Using a System of Tutorials and Groups to Increase Feedback and Teach User Interface Design

Tim Wright **James Noble** **Stephen Marshall**

School of Mathematics, Statistics and Computer Science
Victoria University of Wellington,
PO Box 600, Wellington, New Zealand,
Email: {tim,kjx,marshall}@mcs.vuw.ac.nz

Abstract

Offering a new course on user-interface design presents several problems for a Computer Science department. As well as building student competency in user interface design and user interface evaluation, the course must give timely feedback to students while ensuring that staff and student workload remains manageable. We developed a course that uses a novel system of tutorials and group work to increase feedback about design to students and control both staff and student workload. We plan to extend this approach to other courses where design and group work are important components.

Keywords: User Interface Design, Usage-Centred Design, Project Work, Group Work.

1 Introduction

Many undergraduate Computer Science, Software Engineering, and Informatics programs are starting to include (or already include) courses teaching Human Computer Interaction (HCI) or User Interface Design (UID). These courses are increasingly important for graduates wishing to work in industry where they can expect to spend a substantial part of their employed time designing, implementing, and testing user interfaces — graduates now work in an environment where up to 50% of a software development budget and time is spend on user interface design and testing (Myers & Rosson 1992)).

Our institution recently started offering a Bachelor of Information Technology (BIT). This degree has been popular. Students enrolled in a BIT can major in four areas of information technology, one of which is Software Engineering. Part of the software engineering major mandated teaching a new course on user interface design. The course had several design considerations: as it is part of a software engineering major, we wanted to teach interface design methodologies and as it is a course on design, we wanted to provide students with regular design feedback from teaching and tutoring staff. In this way, the course draws aspects of pedagogy from design, human-computer interaction, and software engineering.

Researchers and industry have developed a wide range of user interface development methodologies. These range from Usability Engineering, where the user's tasks and the user interface are considered at every stage of the software development process (Nielsen 1993), to Goal Directed Design, where the user's goals drive the interface development (Cooper & Reimann 2003), and Interaction Design, which concentrates on the user's needs (Preece, Rogers & Sharp 2002). Usage Centred Design (UCD), the development methodology we chose to teach, is a software development methodology driven by users and the users' tasks (Constantine & Lockwood 1999).

Teaching all methodologies to students, as well as giving students in-depth practical experience developing user interfaces, would take far more than a single trimester third year course. We also believe that it would not be possible to give students regular feedback during the design process to encourage incremental improvements. Rather than give students a broad overview of available user-interface development methodologies, we elected to teach them one user interface methodology in detail: usage centred design. This choice meant that is was possible to design the course assessment using a constructionist view: the projects for the course could closely model the entire usage-centred design development methodology and would let students construct knowledge by carrying out the methodology. Students would also become familiar enough with UCD to understand its strengths and weaknesses and be able to apply this knowledge to user interface design in industry.

To help students gain detailed experience with UCD we used group projects where 3 to 6 students would work together. To increase the amount of feedback staff could give students we used a system of groups and tutorials where each tutorial acted as a design critique session. In these tutorials students would present work to the class and receive feedback from both the class and from teaching staff. Additionally, time was allocated for students to prepare work for presentation. Because students interacted with their group, with staff, and with other members of the class in an organised and time-constrained environment we noticed a both reduction in the number of group problems and an increase in the amount of feedback we could give groups.

In this paper we report on our experience of creating a new course using usage centred design to teach user-interface design using this organisation of groups and tutorials. We begin in Section 2 by describing the development methodology we adopted (usage-centred design) as well as the more traditional approaches to teaching human-computer interaction and user-interface design. Sections 3 and 4 present our design for the course and the structure of the practical work while Section 5 describes the student and staff experience with the course. Section 6 relates this course to other literature on teaching HCI courses and teaching large group projects. Finally, Section 7 reflects on the experience of delivering this course and presents our conclusions.

2 Usage Centred Design

Usage Centred Design (UCD), as proposed by Constantine & Lockwood (1999), is a user-interface development methodology driven by a few abstract models of users, their tasks, and interface content. Usage centred design differs from user-centred design, where users are involved in every stage of interface design, by involving users in limited and highly focused ways.

UCD provides a five step interface design methodology where each step produces a new model derived from the previous model. The five steps and related models are:

1. Identify user roles and the relationship between roles. Produce a user role map and user role descriptions.
2. Identify user tasks and write essential use cases. Produce a use case diagram and essential use cases.
3. Identify interaction requirements and screen layout for each screen. Produce a content model and a canonical abstract prototype.
4. Implement the interface.
5. Test and fix the interface.

UCD fits well in a university course because individual models can be submitted for assessment and feedback from the assessment can be used to both improve the submitted model and the next model for the next piece of assessment. Additionally, as users are only needed in limited and highly focused ways, users can be provided for students to interview at essential points during the process. These points are when students are identifying user roles and tasks.

3 User Interface Design (COMP311)

COMP311 is a third year user-interface design course being taught as part of a software engineering major. It has a total student workload expectation of 10-12 hours per week throughout the course. To help it integrate with other software engineering courses, we decided to teach a software engineering approach to user interface design. Barnes & Leventhal (2001) have previously reported this approach is successful.

The typical student taking our course has experience in Java, C++, and has done a group project (the only prerequisite course, COMP201, contains a significant amount of group work). We expect that most of the students are in the last trimester of their degree, however, in 2004, about 25% of the students were in their second year, and some postgraduate students were also taking the course. Also, in 2004 the course was larger than we expected: we anticipated about 70-80 students would enrol, however 120 students enrolled, making COMP311 the largest third year course in the Computer Science Group at our institution.

For the practical component of the course, we wanted students to follow the entire UCD methodology from identifying user roles and tasks to testing their interface and fixing any problems the test found. The challenge we faced designing a course to teach UCD was how to organise the course to give the students appropriate and timely feedback while not spending all our time marking, and, importantly, not forcing the students to perform an inordinate amount of work. The mechanisms we used follow.

3.1 Group Work

Early in course design we elected to make all practical work done in groups. There are several reasons for this decision: group work skills are increasingly an important part of computer science degrees; user-interface design is done in groups in industry; students have previous group experience; a group can experience more of the UCD methodology than an individual and; students in a group can give each other feedback on design decisions. Some universities are using groupwork as a learning tool in foundation year (Drury, Kay & Losberg 2003).

3.2 Open Ended Group Projects

User interface design is a complex iterative activity where there is no right answer but rather designers try to achieve the best compromise between contradictory design requirements. This complex iterative activity is best supported by an open ended group project. Newman, Daniels & Faulkner (2003) argue that open ended group projects aid learning:

> "Putting students into a 'team' context and facing them with a problem for which there is no 'right' answer helps them think about what they have been taught, internalise some of the material that has been presented in lectures and laboratories, and develop the 'soft', people oriented, skills which are essential in the 'real world'."

COMP311's practical assessment is based around a term long open ended group project where students build a computer interface for an imaginary small company.

3.3 Two hour tutorials

We decided to offer four two-hour tutorials weekly and require all members of a group must attend the same tutorial. We had this requirement so staff could give an entire group feedback on their design. In this way, the tutorial acted pedagogically as a design critique session where groups presented the models or designs they were currently working on and received feedback from both staff and other students in different groups.

3.4 Low Fidelity Models

We required students to present and submit hand drawn models rather than using case tools and were encouraged to use large pieces of paper, post-it notes, and encouraged to make draft versions messy instead of neat. Rettig (1994) and Snyder (2003) describe how this approach helps software designers concentrate on the design rather than focusing on layout and the tool. These models were presented and critiqued in tutorials using a document projector.

3.5 Projects and Milestones

We decomposed the UCD methodology into two projects. Each was done in the same groups and the first project created a design for the second project to implement and test. To ensure we could give timely feedback to students, we set milestones for each project. Milestones were marked on a pass or fail basis in a tutorial. Passing the milestone means that a group must have reached a certain point in the UCD methodology. In the milestone tutorials, groups presented their milestones and received feedback from teaching staff as well as other groups.

3.6 Just In Time

Like Rosson, Carroll & Rodi (2004), we presented "relevant HCI concepts and theories . . . 'just in time' to support the semester-long group projects." We found there was

slight time pressure on the groups and staff to understand the material quickly so that groups could integrate the lecture material into their projects in order to receive feedback on their models in their next tutorial. Nevertheless, most groups successfully learnt a concept one week, presented work based on that concept the next week, and integrated feedback into their model for the following week.

The assessment timetable is shown in Table 1 and the course content schedule is shown in Table 3.

3.7 Usability Evaluations

Evaluating a user interface has become an essential part of the user interface design process. There are many types of evaluations of user interface, ranging from heuristic evaluations, where a group of experts examines a user interface for flaws (Nielsen 1993), to ethnographic studies of user interfaces, where experts record users interacting with a user interface and the user's environment in the user's environment (Suchman 1987).

In COMP311 we required all groups conduct heuristic evaluations of a different group's user interface and to give the results of the evaluation to the group whose interface they evaluated. Each group would then modify their own interface based on the feedback they received. Additionally, some groups were offered the option of carrying out a usability evaluation based on performance in the first project and in the milestones. Usability evaluations are more intensive than heuristic evaluations and typically find different types of problems.

3.8 Weekly Journals

We required each student submit a weekly journal saying what they had done that week with their group. With the journal they had to rate the amount and quality of work done by other group members. We had this requirement so that we could easily determine what has happened in case of group disfunction.

4 Practical Work and Assessment

In the course we wanted students to gain experience with the entire usage-centred design methodology. To do this we designed the assessment to take one problem domain (design computer interface for a pizza company) and do the methodology on that problem. This had 8 steps:

Project 1: User Requirements and Abstract Design

1. Examine potential user roles and user tasks
2. Interview a user to clarify understanding
3. Describe user roles and create a user role-map
4. Describe user tasks using essential use cases and create a use-case map
5. Create a canonical abstract prototype based on the use cases

Project 2: Implementation and Evaluation

6. Create a sample implementation of the computer system
7. Evaluate the user interface
8. Fix the errors uncovered during the evaluation

We decomposed the UCD methodology into two projects. In the first project (abstract design) students had to carry out the first five steps of the methodology. The milestone for the first project was demonstrating that they had completed the first three steps. This milestone was presented two and a half weeks before the project was due. In the second project (implementation), students had to implement their abstract design, test another groups design, and fix their implementation based on the results of another group's test. The first milestone for the second project was to demonstrate that they had a functioning interface and the second milestone was to demonstrate that they had tested another group's interface. The two milestones for project two ensured that there were interfaces for different groups to evaluate and that groups got the results of the interface test back in sufficient time to fix their own user interface.

Week 1	
Week 2	
Week 3	User interview in lecture
Week 4	Milestone due (M1.1)
Week 5	
Week 6	Project 1 due
Two-week mid-semester break	
Week 7	Milestone due (M2.1)
Week 8	Test Interface
Week 9	Milestone due (M2.2)
Week 10	
Week 11	
Week 12	Project 2 due

Table 1: Timetable for Assessment

Milestone 1.1	5%
Project 1	15%
Milestone 2.1	5%
Milestone 2.2	5%
Project 2	15%
Weekly Journal Files	5%
Exam	50%

Table 2: Assessment items and ratings. Milestones are marked as either 'pass' or 'fail' in a tutorial.

5 Experience

As we write this paper, the course has finished and we are currently marking exams. At this stage we have had positive experiences and findings relating to how the groups, tutorials, and milestones interact.

5.1 Groups and Tutorials

We found the organisation of groups and tutorials was very successful. As well as tutorials acting as feedback sessions we noticed several other roles tutorials were playing. First, they were acting as a proxy for a group meeting for some groups so any group problems came to light early. Second, we found that problem students who did not want to participate in group work did not attend tutorials. We stated in the course outline that tutorial attendance was mandatory and regularly failing to attend tutorials would result on removal from a group, making it easy to remove students from their groups well before assignment deadlines. Third, the teaching staff present in a tutorial could act as a mentor for a group and give them advice on group structure as well as feedback on group work.

Organising the tutorials and assigning people to groups was more work than we expected due to the constraint that all members of a group must attend the same tutorial: it took much of the first three weeks of the course.

Week	Tuesday Lecture	Friday Lecture
Week 1	Introduction (ch 1)	Eclipse
Week 2	Usability (ch 2,3)	User Modelling (ch 4)
Week 3	User Tasks (ch 5)	User Interview for Project 1
Week 4	Essential Use Cases (ch 5)	Essential Use Cases (ch 5)
Week 5	Content Models (ch 6)	Canonical Abstract Prototypes
Week 6	Interaction Design (ch 7)	Visual Design (ch 8)
Mid-trimester break		
Week 7	Paper Prototypes (ch 10)	Heuristic Evaluation (ch 16)
Week 8	Usability (ch 1, 2, 3 revisited)	Usability Metrics (ch 17)
Week 9	Usability Testing (ch 18)	Other Testing Methods (ch 18)
Week 10	Help (ch 11)	Novices versus Experts (ch 12)
Week 11	Advanced Interfaces (ch 9, 14)	Web-based Interfaces (ch 14)
Week 12	Review	Exam Review

Table 3: Course content and delivery schedule. Chapters are related readings from the course textbook: Software for Use (Constantine & Lockwood, 1999).

In retrospect, the additional cost of organising timeslots was justified by the increase in group contact time. Having all members of a group in the same room at the same time where they can interact with a staff member created successful, dynamic, interactive tutorials.

An additional advantage was that unlike courses where the groups depended on individual mentors (Noble, Marshall, Marshall & Biddle 2004) all of the groups received feedback from a smaller group of teaching staff. This allowed for greater consistency of feedback and the ability to identify potential problems within the groups much sooner.

Despite this additional feedback, there were still some dysfunctional groups. Three students ($2\frac{1}{2}$% of enrolled students) were removed early from groups because they did not attend tutorials. Two groups ($8\frac{1}{3}$%) had problems large enough that they came and talked to the lecturer. The problems in one of these groups was traced to a personality conflict between two key members. The other group came to see the lecturer after all practical assessment was submitted. Their problems were traced to two group members free riding on the other members' work. Tracing the group members problems was done by talking to the groups as well as reading the weekly journals.

5.2 Milestones

In COMP311, each project contained one or two milestones. The milestones were due between two and four weeks before the project deadline and groups had to present their milestone in their tutorial. The milestones were marked on a pass or fail basis. Having milestones ensured that groups had started the project early and let us give timely feedback that could be incorporated into the final marked version of their project.

The milestones were successful — we gave the students precise requirements of what they needed to present to pass the milestone so all groups who presented in a tutorial passed the milestone. Additionally, groups appeared to enjoy getting feedback on their own work and examining what other groups had done.

An example milestone is shown in Figure 1.

5.3 Feedback From Students

We had two mechanisms to elicit feedback from students. The first was to require students to submit a short reflective discussion on your groups' experiences' with the project. This discussion submitted with the project (and was marked) and was limited to a page. Generally the

In this project you will design (but not implement) a system to help people in a busy pizza delivery business keep track of orders, deliveries, and customers. Some hints to start include:

- Different user roles need different pieces of information. For example, the information requirements of a manager are different to a pizza delivery person.
- Users' tasks are influenced by the tasks of several classes of ersatz users: customers ordering pizzas by phone, fax, email, and through the web.
- You should consider special orders/cases. For example, when a customer receives an incorrect order the business could send them a free pizza as soon as possible. You should think about other special cases.
- The pizza company wants to focus on repeat business, particularly from corporate businesses

Table 4: Specification of program domain.

A satisfactory milestone will contain this information:

- A cover page containing course code and name, team name, tutorial time, group members and their student ID numbers, date, and the name of this milestone.
- A user role map containing eight user roles and identifying focal roles,
- A structured role model for the two most important focal roles, and
- A list of ten candidate user tasks that support the focal roles.

Table 5: List of requirements to pass milestone one for project one (user models and roles). This milestone was presented in a tutorial in week 4.

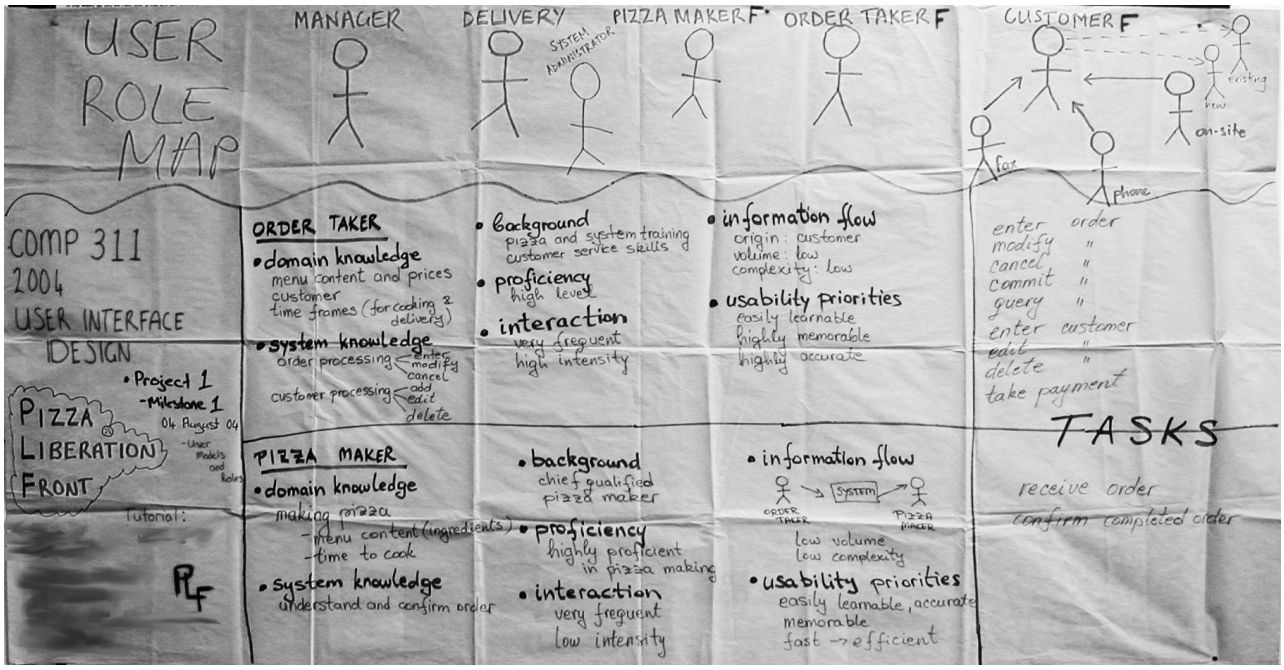

Figure 1: One group's first milestone. This milestone was presented in a tutorial in week 4. The group's names have been obscured to increase student privacy.

feedback we received through this mechanism was positive:

> "We have come to realise that usability and user interface design is not all about pretty images and making something look pretty. It's about supporting and helping the user to carry out there [sic] tasks efficiently, accurately, completely and effectively"

> "The peer evaluation on our interface that was carried out really helped us get a different perspective on what we needed to work on in the overall design. It can be very easy to become too narrow-minded in approach when building a user interface and sometimes, we all . . . can be blinded to certain elements in the process."

The other mechanism was a formal evaluation of tutorials by the University's Teaching Development Centre (UTDC) . The feedback from this mechanism identified several problems with the way we ran tutorials. The major problem was that some students found that the tutorials seemed too long and suggested that groups should work on different projects:

> Because everyone was doing exactly the same thing, it often became very repetitive and quite boring. Maybe if there were a greater number of possible interfaces to design, it would be more interesting to hear what others had to say. As it was, the main reason I went to the tut was to do our presentation, and get comments; the other groups did everything pretty similar, so there wasn't much to learn from their presentations.

On the whole, we feel encouraged by the feedback from the students and plan to use this structure again:

> The whole getting your group to go to the same tutorial (and attend) did make communicatin [sic] issues across to other members easier. Group work was made better for it.

6 Related Work

Much work has examined how groups of students can carry out course-long group projects. First, a comprehensive study of groups performing semester long projects in an HCI course was published by Hartfield, Winograd & Bennett (1992). They identified four problems: scheduling group meetings, time demands on mentors, resource costs, and individual student assessment. In this course we found solutions to all these problems:

Scheduling: Hartfield et al. (1992) wrote "One perennial problem was scheduling group meetings. Finding a time where the group (typically 5 members) can all meet can be a problem, especially since many students also had a part-time job." We avoided this problem by requiring that all members of a group attend the same tutorial.

Time and Availability Demands: Hartfield et al. (1992) found this was a problem for group mentors. We did not use group mentors in COMP311, but found that teaching staff in a tutorial played a similar role.

Resource Costs: Hartfield et al. (1992) wrote "the amount of core 'teaching effort' was possible only with the additional resources provided by the NSF grant and by the fact that one of the instructors was a non-paid lecturer, whose services were volunteered by his company." We avoided this problem by organising tutorials where students could get feedback from other students and teaching staff. We also instituted a series of milestones to ensure we could provide timely feedback to students.

Individual Student Assessment: In similar courses, the group project counted for two thirds of their mark (Hartfield et al. 1992). We reduced the effect of the group project by having it worth 30% of a students mark. Additionally, students had to rank their other group members each week. This data was used during marking to assess skill individually even with the group projects.

Second, Leventhal, Barnes & Chao (2004) describe that the major challenge they had when implementing a term long group project in a Usability Engineering course was specifying the project requirements. They identified four methods to specifying project requirements:

1. Derive user tasks from an existing piece of software
2. Present a list of functions the user interface could support
3. Present an object-based specification describing user-tasks.
4. Present a set of scenarios

We wanted to make the project as close to the real world as possible, and did not use any of those methods. We specified the project requirements using a three step process. First, we presented a small description of the program domain (a small busy pizza restaurant). The specification is shown in Table 4. We got the groups to discuss this project domain and start creating the first model in the UCD process: a map of user roles and how they relate. Based on this preliminary model we provided a user in a lecture that the students could interview to determine actual user roles and user tasks. This interview was done in a wizard of oz style: one of the lecturing staff played the role of the manager of the pizza store. Based on the lecture, students decided which tasks were important to the pizza company and started designing a system based on those tasks.

7 Conclusion

This paper has presented the design of COMP311 (User Interface Design) and some initial experiences from teaching the course. This course was new to our institution in 2004. In this course we wanted to give students in depth experience designing a user interface as well as high levels of feedback so that they could make incremental improvements to their design. We also wanted to achieve these goals without overburdening staff or creating a student workload disproportionate with other courses.

To achieve these goals we used a novel combination of tutorials, groups, and practical work to provide feedback to students in a timely manner without overloading teaching staff or students. The key to this combination is the requirement that all groups attend the same tutorial. This requirement meant that tutorials could act as feedback sessions and group meetings where groups could interact with teaching staff. We found this model works well and plan to extend it to other courses.

References

Barnes, J. & Leventhal, L. (2001), Turning the tables: Introducing software engineering concepts in a user interfce design course, *in* 'SIGCSE'01', pp. 214–218.

Constantine, L. L. & Lockwood, L. A. D. (1999), *Software For Use*, Addison Wesley.

Cooper, A. & Reimann, R. (2003), *About Face 2.0: The Essentials of Interaction Design*, Wiley.

Drury, H., Kay, J. & Losberg, W. (2003), Student satisfaction with groupwork in undergraduate computer science : do things get better?, *in* T. Greening & R. Lister, eds, 'Fifth Australasian Computing Education Conference (ACE2003)', Vol. 20 of *Conferences in Research and Practice in Information Technology*, ACS, Adelaide, Australia, pp. 77–85.

Hartfield, B., Winograd, T. & Bennett, J. (1992), Learning hci design: mentoring project groups in a course on human-computer interaction, *in* 'Proceedings of the twenty-third SIGCSE technical symposium on Computer science education', ACM Press, pp. 246–251.

Leventhal, L. M., Barnes, J. & Chao, J. (2004), Term project user interface specifications in a usability engineering course: challenges and suggestions, *in* 'Proceedings of the 35th SIGCSE technical symposium on Computer science education', ACM Press, pp. 41–45.

Myers, B. A. & Rosson, M. B. (1992), Survey on user interface programming, *in* 'SIGCHI'92', ACM, New York, Monterey, Ca., pp. 195–202.

Newman, I., Daniels, M. & Faulkner, X. (2003), Open ended group projects a 'tool' for more effective teaching, *in* T. Greening & R. Lister, eds, 'Fifth Australasian Computing Education Conference (ACE2003)', Vol. 20 of *Conferences in Research and Practice in Information Technology*, ACS, Adelaide, Australia, pp. 95–103.

Nielsen, J. (1993), *Usability Engineering*, Morgan Kauffman.

Noble, J., Marshall, S., Marshall, S. & Biddle, R. (2004), Less extreme programming, *in* 'ACE 2004 Proceedings'.

Preece, Rogers & Sharp (2002), *Interaction Design*, Wiley.

Rettig, M. (1994), 'Prototyping for Tiny Fingers', *Communications of the ACM* **37**(4), 21–27.

Rosson, M. B., Carroll, J. M. & Rodi, C. M. (2004), Case studies for teaching usability engineering, *in* 'Proceedings of the 35th SIGCSE technical symposium on Computer science education', ACM Press, pp. 36–40.

Snyder, C. (2003), *Paper Prototyping*, Morgan Kaufmann Publishers.

Suchman, L. (1987), *Plans and Situated Actions: The Problem of Human-Machine Communication*, Cambridge: Cambridge University Press.