CS 320

Introduction to Software Engineering
Spring 2017

February 01, 2017

Recap: software development process

Activities and steps

- Requirements engineering
- Design and architecture
- Implementation
- **Verification and Validation**
- Deployment and Maintenance

Recap: verification vs. validation

Verification (did we build it right)

- Does the software meet its specification
- **Static analysis**
  - Reason about the program without executing it
- **Dynamic analysis**
  - Execute the program and observe its behavior

Validation (did we build the right thing?)

- Does the specification reflect the client's needs?

Today

Major software development process models

- Traditional models
- *Agile models*

Major software development process models

Traditional models

- Waterfall model
- Iterative and incremental
- Prototyping
- Spiral model

Agile models

- *XP (Extreme Programming)*
- *Scrum*

There are many more models.

What do all models have in common?

Major software development process models

Traditional models

- Waterfall model
- Iterative and incremental
- Prototyping
- Spiral model

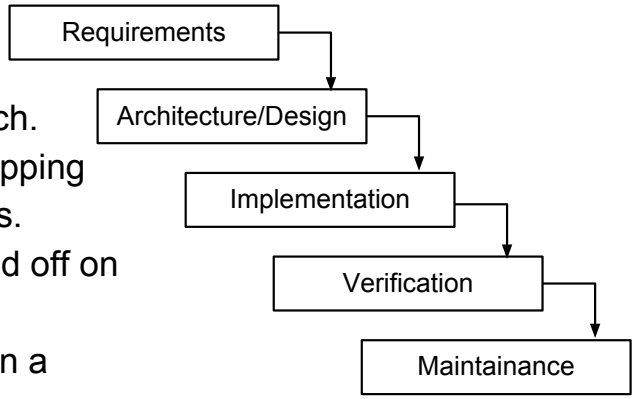
Agile models

- *XP (Extreme Programming)*
- *Scrum*

Same goals: manage risks and produce high quality software.
Same activities and steps (e.g., specification, design, implementation, and testing).

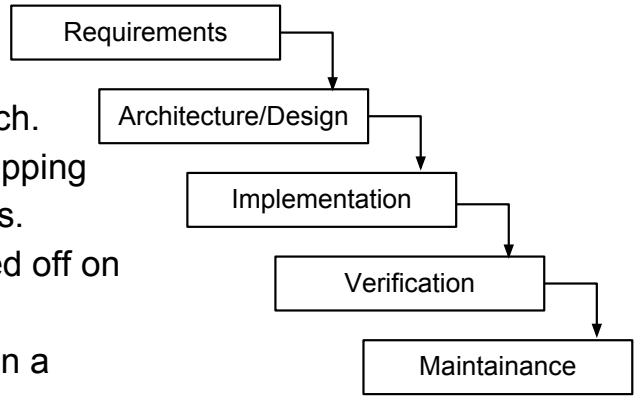
Waterfall model

- Top-down approach.
- Linear, non-overlapping activities and steps.
- Each step is signed off on and then frozen.
- Most steps result in a final document.



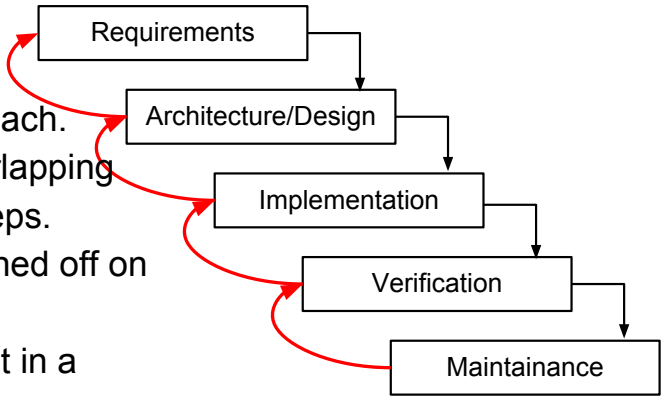
Waterfall model

- Top-down approach.
- Linear, non-overlapping activities and steps.
- Each step is signed off on and then frozen.
- Most steps result in a final document.



What's missing in this simple model?

Waterfall model



- Top-down approach.
- Linear, non-overlapping activities and steps.
- Each step is signed off on and then frozen.
- Most steps result in a final document.
- Backsteps are necessary to allow modifications/re-work.

Waterfall model

Advantages

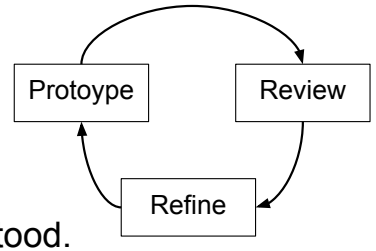
- Easy-to-follow, sequential model.
- Reviews ensure readiness to advance.
- Works well for well-defined projects (i.e., requirements are well understood).

Drawbacks

- Hard to do all the planning upfront.
- Final product may not match the client's needs.
- Step reviews require significant effort.

Prototyping

- Bottom-up approach.
- Problem domain or requirements not well defined or understood.
- Create small implementations of requirements that are least understood.



- Reduces risk as requirements are "explored" before the product is fully developed.
- Developers gain experience when developing the "real" product.

Prototyping: examples

"Throwaway" prototype

- Used to explore/understand an aspect of the system (e.g., GUI design flow, module to test client/server communication).

"Bare-bones" implementation

- Preliminary system that will be built upon to eventually become the final product.

Prototyping

Advantages

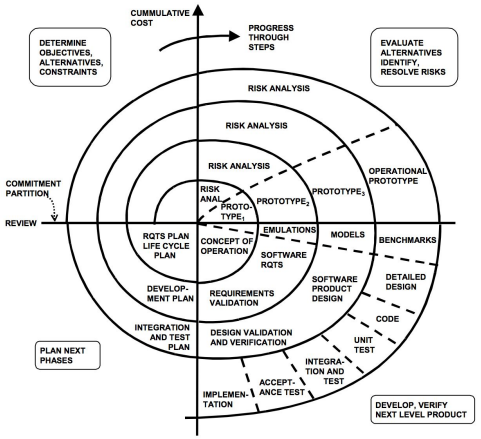
- Client involvement and early feedback.
- Improves requirements and specifications.
- Reduces risk of developing the “wrong” product.

Drawbacks

- Time/cost for developing a prototype may be high.
- Focus may be too narrow (no thinking outside the box).

Spiral model

- Incremental/iterative model (combines the waterfall model and prototyping).
- Iterations called spirals.
- Activity centered:
  - Planning
  - Risk analysis
  - Engineering
  - Evaluation
- Phased reduction of risks (address high risks early).



Boehm, *Spiral Development: Experience, Principles,and Refinements*, CMU/SEI-2000-SR-008

Spiral model

Advantages

- Early indication of unforeseen problems.
- Allows for changes.
- The risk reduces as costs increase.

Drawbacks

- Requires proper risk assessment.
- Requires a lot of planning and experienced management.

Agile models

Agile Manifesto (http://agilemanifesto.org/):

- ***Individuals and interactions*** over processes and tools
- ***Working software*** over comprehensive documentation
- ***Customer collaboration*** over contract negotiation
- ***Responding to change*** over following a plan.

© Scott Adams, Inc./Dist. by UFS, Inc.

Agile models: example

Extreme Programming (XP)

- New versions may be built several times per day with products delivered to customers weekly.
- Adaptation and re-prioritization of requirements.
- All tests must be run for every build and the build is only accepted if tests run successfully (may rely on test-driven development).

What important aspect of
"extreme" programming is missing?

Agile models: example

Extreme Programming (XP)

- Pair programming and continuous code review.
- Pairs and roles are frequently changed.
- Improves communication, and feedback.

Agile models

Basics

- Maintain simplicity.
- Team members choose their own methods, tools etc.
- Continuous customer involvement.
- Expect system requirements to change, focus on incremental delivery.

Agile models

Advantages

- Flexibility (changes are expected).
- Focus on quality (continuous testing).
- Focus on communication.

Drawbacks

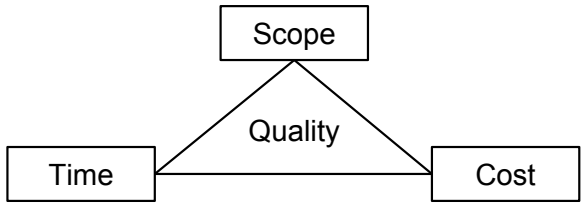
- Requires experienced management and highly skilled developers.
- Prioritizing requirements can be difficult when there are multiple stakeholders.
- Best for small to medium (sub) projects.

What's the best model?

Consider

- The project and task at hand.
- Risk management and quality/cost control.
- Customer involvement and feedback.
- Well-definedness of requirements.
- Experience of management and team members.

Project management triangle (pick any two)



Summary

Software development process models

- **All models have the same goals**: manage risks and produce high quality software.
- **All models involve the same activities and steps** (e.g., specification, design, implementation, and testing).
- **Traditional models:** E.g., Waterfall, Prototyping, Spiral.
- **Agile models:** E.g, Extreme Programming (XP).