model Assumptions…

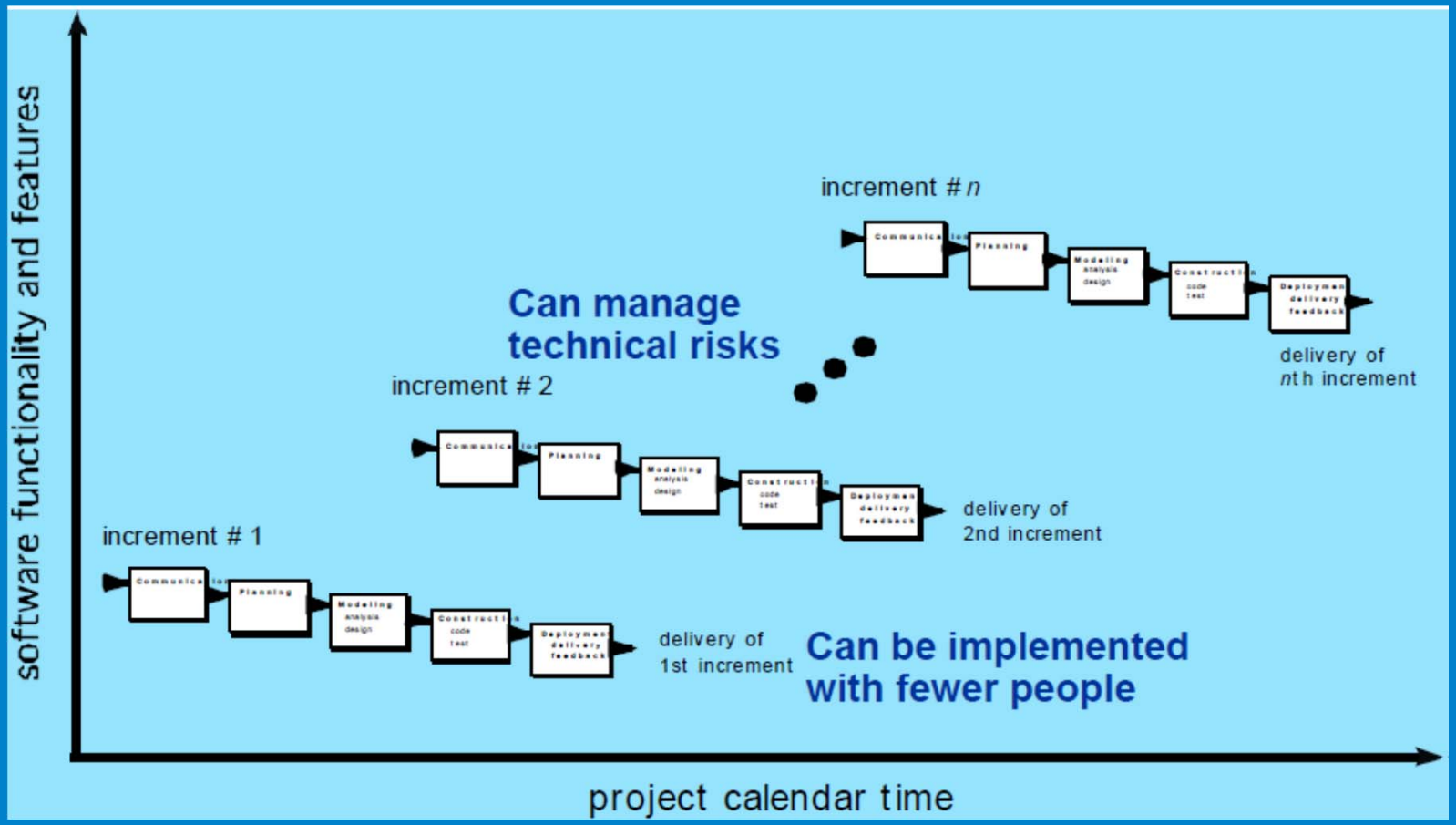
4. **The requirements are compatible with all the key system stakeholders' expectations.**
   - e.g., users, customer, developers, maintainers, investors.
5. **The right architecture for implementing the requirements is well understood.**
6. **There is enough calendar time to proceed sequentially.**

The V Model

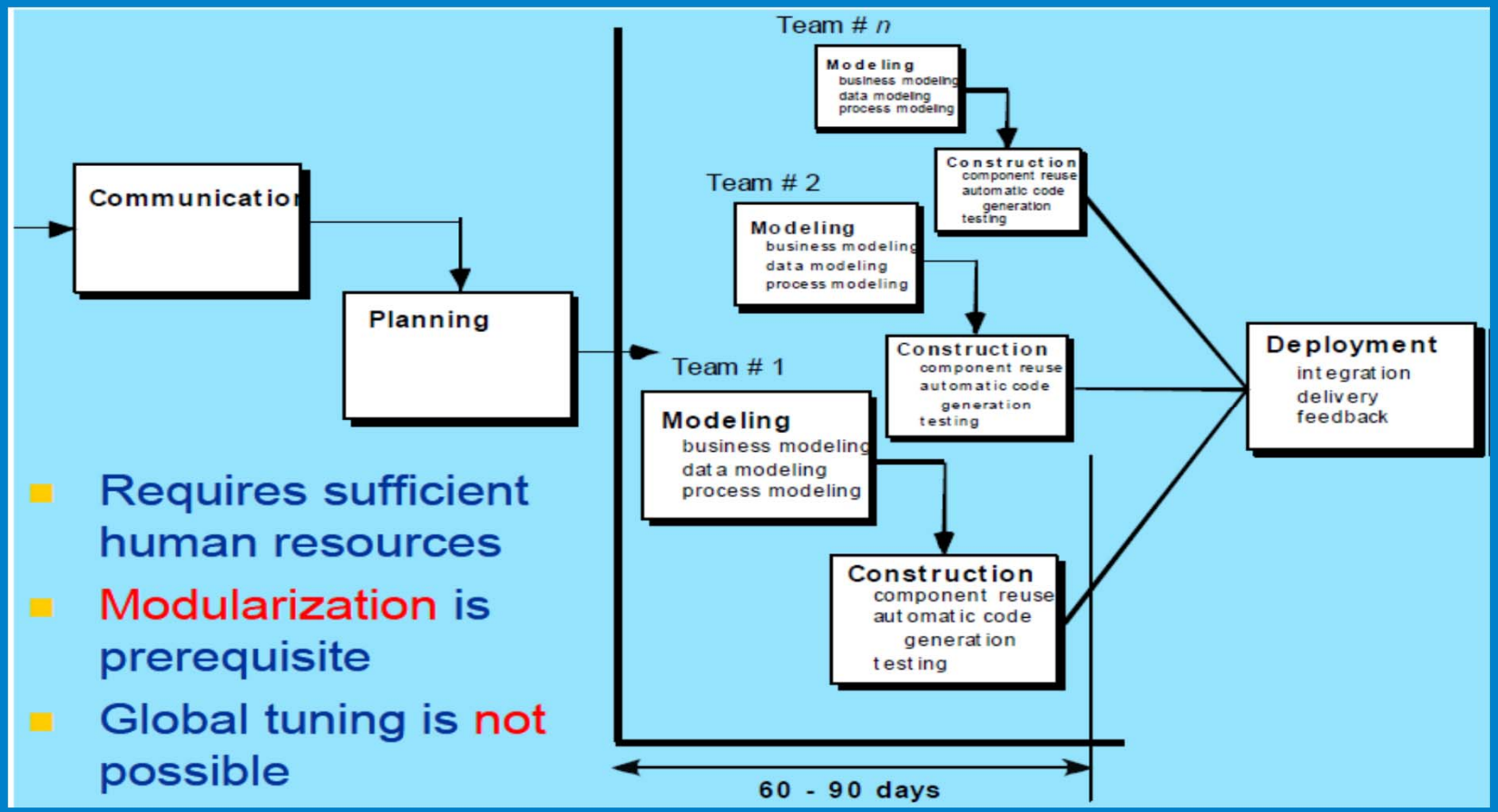
- If we rely on testing alone, defects **created first** are **detected last**



Incremental Models: Incremental



Incremental Models: RAD Model

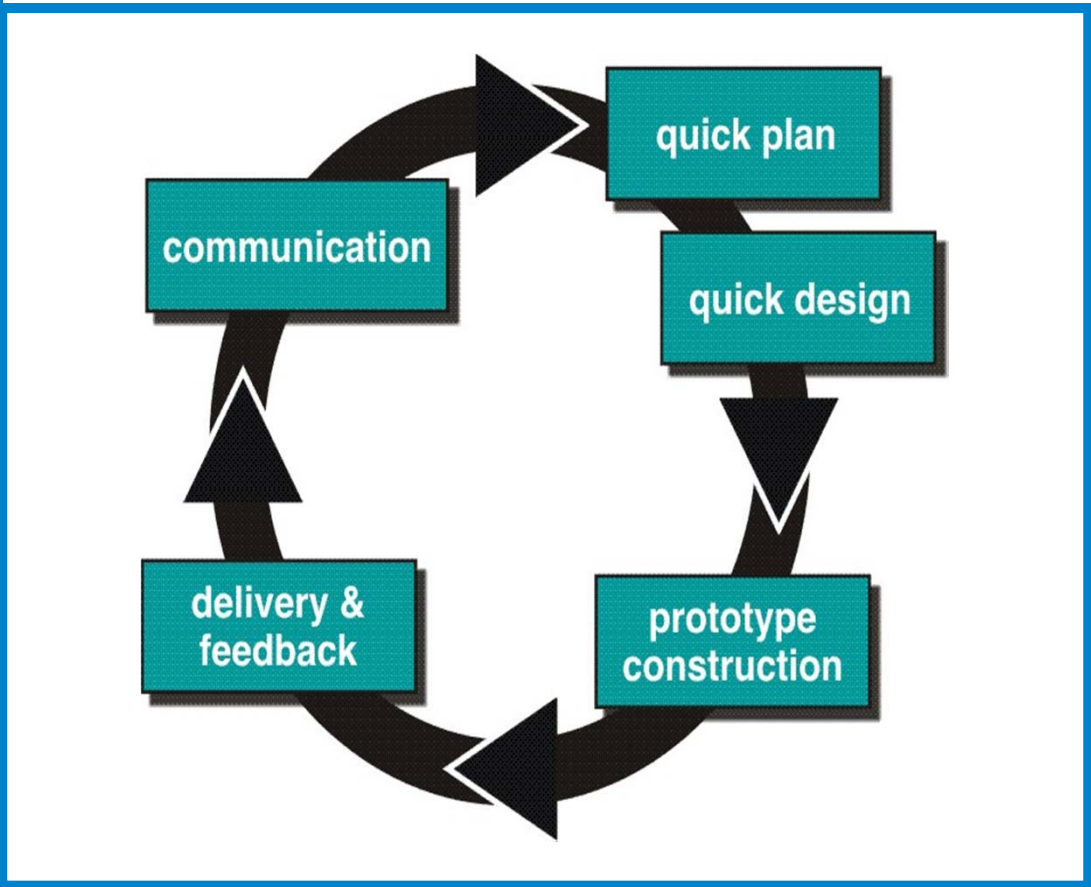


- Requires sufficient human resources
- Modularization is prerequisite
- Global tuning is not possible

Evolutionary Models: Prototyping

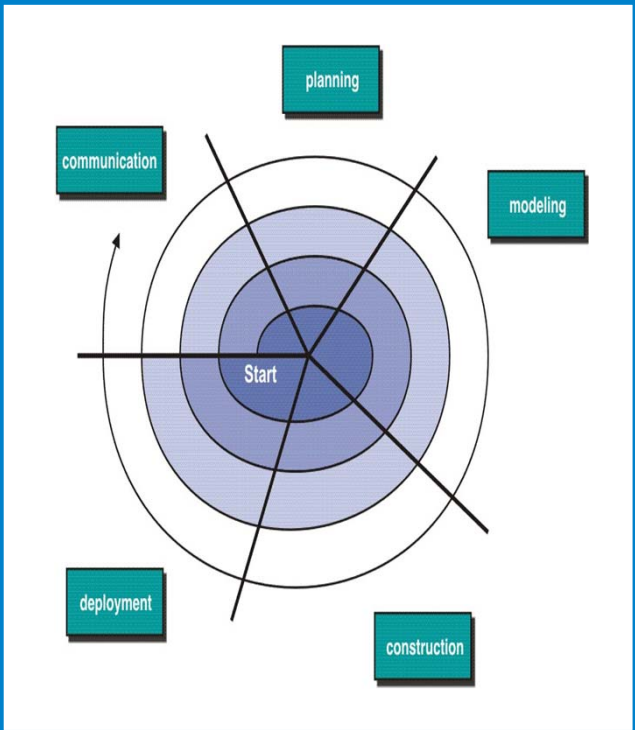
- A prototyping paradigm is the best-fit for the following situations.
  - A customer **does not identify detailed requirements** for SW.
  - SW engineers are **not sure of the efficiency** of an algorithm, usability of SW, and so on.
- In other words, prototyping paradigm helps SW engineers and the customers to **understand what is to be built.**
- The quick design and implementation focuses on a representation of those aspects of the SW that will be visible to the customer.
  - Ideally, the prototype serves as **a mechanism for identifying SW requirements**.

Evolutionary Models: Prototyping…



- Some problems in prototyping paradigm
  - SW engineers **try to modify** the prototype to use as a working version
  - Once the customer see the working prototype, he/she **expects to get** working product soon

Evolutionary Models: The Spiral



- The spiral model was defined by Barry Boehm in his article "A Spiral Model of Software Development and Enhancement" from 1986. This model was not the first model to discuss iteration, but it was the first model to explain why the iteration matters. As originally envisioned, the iterations were typically 6 months to 2 years long. The spiral model (Boehm, 1988) aims at risk reduction by any means in any phase. The spiral model is often referred to as a **risk-driven model**.
- **Definition**: The spiral model, also known as the spiral lifecycle model, is a systems development lifecycle (SDLC) model used in information technology (IT). This model of development **combines the features** of the **prototyping model** and the **waterfall model**. The spiral model is favored for large, expensive, and complicated projects.

The steps in the spiral model

1. The new system requirements are defined in as much detail as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
2. A preliminary design is created for the new system.
3. A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
4. A **second prototype** is evolved by a fourfold procedure:

   (1) evaluating the first prototype in terms of its strengths, weaknesses, and risks;

   (2) defining the requirements of the second prototype;

   (3) planning and designing the second prototype;

   (4) constructing and testing the second prototype.

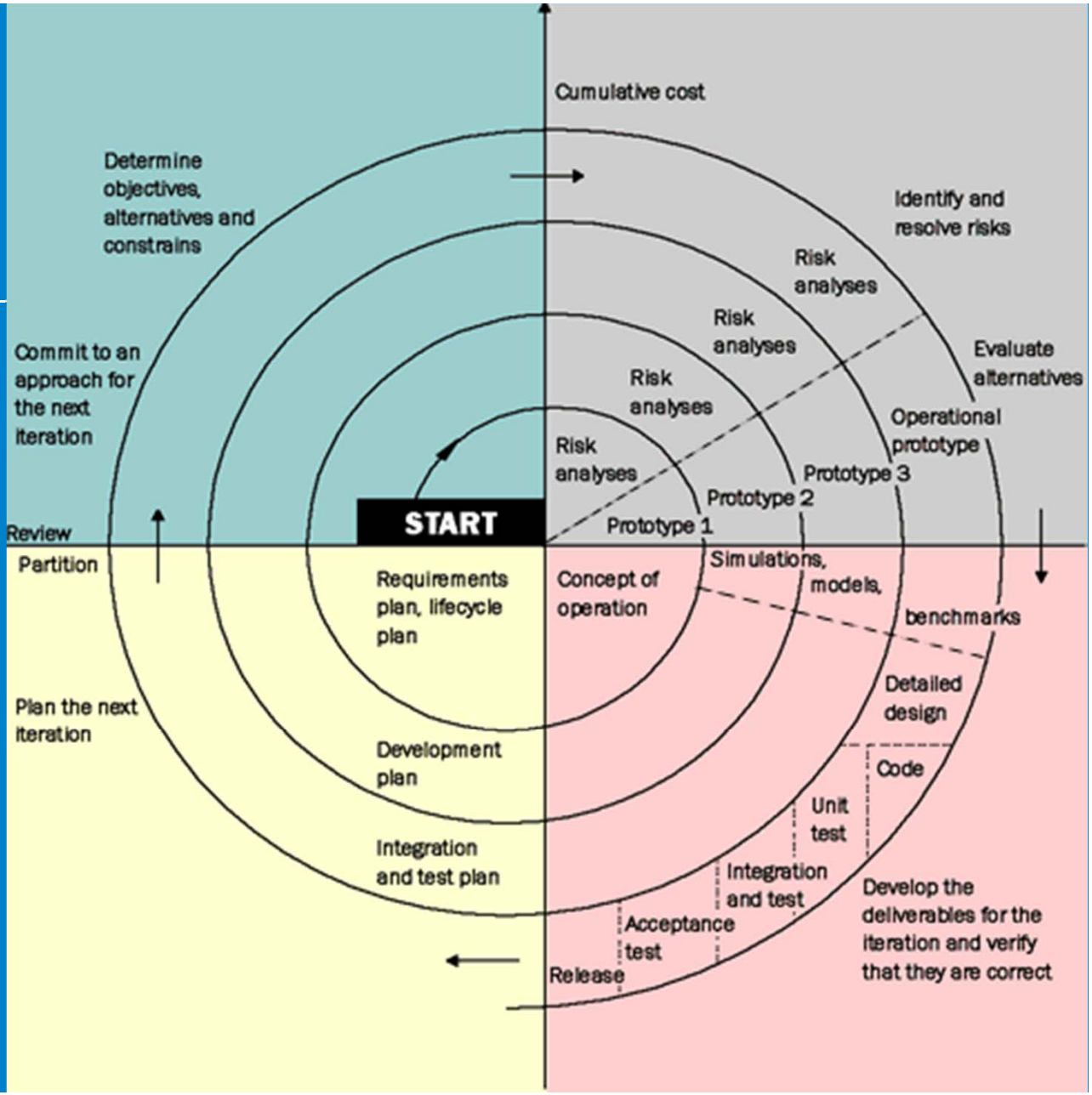
The steps in the spiral model…

5. At the customer‘s option, the entire **project can be aborted**（流产） if the risk is deemed（认为）too great. Risk factors might involve development cost overruns(超出限度), operating-cost miscalculation（误算）, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
6. The existing prototype is evaluated in the same manner as was the previous prototype, and, if necessary, **another prototype** is developed from it according to the fourfold procedure outlined above.
7. The preceding steps are iterated **until** the customer is satisfied that the refined prototype represents the final product desired.
8. The final system is constructed, based on the **refined prototype**.
9. The **final system** is thoroughly evaluated and tested. Routine maintenance is carried out on a continuing basis to prevent large-scale failures and to minimize downtime

Full Spiral Model

- Radial dimension[按射线方向]: cumulative cost to date
- Angular dimension[按螺旋方向]: progress through the spiral

Full Spiral Model ...



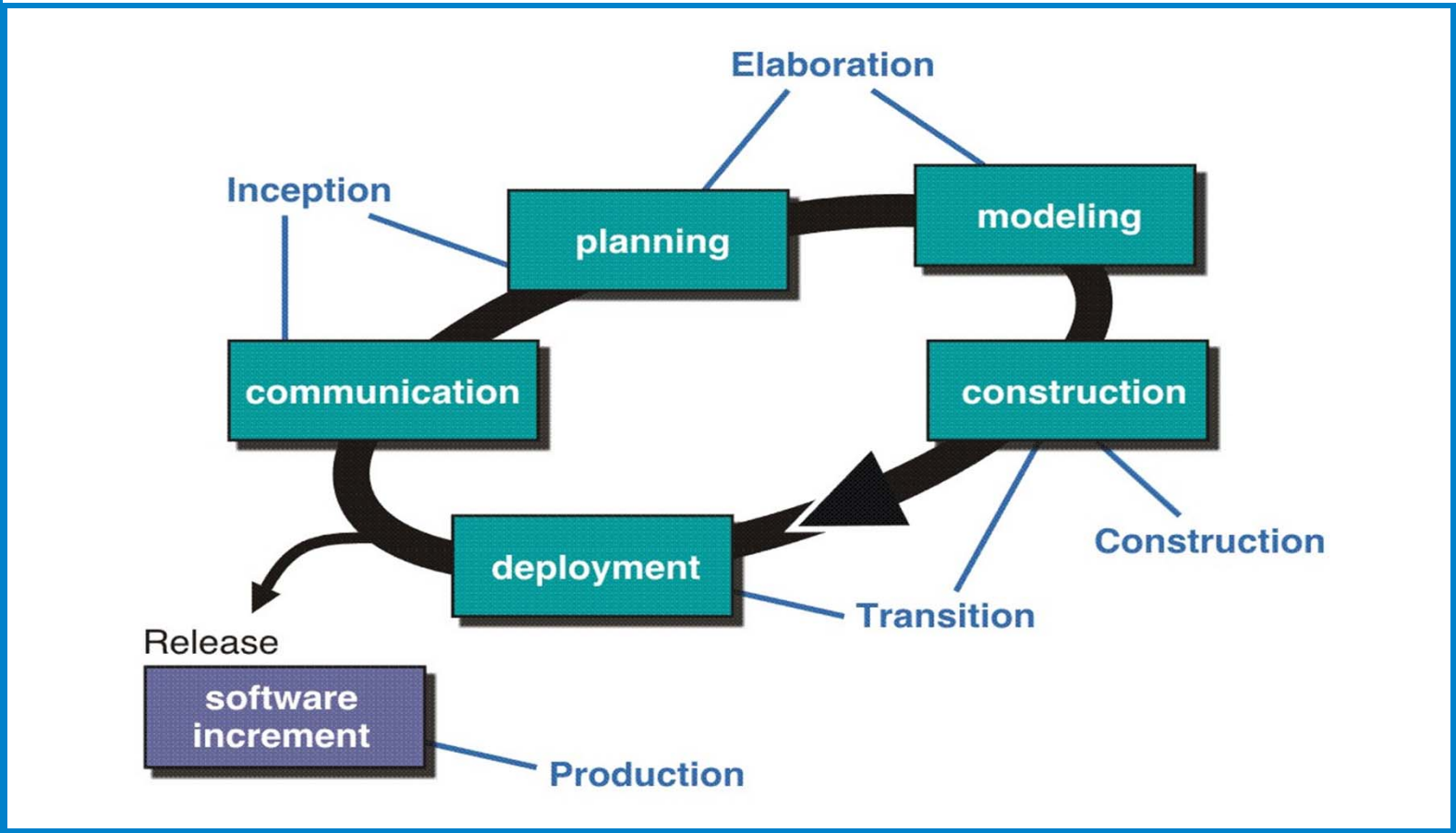
Unified Process Model

A software process that is:

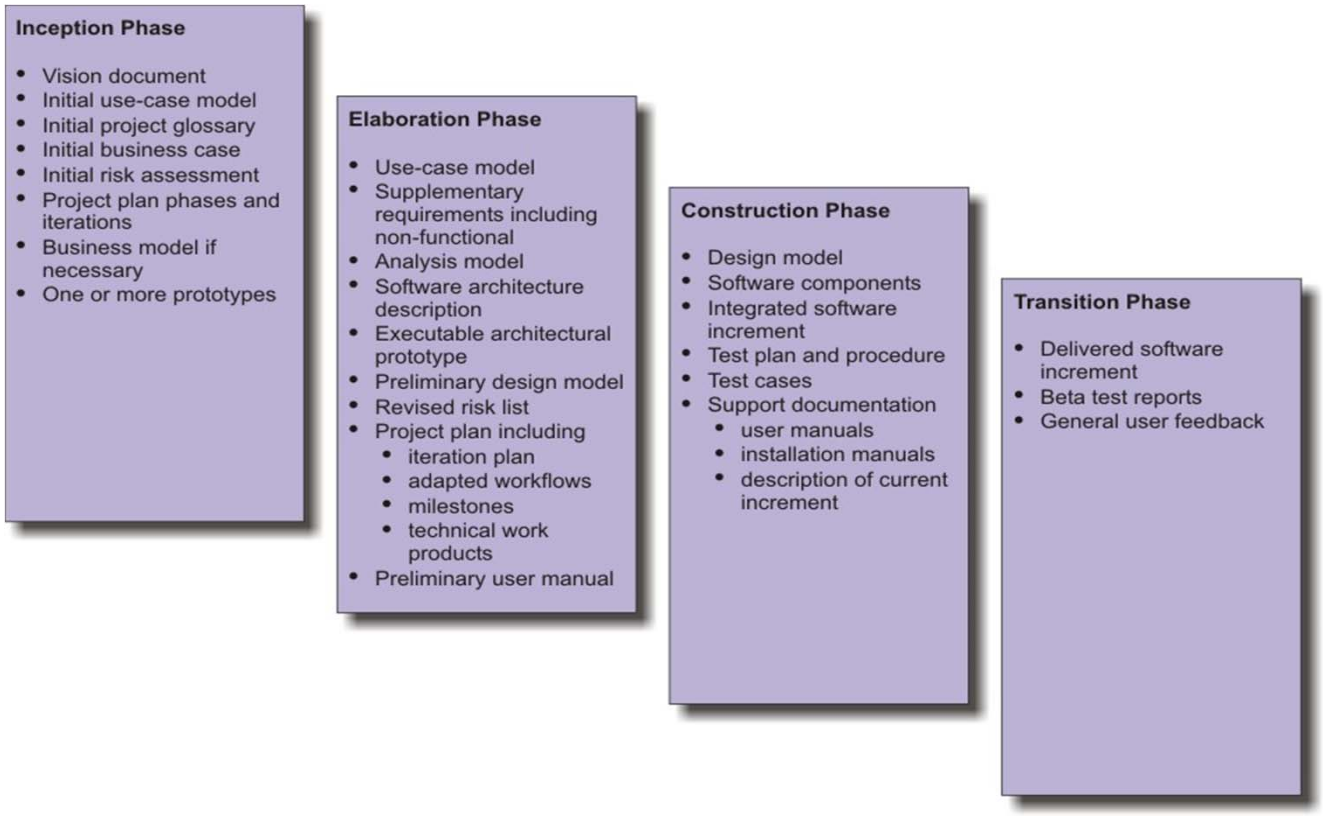
- use-case driven
- architecture-centric
- iterative and incremental

Closely aligned with the
Unified Modeling Language (**UML**)

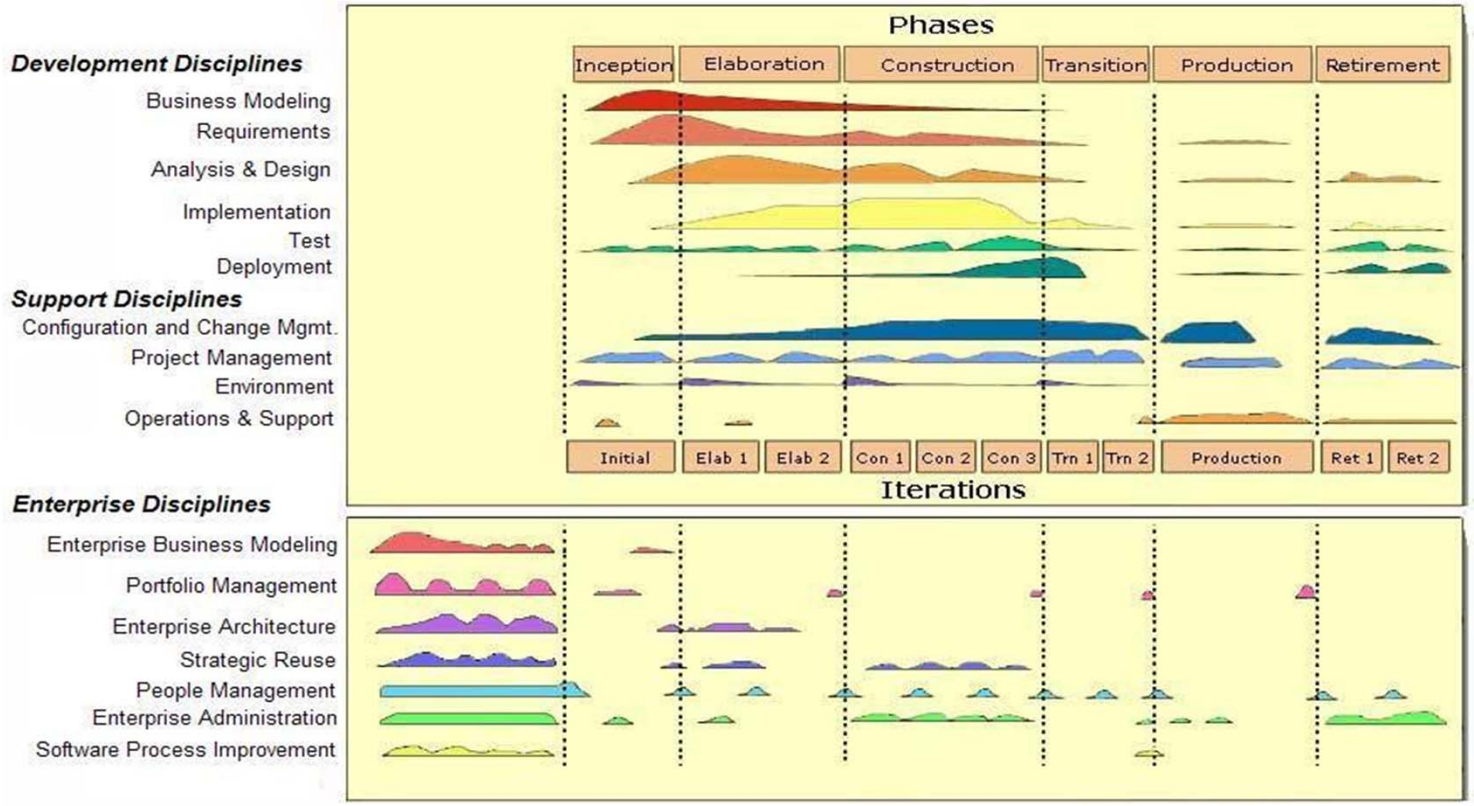
The Unified Process (UP)



UP Work Products



Lifecycle for Enterprise Unified Process



Still Other Process Models

- Component based development—the process to apply when **reuse** is a development objective
- Formal methods—emphasizes the **mathematical specification** of requirements
- AOSD—provides a process and methodological approach for defining, specifying, designing, and constructing **aspects**

Conclusions

- Different life-cycle models
- Each with own strengths[各有所长/寸有所长]
- Each with own weaknesses [各有所短/尺有所短]
- Best suggestion
  - "**Mix-and-match**" life-cycle model