Software Engineering: A Practitioner's Approach, 6/e

Chapter 3
Prescriptive Process Models

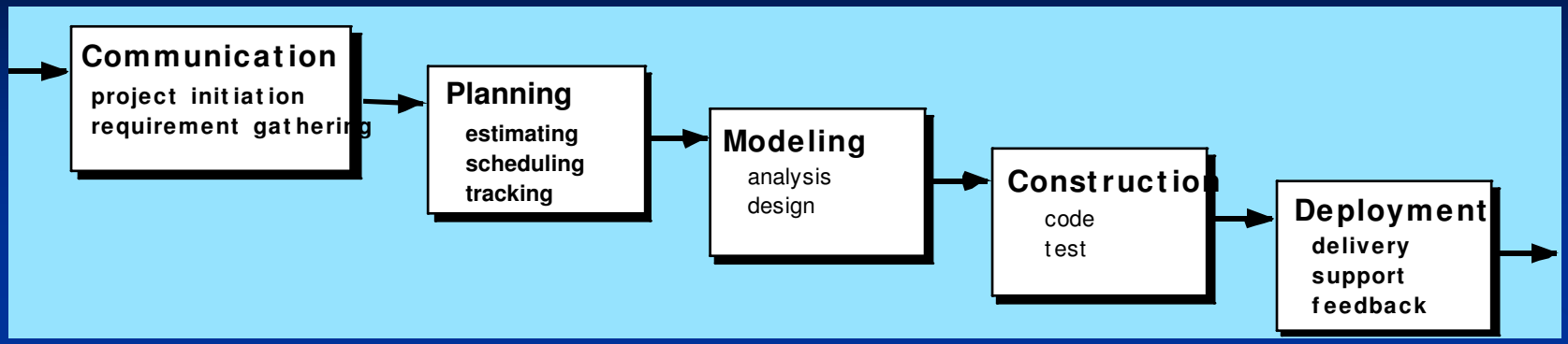
Prescriptive Models

- Prescriptive process models advocate an orderly approach to software engineering

That leads to a few questions …

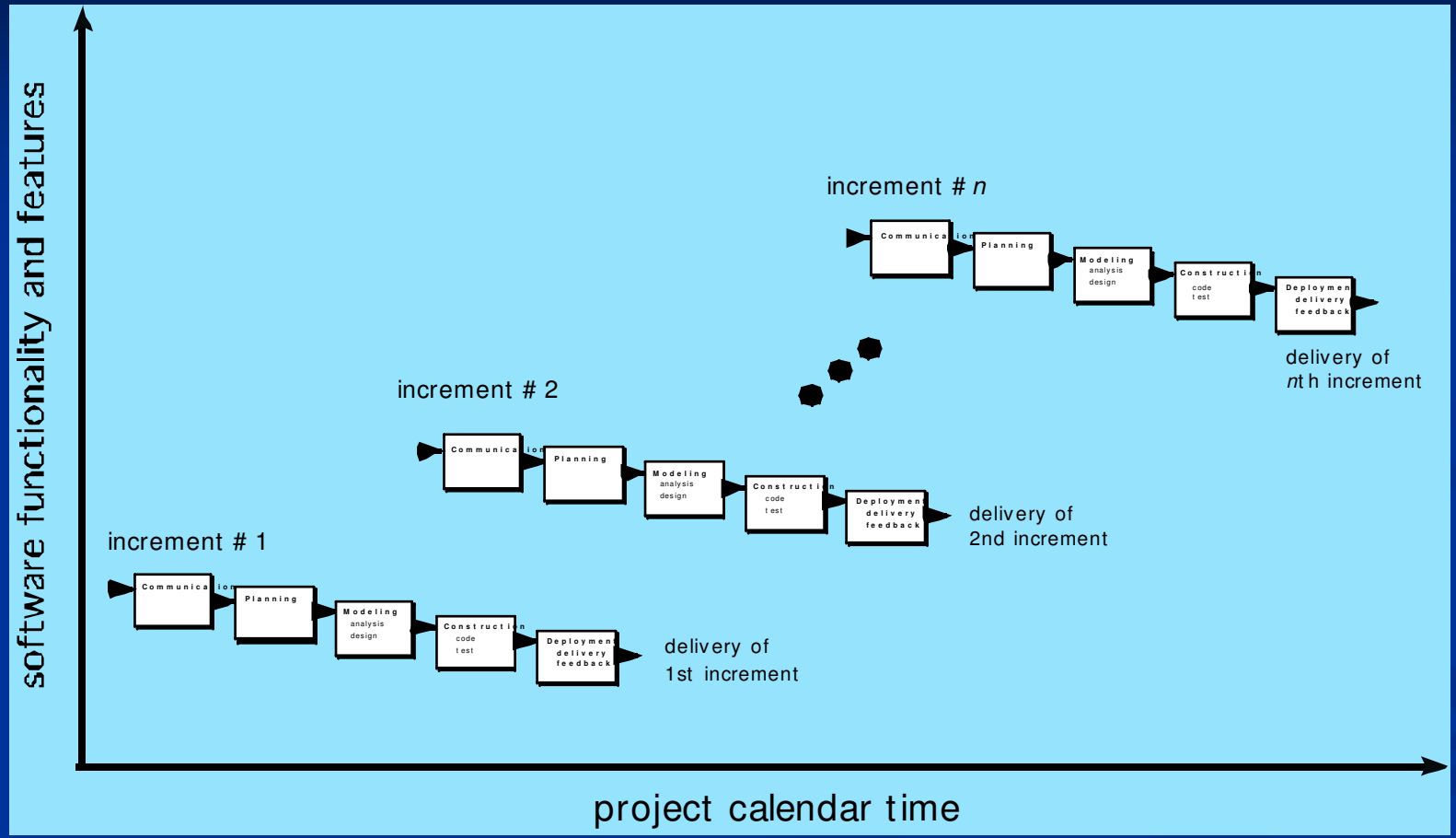
- If prescriptive process models strive for structure and order, are they inappropriate for a software world that thrives on change?
- Yet, if we reject traditional process models (and the order they imply) and replace them with something less structured, do we make it impossible to achieve coordination and coherence in software work?

Waterfall Model



- A.k.a.: Classic Life Cycle
- Useful for problems that are well understood
- Real problems are more complex than that; changes = confusion
- Customer must state all requirements upfront
- Customer must have patience
- Lack of feedback to the customer makes blunders disastrous

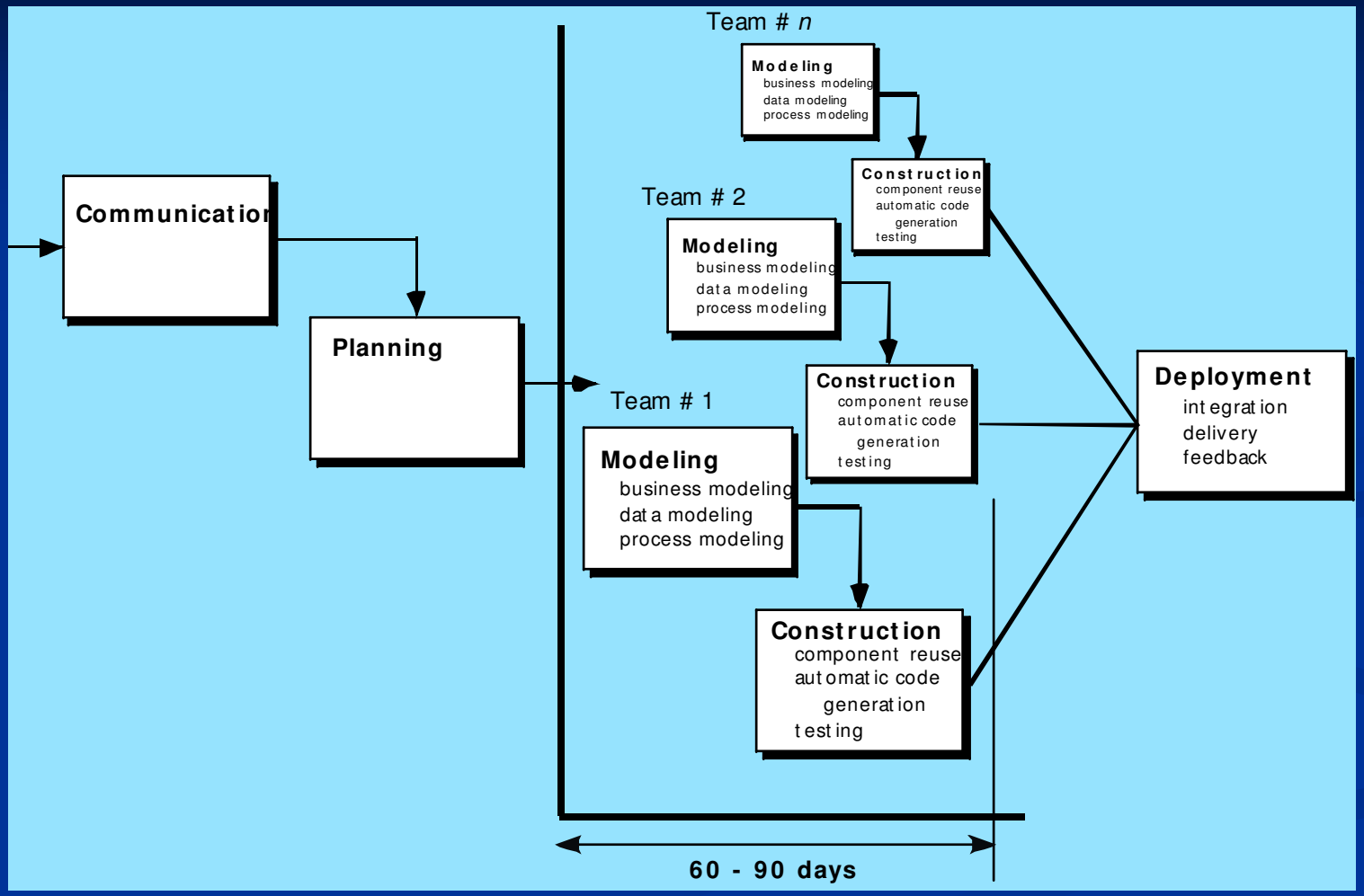
Incremental Model



Incremental Model

- Applies elements of the waterfall model in incremental fashion
- First increment is often a **core product**
- Subsequent elements offer expanded functionality
- Subsequent features (some known, some unknown) are created, while the core product may undergo evaluation
- If a customer requires deadlines that are impossible to meet, a sequence of releases may solve the problem
- Permits for staffing of a team to fluctuate

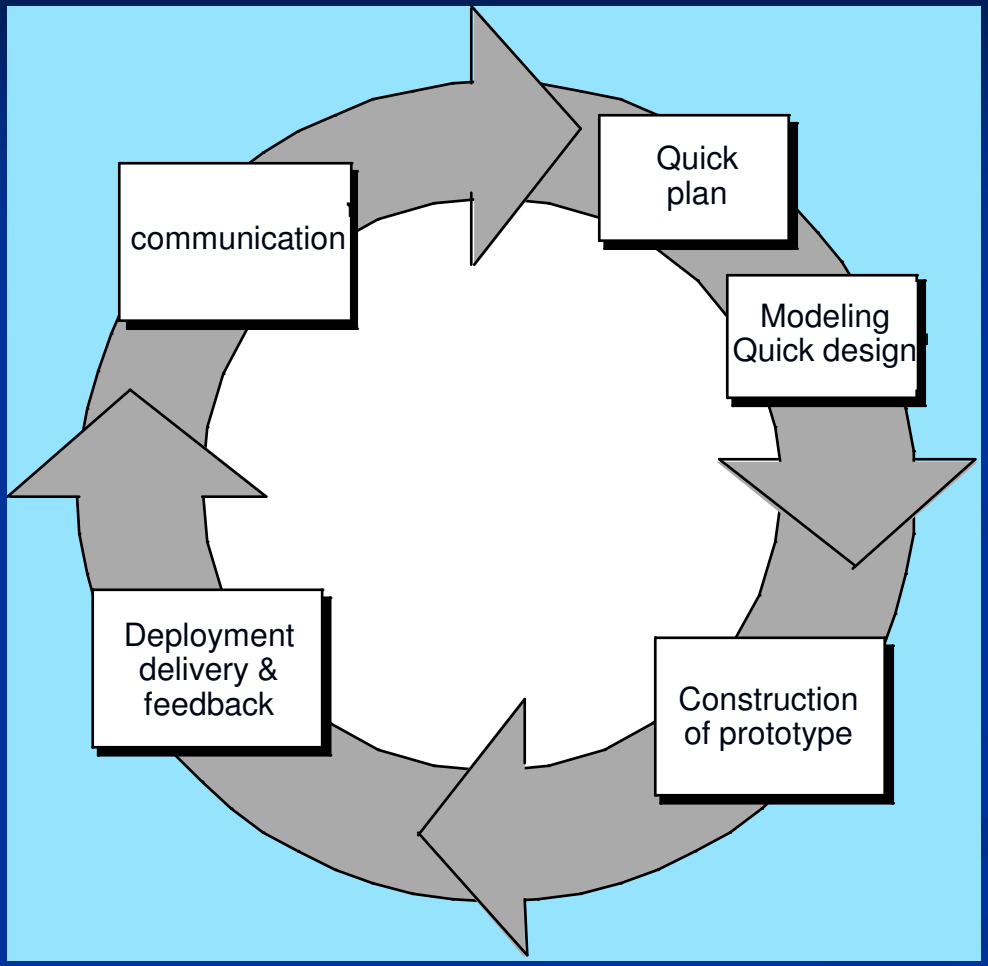
RAD Model



RAD Model

- Emphasizes short development cycle
- If requirements are well understood and project scope is constrained, it may allow rapid creation of S/W systems
- For large projects may require vast human resources (for many RAD teams)
- Developers and customers must be able to cope with fast action
- For systems that are not highly modularized the RAD approach may not work
- If systems require mutual tuning of various components, the RAD approach may not work
- RAD may not be appropriate with technical risks high (new technology)

Evolutionary Models: Prototyping



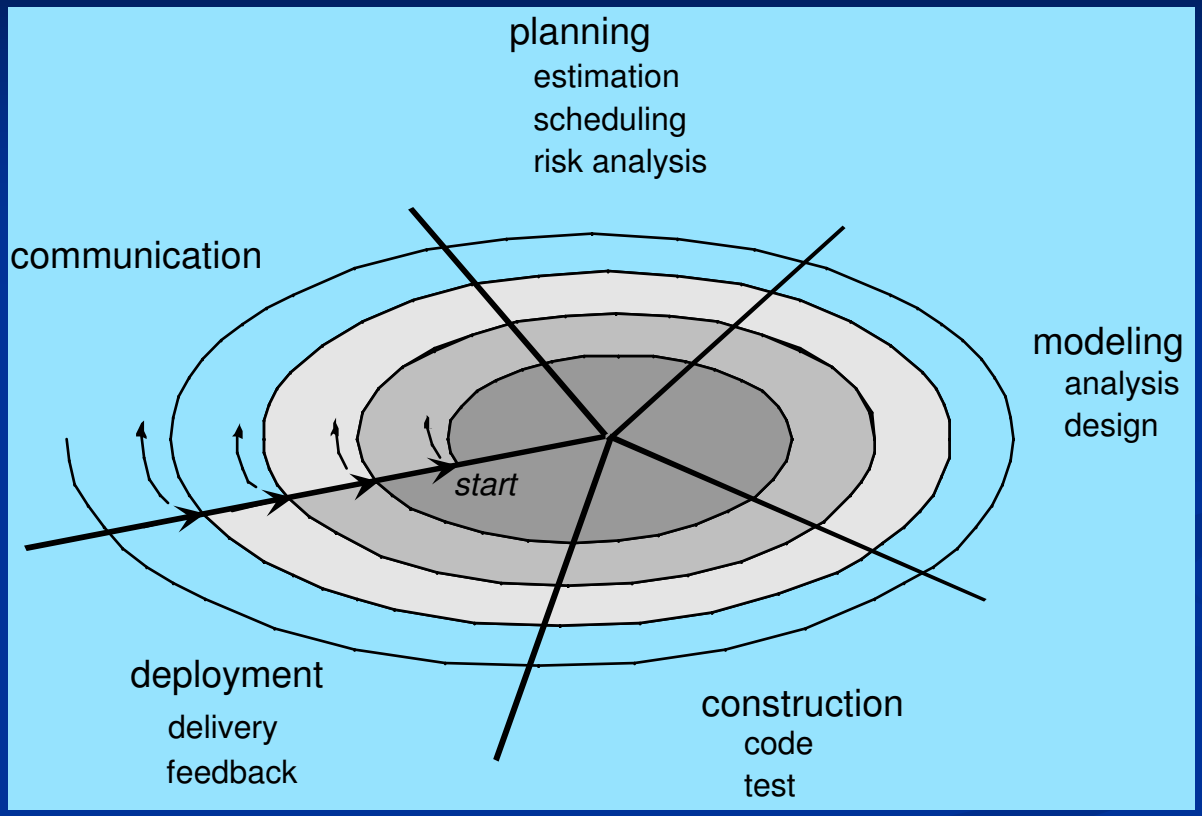
Evolutionary Models: Prototyping

- Useful when the customer has a legitimate need but is clueless about the details
- First prototype may be barely useful: too slow, too big, too unfriendly, etc.

DANGERS:

- The customer may “LOVE” the prototype which is held together with “chewing gum”. When informed that it is a prototype which has to be re-built, he may want “few fixes” only
- The developer may be tempted to deliver the raw prototype as a final product, sacrificing quality for a quick buck

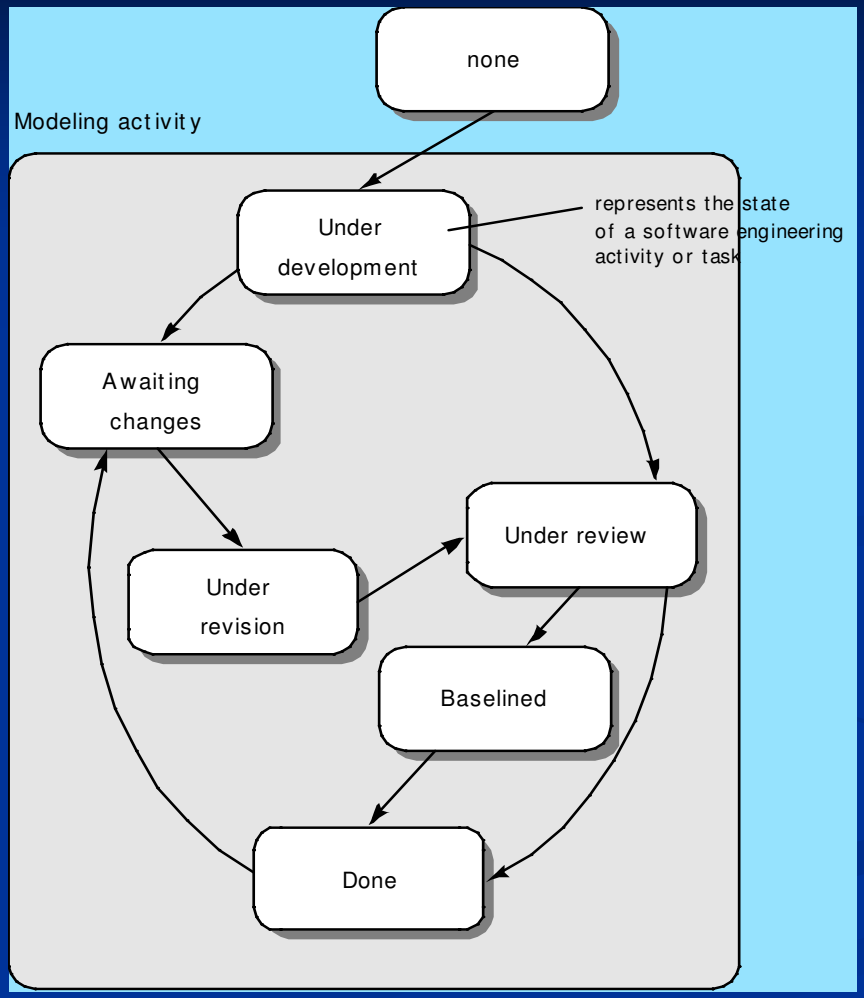
Evolutionary Models: The Spiral



Evolutionary Models: The Spiral

- Combines iterative prototyping with systematic aspects of the waterfall model
- Can be used throughout the S/W life cycle: from conceptualization to maintenance
- Anchor point milestones are used for each evolutionary pass
- Anchor point milestones:
  - Work products
  - Required conditions to be attained
- Realistic to the development of large-scale systems
  - Iteratively reduces risk
  - Allows repeated use of the prototyping approach
- PROBLEM: Customers may not like it, fearing lack of control

Evolutionary Models: Concurrent



Evolutionary Models: Concurrent

- The former image depicted only the modeling activity
- Indeed, concurrent engineering is a set of framework activities such as Communication, Modeling, etc.
- Each of these activities is in one of its states…
- A state is some externally observable mode of behaviour

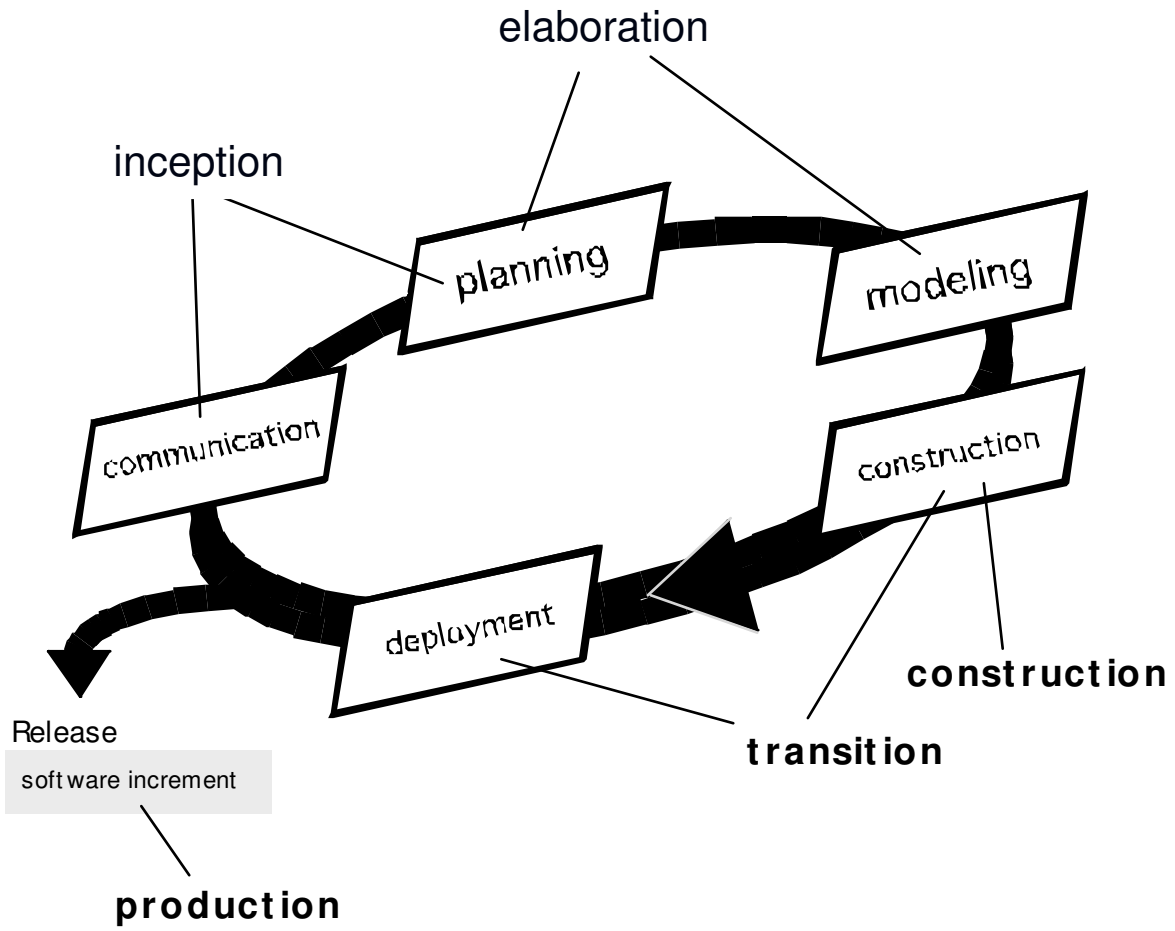
PROBLEMS:

- Prototyping (and other evolutionary processes) pose problems to project planning (too much uncertainty)
- Evolutionary speed may not be optimal: too slow affects productivity, too fast leads to chaos
- Sometimes software processes should focus on flexibility and extensibility, rather than on quality - delivery delays may make us miss the market niche and window of opportunity

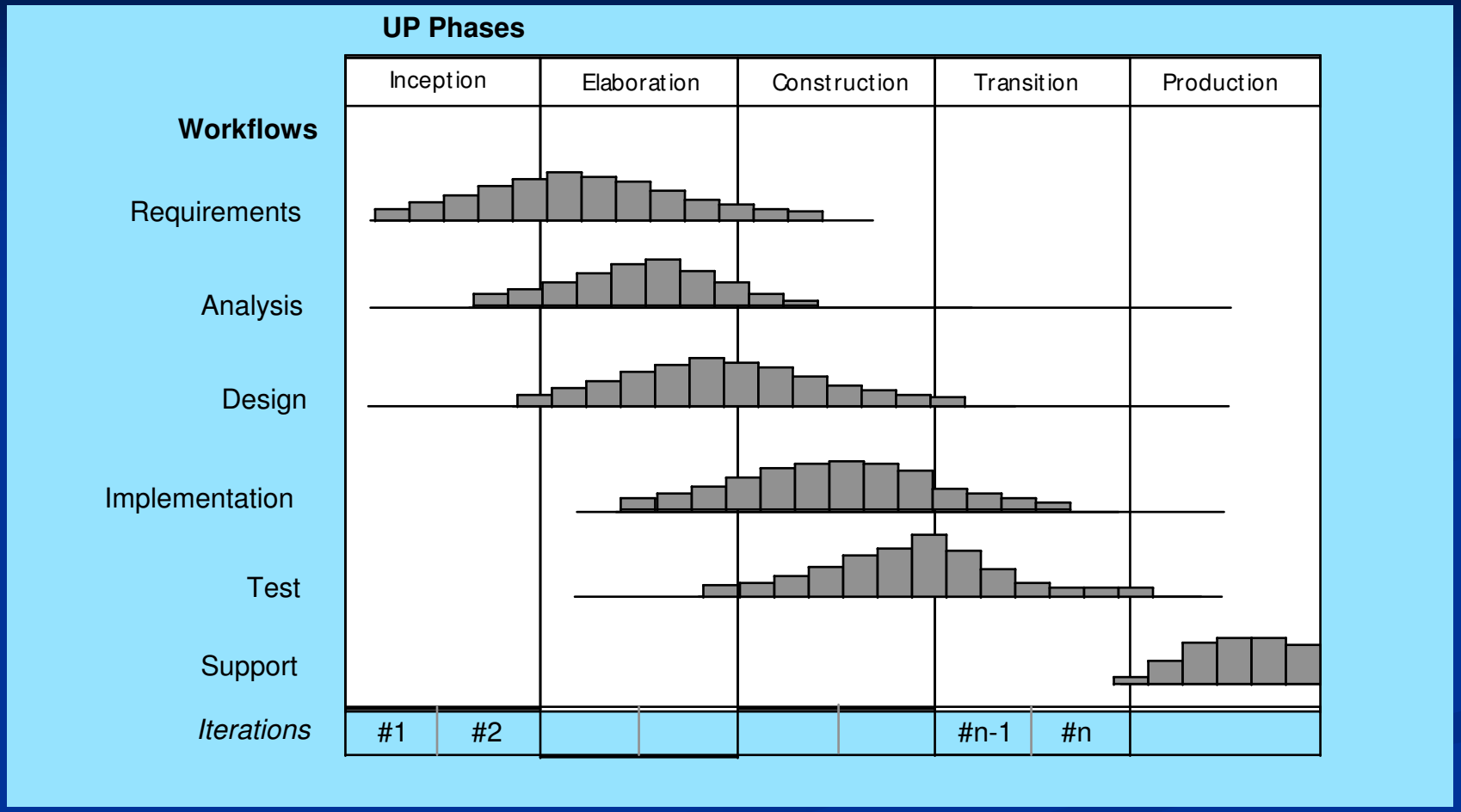
Still Other Process Models

- **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*** like security, fault tolerance, adherence to business rules, systemic: DB mirroring, memory management, concurrency
- **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