[CS3704] Software Engineering

Dr. Chris Brown
Virginia Tech
9/6/2023

Announcements

- **Homework 1 due Friday (9/8) at 11:59pm**
  - Canvas questions (Upload file or input text)

Requirements

What are Requirements?
Types of Requirements
Iterative Requirements Analysis
Course Project

Learning Outcomes

By the end of the course, students should be able to:

- **Understand software engineering processes, methods, and tools used in the software development life cycle (SDLC)**
- **Use techniques and processes to create and analyze requirements for an application**
- Use techniques and processes to design a software system
- Identify processes, methods, and tools related to phases of the SDLC
- Explain the differences between software engineering processes
- Discuss research questions and current topics related to software engineering
- Create and communicate about the requirements and design of a software application

Warm-Up: Discuss

1. ***What is the difference between plan-driven and iterative software engineering processes?***

Requirements

Goal: Understand customer requirements for the software system

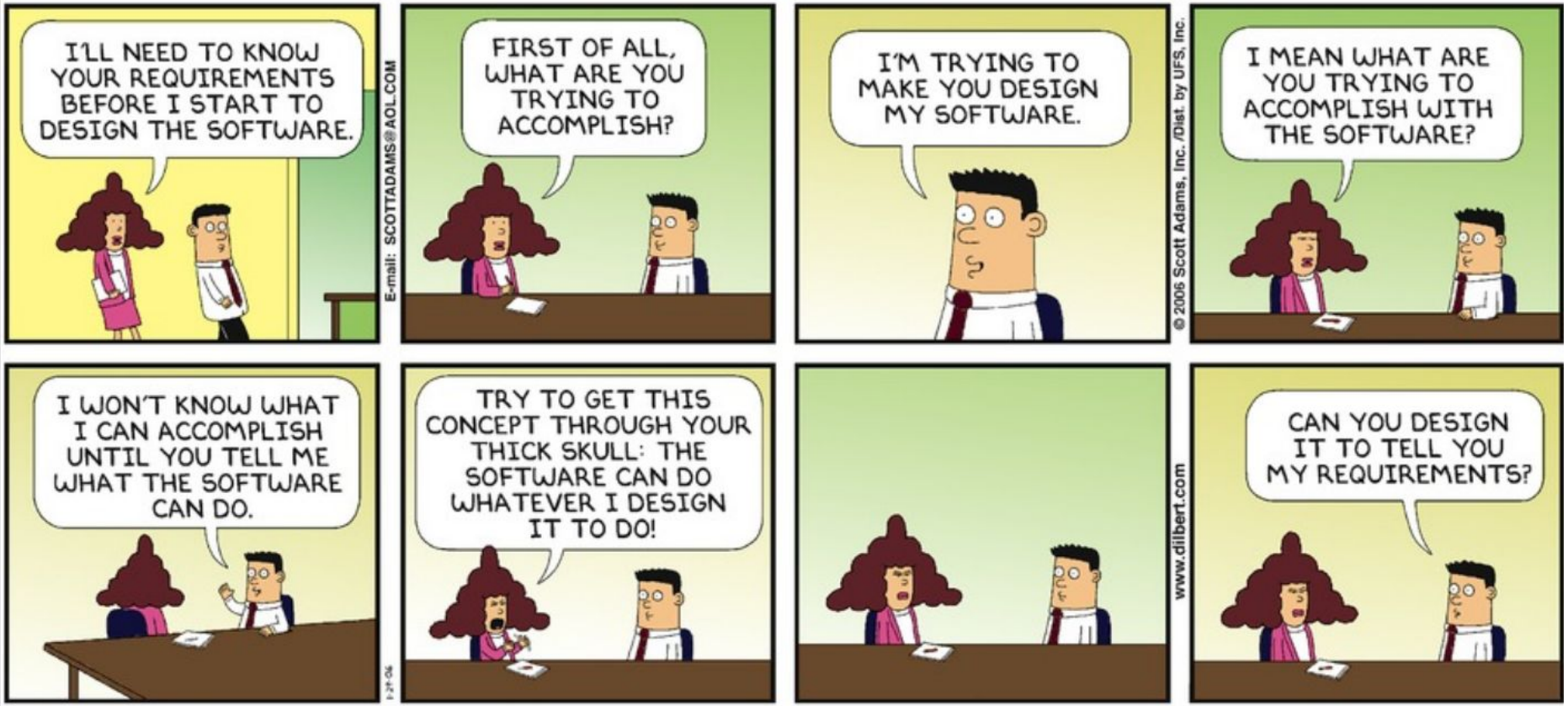
- The *what* of the project

- Very difficult to "get right" the first time and evolve over the course of development
  - Remember the Top 3 reasons for project failure: **(2)** Incomplete and **(3)** Changing Requirements
- *Software Artifacts:* requirements documents, use cases, user stories,...

What are requirements?

Definition: Capabilities and conditions to which the system — and more broadly, the project — must conform. [Larman]

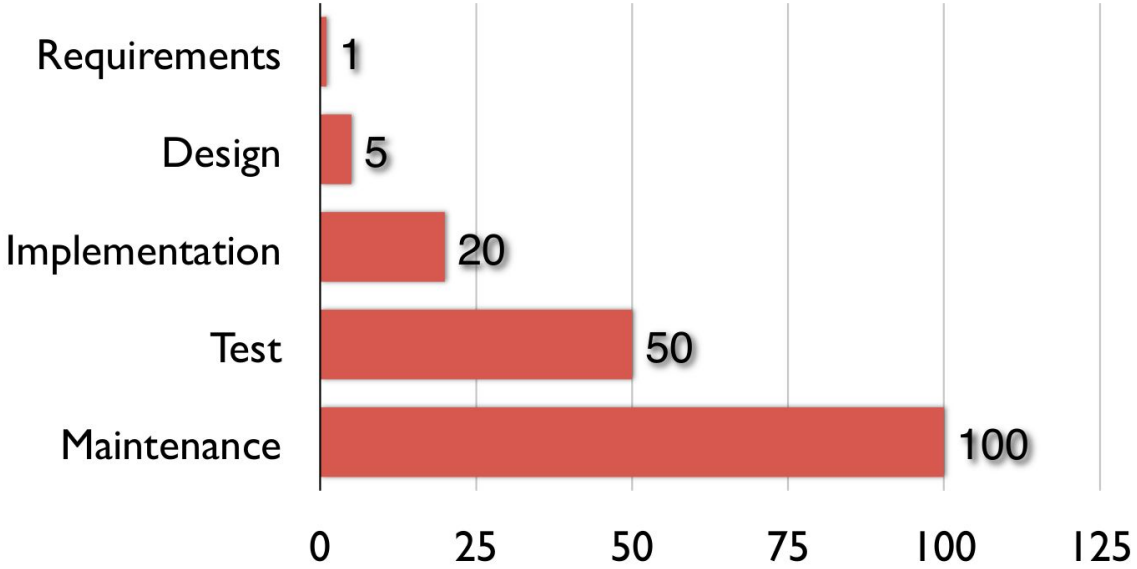
- Focusing on the **WHAT** *not* the **HOW**
- **Should always come *first* in the SDLC**



http://dilbert.com/strip/2006-01-29

Why should we care?

- Much cheaper and easier to modify software and fix bugs earlier in the SDLC.



Why should we care? (cont.)

"*The hardest single part of building a software system is deciding precisely what to build.* ***No other part of the conceptual work is so difficult as establishing the detailed technical requirements****, including all the interfaces to people, to machines, and to other software systems.* ***No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.***"
- Fred Brooks

Requirements are Difficult

Despite the clear benefits of getting requirements set *early*, they are the hardest part of the system to do right because:

[*Discuss:* What are your experiences with software requirements. Why do you think they are difficult?]

- we don't understand everything about the real world that we need to know,
- we may understand a lot, but cannot express everything that we know in code,
- we may think we understand a lot, but our understanding may be wrong,
- we cannot anticipate the understanding and behavior of others,
- requirements change as client's needs change,
- requirements change as technology changes,
- requirements change as clients and users think of new things they want,
- requirements of a system change as a direct result of deploying the system,

…

Requirements Management

- Requirements for software will change
  - First Law!
- High-level requirements are usually set
- **Requirements management:** Set of activities to help teams identify, control, and track requirements and changes to requirements at any time during the SDLC.
  - Analysis
  - Specification

Analysis vs. Specification

- **Analysis:** process of *understanding* the problem and the requirements for a solution
- **Specification:** process of *describing* what a system will do

➔ *Analysis leads to Specification* – they are not the same!

Requirements Analysis is Hard

- Major causes of project failures
  - Lack of user input
  - Incomplete requirements
  - Changing requirements
- **Essential requirements analysis solutions**
  - Classification of requirements
  - Iterative and evolutionary requirements analysis
  - Use Cases

Classification of Requirements

- **Functional:** expresses a function that an application must perform
  – Example: “HokieSpa must read students academic course data for students to view progress towards their degree”
  – All other requirements are *non-functional*
- **Nonfunctional:** does not involve specific functionality, but should be specific, quantifiable, and testable

Non-functional Requirements

- **Usability:** human factors, help documentation, etc.
  - "Text on the display must be visible from 1 meter."
- **Reliability:** failure frequency and recoverability

– "When doing search, the radar should have 28 hours MTBF(mean time between failures)"

- **Performance:** response times, throughput, availability, resource usage, etc.

– "The server response time is <1 sec for 90% of the accesses"

Non-functional Requirements (cont.)

- **Supportability:** adaptability, maintainability, internationalization, configurability
  - "The system should allow frequent and easy changes in the network configuration"
- **Implementation/Constraints:** resource limitations, languages, tools, hardware
  - "Must use Linux and Java"

Requirements Engineering

The process of defining, understanding, documenting, and maintaining requirements.

- Subset of software engineering
- Introduced for the waterfall model (first phase)

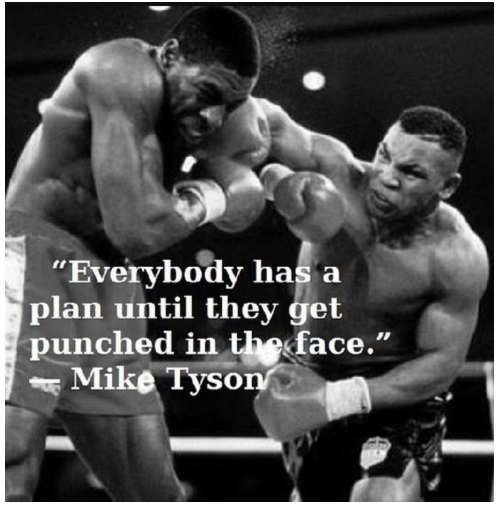
- Inception
- Elicitation
- Elaboration
- Negotiation
- Specification
- Validation

Iterative and Evolutionary Analysis

- **Motivation:** 20-50% of original requirements change.
  - Miscommunication, changing business needs, etc.
- Strategies
  - Specify most architecturally significant and high business-value requirements *before* implementation.
  - Iterations allow quick adaptation and increments of requirements (*Iteration planning meeting*).
  - Brainstorming, interviews, prototypes, ethnographies,...

Reminder: Agile Planning

- The purpose of planning is to understand, not to document.
- Keep is simple
  - Small set of disciplines
  - Avoid unnecessary artifacts
  - Only high-level planning
- Plans are inaccurate



  - **Only *tested code* demonstrates and verifies good design (and requirements)!**

Iterative Requirements Analysis

How can you do requirements analysis iteratively?

- *Inception (~2 days)*

– Identify names of use cases and features, and key non-functional requirements

– 10% are analyzed in detail based on high-risk, high-business-value, and architectural significance

- *Iteration planning meeting*

– Choose a subset of the 10% for implementation, break them down to detailed iteration tasks.

Iterative Requirements Analysis (cont.)

- *Elaboration* (iteration #1)

– Design, implement, and test selected features

– Demo it to collect feedback

– Pick another 10-15% of to analyze in detail (1-2 days *in iteration*)

- *Iteration planning meeting*
- *Elaboration* (iteration #2)

– Repeat iteration #1

…

• *Elaboration* (iteration #3)

… …

At the end of elaboration…

- Requirements are analyzed and written in detail
- Parts of the implementation are done
- Other phases require very little work on use case requirements

Requirements Elicitation

Discovering the requirements of a system.

Techniques for requirements elicitation:

- **Brainstorming**: Gather stakeholders, collect ideas and prune
- **Interviewing:** Formal or informal interviews with stakeholders
- **Ethnography:** A social scientist observes and analyzes how people actually work
- **Strawman/Prototype:** GUI, flow charts of UIs

Requirements Analysis in UP

Introduction of major artifacts (work products):

- **Use Cases:** for functional requirements
- **Supplementary specification:** for non-functional requirements

Definitions

- **Use case:** a written description of using the system to fulfill stakeholder goals.
- **Stakeholders:** Anyone who supports, benefits from, or is affected by a software project that has direct or indirect influence on its requirements.
  - Managers, software engineers, users, clients/customers, marketing, system administrators, testers, etc.

Course Project

Project Details

Develop a project to support teamwork and collaboration in software engineering.

- How you accomplish this goal is up to you and your group.
- Grade is primarily determined by meeting deadlines and your mastery of the SE concepts and skills discussed in class.

Project Goals

Project Learning Outcomes:

- Use techniques and processes to design a software system
- Create and communicate about the requirements and design of a software application

Goal:

- Show an understanding of SE processes, methods, tools, and concepts discussed in class.
- Provide practical experience to get an idea of what would be like to be a *software engineer*.

Project Guidelines

- Can complete any project as long as it meets the overall theme.
  - Project ideas available on GitHub
- Must use a GitHub repository
- Must use a task management and issue tracking system
- Milestone artifacts should be submitted on Canvas and uploaded to your GitHub repo.

Project Deliverables

Milestone (Project Grade %)	Deliverables	Deadline
PM0 (5%)	Group Preferences	Sept 11
	Project Questions	
PM1 (30%)	Lightning Talk	Sept 22
	Project Proposal	
PM2 (15%)	Process I	Oct 13
	Requirements Workshop	
	Requirements Analysis	
	Requirements Specification	
PM3 (15%)	Process II	Nov 10
	High-Level Design	
	Low-Level Design	
	UI Design	
PM4 (35%)	Black Box Test Plan	Dec 11
	Final Presentation	*Dec 4 before class (slides)*
	Final Report	
	Retrospective	

Deadlines at 11:59pm unless otherwise specified. Deliverable details available on GitHub. Course late policy applies to these milestones.

Upcoming Milestones: PM0

The preliminary milestone is a brief survey that consists of:

- Preferences for group members
- Questions about project details based on this class and GitHub documentation

Even if you don't have questions/preferred teammates, you must complete the survey!

Due: Monday, September 11 at 11:59pm

Upcoming Milestones: PM1

- Lightning Talk (Proposal Introduction)
  - **5 min** talk introducing your project to the class
  - Must include problem statement, proposed solution, one example use case, etc.
  - One student or entire group can present.
  - Brief time for Q&A following presentations

Upcoming Milestones: PM1 (cont.)

- Project Proposal
  - Document explaining your project idea and which SE process model your team will try to utilize.
    - Choose your SE process wisely, it will impact future milestone submissions for your project!
  - Proposal and lightning talk will be reviewed by instructor to approve your project idea! You must get approval before continuing to PM2.

Next Class…

- **Practice Discussion talk by instructor**
  - **Sign-ups for discussion presentation will be in class on Friday!**
- **HW1 (due Friday 9/8 at 11:59pm)**

- **PM0 (due Monday 9/11 at 11:59pm)**
- **PM1 (due Friday 9/22 at 11:59pm)**
- **Requirements Analysis next week**