

# Course Logistics

CS445/ECE451/CS645

Software Requirements Specifications & Analysis

Winter 2026

Byron Weber Becker

[bwbecker@uwaterloo.ca](mailto:bwbecker@uwaterloo.ca)

# Today's Agenda

- Overview of “Reqs and Specs”
- AI
- Project & Deliverables
- Weekly Schedule
- Teamwork
- Grading Scheme
- Course Text and Resources
- Forming Teams

# Overview of “Reqs and Specs”

“The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as 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, “No Silver Bullet – Essence and Accident in Software Engineering”, *IEEE Computer*, 1987

# Overview of “Reqs and Specs”

- determining what software to build
- communicating those decisions
  - to the rest of the development team
  - other stakeholders
- This will be done in teams.
- Ultimate deliverable is a “Software Requirements Specification” or SRS.

# Course content

## Requirements

- Scoping
- Stakeholder analysis
- Elicitation
- Reqs reference model
- Negotiation
- Quality requirements
- Prioritization
- Risk analysis
- Validation and verification
- Safety analysis
- Cost estimation

## Modelling

- Lean canvas
- Software modelling:
  - Use cases and scenarios
  - User stories
  - Workflow models
  - Domain models (UML)
  - Constraints (OCL)
  - State machines (UML)
  - Navigation map
  - User interfaces
  - Temporal logic
- Software requirements specification

# Overview of “Reqs and Specs”

- What does an SRS look like in the age of artificial intelligence?

Sunday	Monday	Tuesday
1 Wk05	2 Use cases, User stories Analyze Prob Fit interviews Writeup brainstorming	3 Lightweight Models
8 Wk06	9 Updated use cases, Workflow models, Quality attributes	10 Requirements Analysis
15 Reading Week	16 Family Day	17 Reading Week
22 Wk07	23 Risk analysis Quality require interviews Require priority interviews	24 Elaboration

February

Write a command line program in Scala3 that reads a configuration file and produces one or more HTML monthly calendars.

Each monthly calendar will be a grid with seven columns (one for each day of the week starting with Sunday). There will be a row for each full or partial week in the month. Partial weeks will need to be padded so that days line up correctly.

Each monthly calendar will be in an HTML DIV element. That DIV element will contain the month name (eg January) and year (eg 2026) on one line followed by the grid.

Each day in the grid will have the date (eg "1" for January 1, etc) and annotations taken from the configuration file. Each day in the grid should have the same width. The height should be approximately the same but grow as needed to accommodate the annotations.

The config file will include one or more lines in the following format: MONTH DAY [CLASS]: ANNOTATION

where:

- MONTH is a month name abbreviation (Jan, Feb, Mar, Apr, May, etc)

- DAY is the date of the month (eg 1, 05, 21); leading zeros should be allowed.
- CLASS is optional. It's a string representing the HTML class to use to display the annotation. This allows for adding colour or bold or something.
- ANNOTATION is a string to be placed in the calendar grid cell identified by the MONTH and DAY.

The beginning of the config file should have two lines representing the starting and ending year and month for the calendar (inclusive):  
start: YYYY-MM  
end: YYYY-MM

Use SASS or LESS for the css code that you need. Put it in a file named `scal.less` or `scal.sass` as appropriate. Add instructions to the build script to transpile it into CSS.

The program should be written in Scala3.  
Use braces; avoid the significant indentation syntax.  
Use the com.lihaoyi.scalatags library for generating HTML.  
Use the com.github.scopt library for parsing command line arguments.  
Document your code.  
Use the mill build tool.

# Artificial intelligence

- How does AI affect the discipline of requirements engineering?
- How does AI affect **learning** the discipline of requirements engineering?

# Artificial intelligence



## **NO AI**

### **AI must not be used**

The assessment is completed entirely without AI assistance. This level ensures that students rely solely on their knowledge, understanding, and skills.



## **AI-ASSISTED EDITING**

### **Include original work/drafts**

AI can enhance student work for clarity and quality of final output, but cannot create new content.



## **AI AS A LEARNING TOOL**

### **Use AI to prepare, review and study**

Use AI to learn skills and knowledge related to the course content and to prepare for course assessments.



## **AI OUTPUT EVALUATED**

### **Use AI as instructed; cite AI-content**

AI completes task parts; students must discuss and critically evaluate AI output.



## **AI FOR IDEA GENERATION**

### **No AI content in submission**

AI can be used to enhance brainstorming, structure creation, and idea generation to improve work.



## **FULL AI**

### **AI use is integral to assessment**

Use AI as a 'co-pilot' to enhance creativity and meet assessment requirements.

# AI: What does an LLM “think”?

Prompt:

I am a student in a fourth-year course on software requirements specifications. The instructor just said the course is about “deciding what software to build and communicating those decisions to the development team and other stakeholders.”

We will be discussing requirements elicitation, brainstorming features, prioritizing tasks, and risk analysis. We'll be identifying use cases, writing user stories, drawing UML diagrams, etc. It all culminates in a massive Software Requirements Specification document.

As a student, what are appropriate uses of Gen AI? What are the skills in this course that I absolutely need to know and do for myself vs. those that are amenable to AI use?

# AI response... (1 of 4)

This is a thoughtful question. Here's how I'd think about it:

## **Skills you absolutely need to develop yourself:**

The core competency of this course is learning to *think like a requirements engineer*—to ask good questions, identify what's actually needed versus what's wanted, and communicate clearly with stakeholders who may have conflicting priorities. These are fundamentally human skills that require practice.

You need to do your own elicitation and brainstorming. When you're talking to stakeholders or domain experts (even hypothetical ones), *you* need to ask clarifying questions, probe assumptions, and discover the implicit requirements. If you use AI to generate requirements, you skip the learning experience of recognizing gaps and contradictions in your understanding. This is where the real value lives.

Similarly, you need to practice prioritization and risk analysis yourself. These involve making judgment calls with incomplete information—deciding what trade-offs matter, what could go wrong, which risks are acceptable. That decision-making muscle is built through doing it, not delegating it.

# AI response... (2 of 4)

## **Where AI can appropriately help:**

Once you've identified your requirements, AI can help you polish your communication. For example, you might use it to improve the clarity of a user story you've already written, or to catch inconsistencies in how you've phrased similar requirements across your document. It can help you understand UML diagram syntax or suggest how to restructure a section for clarity.

AI can also be useful for *testing your thinking*. After you've drafted a use case, you could ask an AI to identify ambiguities or missing steps—then you decide whether those observations are valid. This is different from having AI generate the use case in the first place.

Creating the actual SRS document involves a lot of technical writing—formatting, organizing, ensuring consistency, catching typos. These are places where AI assistance seems reasonable and doesn't undermine your learning.

## AI response... (3 of 4)

### **The red flags to avoid:**

Don't use AI to avoid doing the hard thinking work. If you prompt ChatGPT to "generate 20 user stories for a student registration system," you've bypassed the elicitation and discovery phase entirely. You won't understand *why* those stories matter or what you're missing.

Similarly, generating a risk analysis without actually thinking through failure modes means you're not developing the judgment that makes requirements engineers valuable.

# AI response... (4 of 4)

## **A practical framing:**

Think of it this way: AI should enhance your capabilities, not replace your decision-making. You should be able to explain and defend every significant requirement and design choice in your SRS using your own reasoning. If you can't do that—if the AI generated something and you just included it—that's a sign you've crossed a line.

Your instructor is ultimately looking for evidence that you can think through a problem systematically and communicate it clearly. That's what the grade should reflect.

# Course project

- Produce a software requirements specification (SRS) for software of your choice.
- Team project (4-5 members on a team).
- Requirements:
  - Rich enough to support 3+ different and involved use cases.
  - Implementation would take 12-20 months.
  - Includes interesting interactions with users.
  - NOT a façade to existing services.
  - Software must have some degree of design originality – the more the better.

# Weekly deliverables

Jan 12	Ethics training; project team information form
Jan 19	Team contract, Lean canvas model, problem fit questions
Jan 26	Stakeholder analysis matrix, personas, feedback on problem-fit interviews
Feb 2	Use case diagram, user stories, feature brainstorming writeup, analysis of problem-fit inter...
Feb 9	Use case diagram (updated), workflow models, quality attributes, team health check
Feb 23	Analysis of requirements prioritization interviews, analysis of quality interviews, risk analysis
Mar 2	Requirements, specifications, interface phenomena, assumptions, scenarios
Mar 9	Domain model
Mar 16	UI sketches, solution-fit hypothesis questions, feedback on questions
Mar 23	Domain model (updated), navigation map, <a href="#">grad lecture</a>
Mar 30	Analysis of solution-fit interviews, cost estimates
Apr 6	Software Requirements Specification

# Weekly Rhythm

Friday	Recorded lectures and weekly ToDos are released. View lectures before the team meeting.
Tuesday	Team meeting – in class, 2:30-4:20. Attendance is required (but don't come if you are sick).
Thursday	Optional team meeting, as needed (decided by you, not me). Peer evaluations and team health checks due.
Friday	Optional team meeting, as needed (decided by you, not me). Quizzes (Jan 30, Feb 27, Mar 20).
Monday	Deliverables due, 8:59pm.

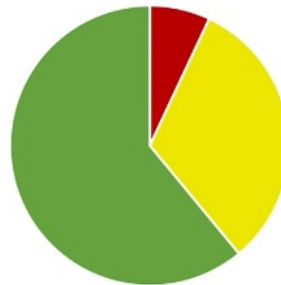
# Lectures

- Recorded by Jo Atlee.
- About 1 hour per week.
- Videos and slides available on the web site.

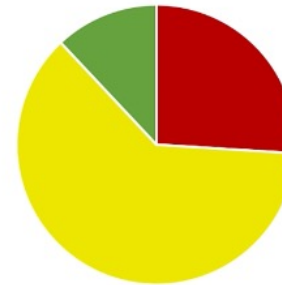
## Standish Group Report - CHAOS



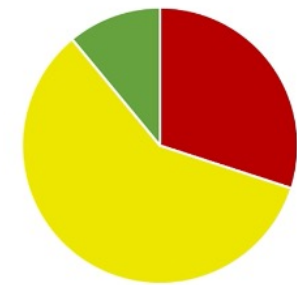
Annual survey of IT executives shows that most software projects are not successful.



Small Projects



Medium Projects



Large Projects

**Success** – satisfactory functionality, on-time, on-budget

**Challenged** – “completed”, but over-budget, over-time, missing key functionality

**Failed** - cancelled

3:57 / 13:42

[https://www.standishgroup.com/sample\\_research\\_files/CHAOSReport2015\\_Final.pdf](https://www.standishgroup.com/sample_research_files/CHAOSReport2015_Final.pdf)

# Team meetings

- Occur in the regularly scheduled Tuesday class time – nearly 2 hours.
- Instructor and TA will be present to answer questions and coach.
- Everyone is expected to attend (unless you're sick).
- Schedule additional meetings if/when needed.
- [Detailed ToDo](#) lists released on Fridays.

# Quizzes & Exams

- Quizzes/midterm exams on three Fridays in the tutorial slot (12:30-2:20).
- Final exam to be scheduled by the Registrar.
- All exams will be in person (unless mandated otherwise).
- Lots of practice questions at Resources → Exams.

# Grad students

- 20 minute lecture (recorded)
- 10-15 page supporting paper
- Topic:
  - Related to requirements engineering
  - Your choice but must be approved by me
  - Worth 15 points on your final grade
- Grad lecture material will be on the final exam for all students!

# Late days

Each team has 10 Late Days.

- Applies to each deliverable (including SRS) except the first.
- Can be applied in (almost) any manner (1 late on each of 10 deliverables, 5 + 2 + 3, etc.)
- At most 5 late days on any one deliverable.

If you plan to use a late day, tell your TA before the deliverable deadline. Give an estimate of how many late days you'll use.

# Grading scheme

- Points, not percentages!
- If we're forced on-line for exams, the weight of the final exam reduces to 10; affected quizzes will be dropped.

<b>Teamwork</b>	CS445 ECE451	CS645
Ethics Training	--	
Team Contract and Surveys	3	
Weekly Team Meetings	10	
Peer Evaluations	4	
Feedback to Other Teams	3	
<b>Deliverables</b>		
Weekly Deliverables	35	
Final SRS Document	20	
<b>Quizzes/Exam</b>		
Quiz 1	5	
Quiz 2	5	
Quiz 3	5	
Final Exam	25	
<b>CS645 only</b>		
Grad Lecture (oral + written)	--	15

# Teamwork!

Initiatives to help teams stay on track:

- Team contract
- Rotating leadership
- Team health-check surveys (3)
- Peer evaluations (6)
- Weekly in-person team meetings, coaches available

Other team stuff:

- Buddy team activities (3)
- Communication on MS Teams
  - Team channel
  - Direct messages

# Microsoft Teams

- Everyone, all term:
  - Announcements
  - Questions
- Short term; getting started
  - Introductions
  - Project Pitches
- Per team channels
  - Team-specific communication (between yourselves)
  - Common storage for files, etc.
- Direct message anyone

# Course text

- Lectures draw from 10-14 books and references.
- *Most* are available for free via the University's subscription to O'Reilly's learning platform.
- See [Resources](#) → [References](#) for instructions to access.

## If you get sick...

- Please, ***stay home*** and get better; protect others.
- Inform your team about what you can and cannot do so they can adjust.
- **Let us know** via official channels. See web site Syllabus → Help & Accommodations (Accommodations for illness...) for more info.

# Course web page

- <https://student.cs.uwaterloo.ca/~cs445>
- <https://student.cs.uwaterloo.ca/~cs445/Winter2026/index.html>
- Syllabus
- Project (peer evaluations)
- Schedule (weekly to-do lists)
- Resources (references, tools, examples, exams)

# Get started!

- After class:
  - Watch lecture videos
  - TCPS2 Ethics Training
  - Review course web site
  - Find a team!
- Right now:
  - Get into groups of about 3. Introduce yourselves. Talk about what you're hoping to get out of the course. If you have a project idea, float it.
  - In X minutes I'll encourage you to find a new group of 3. Rinse and repeat.
  - DO NOT commit to teams now. Watch the Teamwork video first. Use MS Teams to introduce yourselves, pitch projects, and form teams.