University course in software requirements, teacher's notes

This is a plan for a 12 week course in software requirements, based on three hours lectures and discussions per week. During the first half of the course, the students learn the various notations and techniques and apply them in a simple case project. During the second half of the course, they try out their knowledge in a real project. The course is based on Lauesen: Software Requirements - Styles and Techniques, Addison-Wesley 2002. See also the notes in the book, pages xii and xiii.

Course schedule

Week 1. Chapter 1. Basic concepts and project types.

During lectures, discuss some of the issues in the exercise chapter (Chapter 16), for instance 1.1 and 1.4 to 1.7.

Run exercise 1.9 as an in-class exercise (study pieces of existing requirements specifications).

Home work: Exercise 1.8 (preferably case study 16.1, Journal Circulation, or 16.2, Ticket Machine).

Week 2. Chapter 2 and parts of Chapter 3. Data requirements and functional requirements.

If students have preliminary data modelling expertise, Chapter 2 should be handled briefly. Otherwise two half weeks should be dedicated to it (not included in this plan).

Chapter 3 should be handled with emphasis on sections 3.6 to 3.12 (tasks and use cases). The task descriptions seem easy, but students (and many developers) fail to define good tasks. They tend to make each function in the system a separate task (the typical use-case approach). I deal with the problem in this way: After explaining the task concept (page 93), I invite the students to list the tasks for an email system - the client side only. I write down their suggestions on the blackboard. They come up with a long list of tasks: compose a mail, send a mail, read a mail, file a mail, delete a mail, etc. I point out that this would give a very long requirements document. Analysts would feel very productive, but would miss the whole point because the system is not going to support these tasks in isolation. Nobody would start their email system just to file a mail. Instead the system has to support the complex task of receiving a whole bunch of mails, delete some, file some, reply to some, etc. Each of the student's "tasks" would become an optional subtask, and the task document would shrink from 10 pages to one. My book *User Interface Design - A Software Engineering Perspective* discusses this example in detail (Chapter 11).

During lectures, discuss some of the issues in Chapter 16.

Home work: 2.4 and 3.7 (preferably case study 16.1, Journal Circulation, or 16.2, Ticket Machine). Note: Exercise 3.7, question c, says that high-level tasks are "artificial" for the journal circulation. All wrong! The life cycle of a company subscription and the life cycle of a specific journal copy are good high-level tasks. See the suggested solution available for teachers.

Week 3. The rest of Chapter 3 and Chapter 4. Functional details.

Depending on how much UML is emphasized, one or two weeks may be spent on this (not included in this plan).

Home work: 4.5 (preferably case study 16.1, Journal Circulation, or 16.2, Ticket Machine).

Industry projects: At this point students should start looking for a project in industry. They should try to establish contacts on their own and develop skills in finding the right people. This is an important skill in requirements engineering. Most companies have some needs - large or small - that they would like to support with IT. However, they never get down to writing the requirements. The students might sell themselves as "an opportunity to define the requirements". The project might be anything, for instance a medium sized company considering an administrative system, a government department wanting a web site, the

university wanting a student administration system, the tennis club wanting a booking system. Remember that there is no promise of delivering the system, only a promise of writing the requirements specification.

Week 4. Chapter 6. Quality requirements. (Skip Chapter 5 for now.)

Home work: 6.9 (preferably case study 16.1, Journal Circulation, or 16.2, Ticket Machine). Exercise 6.11 may be a good supplement.

Industry project: Monitor that the students make progress in establishing contacts.

Week 5. Chapter 7. Requirements in the product life cycle.
Run exercise 7.5 as an in-class exercise.

Home work:
Exercise 7.6 (not in the book): For the Journal Circulation project or the Ticket Machine, write a plan for design, verification, and delivery of the product. In particular, describe in some detail how you would verify each requirement.

Week 6. Chapter 8. Elicitation.
Run exercise 8.5 as an in-class exercise (for instance case study 16.5). If there are more than 13 students, make two or more focus groups.

Home work: 8.4 (preferably the industry project, but might also be case study 16.1, Journal Circulation, or 16.2, Ticket Machine).

Industry project: Check that the projects are in a state where an elicitation plan can be made (exercise 8.4).

Week 7. Chapter 9. Checking and validation.
Homework: 9.6 (preferably the student's own requirements for the case project, but the Midland Hospital spec might be used too as explained in the exercise text).

Industry project: From this week and on, arrange for two or more project teams to present some results of their industry project (preferably some requirements), their concerns and their plans for the next weeks. The audience (or a pre-selected other team) should play the role of customers asking questions about the requirements or the business consequences. Every now and then they may change to the role of supportive sparring partners. (See the hints on page xiii of the book.)

For the rest of the weeks, the pattern might be a supplementary topic followed by student presentations of their industry projects, in the same way as in week 7.

Suggestions for supplementary topics

1. Contracts and law, for instance as one or two guest lectures by a lawyer working in the IT area.
2. Chapter 5 and Lauesen (2004): COTS Tenders and Integration Requirements (available from Lauesen's home page)
3. User interface design, for instance based on Chapters 5-9 in Lauesen: User Interface Design - A Software Engineering Perspective. These chapters show how to get from domain-level requirements to a good user interface, i.e. a prototype with process descriptions for all buttons and controls (mini-specs).
4. Selected papers, for instance picked from the book's reference list. For each paper, discuss how it relates to the industry projects.

Exam

Oral exam based on the student's requirements specification in the industry project.