CSE 403
Software Engineering
Spring 2020

Software development life cycle

April 06, 2020
Logistics

- Two sections this week (project proposal pairs).
- Team up on Canvas by tonight (optional); random assignment by default.
- Zoom: pre-assigned breakout rooms based on Canvas groups.
Why use COBOL on a mainframe (40 years later)?

- Reliable infrastructure and a well-tested system.
- Very few attacks/hacks (who can even write COBOL?)

Why change?

- Technical debt and lack of skilled engineers.
- Almost impossible to improve and debug (at this point).
Today

- Ad-hoc software development
- Software development life cycle
  - Traditional models
  - Agile models
- What’s the best model (for your course project)?
Software development: the high-level problem

Specification → ??? → Source code
Software development: code and fix

One solution: “Here happens a miracle”
Software development: code and fix

One solution: “Here happens a miracle”
Software development: ad-hoc or systematic?

Pros: Ad-hoc
● No formal process.
● Easy, quick, and flexible.

Cons: Ad-hoc
● Might lack important tasks such as design or testing.
● Doesn’t scale to multiple developers.
● How to measure effort and progress?
Software Development Life Cycle (SDLC)
The software development life cycle (SDLC)

SDLC: produce software through a series of stages
- From conception to end-of-life.
- Can take months or years to complete.

Goals of each stage
- Define a clear set of steps to perform.
- Produce a tangible item.
- Allow for review of work.
- Specify actions to perform in the next stage.
Life-cycle stages

Virtually all SDLC models have the following stages
• Requirements
• Design
• Implementation
• Testing
• Maintenance

Key question:
How to combine the stages and in what order?
Major SDLC models

Traditional models
● Waterfall model
● Prototyping
● Spiral model
● ...

Agile models
● XP (Extreme Programming)
● Scrum
● ...

All models have the same goals:
Manage risks and produce high quality software.
Traditional SDLC models
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 to correct mistakes.
Waterfall model

Advantages
● Easy-to-follow, sequential model.
● Reviews ensure readiness to advance.
● Works well for well-defined projects (requirements are clear).

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.

Prototype ➔ Review ➔ Refine ➔ Prototype
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).

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 SDLC models
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.
Agile models: XP

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).
Agile models: XP

Extreme Programming (XP)
- Pair programming and continuous code review.
- Pairs and roles are frequently changed.
Agile models: XP

Extreme Programming (XP)

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

(Scene from Bug Hunting:

I wonder where that bug can be. 
Aha! Here it is.

Whoever coded that should be shot!

I coded that. I'm sorry I didn't mean it.)
Agile models

Basics

- Maintain simplicity.
- Team members choose their own methods, tools etc.
- Continuous customer involvement.
- Expect 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 SDLC model?
What model would you choose and why?

- A system to control anti-lock braking in a car.
- A hospital accounting system that replaces an existing one.
- An interactive system that allows airline passengers to quickly find replacement flight times (for missed or bumped reservations) from airport terminals or an app.
What’s the best SDLC model?

Project management triangle (pick any two)

Consider
- The project and task at hand.
- Well-definedness of requirements.
- Risk management and quality/cost control.
- Customer involvement and feedback.
- Experience of management and team members.
Summary: SDLC 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).
- **All models have advantages and drawbacks**.
- **Traditional models**: E.g., Waterfall, Prototyping, Spiral.
- **Agile models**: E.g., Extreme Programming (XP), Scrum.