

CSE 403



Software Engineering

Spring 2023

#4: Software development life cycle

What is UP?

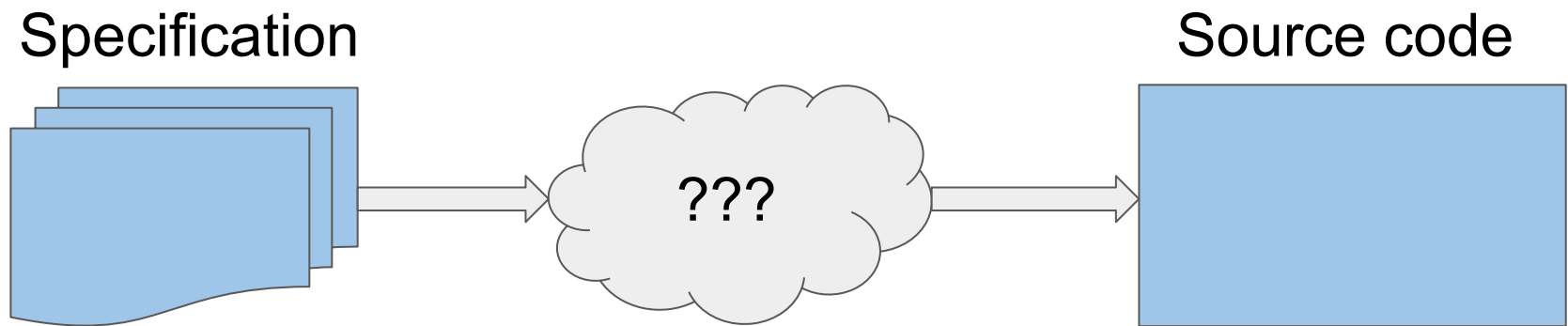
Week 1

-  Course & Projects bootstrapped
-  Discussed SW project good practices (Joel's Test)

Today

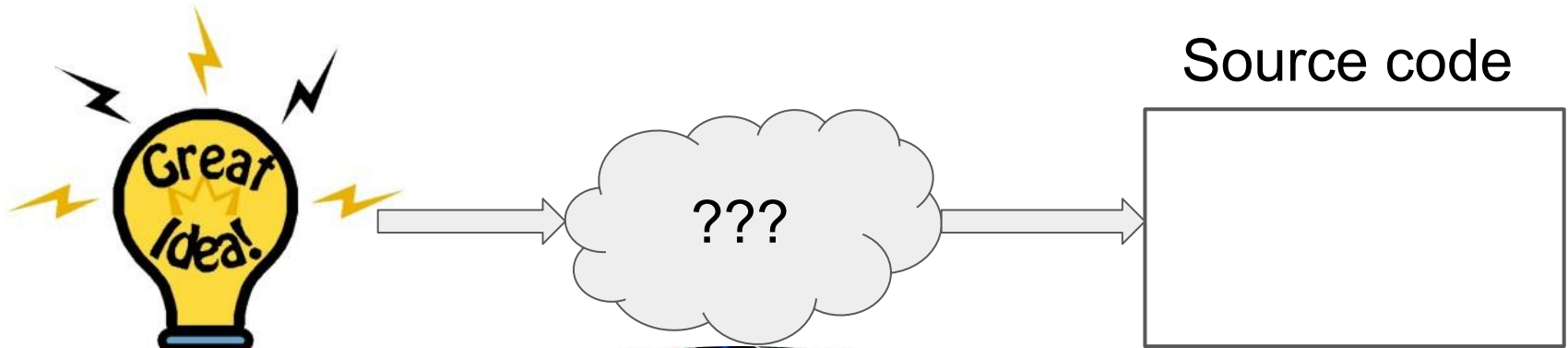
-  Submit Project Ideas 
 - (50% done!!!)
- Thoughts about SW Engineering Life Cycle
 - The problem
 - Traditional Models
 - Agile Models
 - What is the best for your project?

Software development: the high-level problem



Software development: code and fix

One solution: *“Here happens a miracle”*



Software development: ad-hoc or systematic?

Pros: Ad-hoc

- ...

Cons: Ad-hoc

- ...



Software development: ad-hoc or systematic?

Pros: Ad-hoc

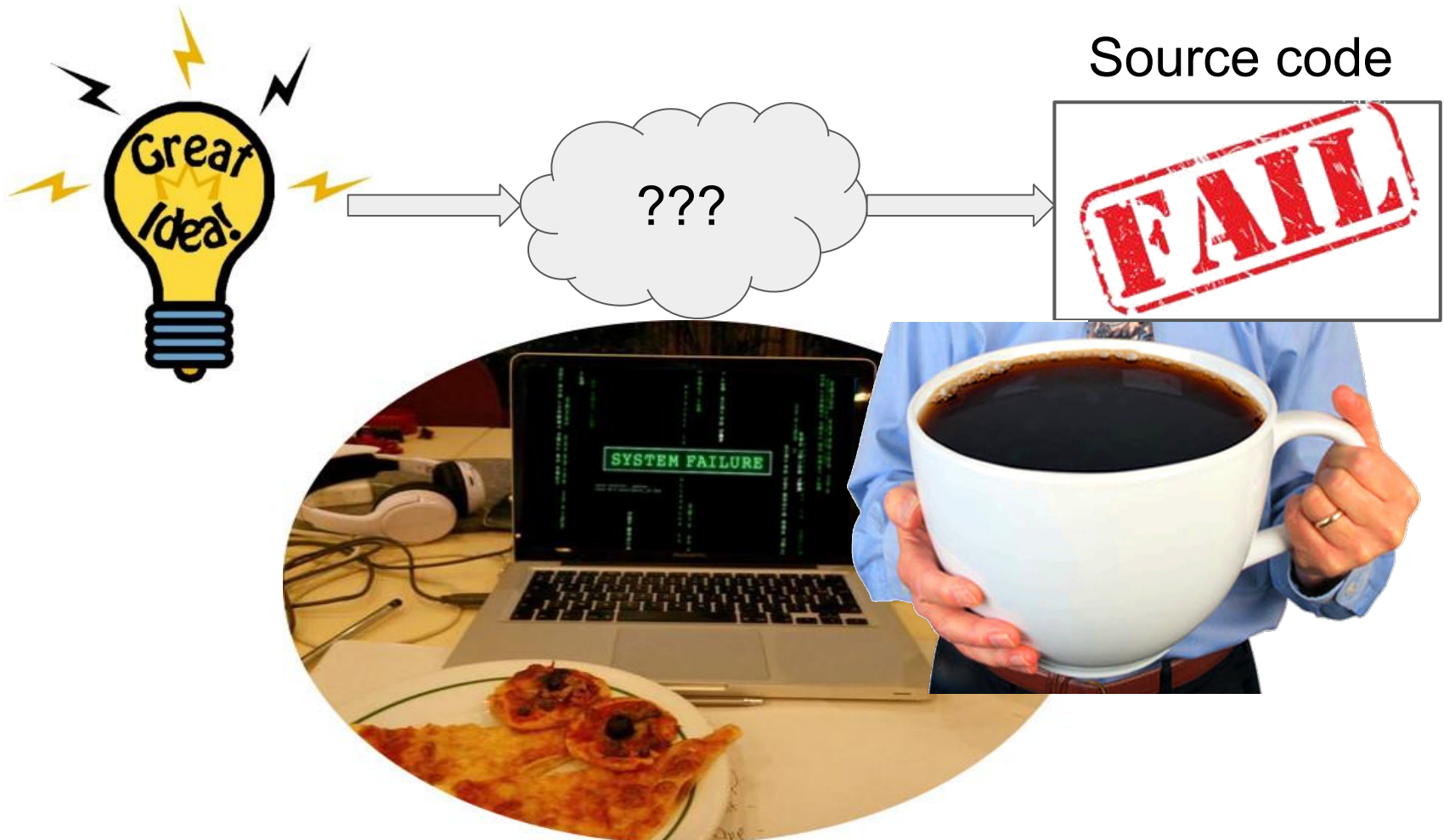
- No formal process and onboarding costs.
- Easy, quick, and flexible.

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: code and fix

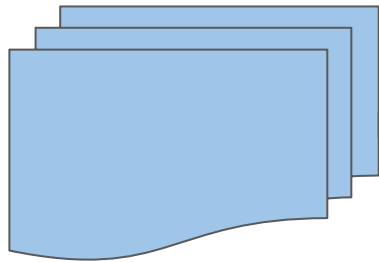
One solution: *“Here happens a miracle”*



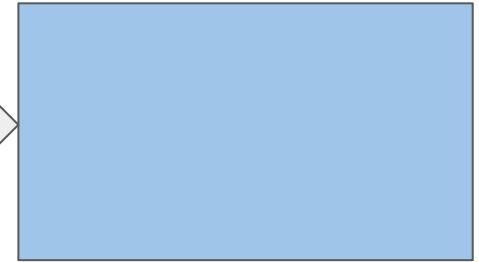
Software development: code and fix

The Engineering way: *“Can we do better given the context?”*

Specification



Source code



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.

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 **actions** to perform.
- Produce **tangible (trackable) items**.
- Allow for **work revision**.
- **Plan actions** to perform in the next stage.

Life-cycle stages

Virtually all SDLC models have the following stages

- Requirements
- Design
- Implementation
- Testing
- Maintenance

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?
- What is the focus on each of those stages?
- How quickly are you going through them?

Major SDLC models

Traditional models

- Waterfall model
- Prototyping
- Spiral model
- ...

Agile models

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

Major SDLC models

Traditional models

- Waterfall model
- Prototyping
- Spiral model
- ...

Agile models

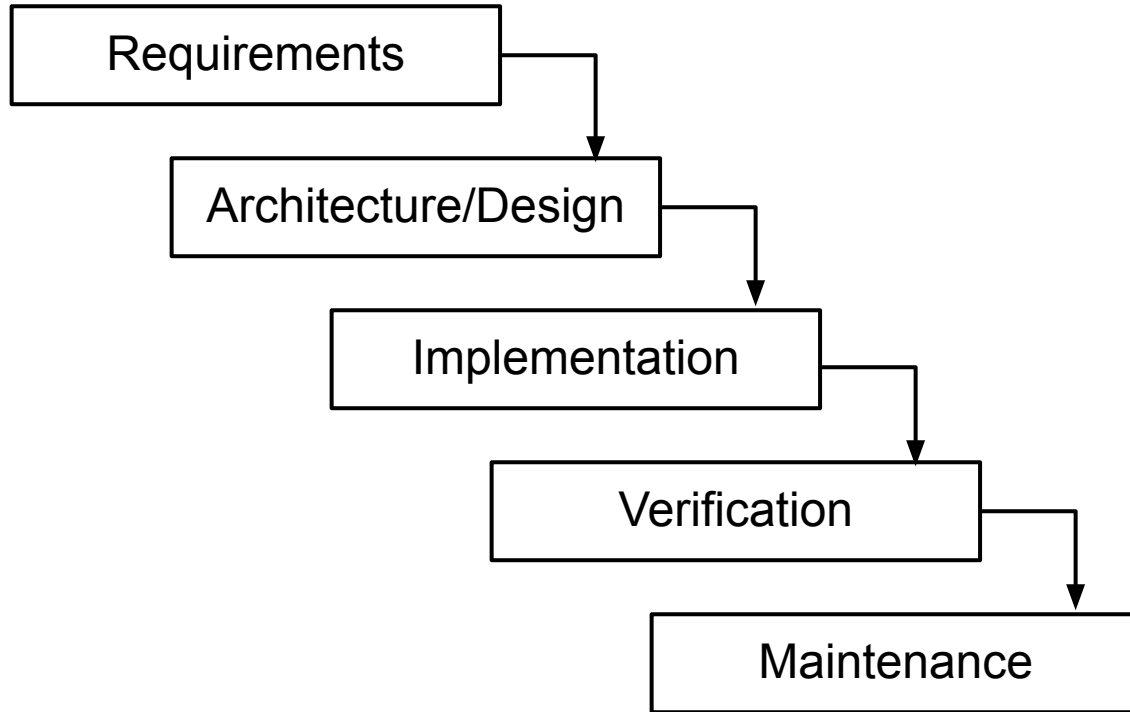
- *XP (Extreme Programming)*
- *Scrum*
- ...

All models have the same goals:

Manage risks and produce high quality software.

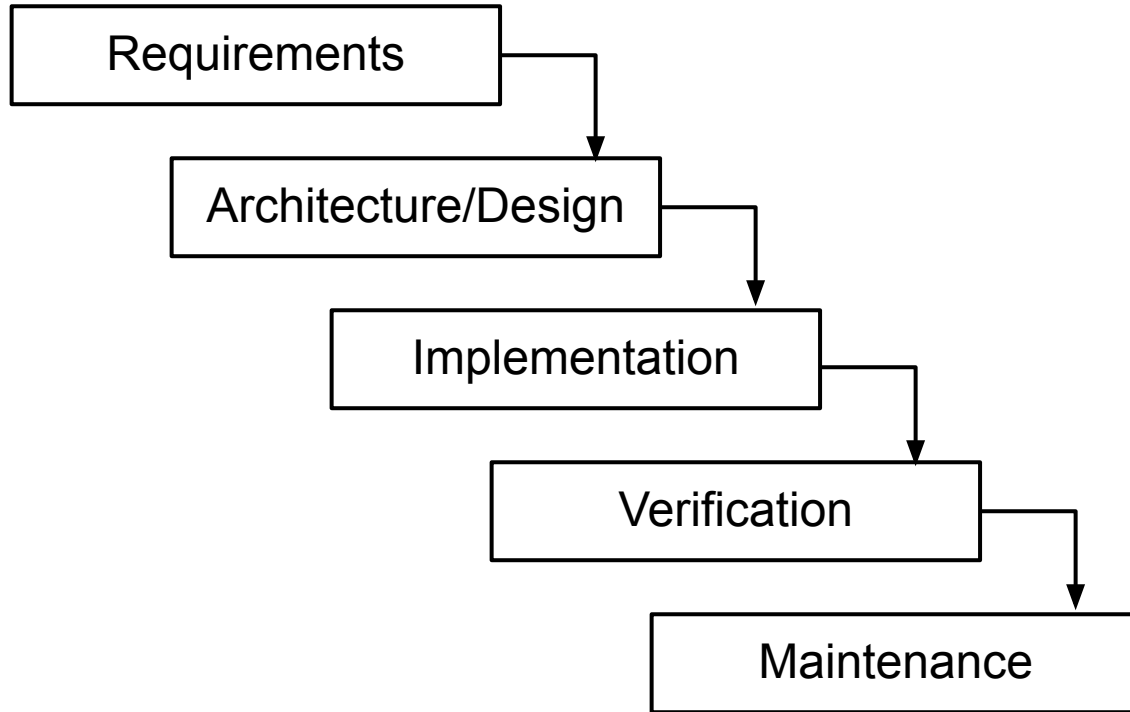
Traditional SDLC models

Waterfall model



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

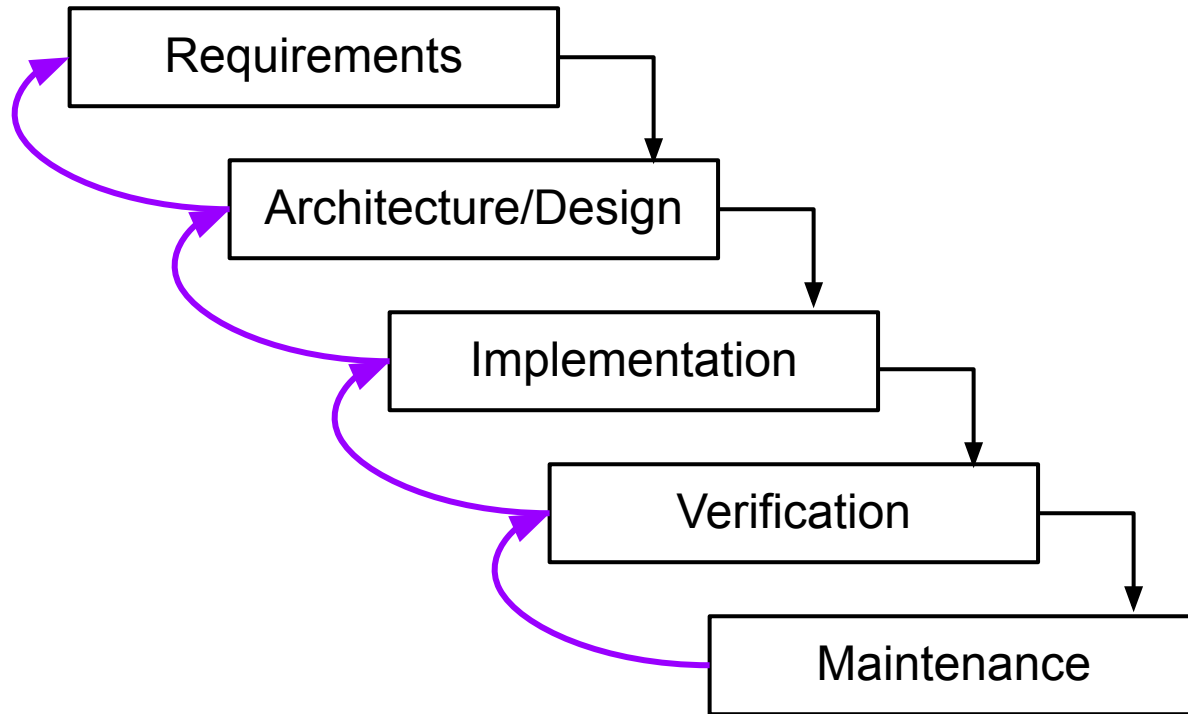
Waterfall model



- Top-down approach.
- Sequential, 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



- 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

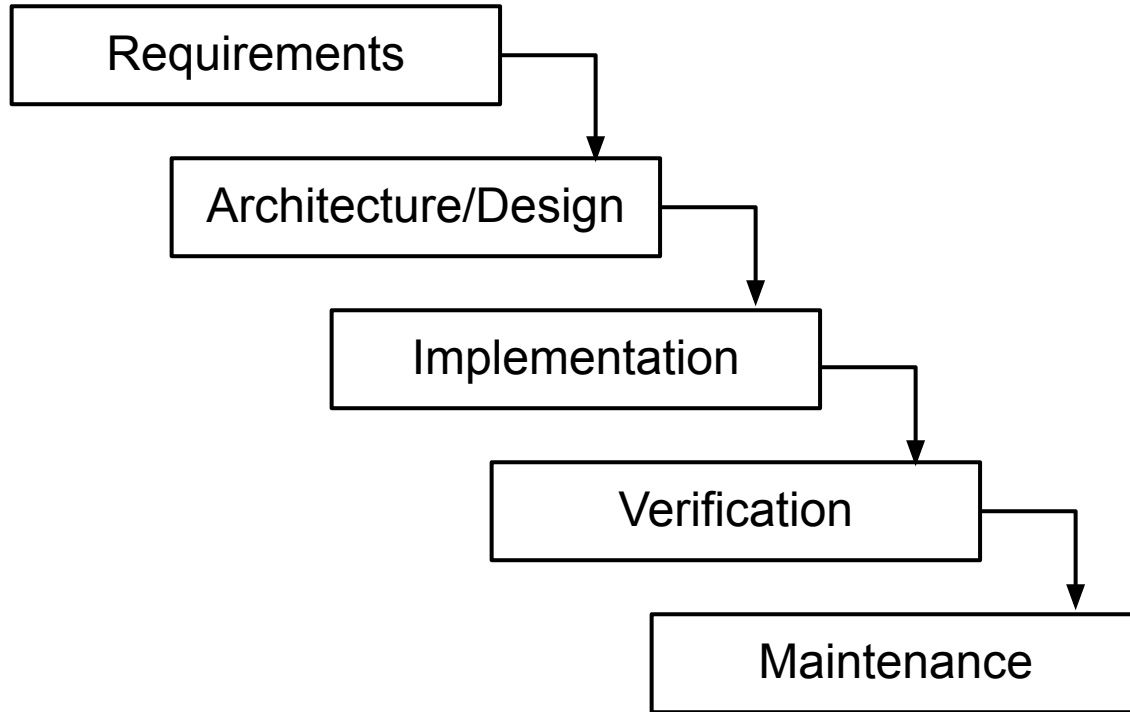
Advantages

- 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.

Waterfall model

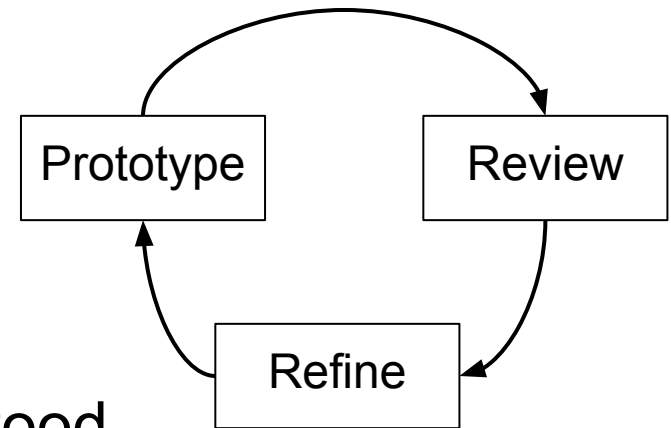


- Top-down approach.
- Sequential, non-overlapping activities and steps.
- Each step is signed off on and then frozen.
- Most steps result in a final document.

In which contexts this can work well?

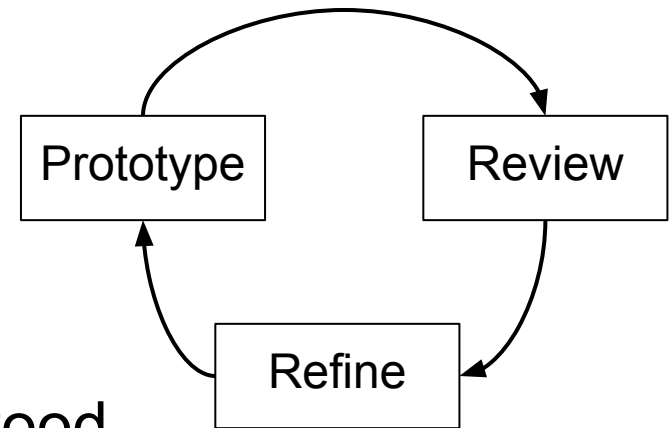
Prototyping

- 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

- 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.



In which contexts this can work well?



- Tasks:
1. Open the Bill to you
 2. Review the account file
 3. Review the account number
 4. Place the bill to you
 5. Be - manage the bill to you

Bill to you
Bill to you
Bill to you

Prototyping: Cool, uhu?

Advantages

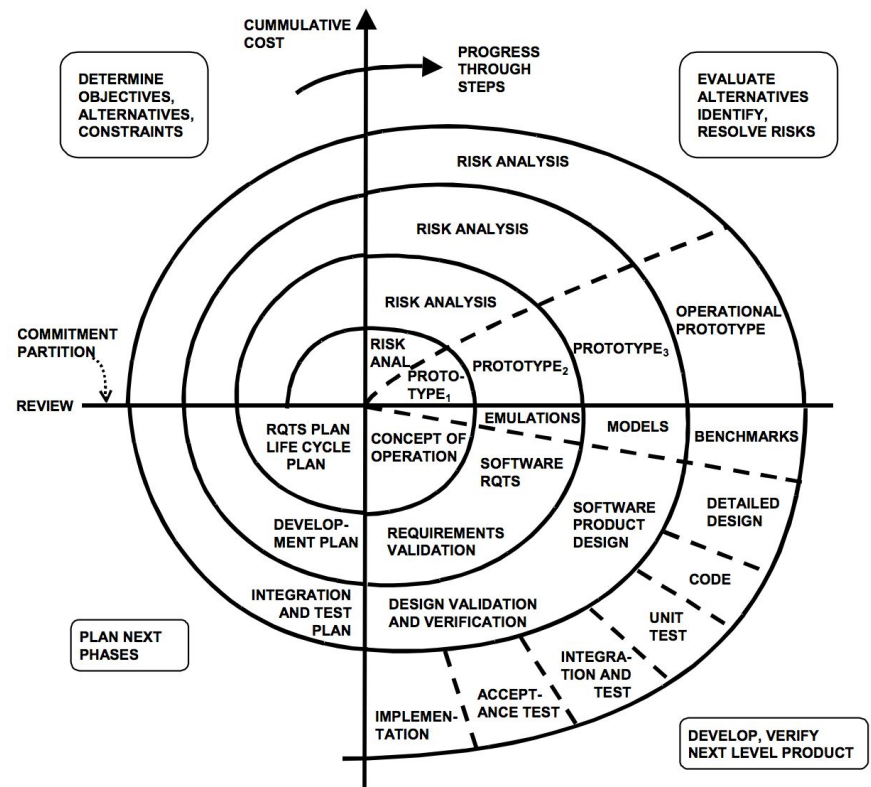
- Client involvement and early feedback.
- Improves requirements and specifications.
- Reduces risk of developing the “wrong” product.

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:
 - Specify
 - Risk analysis
 - Build & Evaluate
 - Plan
- **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

- Harder to run!
- 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 models



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

- Argument: the world is too uncertain, we have to be flexible and responsive to changes!

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)

Agile models: XP

Extreme Programming (XP)

- Shared code ownership
- New versions may be built several times per day (CI)
- All tests must be run and pass for every build
 - test-driven development is highly desirable
- Products delivered to customers weekly.
- Adaptation and re-prioritization of requirements.

Agile models: XP

Extreme Programming (XP)

- Shared code ownership
- New versions may be built several times per day (CI)
- All tests must be run and pass for every build
 - test-driven development is highly desirable
- Products delivered to customers weekly.
- Adaptation and re-prioritization of requirements.

Intense!

Agile models: XP

Extreme Programming (XP)

- Pair programming and continuous code review.



Agile models: XP

Extreme Programming (XP)

- Pair programming and continuous code review.

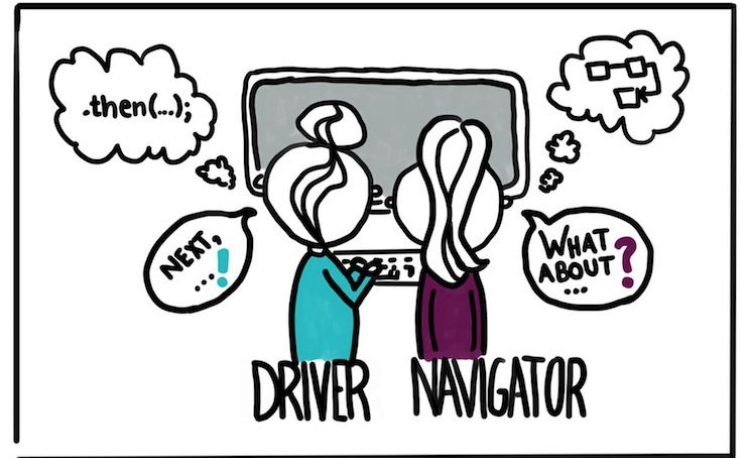


Anyone ever used it?

Agile models: XP

Extreme Programming (XP)

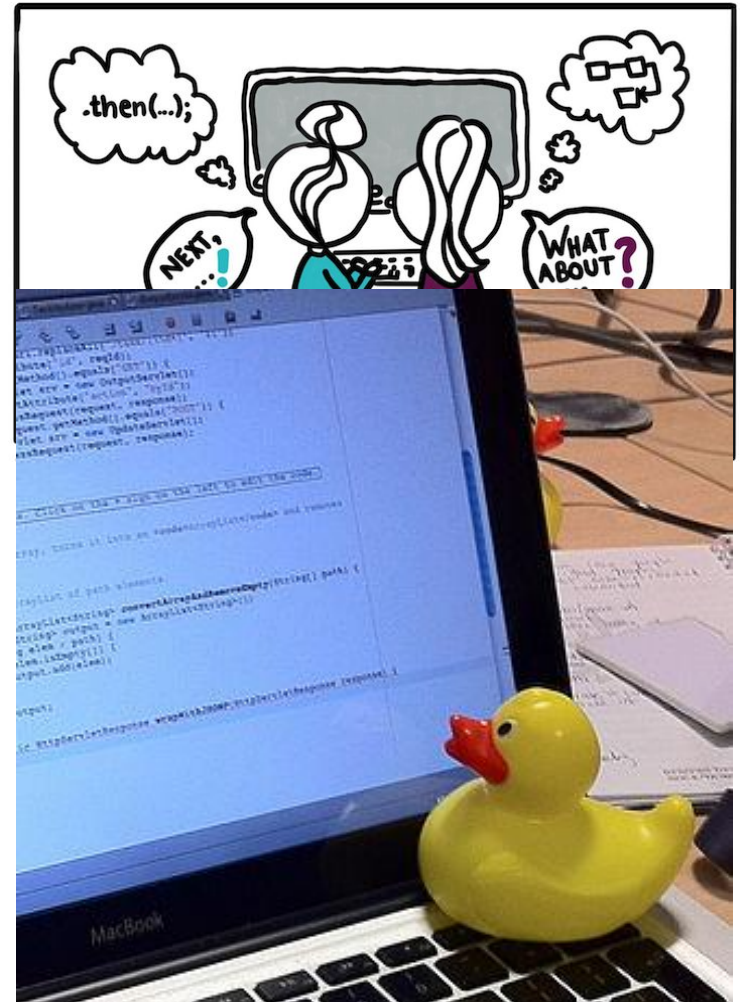
- Pair programming and continuous code review.
- Pairs and roles are frequently changed.



Agile models: XP

Extreme Programming (XP)

- Pair programming and continuous code review.
- Pairs and roles are frequently changed.



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

Basics

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

Any takers?

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?

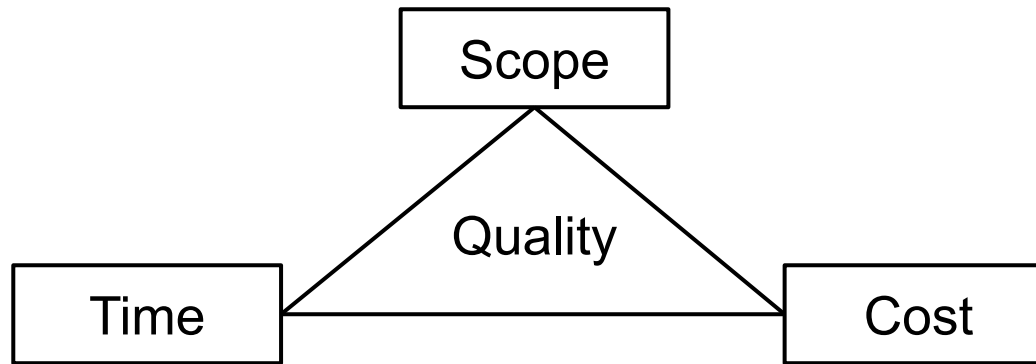
What model would you choose and why?



- 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)



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.

What's next?

WEEK 2

04/03 L: Dev. Cycle

DUE: [PP_1.1!!!](#)

04/04 T: Proposals

DUE: [PP_1.2!!!](#)

04/05 L: Requirements

[Project Requirements \(PR\)](#)

04/06 P: Requirements

04/07 L: Use-Cases

What's next?

WEEK 2

04/03	L: Dev. Cycle	DUE: PP_1.1!!!
04/04	T: Proposals	DUE: PP_1.2!!!
04/05	L: Requirements	Project Requirements (PR)
04/06	P: Requirements	
04/07	L: Use-Cases	

Question, please!