CS645

Software Requirements Specification and Analysis

Winter 2023

Calendar Description

Introduction to the requirements definition phase of software development. Models, notations, and processes for software requirements identification, representation, validation, and analysis. An important component of the course is a group project: the software requirements specification of a large software system.

Required Background

- Knowledge of finite-state machines (e.g., as used in CS 241 or ECE 251)
- Experience with object orientation (e.g., experience with OO programming languages like Java or C#)
- Knowledge of propositional and predicate logic (e.g., as used in CS 245 or ECE 103)

Overall Goals

CS 645 exposes students to the problem of determining, or deciding, *what* a proposed software system should do. That is, it focuses on identifying the problem to be solved by software. In this sense, the course differs from other CS courses, focusing on knowledge and techniques that lead to software solutions. There are some non-technical aspects of the course concerning communication and negotiation. However, most of the course covers technical approaches to the requirements problem, such as notations for modelling and documenting requirements, strategies for prioritizing requirements, and techniques for analyzing and verifying documented requirements.

Learning Outcomes

- Ability to separate requirements from specifications from domain assumptions and argue that a specification + domain assumptions satisfy requirements
- Ability to elicit requirements using different strategies
- Ability to model requirements in a variety of modelling paradigms
- Ability to elicit and document non-functional requirements such that the to-be-developed software can be objectively tested for their satisfaction
- Ability to prioritize requirements using AHP

- Ability to explain and compare different techniques for validating and verifying a requirements document

General Overview of Topics to be Covered

The course covers the four major phases of the software requirements process: elicitation, modelling, analysis (e.g., prioritizing requirements, detecting conflicting requirements), and documentation. The phases are broken down into 12 topics, each of which is 1-3 lectures long. Each topic focuses on a requirement-related problem to be solved when developing a software system or on a particular technique for addressing a requirements-related problem. Listed below are the required topics to be covered in every offering of this course, and the expected number of lecture hours needed to cover the topic. The subjects need not be taught in the order they are listed below. The only real constraint on topic order is that each project-related topic should be covered at least two weeks before the corresponding project deliverable is due.

1. Introduction (1 hour)
Course Philosophy and logistics. Why is requirements analys hard?

2: Requirements Engineering Reference Model (1.5 hours)
Context diagram. Identifying or determining a system's interfaces. Monitorable inputs and controllable outputs. Requirements vs. specifications.

3: Domain Modelling (4.5 hours)
Modelling of the system's environment. Entity-relationship diagrams. UML Class and object diagrams. Constraints on the domain model. OCL

4: Functional Modelling (2 hours)
Functions as a modelling notation. Pre- and post-conditions. Use-case diagrams. Functions over a class diagram.

5: Behavioural Modelling (7 hours)
Modelling dynamic behaviour of a software system. Use cases, scenarios, sequence diagrams. Extended state machine models, including state hierarchy, concurrent regions, communication, activities, history. UML State Machines.

6: Constraint Modelling (2 hours)
Temporal Logic. Specification patterns.

7: Model Integration (1 hours)
Model integration. Composition of models. Consistency among overlapping models. Feature interactions.

8: Requirements Elicitation (4.5 hours)
Stakeholders, sources of requirements, strategies for identifying requirements.

9: Requirements Analysis (4.5 hours)
Requirements triage and prioritization. Cost-benefit analysis. Analytic Hierarchy Process. Conflicts and negotiation. Cost Estimation. Risk Analysis.

10: Quality Requirements (2 hours)
Nonfunctional requirements (e.g., performance, security, useability, maintainability). Fitness criteria. User-interface requirements.

11: Validation and Verification (2 hours)
Walkthroughs, inspections, and technical reviews of requirements documents. Executable specifications. Automated analysis. Verifying that specifications satisfy requirements.

12: Documenting Requirements (2 hours)
Expected contents and organization of a Software Requirements Specification (SRS). Non-traditional means for documenting requirements (e.g., user manuals, test cases).

13 Other Topics (1 hour)
Guest lectures by graduate students enrolled in the graduate-level version of the course.

Course Reference

- Gause and Weinberg, Exploring Requirements: Quality Before Design, Dorset House, 1989.
- Gause and Weinberg, Are Your Lights On? How to Figure Out What the Problem REALLY Is?, Dorset House, 1990.
- Rumbaugh, et.al., Object-Oriented Modeling and Design, Prentice-Hall, 1991.
- Davis, Software Requirements: Objects, Functions, & States, Prentice-Hall, 1993.
- Braek and Haugen, Engineering real-time systems: an object-oriented methodology using SDL, Prentice-Hall, 1993.
- Jackson, Software Requirements and Specification, ACM Press, 1995.
- Ellsberger, Hogrefe, Sarma, SDL, Prentice Hall, 1997.
- Larman, Applying UML and Patterns, Prentice Hall, 2004.
- Kotonya and Sommerville, Requirements Engineering, Wiley, 1998.
- Robertson and Robertson, Mastering the Requirements Process, Addison-Wesley, 1999.
- Rumbaugh, Jacobson, and Booch, The Unified Modeling Language Reference Manual, Reading, MA, 1999.
- Leffingwell and Widrig, Managing Software Requirements, Addison Wesley, 2000.
- Maciaszek, Requirements Analysis and System Design, Addison Wesley, 2001.
- Bray, An Introduction to Requirements Engineering, Addison-Wesley, 2002.
- Robertson and Robertson, Requirements-Led Project Management, Addison-Wesley, 2005.
- Nuseibeh and Zave, Software Requirements and Design: The Work of Michael Jackson, Good Friends, 2010.

- van Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley, 20009.
- Pohl, Requirements Engineering: Fundamentals, Principles, and Techniques, Springer, 2010.
- Brooks, The Design of Design, Addison Wesley, 2010.
- Karl E. Wiegers and Joy Beatty, Software Requirements, 3ed., Microsoft Press, 2013.
- Ash Maurya, Running Lean, 2ed, O'Reilly, 2012.
- Steve Adolph, Paul Bramble, Alistair Cockburn, and Andy Pols, Patterns for Effective Use Cases, Addison-Wesley Proessional, 2002.
- Mike Cohn, User Stories Applied: For Agile Software Development , Addison-Wesley Professional, 2004.
- Richard Banfield, C. Todd Lombardo, and Trace Wax, Design Sprint , O'Reilly Media, Inc., 2015
- Craig Larman, Applying UML and Patterns, 3ed., Prentice Hall, 2004.
- Lenny Delligatti, SysML Distilled: A Brief Guide to the Systems Modeling Language, Addison-Wesley Professional, 2013.
- Steve McConnell, Software Estimation: Demystifying the Black Art, Microsoft Press, 2006.
- Steve McConnell, Rapid Development: Taming Wild Software Schedules, Microsoft Press, 2996.
- Alan Davis, Just Enough Requirements Management: Where Software Development Meets Marketing , Dorset House Publishing, 2005.

Marking Scheme (CS445)

Project	40%
Class-Activities	10%
(During tutorials)	
Final exam	50%
	100%

You need to pass the final exam to pass the course.

Graduate Students

Graduate students in this course (i.e., students enrolled in CS645) are expected to do a small library-research project on top of the normal course work. The deliverables of this project are

- A 25-minute lecture delivered in class
- A 15-20 page written paper, complete with references

The project can be on any topic related to requirements engineering, or possibly a more general software engineering topic. Example topics include requirements notations not covered in class (e.g., Alloy, SCR, SDT), requirement-phase activities (e.g., conformance to privacy laws, security requirements, requirements for product lines, and goal-oriented requirements modelling), and other research areas of software engineering (SEI's CMM, concept analysis).

Topics must have the instructor's approval.

The graduate lecture is to be just that: a lecture on a course-related topic aimed at an audience who is familiar with requirements-engineering activities and techniques. The topic is expected to be an overview, based on *several* sources (journals, conference papers, textbooks, etc.). Electronic copies of graduate-student lectures will be placed on the course web-page for study (the material may be on the final exam).

For graduates' students in CS645, the final exam and the project are scaled to 90% and 10% of your grade is the evaluation of your lecture and report.