CS 520
Theory and Practice of Software Engineering
Fall 2022

Software requirements

September 8, 2022
Recap: Logistics

● Will meet in person on Tuesday and Thursday, 10 AM – 11:15 AM and will also be recorded
  ○ Lectures, in-class exercises, final project fairs

● Course schedule and policies on web site:
  https://people.cs.umass.edu/~hconboy/class/2022Fall/CS520/

● Course materials (e.g., slides, recorded lectures, assignments) available through Moodle:
  https://umass.moonami.com/course/view.php?id=31597

● Q&A forums for assignments via Piazza:
  https://piazza.com/umass/fall2022/CS520/home
Recap: Software Engineering

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

Why is it important?
- Software is everywhere and complex.
- Software defects are expensive and range from annoying to life threatening.

Goals
- Decompose a complex engineering problem.
- Organize processes and effort.
- Improve software reliability.
- Improve developer productivity.
Recap: Software Engineering

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

Why is it important?
- Software is everywhere and complex.
- Software defects are expensive and range from annoying to life threatening.

Goals
- Decompose a complex engineering problem.
- Organize processes and effort.
- Improve software reliability.
- Improve developer productivity.
Stakeholders

- “individuals and organizations who are actively involved in the project, or whose interests may be positively or negatively affected as a result of project execution or successful project completion”

[Project Management Institute (PMI®), 1996]
Example 1: Electronic gradebook

- **Users**: Students, course support (e.g., Instructor, TA, grader), Registrar’s Office, tech support
- **Developers**: Designers, programmers, testers, upper management
- **UI experts**: Human Factors, HCI (Human-Computer Interaction)
- **Agencies and organizations**: ADA (Americans with Disabilities Act), FERPA (Family Educational Rights and Policy Act), colleges and universities (e.g., UMass system)
Example 2: US online banking app
What is a software requirements specification?

- Documents the assumptions about, features requested, and behavior of a given software application excepted by the users
- Defines a set of requirements that must be satisfied by the software application
What is a software requirements specification?

- Documents the assumptions about, **features requested**, and behavior of a given software application excepted **by the users** (and more generally **the stakeholders**)

- Defines a set of requirements that must be satisfied by the software application
User story: Overview

- An informal, natural language description of a feature of a software system written from the perspective of an end user.

- This description should generally specify:
  - who wants the given feature
  - what that feature needs to do
  - why the feature is wanted

User story: Common templates

- As a <role> I can <capability>, so that <receive benefit>
- In order to <receive benefit> as a <role>, I can <goal/desire>
- As <who> <when> <where>, I want <what> because <why>
Example 1: Bank manager user stories

- As a Bank manager I can open a new bank account, so that we increase our customer base.

- As a Bank manager I can ask for a list of the existing bank accounts to provide services for those accounts.

- As a Bank manager I can close an existing bank account.
Example 2: Customer user stories
Example 3:
Credit card company user stories
What is a software requirements specification?

- Documents the assumptions about, features requested, and behavior of a given software application excepted by the users
- Defines a set of requirements that must be satisfied by the software application
Two key types of requirements

- **Non-functional requirement**: A quality constraint on the software application (often called the ‘ilities’), e.g., understandability

- **Functional requirement**: An intended (or unintended) behavior of the software application, e.g., Initially, the electronic gradebook needs to allow registered users to login to it.

**NOTE** There are other types of requirements to describe assumptions, features, and usage scenarios (e.g., UML use cases).
Two key types of requirements

- **Non-functional requirement**: A quality constraint on the software application (often called the ‘ilities’), e.g., understandability

- **Functional requirement**: An intended (or unintended) behavior of the software application, e.g., Initially, the electronic gradebook needs to allow registered users to login to it.

*NOTE* There are other types of requirements to describe assumptions, features, and usage scenarios (e.g., UML use cases).
Example:
Agile (or Agility) non-functional requirements

- Debuggability
- Extensibility
- Portability
- Scalability
- Securability
- Testability
- Understandability

Requirements Engineering:
Understandability Example
Requirements Engineering:
Debuggability Example
Requirements Engineering: Two key types of requirements

- **Non-functional requirement**: A quality constraint on the software application (often called the ‘ilities’), e.g., understandability
- **Functional requirement**: An intended (or unintended) behavior of the software application, e.g., Initially, the electronic gradebook needs to allow registered users to login to it.

**NOTE**: There are other types of requirements to describe assumptions, features, and usage scenarios (e.g., UML use cases).
Requirements Engineering: Phases

1. Elicitation
2. Specification
3. Analysis
4. Management

Requirements → Architecture & design → Implementation
Requirements Engineering: Phases

1. Elicitation
2. Specification
3. Analysis
4. Management

Requirements <-> Architecture & design <-> Implementation