Software Process
(*process models*)

Categories of Process Models

- Prescriptive model
  - Sequential model
  - Incremental model
  - Evolutionary model
- Specialized process model
  - Component-based model
  - Formal method model
  - Aspect-Oriented process model
- Unified process model
  - Software development with UML

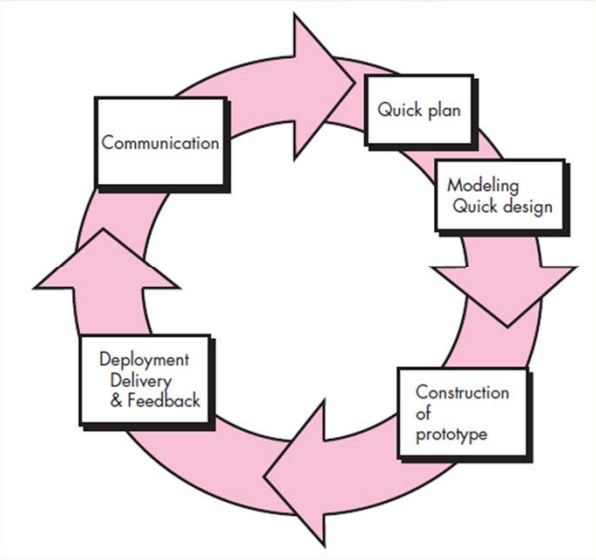
Prescriptive Process Models (2)

Evolutionary Models

- Evolutionary processes produce an increasingly more complete version of the software with each iteration.
  - Like incremental process model, they are iterative.
  - Unlike incremental process model, each iteration does not necessarily produce a full-functional product.
- Examples:
  - Prototyping model
  - Spiral model

Subtle differences

Prototyping



Rapidly revise requirements and designs before going for operational product

Design pattern

Key in the scenario: Define the requirements details.

Starts with communication. Quick design: Focus on aspects that are visible to end users.

Prototype delivered, evaluated by stakeholders, who provide feedback and used for further refinement on the requirements.

Prototyping

- A software process to develop consensus on requirements through iterations of prototypes.
  - Used for situations where customers don't have a clear specification for software.
  - Each iteration will produce a prototype that best presents developers' understanding on what customers want.
  - Customers provide feedbacks based on prototypes.
  - Final products are usually built using technologies different from those used in prototypes for better quality.
    - In throwaway prototyping, prototypes are used only as references for requirements and designs.

Prototyping

- Strength
  - Prototyping has an integrated and iterative "communication" stage for feedback.
    - Each prototype demonstrates engineers/analysts' understanding.
    - Customers will communicate their feedbacks on prototypes.
  - Better risk management.
    - Avoid costly revision after product release due to misunderstanding of requirement.
  - Prototyping may be used as "proof-of-concept", to measure the performance of algorithms and data structures.

research

Prototyping

- Disadvantages
  - may provide the false image of “quick fix” to engineers and customers, causing unrealistic expectations from both sides.
    - Prototype is built to demonstrate the feasibility, not quality.
    - Finish product is often rebuilt using different techniques to ensure quality and maintainability.
      - E.g. use Visual Basic instead of C++ when prototyping GUI application, which compromise efficiency and portability.

Stakeholders and engineers like prototyping, but provides false sense of quality.

Customers expect to see the product fast.

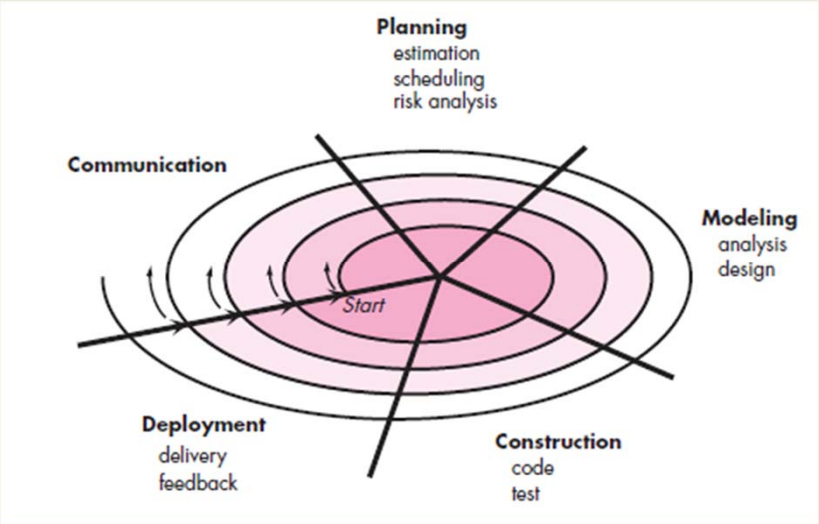
Engineers might ignore the problems that are there at the beginning: inefficient algorithms, inappropriate programming languages.

Make it clear at the beginning.

Prototyping

- Use scenarios
  - Standalone model
  - Assistive technique
- Benefits
  - Requirements identification
    - process (phase) pattern example
  - Reuse
- Problems
  - Quality

Spiral Model



- Each cycle will produce artifacts which will be one step closer to the finish product than previous step.
- Anchor points will serve as milestones to check the progress of software development.

Risk driven!

Multi-stakeholder concurrent engineering. Two features: cyclic approach; a set of anchor point milestones.

Divide into a set of framework activities.

Spiral Model

- Strength
  - Delivers initial value early
    - Focus on high-priority functionality
  - Better risk management
    - Improvement and bug fixing can be handled during iterations.
  - Frequent requirements refinement
    - Uses feedback from one iteration to refine requirements for the next.
    - The deliverable at the end of each circle is NOT necessarily a functional product.

spiral flow can be applied across the entire lifecycle, from product inception to maintenance.

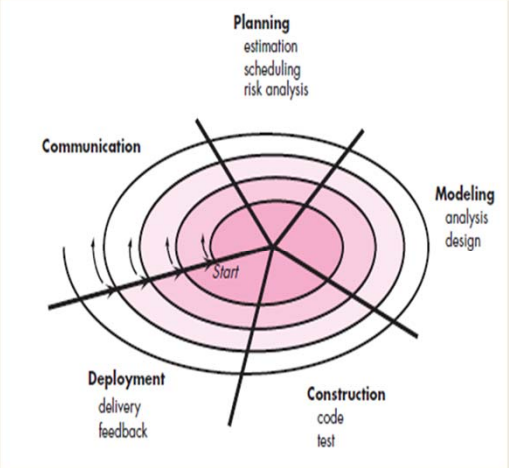
“concept development project” -> “new product development project” -> “product enhancement project”

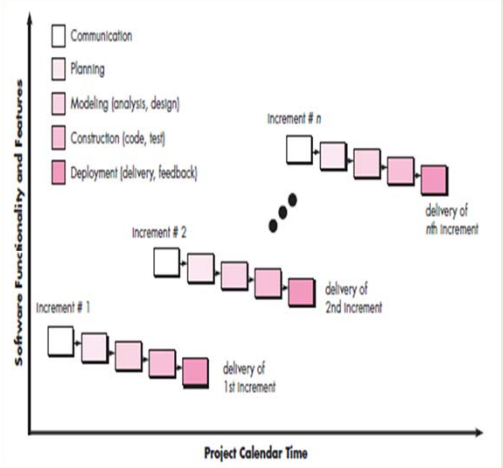
Suitable for large-scale systems and software. Uses prototyping as a risk reduction mechanism, but also allows the engineers to apply prototyping technique at any state during the evolution.

Spiral Model

- Weakness
  - Hard to control the timeline of software development.
    - Not suitable for projects demanding rigid timeline control and strict deadline.
  - Relying on risk-assessment expertise.
    - Questions like “should we move to next iteration of development?”

Comparison with Incremental Model





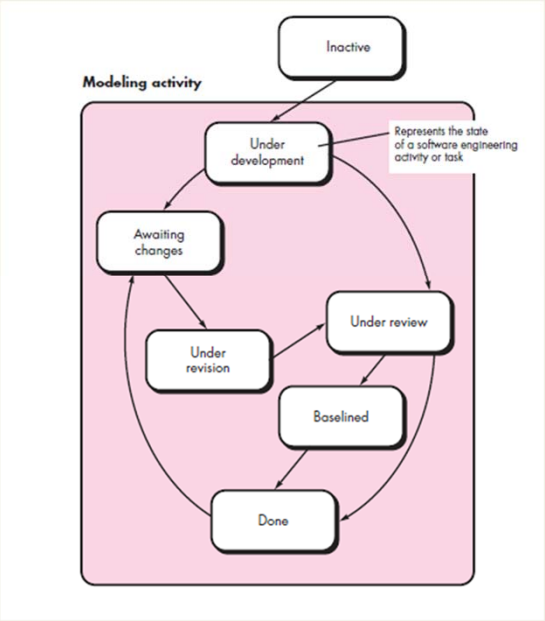
- Nature: iterative versus incremental
- Deliverable: early versus partial
- Focus: risk reduction/quality control versus delivery of operational product

Extreme case: requirements and design are only done once at the beginning. The key here is increment.

But for evolutional models, the key is evolution. For example, the coupling between successive versions in evolutional models are much looser.

Concurrent Development Model

- Development activities are modeled as finite state machines (finite automata).
  - States indicate the progress of an activity.
  - E.g. modeling activity on the right.
- The entire software process is modeled by many finite automata that operate concurrently.
  - The overall state of software process is a vector of states of each individual finite state model.
  - E.g. the current state of software process is given by the progress on design, implementation, verification and validation etc.



Defines a series of events that will trigger the transitions from state to state.

Features	Incremental	Spiral	RAD	Waterfall
Requirement	Beginning	Beginning	Time boxed release	Beginning
Planning	Yes	Yes	Not require	Yes
Documentation	Yes	Yes	Not necessary	Yes
Handle large project	Not necessary	Yes	Not necessary	Not necessary
User involvement	Intermediate	High	At beginning only	At beginning only
Returning to early phase	Yes	Yes	Yes	No
Cost	Low	High	Low	Low
Modifications	Easy	Easy	Easy	Difficult
Duration	Long	Long	Short	Long
Testing	After each iteration	After each iteration	After coding	After coding
Risk	Low	Medium to high	Low	High
Maintenance	Easy	Hard	Easy	Easy
Reuse	Up to some extent	Up to some extent	Yes	Up to some extent
Framework	Iterative and Linear	Iterative and Linear	Linear	Linear

http://www.learnerswindow.com/compare-incremental-vs-spiral-vs-rad-vs-waterfall-model/
http://www.guru99.com/compare-waterfall-vs-incremental-vs-spiral-vs-rad.html