SE 1: Software Requirements Specification and Analysis

Lecture 1: Introduction and Administration

Nancy Day, Davor Svetinović

uw.cs.cs445
Welcome

This course is known as . . .

\[
\{ 
\begin{array}{ll}
E&CE 451 \\
CS 445 \\
CS 645 \\
SE463 \\
\end{array}
\}
\]

SE 1

Software Requirements Specification and Analysis

It is the first of three project courses on software engineering:

- **E&CE452/CS446/SE464 (SE 2):** Software Design and Architecture.
Introductions

- Instructors:
  - Nancy Day, DC2335
  - Davor Svetinović, DC3334C

- Project TAs:
  - Shahram Esmaeilsabzali
  - Kristina Hildebrand

Contact information is available on the course web page.
Today’s Agenda

- Introduction
  - What is Software Engineering (SE)?
  - What is Requirements Engineering (RE)?

- Administration
  - Outline, format
  - Textbooks
  - Project, CASE tools
  - Grading Scheme
  - Web pages, newsgroup
What is Software Engineering?

[Software engineering is] the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.


Software maybe only one part of a computer-based system (CBS).
What are “requirements”? 

Requirements are the desired goals or behaviour of the system.

- **What** are we trying to accomplish?
- What does the customer want?

“Software requirements are about writing the right expectations in the right way.” (Lauesen p. ix)
What is a “specification”?

A requirements specification is a description of the proposed behaviour of a computer-based system (CBS).

- What is assumed of the environment in which the CBS operates?
- Given these assumptions, what shall the CBS do?
- What are the boundaries of this CBS?

We will concentrate of software requirements specifications (SRS), but we need to remember that often software is just one part of the system.

The requirements specification can often form a contract between the customer and supplier.
SE Life Cycle: WaterFall Model
Prototyping Model

- REQUIREMENTS
- DESIGN PROTOTYPE
- BUILD PROTOTYPE
- TEST PROTOTYPE

- DOCUMENT REQUIREMENTS
- DESIGN
- CODE
- TEST
- INTEGRATE
Incremental Model
Evolutionary/Iterative

TIME

REQUIREMENTS  DESIGN  CODE  TEST  INTEGRATE

REQUIREMENTS  DESIGN  CODE  TEST  INTEGRATE

REQUIREMENTS  DESIGN  CODE  TEST  INTEGRATE
Determine objectives, alternatives, constraints

Evaluate alternatives;
identify, resolve risks

Risk analysis

Simulations,
Models,
Benchmarks

Develop, verify next level product

Plan next phase

The **Rational Unified Process** (referred to as UP in Larman) is an *evolutionary model* of software development that uses UML.

Each of these phases may contain several iterations. UP recommends having iterations that last only 2-6 weeks.
Rational Unified Process

1. **Inception:**
   - vision
   - scope
   - consider project feasibility
   - commonly-used technique: use cases

2. **Elaboration:**
   - elaboration of requirements (more diagrams)
   - resolution of high risks
   - begin core architecture

3. **Construction:** implementation

4. **Transition:** testing, deployment

Some aspects of requirements, design, implementation, and test are done in most phases.
Project Types

- In-house development
- Product development
- Time-and-material based development
- COTS purchase
- Tender/Contract
Uses of Requirements

CUSTOMER
CERTIFICATION, ETC.

REQUIREMENTS

MAINTENANCE

TESTING/VERIFICATION

DESIGN

IMPLEMENTATION
Requirements Engineering

Elicitation – collect information about requirements

Requirements Analysis – understanding/modeling desired behaviour

Specification – documenting behaviour of proposed software system

Validation – checking whether documented specification accomplishes customer’s requirements
Why write requirements?

According to F. Brooks:

“The hardest single part of building a software system is deciding precisely what to build . . . No other part of the work cripples the resulting system if it is done wrong. No other part is more difficult to rectify later.”
Why write requirements?

- Requirements errors are expensive to fix

<table>
<thead>
<tr>
<th>Stage in Which Error is Detected</th>
<th>Relative Cost to Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>1</td>
</tr>
<tr>
<td>Design</td>
<td>~5</td>
</tr>
<tr>
<td>Coding</td>
<td>~10</td>
</tr>
<tr>
<td>Unit Test</td>
<td>~20</td>
</tr>
<tr>
<td>Acceptance Test</td>
<td>~50</td>
</tr>
<tr>
<td>Maintenance</td>
<td>~200</td>
</tr>
</tbody>
</table>
Why write requirements?

- ~80% of all software errors are requirements errors.
  - These are software errors detected after unit testing – that is, in integration testing, system testing, and after the software is released.
  - Most errors can be traced to unknown, wrong, or misunderstood requirements.
Why write requirements?

- Requirements usually affect large portions of the implementation; they are rarely encapsulated into modules.

- Requirements errors may be fundamental assumptions built into the design or code, e.g., each phone has 1 phone number.

- Expensive requirements errors are often not fixed; they become “features”.

- Devoting 25% of the system development budget on requirements reduces the cost overrun from ~80% to ~ 5%.
Why write requirements?

- If there are no requirements, how will you evaluate the success or failure of the project?
- If there are conflicting requirements, how can you build a product that satisfies all of them?
Types of Requirements

Three major categories:

- **Functional/Data**
  - What is the system supposed to do?
  - Format of input/output
  - Mapping from input to output (possibly over time)

- **Non-functional**
  - Process: standards, delivery, etc.
  - Product: usability, efficiency, reliability, portability
  - External: cost

- **Context/environment**
  - Range of conditions in which the system should operate
Course Goal

Goal: learn how to write a software requirements specification that is,

- clear/communicable/understandable
- unambiguous
- a useful reference/concise
- checkable (complete, consistent)
- testable/verifiable/measurementable
- traceable
- does not describe the implementation

You may have to balance attributes such as completeness and understandability.
Unambiguous Requirements

Example: For a cruise control system, “if the current speed is greater than the set speed, then the system shall decelerate the vehicle to the set speed within 10 seconds”.

Consistent Requirements

Vancouver Airport:

International Departures  U.S. Departures
Course Outline

- Software Requirements Specifications (SRSs)
- Requirements Elicitation
- Requirements Specification Notations
  - UML
  - SDL
  - Algebraic Specifications
  - Temporal Logic
- Non-functional Requirements
- User Interfaces
- Validation (inspections, formal analysis)
- Cost Estimation

Please see the course web page for details of the schedule.
Course Format

CS445:
Lectures:  Tuesday/Thursdays 8:30-9:50 RCH 307
Tutorials:  Thursday 2:30-4:20 RCH 306
Tutorials will provide additional examples and exercises. They can also be used for questions about the project. There will be a tutorial this week. Some tutorials will not be used.
Course Textbooks: Required

Available at Bookstore:


- Course notes (collection of papers and chapters extracted from other texts)

Textbook errata are listed on the course web page. Please let us know about any other errors that you find.

DC Library:

- *IEEE Recommended Practice for SRSs*, 1993 (UW 1378)
Course Textbooks: Additional Refs

Here are other texts that you might find useful:

Course Resources

For (almost) every lecture, there will be:

- possibly lecture notes and extra handouts (which will be available on the course web page)
- **required** readings from the course textbooks

You are responsible for the material in the lectures and required readings on exams.
Course Project

You will write a Software Requirements Specification (SRS) for a small telephone exchange and its associated information system. Doing so will allow you to apply the software engineering principles and techniques discussed in lecture to the problems of eliciting, documenting, and validating the specification of a nontrivial software system.

To give better coverage of various specification techniques and notations, we have designed the course project to incorporate techniques used to develop real-time software for embedded systems and object-oriented techniques used to develop information systems.
Project Documents

The following documents are available on the “Projects” course web page:

- *Overview of the Course Project*
- *Hardware Interface Description*
- *SE1: Course Project*
Project

Terminal

System Console

Server

Your Code

Sockets

Phone Subsystem

Your Code
Phone Interface

IPPhone

...
Project

At the highest level, your system must consist of a Server and several Phone Subsystems. The Server consists of one or more processes and possibly a database. Each Phone Subsystem amounts to a process that uses the API described in the Hardware Interface Description document to interface with a single phone.

You are free to distribute or centralize the responsibilities of your system over your Server processes and the Phone Subsystems as you see fit. However, communication between your Server and each Phone must occur over sockets. The reason for this restriction is that, conceptually, the Phone Subsystem is connected to your system via an IP network. Thus, sockets are the default form of inter-process communication.
Project Requirements Overview

Basic Call Processing:

- call from one phone to another using 4-digit phone numbers.
- convert the 4-digit phone number to an IP address to locate the destination phone.

System Console: Graphical interface to your system. Used by an Administrator.

User Accounts: For someone to use a phone, that phone must be assigned to a user. This user will need a phone number which must be published in someway to allow dialed number translation to occur.
Project Requirements Overview (con’d)

Maintenance:

- Run system tests periodically and display failures to the Administrator.
- Allow the Administrator to limit the number of calls allowed in your system.
- Display hardware status to the Administrator.

Billing:

- Each user is sent a bill showing all charges incurred during a defined billing period. To facilitate this your system needs to keep a record of every established call.
- An Administrator can change the amounts charged for a call by adding new billing plans or editing existing ones.
Project Specification Notations

- Call Processing → UML
- Administrative Functions → UML
  - User Account Management
  - Maintenance
  - Billing
- System Console → Any drawing tool/language, e.g., Visio, xfig, Visual C++, tcl/tk, etc.

You will use CASE tools to produce versions of a SRS for the telephone exchange.
Project Format

- You will work in groups of 3-4 (4 is preferable).
- Each group will work with one TA, who will act as your customer.
- You should meet with the TA preferrable once per week.
- Every project will be different in the details!
- *Usually*, everyone in a group receives the same grade for the project.
# Project Deliverables

<table>
<thead>
<tr>
<th>Deliverable</th>
<th>Worth</th>
<th>Week Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walkthroughs of use cases</td>
<td>5%</td>
<td>5</td>
</tr>
<tr>
<td>First Partial SRS</td>
<td>10%</td>
<td>8</td>
</tr>
<tr>
<td>Walkthroughs</td>
<td>3%</td>
<td>11</td>
</tr>
<tr>
<td>Critique of another group</td>
<td>2%</td>
<td>11</td>
</tr>
<tr>
<td>Final SRS</td>
<td>20%</td>
<td>13</td>
</tr>
</tbody>
</table>

In total, the project is worth 40% of your final grade.

Details on what belongs in each deliverable and the exact due dates are available on the course web page.

The TA who is your customer, will also grade your deliverables.
Project: Lateness Penalties

Deliverables are to be handed in to the Head Project TA (Davor) AND submitted electronically to cs445@student.cs.uwaterloo.ca.

<table>
<thead>
<tr>
<th>Lateness</th>
<th>Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wed 2pm</td>
<td>on time</td>
</tr>
<tr>
<td>Wed 2:15pm</td>
<td>15 mins.</td>
</tr>
<tr>
<td>Wed 3pm</td>
<td>1 hour</td>
</tr>
<tr>
<td>Thu 10am</td>
<td>20 hours</td>
</tr>
</tbody>
</table>

After 20 hours, your deliverable will receive no marks but will be graded in order to give you feedback.
How to Choose a Team

- Groups of size 4 (some may have to be groups of 3)
- Similar work habits
- Similar goals and expectations
- Similar or balanced abilities
- One member with good English and communication skills
- Similar workloads and resources

GOAL:

- to maximize equitable distribution of the workload
- minimize resentment

You will likely be working with the same group for three courses!
Meetings with Your Customer (TA)

- The purpose of these meetings is to find out the requirements of the telephone exchange system.

- You, the team members, should take minutes of your meetings with the TA so you have documented what was agreed to.
  - These minutes should typed up and sent by e-mail to the TA. You and the TA should iterate on the text of these minutes until you agree on the contents.
  - After that, both you and the TA will keep a copy of the minutes with the words “Agreed to by group and TA” added.

- Meetings with your TA later in the term will involve presentations of your requirements and inspections.
Course Software

- Computer-Aided Software Engineering (CASE) tools: specialized software tools to help document specifications using graphical notations; include graphical editors, repository, navigation facilities, rudimentary checking
  - MagicDraw for UML
- Document Processing Tool (\LaTeX, FrameMaker, HTML, Word, etc)
  - must be able to incorporate PostScript files
- Drawing/UI tool: MS-Visio, xfig, tcl/tk, etc.
- Document management tools: CVS, RCS
### Grading Scheme

<table>
<thead>
<tr>
<th>%-age</th>
<th>Item</th>
<th>Week Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>40%</td>
<td>Course Project</td>
<td></td>
</tr>
<tr>
<td>5%</td>
<td>Walkthrough of use cases</td>
<td>5</td>
</tr>
<tr>
<td>10%</td>
<td>First Partial SRS</td>
<td>8</td>
</tr>
<tr>
<td>3%</td>
<td>Walkthroughs</td>
<td>11</td>
</tr>
<tr>
<td>2%</td>
<td>Critique</td>
<td>11</td>
</tr>
<tr>
<td>20%</td>
<td>Final SRS</td>
<td>13</td>
</tr>
<tr>
<td>10%</td>
<td>Midterm Exam</td>
<td>7</td>
</tr>
<tr>
<td>50%</td>
<td>Final Exam</td>
<td></td>
</tr>
<tr>
<td>10%</td>
<td>Graduate tutorial (oral &amp; written)</td>
<td>12</td>
</tr>
</tbody>
</table>

You must pass the final exam to pass the course. The midterm exam (1.5 hours) will be Thursday, February 16th during the tutorial.
Yes, graduate students must do 10% more! They must give a lecture and write a report that is due the last day of class.

Graduate students should send me e-mail regarding selecting a topic.

All students must attend the grad lectures. The content will be on the exam.
Course Web Page


Look here for announcements, updates about the project, solution sets to old exams.
Course Newsgroup and Email

Course newsgroup: uw.cs.cs445

Primarily used for class announcements and questions about lecture material; not for project clarifications — questions about the projects should be directed to the TAs.

E-mail to Instructor and TAs

- cs445@student.cs.uwaterloo.ca (read by instructors)
- nday@cs.uwaterloo.ca (Nancy Day)
  dsvetinovic@swag.uwaterloo.ca (Davor Svetinović)

Note: Please send email from your Waterloo account. Email sent from non-Waterloo email accounts is likely to be filtered as spam.

All email must have the subject line: [cs445] ...
Credits

The lecture slides and other course materials have been prepared by:

- Joanne Atlee (original course designer)
- Dan Berry
- Nancy Day
- Mike Godfrey
- Davor Svetinović
Expectations

- Attend lectures
- Complete readings as per schedule
- Attend all tutorials (very important for project!)
- Regularly read the newsgroup
- Work effectively in groups of four
- Read the project documentation
- Understand all of the project you hand in
Questions (or Complaints)

- If you have a question (or complaint) about a lecture, please direct it to the lecturer. If you have a question about a lecture, it is best to ask it during the lecture when a lot of people may benefit from the answer.
- If you have a project-related question (or complaint), please direct it to your Project TA.
- If you have a project-related question (or complaint), and your project TA cannot answer it satisfactorily, then please direct it to the Davor and Nancy.
First Assignment

Due: Friday 13 Jan 2006 2:00pm

- Form a team.
- Send to cs445@student.cs.uwaterloo.ca e-mail in plain ASCII (no .html, .doc, .pdf, etc. files) with the following for each team member:
  1. name (family“, ” private),
  2. student ID,
  3. UW userid (just one word and no “@site”), and
  4. preferred e-mail address

in that order on one line with a space (and no other punctuation) between each pair of items
First Assignment

Example:

Fine, Larry 234567891 larry larry@threestooges.org
Howard, Curley 345678912 curley curley@threestooges.org
Howard, Moe 123456789 moe moe@threestooges.org

If the message is not in the specified format, it will be returned for correction!

The assignment of groups to project TA will be posted to the newsgroup. Once this is done, it is your responsibility to contact your project TA and arrange a customer and TA meeting.
Summary

- Introduction
  - What is Software Engineering (SE)?
  - What is Requirements Engineering (RE)?

- Administration
  - Outline, format
  - Textbooks
  - Project, CASE tools
  - Grading Scheme
  - Web pages, newsgroup

Required readings: None
Next Lecture

Topic: Requirements Elicitation

Readings:

- Lauesen, Ch. 8 (course pack)
- Berry, *Ignorance* (course pack)

QUESTIONS?