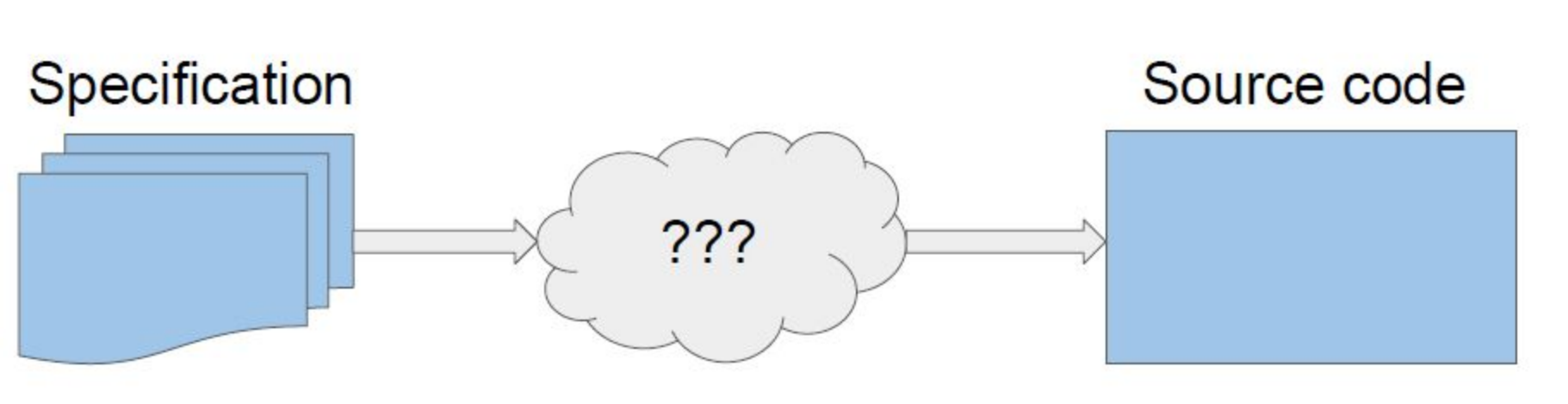
Lesson 2: Software development life cycle

CS 438
Ali Aburas PhD

Today's Goals

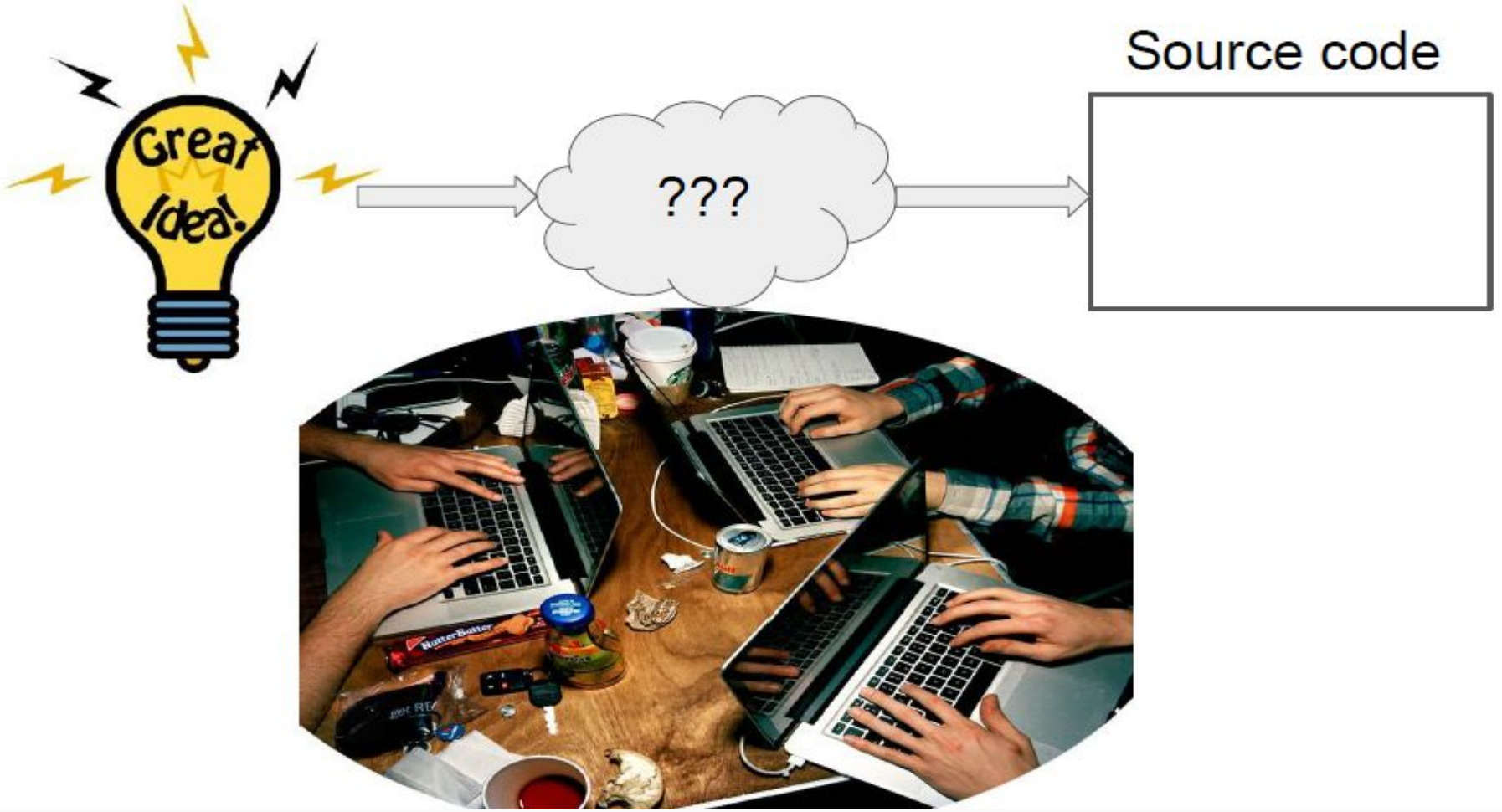
- Thoughts about SW Engineering Life Cycle
- Traditional Models
- Agile Models

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

- 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)

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

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?

Requirement Analysis

- Requirements Analysis phase defines the purpose/function of the system from the client/customer.
- Why is this phase so important?
  - Cost of late correction
  - Failures to understand the requirements led the developers to build the wrong system
- How can we collect the right requirements?
  - **Requirements Elicitation**: collection of requirements from the customer/users and other sources.
  - **Requirements Analysis**: involving the study and understanding of the collective requirements.
  - **Requirements Specification**: laying out function and non-functional requirements.
  - **Requirements Validation**: validating the requirements to make sure that they are complete, consistent and unambiguous

Design

- Design describes the system from the software developer's viewpoint

Developers define the design goals of the project and decompose the system into smaller subsystems that can be realized by individual teams

- The result of this phase can be:
  - Architectural design
    - Figuring out the overall structure of the system
      - What components should be in the system?
      - How should the components be connected?
    - Program design
      - Figuring out how code should be organized
    - How should each component’s code be distributed among classes and/or functions?

Implementation

Finally, we get to write some code!

- The code may be written by the development team, acquired from elsewhere, or modified from existing components.
- Implementation also may include:
  - Writing comments
  - Helping fellow engineers with their coding
  - Answering questions
  - Reading colleagues' code, documentation, etc (i.e., code review)

Testing

Testing phase aims to check the software system meets its specifications and fits its intended purpose/use.

- **Static testing** involves verification, whereas dynamic testing involves validation

- **Verification** is the process of answering the question, are we building the system right?. And checking whether the software system conforms to specifications, and the software system is well-engineered.

- **Verification** uses methods like reviews, audits, and inspections, and walkthroughs.

- **Validation** is the process of answering the question, are we building the right system?, and ensuring that the software system meets the customer expectations and requirements

- **Validation** can be performed at different levels: **Unit testing**, the developers test units individually.**Integration testing**, the developers test the interaction between different units. **System testing**, the developers test the system as a whole. **Acceptance testing**, the client tests the final version of the system against the requirements.

Maintenance

● Maintenance phase aims to identify and fix any errors and problems that occur after delivery the software to the users.

● There are three kinds of maintenance activities:

- Corrective maintenance to eliminate problems with the code.
- Evolution maintenance to add new functions/feature.
- Adaptive maintenance to adapt to a changing technical environment.

● During maintenance every time you modify the software you have to regression test the software (regression testing).

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?

Why do we need project development methodology?

Project development methodologies provide a structured approach to software development that helps ensure that projects are completed **on time**, **within budget**, and to the **required quality**.

Major SDLC models

Traditional models

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

Agile models

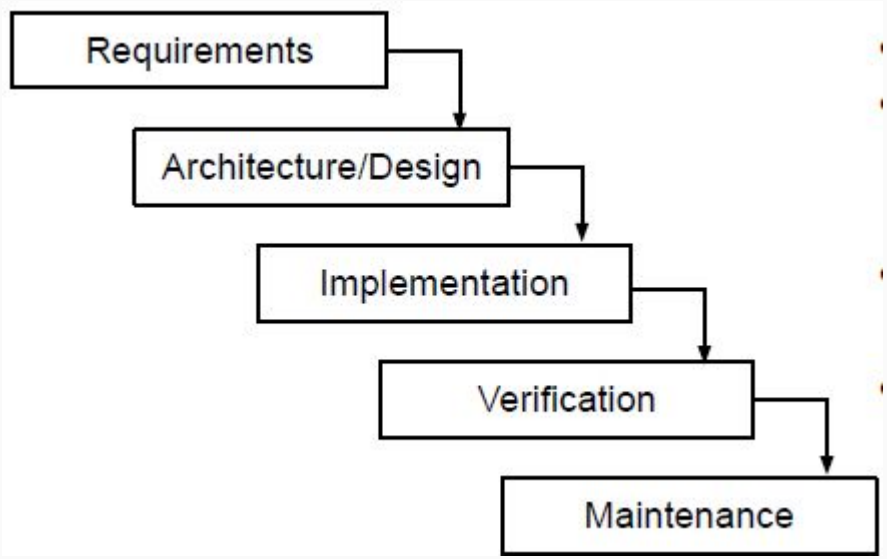
- *XP (Extreme Programming)*
- *Scrum*
- ...

All models have the same goals:

Manage risks and produce high quality software.

Traditional SDLC models

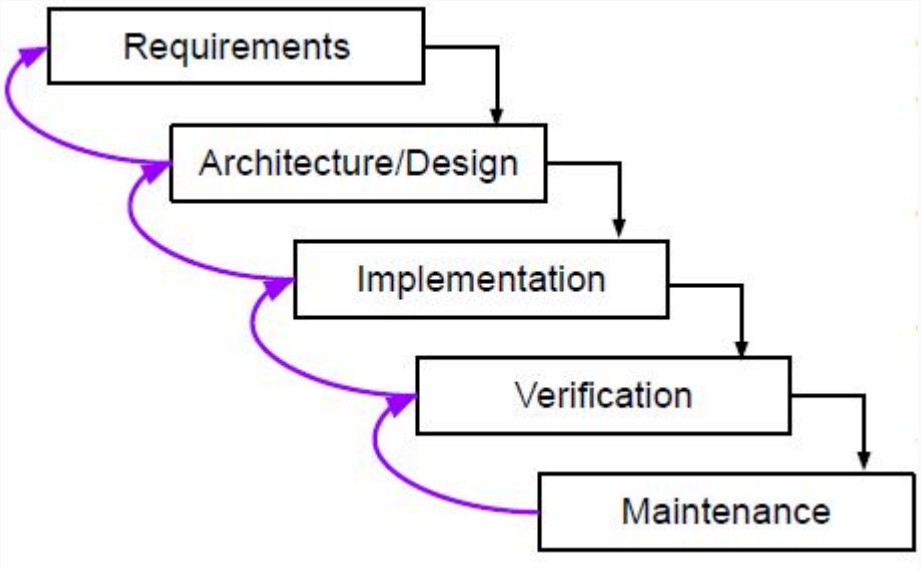
Waterfall model



- 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

Conceptually very clean, but what's missing?

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
- **Backsteps to correct mistakes.**

Waterfall model

Advantages

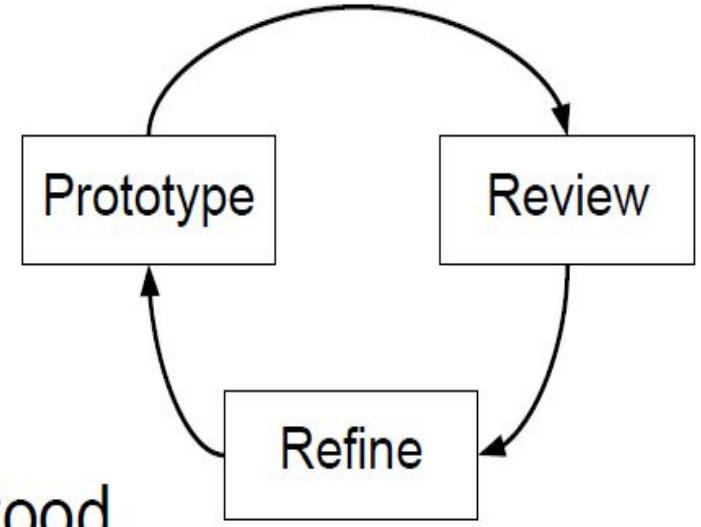
- Easy-to-follow, sequential model.
- Reviews ensure readiness to advance.
- Works well for well-defined projects (requirements are clear).

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

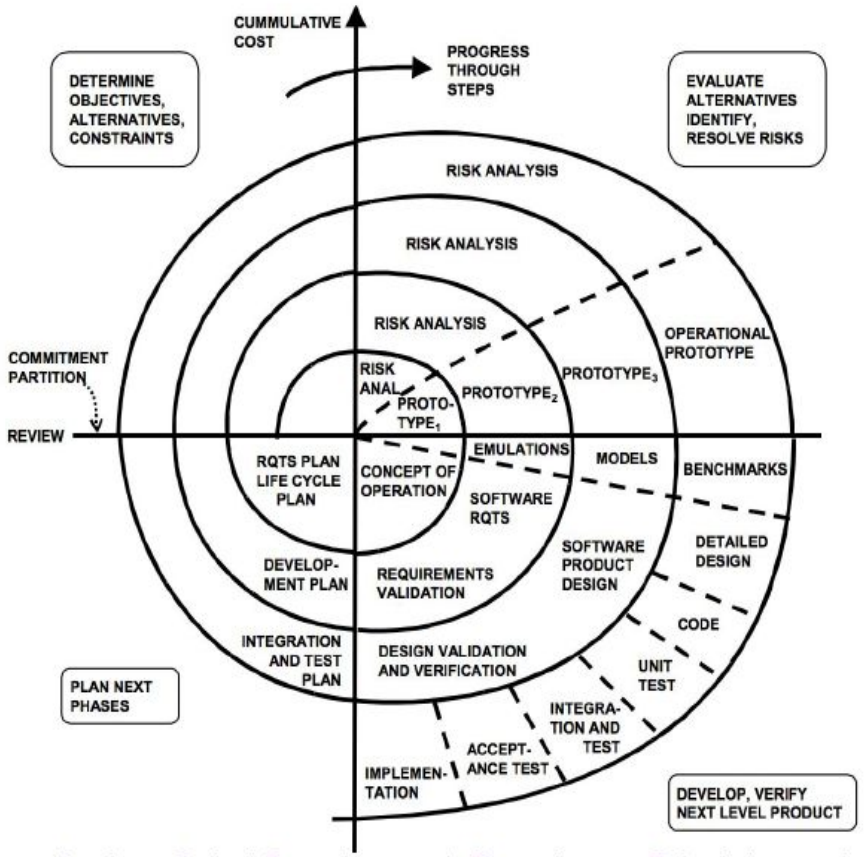
- Client involvement and early feedback.
- Improves requirements and specifications.
- Reduces risk of developing the “wrong” product.

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:
  - 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.

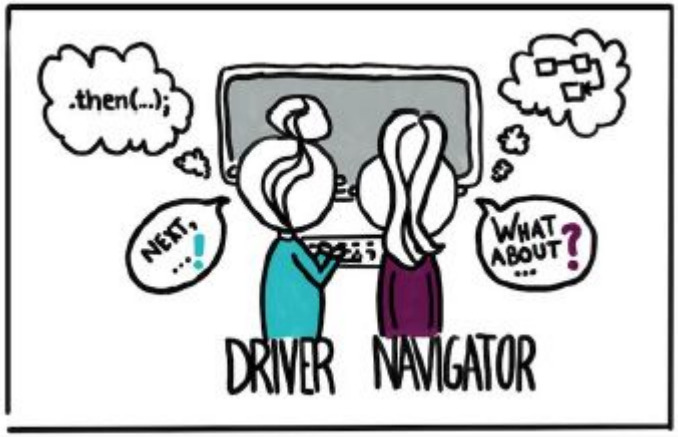
Agile models

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



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 model would you choose and why?

- A system to control anti-lock braking in a car. (Waterfall Model)
- A company accounting system that replaces an existing system.(Waterfall Model)
- A virtual reality system to support software maintenance (Spiral Model)
- A new operating system (Spiral Model)
- Developing an eco friendly - Amazon.com (Spiral Model)
- A new company developing a web based shopping service

(Agile/Evolutionary Prototyping Model)

What model would you choose and why?

- A system to control anti-lock braking in a car. (Waterfall Model)
- A company accounting system that replaces an existing system.(Waterfall Model)
- A virtual reality system to support software maintenance (Spiral Model)
- A new operating system (Spiral Model)
- Developing an eco friendly - Amazon.com (Spiral Model)
- A new company developing a web based shopping service

(Agile/Evolutionary Prototyping Model)