

## University course in user interface design, teacher's notes

This is a plan for a 12 week course in user interface design. The plan is the result of several year's iterative improvement of the teaching method. The course is based on Lauesen: User Interface Design - A Software Engineering Perspective, Addison-Wesley, 2005. See also the notes in the book, pages xiv and xv, and the author's home page, <http://www.itu.dk/people/slauesen/>. You may contact the author at [slauesen@itu.dk](mailto:slauesen@itu.dk).

Here is the weekly teaching pattern:

- Three hours lecture with discussions and drills. The students are not expected to read the book chapters before the lecture. The lecture is based on explaining and discussing the examples in the book figures (available as Power Point slides). There are very few traditional bullet-point slides.
- Two hours in-class exercise where the students start on their homework exercise under guidance.
- Homework where they complete their exercise and read the week's parts of the book.

Our classes are 30-50 students, which allows for some fine in-class discussions. We have good experience with letting the students work in groups if they like, but insist on them writing solutions individually. The step from solving the problem in a group to writing down the result is quite big. The home work is optional, and each week around 50% of the students voluntarily turn in a solution. They know that the written exam questions are similar to the weekly exercises, and if they don't do a reasonable amount of homework, they have little chance to pass the exam.

### Course schedule

Week 1. Chapter 1. Usability.

During lectures, make sure that you cover the basics of usability testing as it is needed for the homework. Also prepare the homework by discussing the plan for the usability test and examples of good test tasks. Find a suitable web site for the students to test. In order to give the students constructive feedback and compare their solutions, it is best that all of them test the same site.

Home work: Exercise 1.1 (usability requirements to an existing web site) and 1.2 (carry out a usability test of the site to see whether it meets the requirements).

Week 2. Chapter 2 (prototyping and iterative design, 2.5 may be handled later). Sections 3.1 to 3.3 (data presentation).

The previous exercise dealt with usability measurement and usability testing of an existing system. This week the students develop a paper mockup, based on their intuition. Next they test it for usability. The aim is to make them experience that it can be done quite easily and gives a lot of surprising insights.

During lectures, show them how to test a mockup for usability. Use an existing paper mockup and run a usability test of it while the students observe (*don't* use IT students as test subjects). I use the hotel system from the book (hand-drawn mockup) on plastic overheads so that everybody can watch what goes on. I invite a suitable test person, for instance a secretary with little IT experience, or a humanities student. The test person uses an overhead marker as mouse and keyboard. After a minute, the test person feels confident with this setup. It is best to use water-soluble markers so that "erroneous typing" can be erased. The students record usability problems and procedural errors made by the tester (me). Each semester I run this usability test with a new person - and I am still a bit nervous every time. We find largely the same usability problems each time, but there are also individual differences.

Home work: Exercise 2.1 (make a prototype for the work-hour registration) and 2.3 (usability test of it; skip question d). If time allows, ask them to do exercise 2.2 (heuristic evaluation) and question d of exercise 2.3 (comparing heuristic evaluation and usability testing).

Week 3. Sections 16.1 to 16.5 (data modelling). Sections 3.4 to 3.5 (data presentation).

Most students find data modelling difficult, but after struggling with it for a couple of weeks, nearly everybody masters it and finds it very useful. It takes time. Don't rush through it, but take a bit a week. Use roughly half of the time on data presentation, which is much more relaxing.

I have developed a way to "rescue" students who make their data models all wrong. Usually the problem is some detail they have misunderstood or do wrongly. I invite them on an individual basis for a think-aloud test with me. They get a small data example and try to model it while thinking aloud. (Often I take a real invoice and let them create a model of all the data they see on the invoice.) I listen and sometimes ask a gentle question about what they try to do. Within half an hour I have noticed one or a few basic misunderstandings. I then explain my observations to the student, who mostly is quite surprised. *Oh!* he says, *that is what I do!* With some more help, he is now able to model data - at least on an acceptable level. The misunderstandings are quite individual. One example is asking the cardinality questions in the wrong way: *One A is related to more B's. More B's are related to one A. Thus we have a 1:m relation.* This misunderstanding is so common that it is explained in the book. Another is placing all the boxes on a line, as if they had something to do with a sequence of things to do. A third is interpreting the same box as a class of types in one connection and as a class of objects in another connection (the *service* box in the hotel system is an example).

Home work: Exercises 16.1 and 16.2. You may also ask for 3.1 (gestalt laws applied on a web site).

Week 4. Sections 16.6 to 16.12 (data modelling, you may skip 16.10). Sections 3.6 to 3.7 (the rest of data presentation).

Take the opportunity to explain how a data model relates to the data presentation. The hotel mockup (page 51) shows simple and medium complex examples that can easily be related to the hotel data model on page 497. The planning screens and data model on pages 99-101 provide a complex example.

Usually everybody is exhausted at this stage of data modelling and data presentation. Use the Elephant experiment as a refresher - and to illustrate the surprising law of parallel movement (see the web site).

Home work: Exercise 16.3 (library) and 16.4 (marriage). Don't use sub-classing for the library, but use it for *marriage* as a bit of practice. As explained in the book, sub-classing is rarely useful when modelling the real world.

Exercise 3.3 (data presentation for flight collision). Let one or two groups present their solution as an understandability test during the next lecture. They must *not* explain anything about their screen (very hard to keep them away from this). They should invite the other students to explain what they believe the screen shows. This is a simplified usability test that deals only with understandability of the data presentation.

Week 5. Chapter 4 (mental models and interface design). Chapter 5 (domain description; parts of section 5.5 may be handled later).

Usually, I start the mental model part by illustrating one of Piaget's experiments with 3 and 6 month old babies. The 3 month baby watches a teddy bear on an electrical train, but loses interest when it disappears behind a screen, while the 6 month baby expects it to reappear. The 6 month baby has developed "object permanence".

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 145), 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 task 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. (Chapter 11 of the book shows this example in detail.)

Home work: Exercise 5.1 (design project, domain analysis). Choose the COP project (Circulation of Periodicals, pp. 558-559). Suggested solutions available for teachers.

Week 6. Chapter 6 (virtual windows).

Virtual windows seem easy but are hard to make right. It takes time and we practice several times during the next weeks. The key part during the first lecture is sections 6.1, 6.2, 6.3 and 6.7.

The chapter also introduces the *defect* list, to the surprise of many students. *Do you really mean that we have to list the defects and later decide which ones to repair- and even ship products with defects?* they ask. *Yes, it is good professional practice.* This is an opportunity to discuss professional practice versus academic ideals.

Home work: Exercise 6.1 (virtual windows, COP project). You may omit questions c, e and f. Suggested solutions available for teachers. Remind the students that it is alright to make the windows with pencil and eraser. It is even a professional way to work.

Week 7. Recap and reflection.

At this point in time, the students have produced their first virtual windows - not too successful - like the first programs students make. Before continuing with functional design, it is necessary to review what they have done until now. In this lecture I skip the overheads entirely and use only blackboard. Usually I let the students control the lecture through their questions and concerns. Issues to review and discuss are typically: usability factors versus usability requirements; good and bad tasks; domain versus system; the relationship between data model and virtual windows; various ways to present data for a 1:m relation, an m:m relation, and an enumeration type.

You need the rest of the week to give the students individual feedback on their first virtual windows. Meanwhile let them do an exam exercise as home work.

Home work: Written exam, January 2001 (pp. 567-570). This exam can be made on the basis of what the students have learned until now. It contains two exercises: (1) improving the development methods in a software company. (2) designing the user interface to a knowledge management system. Warn the students that since this is their first exam exercise, they will spend more time on it than the four hours set off for it during the real exam. Arrange for one of the students to take an "oral defence" of his solution during the next lecture.

Week 8. Oral defence of an exam solution. Chapter 7 (function design; sections 7.3 and 7.9 may be handled later).

Oral defence: Hand out a copy of the "defence" solution to all the students and give them around 30 minutes to read it. Next let the students act as assessors and ask questions to the defender. Reject questions of the type "I would rather do it this way . . .". This doesn't help the defender see what is wrong with his solution. The discussions often turn into something where the teacher has to recap a bit from the book.

The whole defence can easily take half of the lecture time. Make sure you cover the important parts of Chapter 7 in order for the homework to be made.

Home work: Exercise 7.1 (function design, COP project). You may omit questions c and f.

Week 9. Chapters 8 and 9 (prototyping, defect correction, virtual-window design versus object-oriented approaches). Start programming a complex user interface (only for students who have never programmed a real user interface).

This lecture is of the discussion type. According to the pattern above, the homework should be to transform the COP design to a prototype, usability test it, etc. In practice this is trivial for

the students at this point, and they are not motivated. Their virtual windows and function design are very close to a prototype, but they realize that it will not be a good one since it is their first design. So the students would rather do the whole thing on a new project.

Exercise after the lecture: On our courses, some students have experience with user interface programming. They can readily apply the virtual windows method. Other students are learning programming while they take the user interface design course. At this point they are able to appreciate the object-oriented aspects of the virtual window method, and they are interested in programming a real, complex user interface. We let them practice with the Access booklet (available on the web site).

Home work: Read one or two scientific papers and prepare for a discussion next time. We often use Molich and Nielsen (1990): Improving a human-computer dialogue, versus Bailey et al. (1992): Usability testing vs. heuristic evaluation. The two papers give rise to an interesting discussion about methods, comparisons, and "truth".

Week 10. Discussion of the papers. Maybe lecture on book sections skipped during the previous sections.

Exercise after the lecture: More programming of a user interface.

Home work: One of the earlier written exams. You can use the one from January 2000 (pp. 563-566). I have many other exam exercises and will gradually make them available in English. Arrange for one of the students to take an "oral defence" of his solution during the next lecture.

Week 11. Oral defence of the exam solution. Chapter 12 (IT support of a new system). Other things to recap.

Exercise after the lecture: More programming of a user interface.

Home work: Read yet another exam exercise, but don't try to make the solution. (At this time of the semester, our students are finishing some demanding projects on other courses. They are not willing to spend much time on this course. We don't try to oppose nature ;-)

Week 12. Solution of the exam exercise.

We jointly develop the solution at the blackboard. This takes at least 4 hours and leads to many interesting discussions.

### **Exam**

Written exam similar to the two shown in the book. In our exams, students use pencil and eraser to maximize their speed. Few students are able to make reasonable virtual windows as fast with a computer.