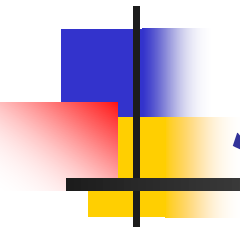
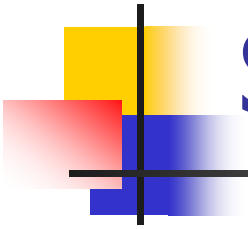


COMP 354: Introduction to Software Engineering

Software Process Models

Based on Chapter 2 of the textbook

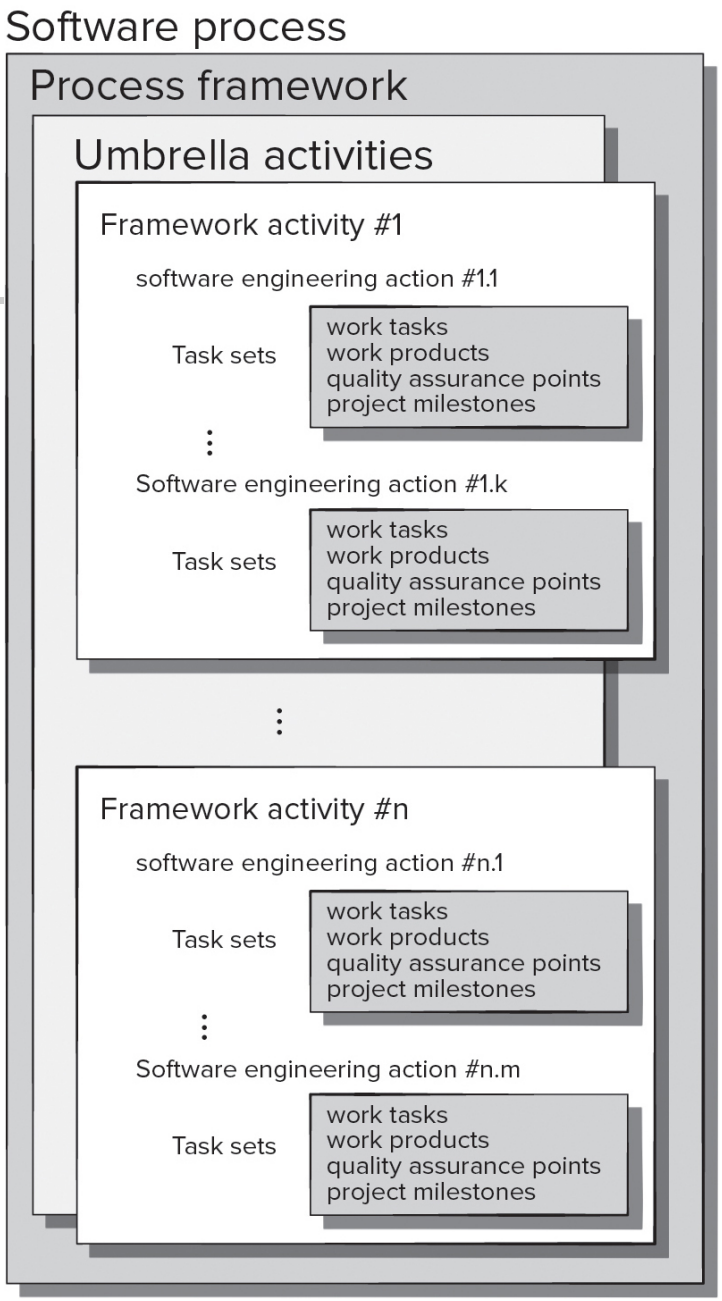
Essence of Software Engineering Practice

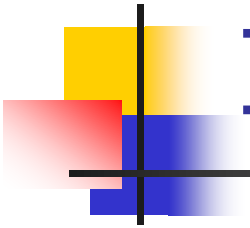


1. Understand the problem (communication and analysis).
2. Plan a solution (modeling and software design).
3. Carry out the plan (code generation/construction).
4. Examine result for accuracy (testing & quality assurance/deployment).

Generic Process Model

- Framework Activities
  - Communication
  - Planning
  - Modeling
  - Construction
  - Deployment
- Umbrella Activities
  - Project Tracking and Control
  - Risk Management
  - Quality Assurance
  - Configuration Management
  - Technical Reviews

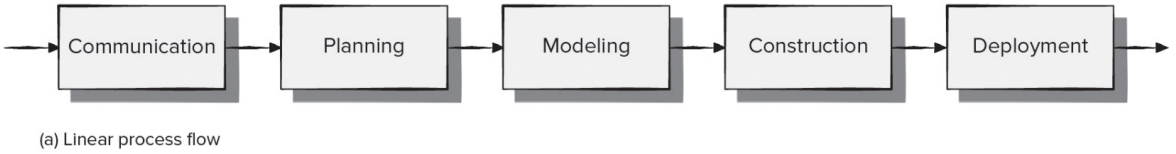




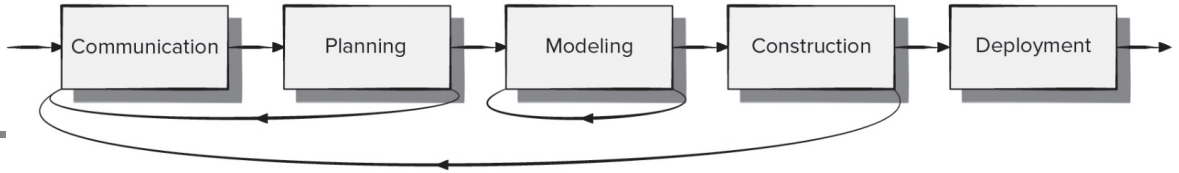
Identifying a Task Set

- A task set defines the actual work to be done to accomplish the objectives of a software engineering action.
- A task set is defined by creating several lists:
  - A list of the tasks to be accomplished.
  - A list of the work products to be produced.
  - A list of the quality assurance filters to be applied.

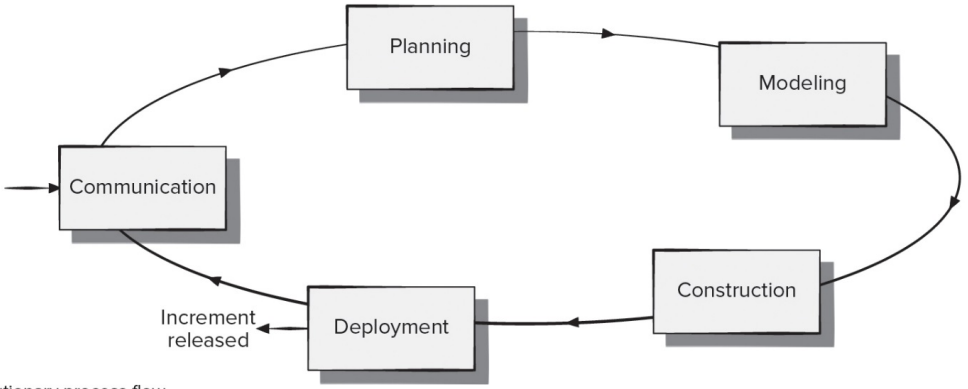
Process Flow



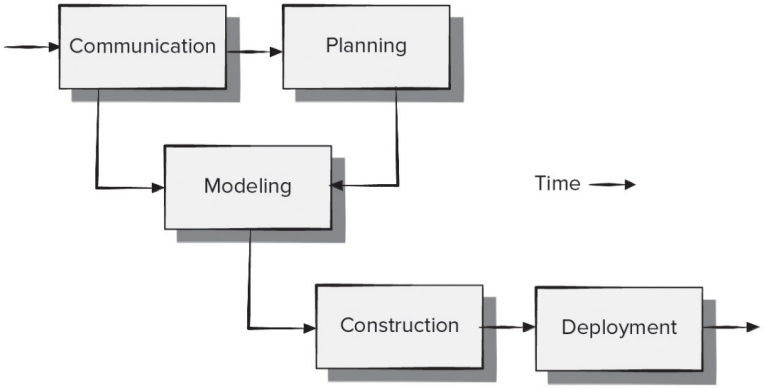
(a) Linear process flow



(b) Iterative process flow

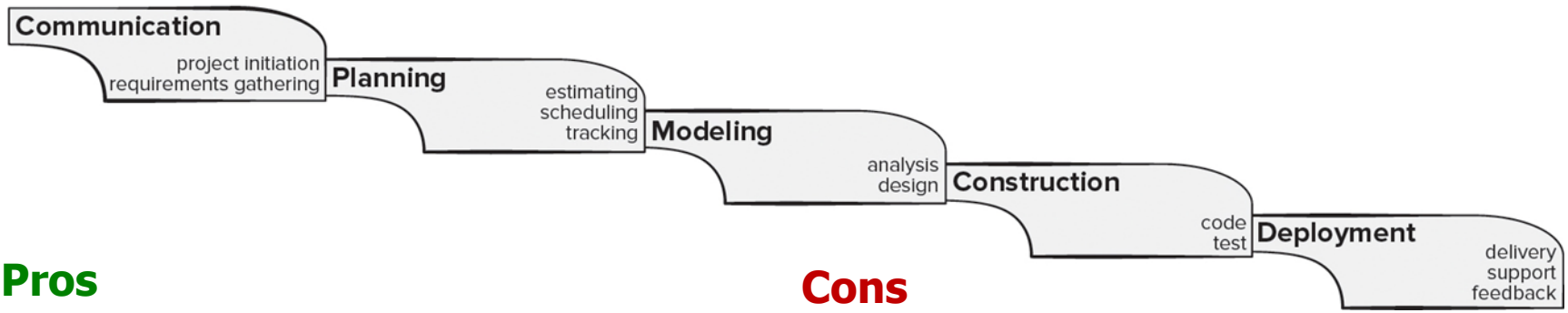


(c) Evolutionary process flow



(d) Parallel process flow

Waterfall Process Model



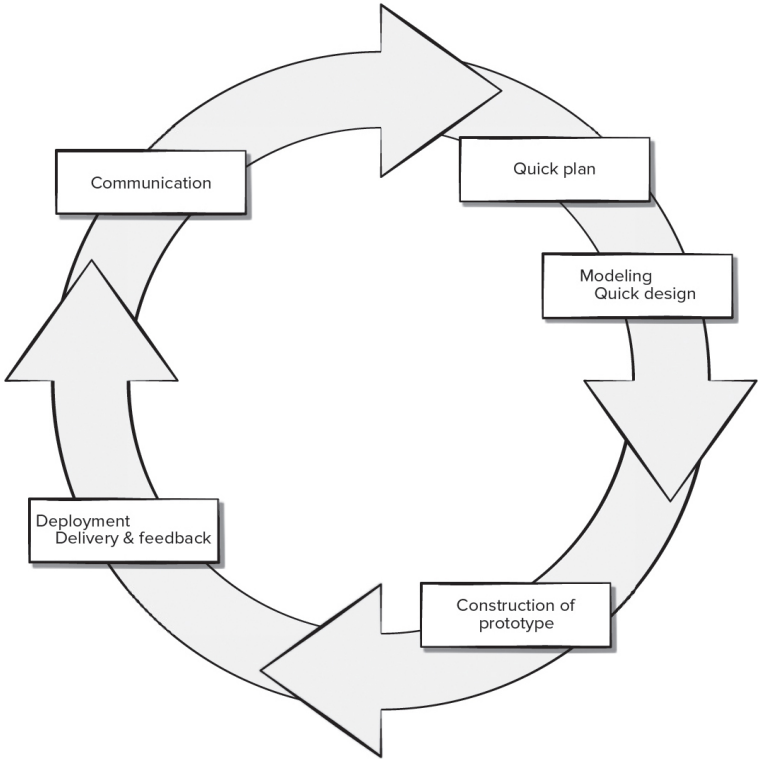
Pros

- It is easy to understand and plan.
- It works for well-understood small projects.
- Analysis and testing are straightforward.

Cons

- It does not accommodate change well.
- Testing occurs late in the process.
- Customer approval is at the end.

Prototyping Process Model



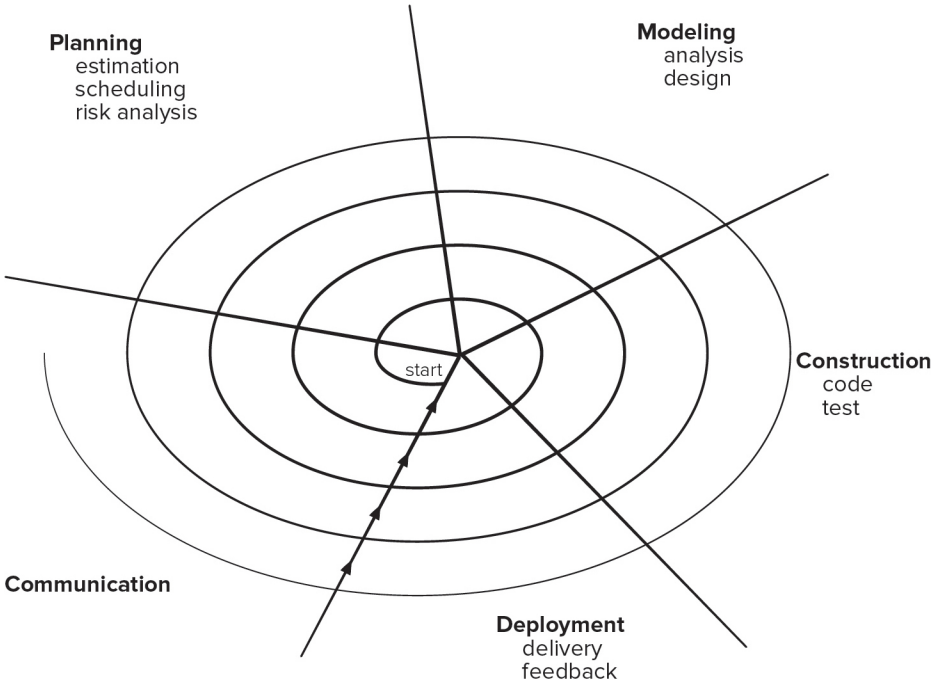
Pros

- Reduced impact of requirement changes.
- Customer is involved early and often.
- Works well for small projects.
- Reduced likelihood of product rejection.

Cons

- Customer involvement may cause delays.
- Temptation to "ship" a prototype.
- Work lost in a throwaway prototype.
- Hard to plan and manage.

Spiral Process Model



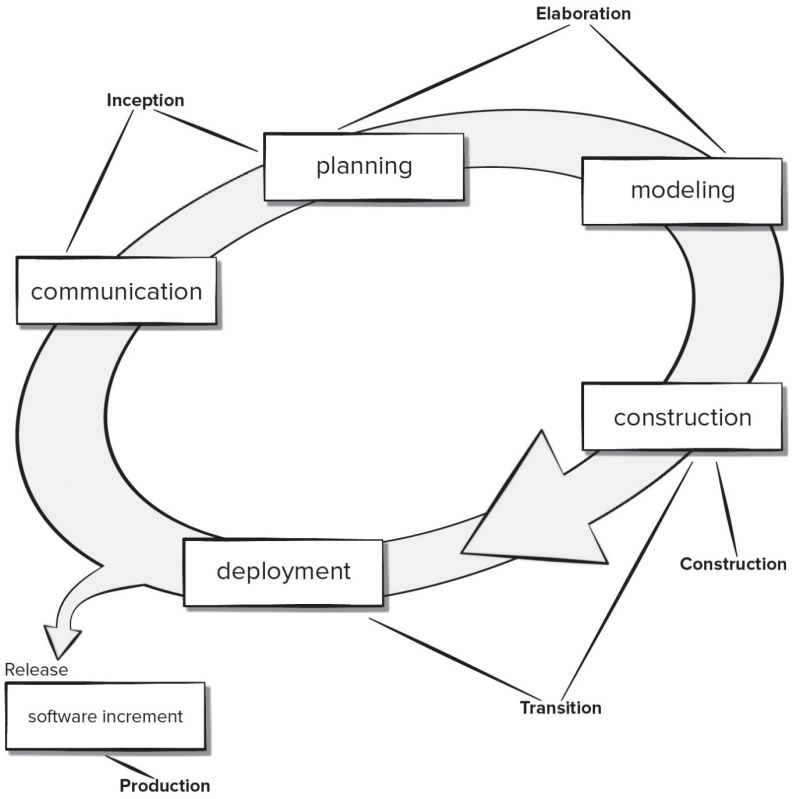
Pros

- Continuous customer involvement.
- Development risks are managed.
- Suitable for large, complex projects.
- It works well for extensible products.

Cons

- Risk analysis failures can doom the project.
- Project may be hard to manage.
- Requires an expert development team.

Unified Process Model



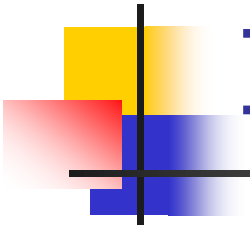
Pros

- Quality documentation emphasized.
- Continuous customer involvement.
- Accommodates requirements changes.
- Works well for maintenance projects.

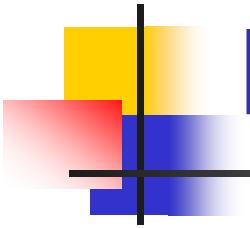
Cons

- Use cases are not always precise.
- Tricky software increment integration.
- Overlapping phases can cause problems.
- Requires expert development team.

Process Assessment and Improvement



- The existence of a software process is no guarantee that software will be delivered on time, or meet the customer's needs, or that it will exhibit long-term quality characteristics.
- Any software process can be assessed to ensure that it meets a set of basic process criteria that have been shown to be essential for successful software engineering.
- Software processes and activities should be assessed using numeric measures or software analytics (metrics).



Prescriptive Process Models

- Prescriptive process models advocate an orderly approach to software engineering.
- *That leads to two questions:*
  - If prescriptive process models strive for structure and order, are they appropriate for a software world that thrives on change?
  - If we reject traditional process models and replace them with something less structured, do we make it impossible to achieve coordination and coherence in software work?