Chapter 2

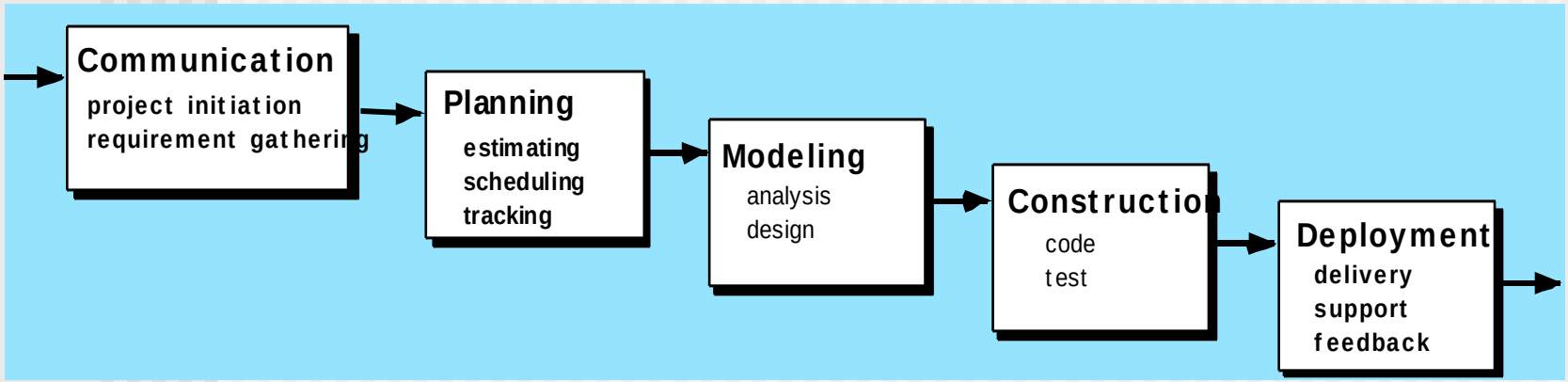
Process Models

Slide Set to accompany

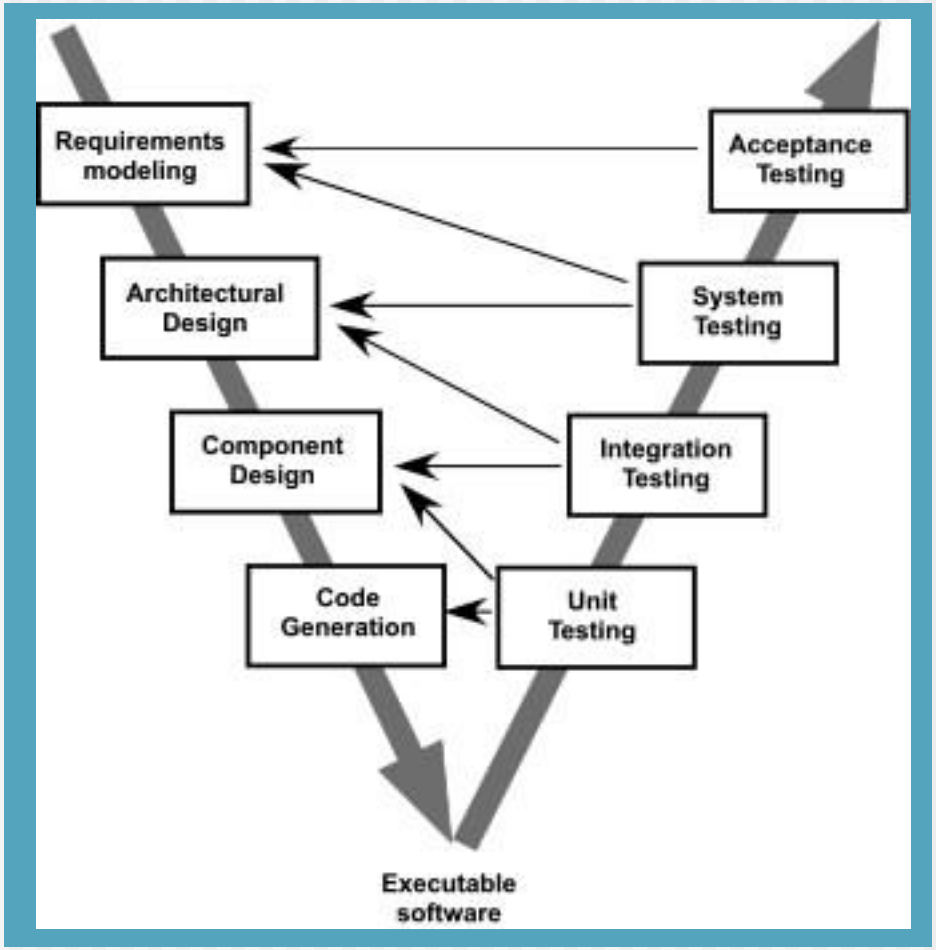
Software Engineering: A Practitioner's Approach, 7/e

by Roger S. Pressman

The Waterfall Model



The V-Model



KEY POINT

The V-model illustrates how verification and validation actions are associated with earlier engineering actions.

The Waterfall Model

Why does the waterfall model sometimes fail?

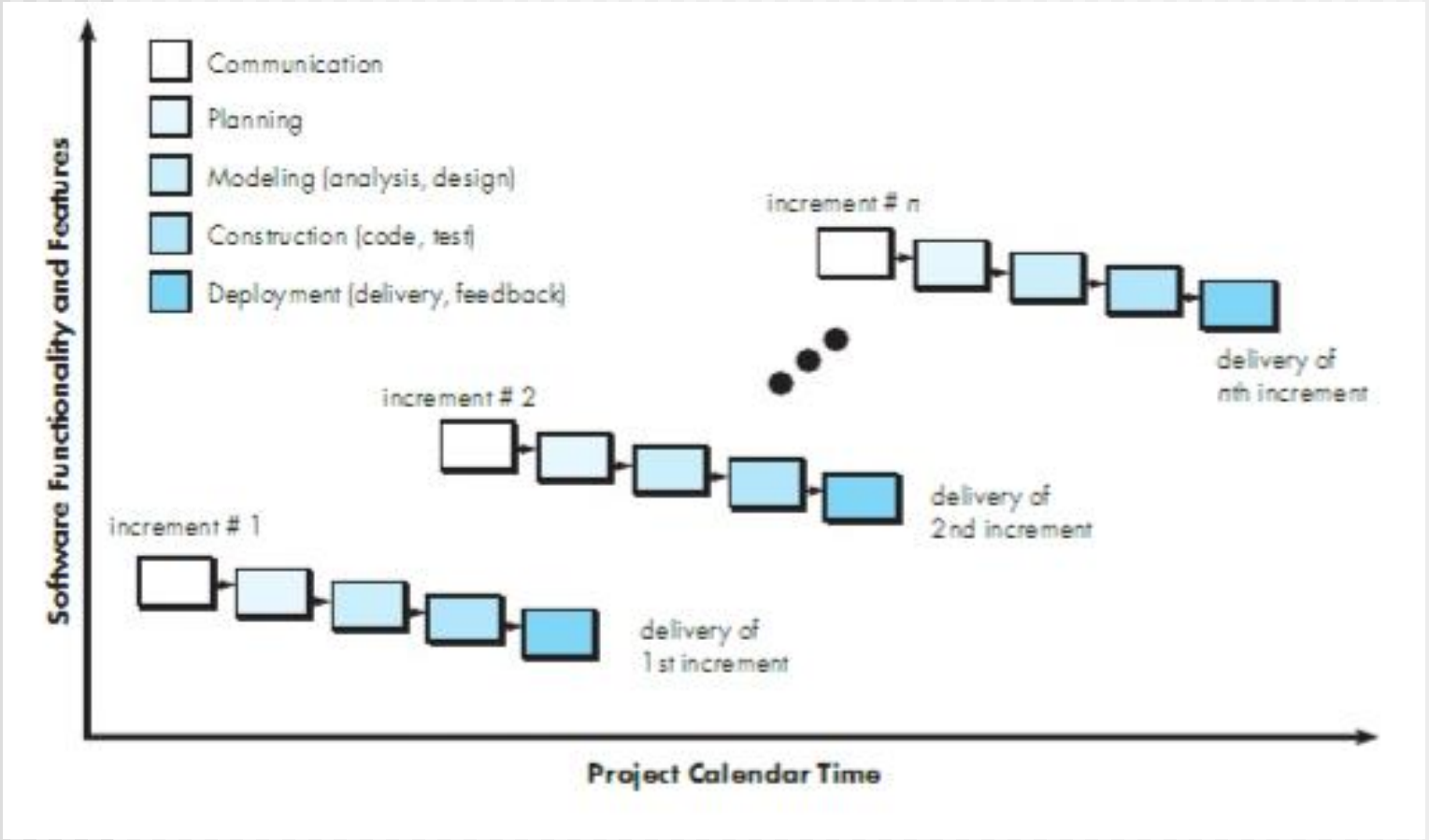
1) **Real projects rarely follow the sequential flow that the model proposes.**
2) **It is often difficult for the customer to state all requirements explicitly.**
3) **The customer must have patience.**

- Bradac [Bra94] found that the linear nature of the classic life cycle leads to "blocking states" in which some project team members must wait for other members of the team to complete dependent tasks.
- However, it can serve as a useful process model in situations where requirements are fixed and work is to proceed to completion in a linear manner.

The Incremental Model

- The incremental model delivers a series of releases, called **increments**, that provide progressively more functionality for the customer as each increment is delivered.
- When an incremental model is used, the first increment is often a *core product.* That is, basic requirements are addressed but many supplementary features (some known, others unknown) remain undelivered

The Incremental Model



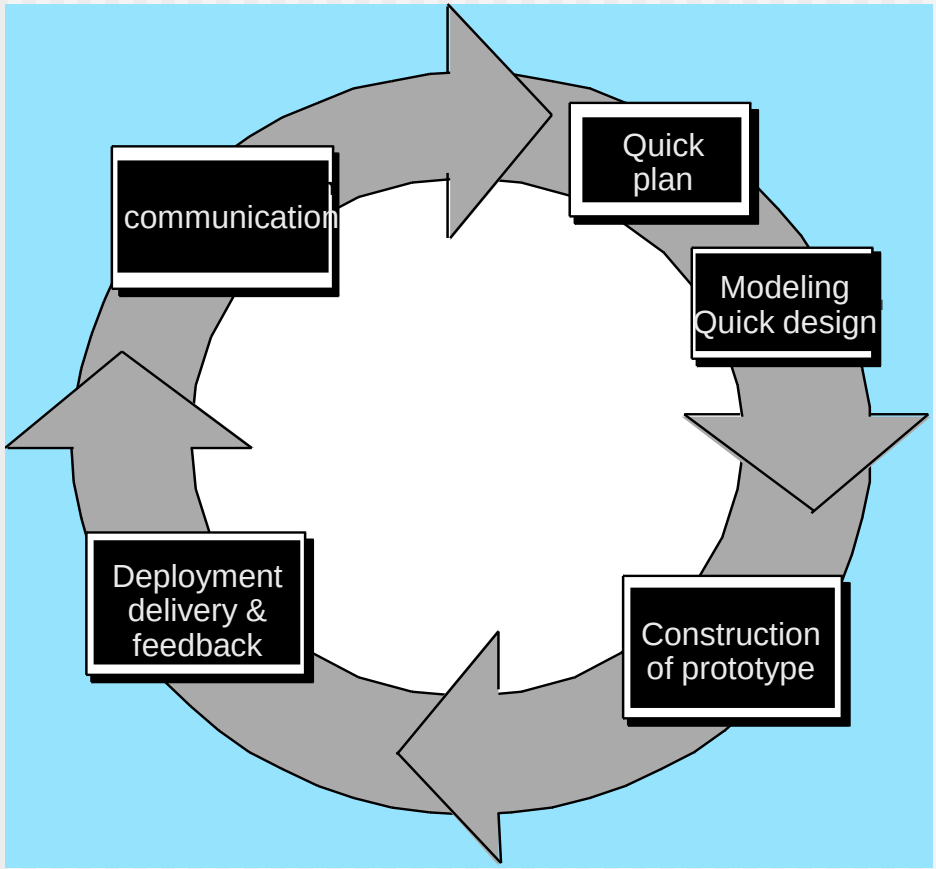
Evolutionary Models: Prototyping

Often, a customer defines a set of general objectives for software, but does not identify detailed requirements for functions and features. 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.

- 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.
- The prototyping paradigm assists you and other stakeholders to better understand what is to be built **when requirements are fuzzy**.

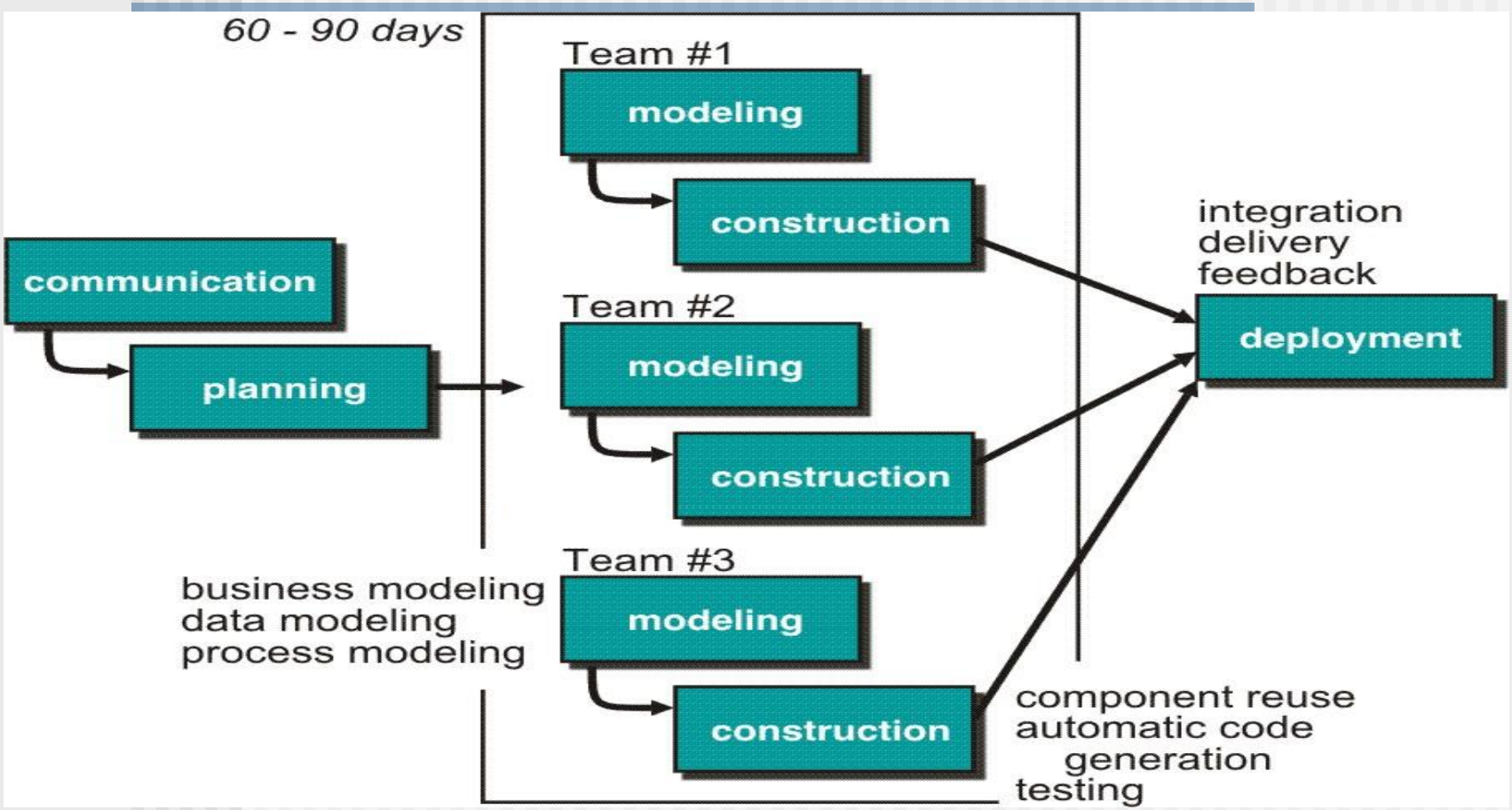
Evolutionary Models: Prototyping



Evolutionary Models: Prototyping

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.
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.

Rapid Application Development (RAD) Model



Makes heavy use of reusable software components with an extremely short development cycle

RAD model

- **Communication** – to understand business problem.
- **Planning** – multiple s/w teams works in parallel on diff. system.
- **Modeling** –
  - **Business modeling**
  - **Data modeling**
  - **Process modeling**
- **Construction –** it highlighting the use of pre-existing software component.
- **Deployment –** Deliver to customer basis for subsequent iteration.
- RAD model emphasize a short development cycle.
- “High speed” edition of linear sequential model.
- If requirement are well understood and project scope is constrained then it enable development team to create “ fully functional system” within a very short time period.

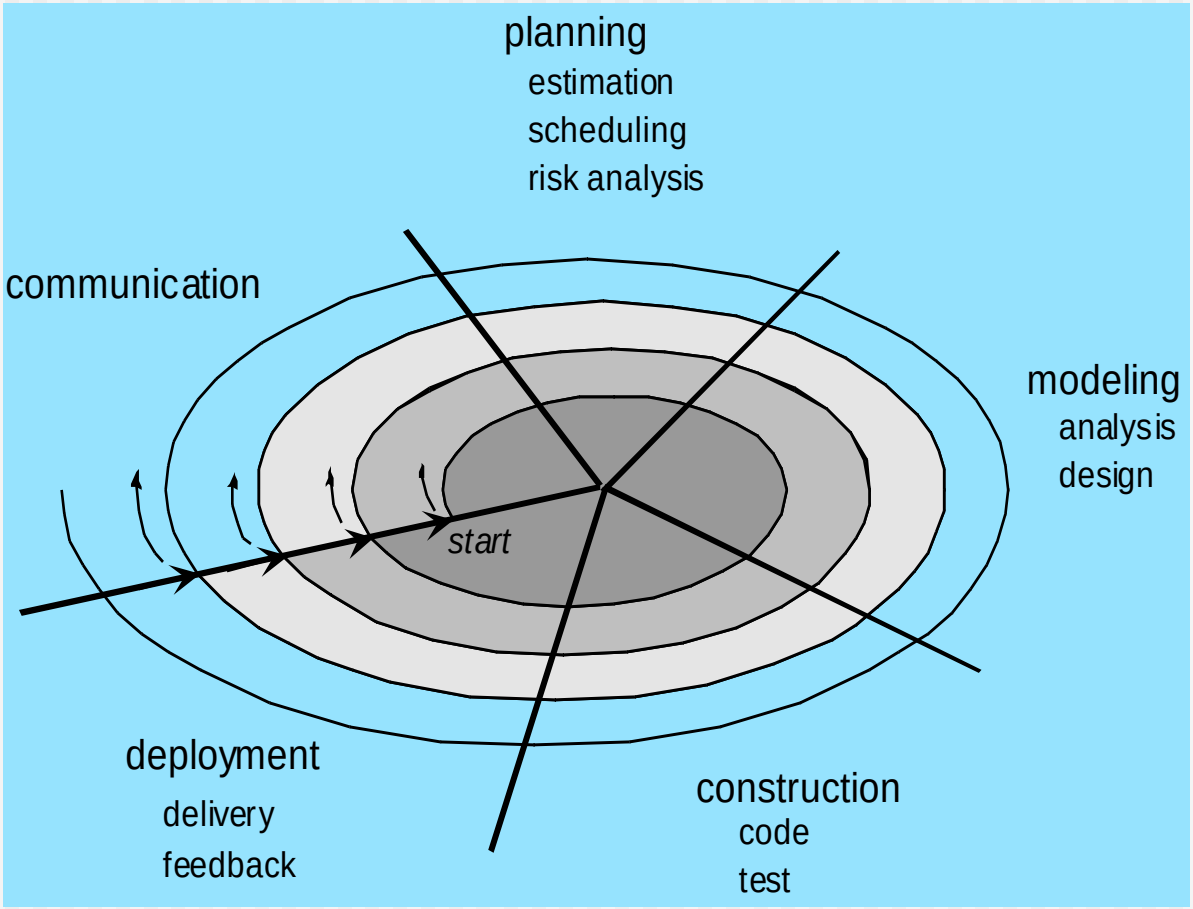
RAD model

- If application is modularized (“Scalable Scope”), each major function to be completed in less than three months.
- Each major function can be addressed by a separate team and then integrated to form a whole.

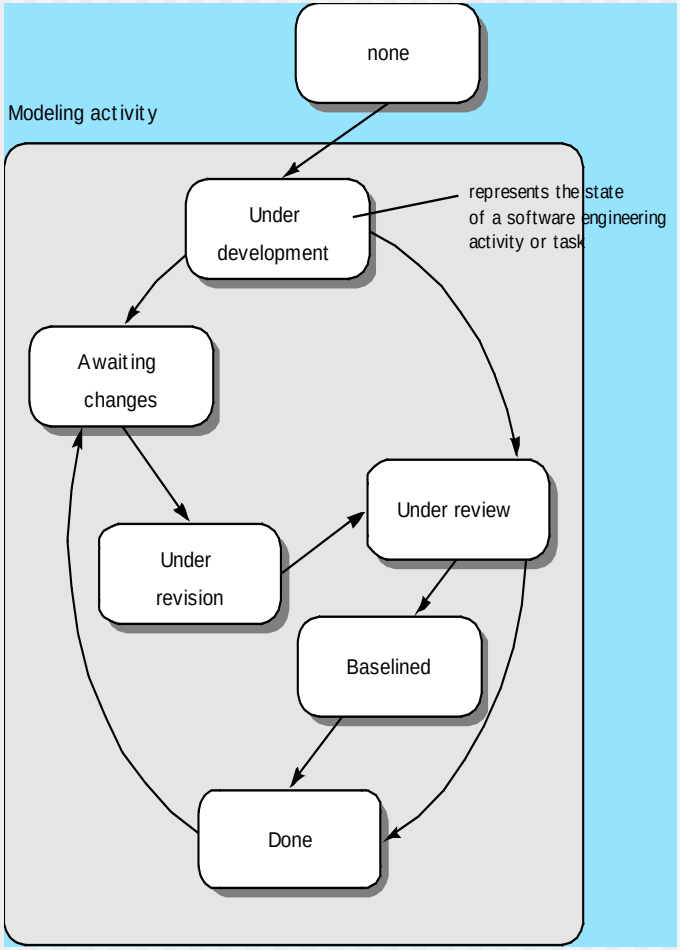
Drawback:

- For large but scalable projects
  - RAD requires sufficient human resources
- Projects fail if developers and customers are not committed in a much shortened time-frame
- Problematic if system can not be modularized
- Not appropriate when technical risks are high (heavy use of new technology)

Evolutionary Models: The Spiral



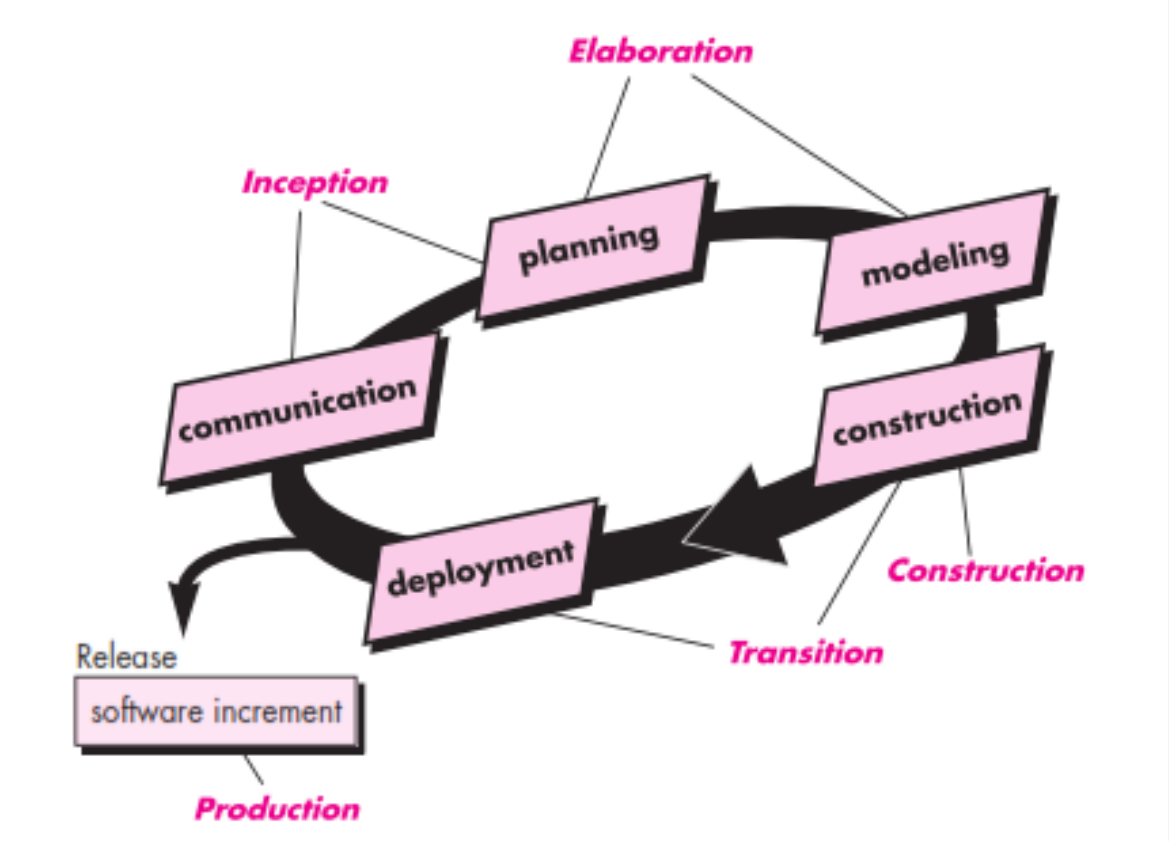
Evolutionary Models: Concurrent



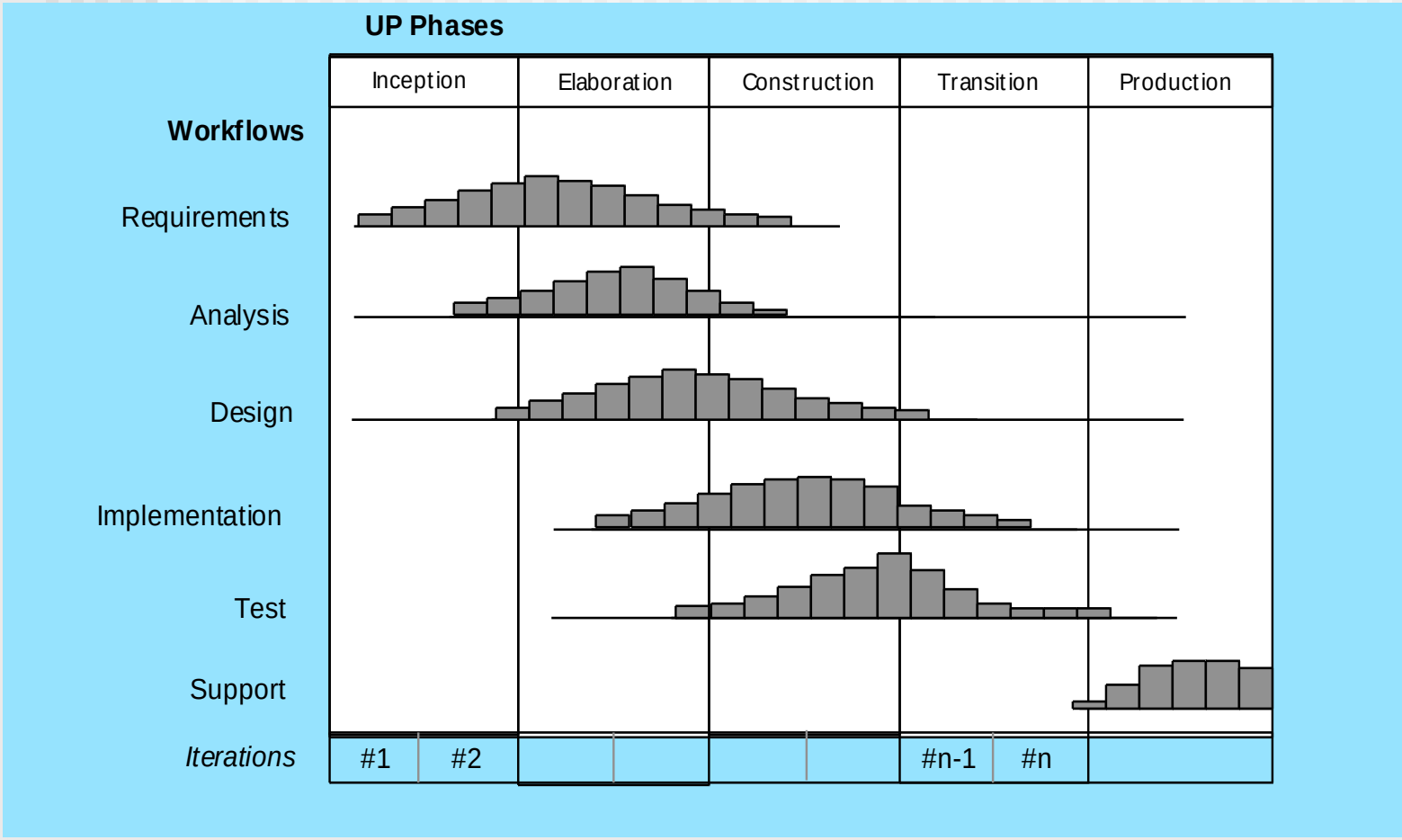
Still Other Process Models

- 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*
- Unified Process—a "use-case driven, architecture-centric, iterative and incremental" software process closely aligned with the Unified Modeling Language (UML)

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