CS 320
Introduction to Software Engineering
Spring 2017

January 25, 2017
Recap

What is Software Engineering?
● The complete process of specifying, designing, developing, analyzing, and maintaining a software system.

Why is it important?
● Decomposes a complex engineering problem.
● Organizes processes and effort.
● Improves software reliability.
● Improves developer productivity.
Recap

Software development process: ad-hoc or systematic?

Pros: Ad-hoc
- No formal process, no overhead. "Brain to keyboard"
- Easy, quick, and flexible.

Cons: Ad-hoc
- Might lack important tasks such as design or testing.
- Doesn’t scale to multiple developers.
- Impossible to measure effort and progress.
Today

- What are the activities and steps in a systematic software development process?
- What are the definitions of all these terms and what are intuitive examples?
Today

● What are the activities and steps in a systematic software development process?
● What are the definitions of all these terms and what are intuitive examples?
Software development process

Activities and steps

Suppose you picked a class project…now what?
Software development process

Activities and steps
- Requirements engineering
- Design and architecture
- Implementation
- Verification and Validation
- Deployment and Maintenance
Example project: smart fridge

Scenario
- Dinner/party time
- On the way home
- Is the fridge stocked?
Example project: smart fridge

Scenario
● Dinner/party time
● On the way home
● Is the fridge stocked?

Solution
● DIY smart fridge
● Realtime data
● Mobile app
Software development process

Activities and steps

- Requirements engineering
- Design and architecture
- Implementation
- Verification and Validation
- Deployment and Maintenance
Software development process

Activities and steps
- Requirements engineering

Group discussion
- How to gather requirements?
- What are potential requirements for our smart fridge example?
Requirements engineering

The process of eliciting, analyzing, documenting, and maintaining requirements.

Types of requirements

- Functional requirements
  - E.g., input-output behavior

- Non-functional requirements
  - E.g., security, privacy, scalability

- Additional constraints
  - E.g., programming language, frameworks, testing infrastructure
Requirements engineering

The process of eliciting, analyzing, documenting, and maintaining requirements.

Common mistakes and challenges
● Implementation/design details instead of requirements
● Unclear scope and unclear requirements
● Changing/evolving requirements
Requirements engineering

The process of eliciting, analyzing, documenting, and maintaining requirements.

Possible strategies for eliciting requirements

- Interviews
- Observations
- Use cases
- User stories
- Prototyping
Software development process

Activities and steps
- Requirements engineering
- Design and architecture

Group discussion
- Given our requirements on the whiteboard, how would you design the system?
- What is the difference between design and architecture?
Software architecture vs. design

Architecture (what is developed?)
- High-level view of the overall system:
  - What components do exist?
  - What type of storage etc.?

Design (how are the components developed?)
- Considers individual components:
  - Data representation
  - Interfaces, Class hierarchy
  - ...

Software architecture: examples

Pipe and Filter

A,CS320,Joe
A,CS529,Joe
B,CS320,Jane
...

grep CS320 grades.csv | cut -f 1 -d ',' | sort | uniq -c

28 A
12 B
...

Software architecture: examples

Pipe and Filter

A,CS320,Joe
A,CS529,Joe
B,CS320,Jane
...

The architecture doesn’t specify the design or implementation details of the individual components (filters)!
Software architecture: examples

Client-server / n-tier

Simplifies reusability, exchangeability, and distribution.
Software architecture: examples

Model View Controller (MVC)

Separates data representation (Model), visualization (View), and client interaction (Controller)
Software architecture vs. design: summary

Architecture and design
- Lowers complexity: separation of concerns, well defined interfaces
- Simplifies communication
- Allows effort estimation and progress monitoring
What’s next?

Activities and steps

● Requirements engineering
● Design and architecture
● Implementation
● Verification and Validation
● Deployment and Maintenance

More on these activities and steps.
How to organize these steps in traditional and agile software development processes?