CSE 403

Software Engineering

Winter 2023

Software development life cycle

Recap: The Joel Test

1. Do you use **source control**?
2. Can you make a **build [+ release] in one step**?
3. ~~Do you make daily builds?~~
   Do you **use CI (clean main branch)**?
4. Do you have a **bug database**?
5. * Do you **fix bugs before** writing **new code**?
6. Do you have an **up-to-date schedule**?
7. Do you **have a spec**?
8. ** Do programmers have **quiet working conditions**?
9. * Do you **use the best tools** money can buy?
10. ~~Do you have testers?~~
    Do you do **automated testing** AND do you **have testers**?
11. * Do new candidates **write code during** their **interview**?
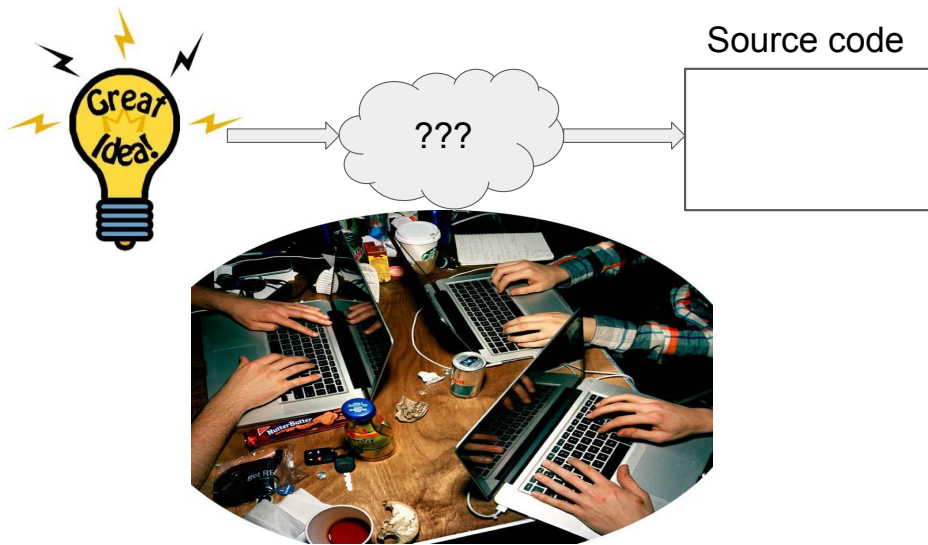12. Do you do **hallway usability testing**?

This week

- Software development life cycle
  - Traditional models
  - Agile models
  - What's the best model (for your course project)?
- Requirements
- Teams and Scrum
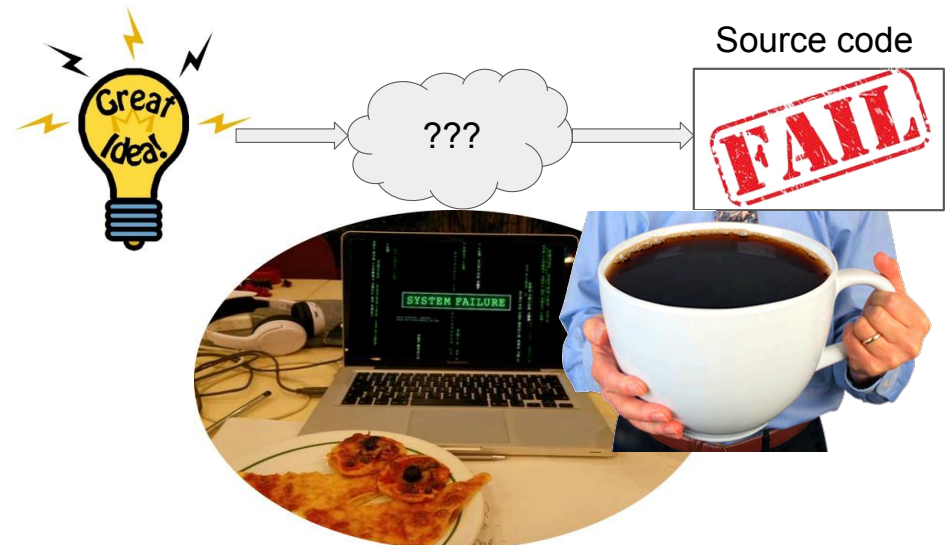

Software development: the high-level problem

Specification → ??? → Source code

Software development: code and fix

One solution: "*Here happens a miracle*"

Source code

Software development: code and fix

One solution: "*Here happens a miracle*"

Source code

Software development: ad-hoc or systematic?

Pros: Ad-hoc

● ...

Cons: Ad-hoc

● ...

Software development: ad-hoc or systematic?

Pros: Ad-hoc

● No formal process and onboarding costs.
● Easy, quick, and flexible.

Cons: Ad-hoc

● Might lack important tasks such as design or testing.
● Doesn't scale to multiple developers.
● Difficult 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 questions:
- How to combine the stages and in what order?
- How does this differ for *traditional vs. agile* models?

Major SDLC models

Traditional models
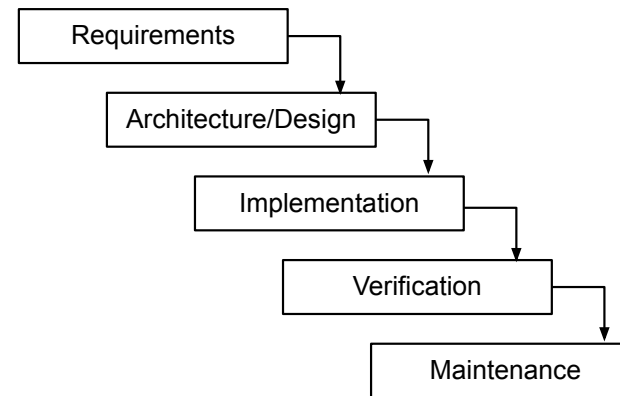- Waterfall model
- Prototyping
- Spiral model
- ...

Agile models
- *XP (Extreme Programming)*
- *Scrum*
- *...*

All models have the same goals:

Manage risks and produce high quality software.
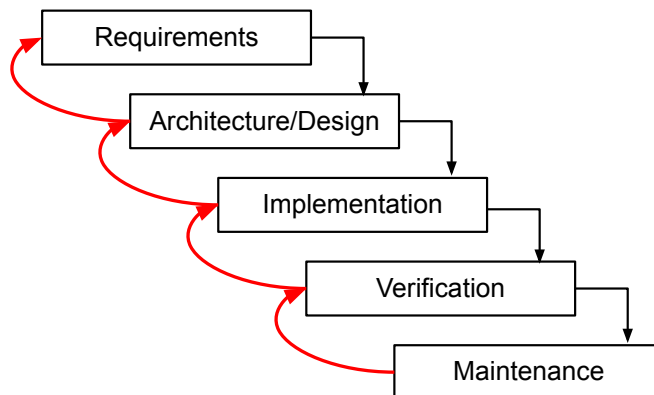
Traditional SDLC models

Waterfall model

| Requirements |
| Architecture/Design |
| Implementation |
| Verification |
| Maintenance |

- 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.

Conceptually very clean, but what's missing?

Waterfall model

| Requirements |
| Architecture/Design |
| Implementation |
| Verification |
| Maintenance |

- 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.
- Requirements are "explored" before the product is fully developed.
- Developers gain experience when developing the "real" product.
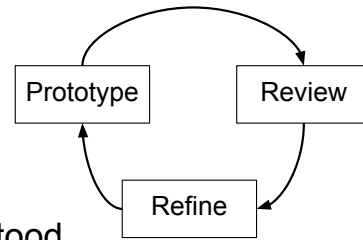

Prototyping

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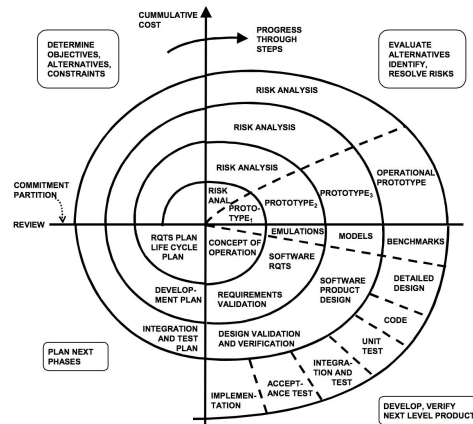
Boehm, *Spiral Development: Experience, Principles,and Refinements*, CMU/SEI-2000-SR-008
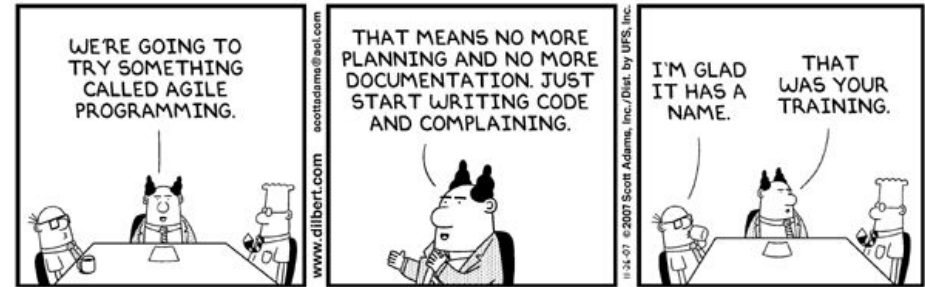
Spiral model

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 SDLC models

Agile models

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

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.
- All tests must be run and pass for every build (may be combined with test-driven development).
- Adaptation and re-prioritization of requirements.
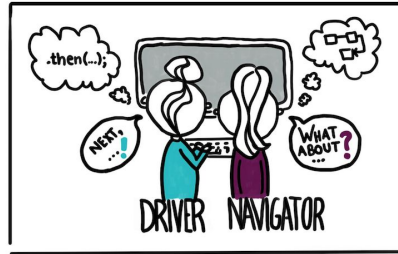
Agile models: XP

Extreme Programming (XP)
- Pair programming and continuous code review.

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.

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 control system for 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 flights (for missed or bumped reservations) from airport terminals or a mobile app.

What's the best SDLC model?

Project management triangle (pick any two)

```
              Scope
               / \
              /   \
         Quality   \
        /           \
     Time----------Cost
```
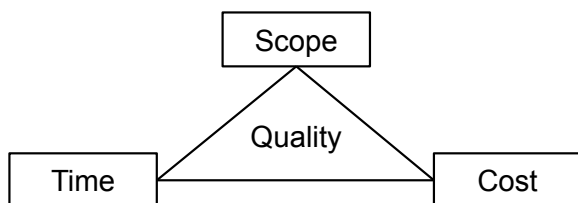
Consider
- The project and task at hand.
- Well-definedness of requirements.
- Risk management and quality/cost control.
- Customer involvement and feedback.
- Experience of management and team members.

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.