# Conceptual Design and Prototyping to Explore Creativity

Manuel J. Fonseca, Joaquim A. Jorge, Mário R. Gomes, Daniel Gonçalves and Marco Vala
Dep. of Information Systems and Computer Engineering,
INESC-ID/IST/Technical University of Lisbon
mjf@inesc-id.pt, jaj@inesc-id.pt, mario.gomes@tagus.ist.utl.pt,
daniel.goncalves@inesc-id.pt, marco.vala@tagus.ist.utl.pt

**Abstract**. Many approaches to teaching HCI focus on either user requirements or prototyping. However, these two techniques do not provide enough tools for students to explore the design space in breadth at early stages of conception. Indeed, even when these two approaches are combined, students still lack tools to explore the design space and bridge the gap from requirements to prototyping. In this paper, we describe the way we teach Human Computer Interaction, stimulating students to be creative during interface design. To that end we added course materials on conceptual design and scenario based interaction, combined with the exploration of different low fidelity prototypes, which we believe increase both the usability of student-developed prototypes and foster learner creativity. To illustrate this we present some of the best examples of interactive prototypes designed and developed by students attending our HCI course in the context of Information Systems and Computer Engineering (ISCE) curriculum at the Technical University of Lisbon, Portugal. While the current approach seems to elicit positive responses and draw encouraging remarks from students, work remains to be done in emerging interface paradigms and more formal evaluation on how this approach positively affects student outcomes.

## 1 Introduction

Currently, many Human Computer Interaction courses in most undergraduate IT curricula all over the world, adopt the iterative method for interaction design (see Figure 1). However, most common methodologies pose problems in the first steps in the interaction design process. The usual approach is to go straight from task analysis to prototyping. We think that this presents problems in interface development methodology. Indeed, going directly from task analysis to paper prototyping limits student creativity, since they are forced to start thinking in terms of interaction styles

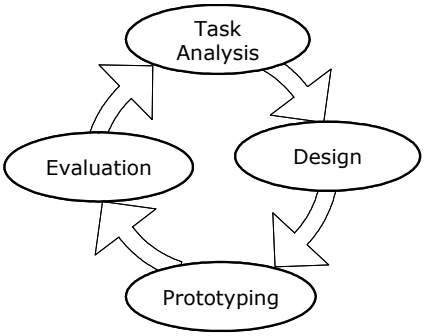


**Fig. 1.** The iterative development cycle.

and screen layout instead of focusing on solving user needs. As a consequence, both the quality and creativity of coursework suffers and students find it difficult to explore the design space.

A summary analysis of existing courses on Human Computer Interaction reveals different approaches in communicating the HCI discipline to students. Indeed some courses structure subject matter around scenarios [9]. Other courses focus on task analysis instead [4]. A third group combines task analysis, scenarios and prototyping. However, few if any of the syllabi surveyed include conceptual design in the overall cycle of developing an interactive system.

From our experience, introducing Conceptual Models and Scenario-Based Design in the syllabus in combination with the other techniques indicated allows students to break the creativity gap from task analysis to prototyping. We added a module about conceptual modeling between task analysis and prototyping. Creating conceptual models during this phase of the interface development forces students to think about concepts and actions that their system will offer to users, instead of being worried about screen layout and color schemes as discussed in [5]. Moreover, the selection of metaphors to include in the conceptual model, stimulates students to be creative in the analogies they choose, and allows them to explore the design space more thoroughly than they would had if they started by directly sketching low-fidelity prototypes after requirements analysis. Arguably a better conceptual model when applied thoroughly may well make the final system more familiar to users, and consequently easier to learn and use.

Additionally to the conceptual design, we also explore three different scenarios in our course syllabus. First, problem scenarios describe how actually users perform tasks. Second, activity scenarios describe how users will perform tasks using the concepts of our conceptual solution. Third, interaction scenarios describe how users will interact in detail with the implemented solution. Additionally, in the most recently taught semester (Spring 2007), we also asked students to design three alternative prototypes, before they start developing the final design.

We present the Introductory HCI course which is taught to ISCE students on the third year of a five year Computer Science and Engineering undergraduate degree taught at *Instituto Superior Técnico* (IST), the school of Engineering of the Technical University of Lisbon, which is the oldest engineering school in Portugal. The course has evolved considerably over the years since its inception in 1992. The pedagogical

approach described herein is the result of curricular changes started in the 2002/2003 academic year. The one-semester course is currently taught to 280+ ISCE students across the two campi of IST, spaced 30km apart. In what follows, we describe the program as currently taught at IST with special emphasis on course structure and coordination between recitation, laboratory classes and project development. To illustrate results, we show some exemplar prototypes developed as coursework by students followed by discussion, conclusions and proposals for future developments.

## 2 Teaching Human Computer Interaction

At the core of the subject matter of any HCI course lies the design, development and evaluation of Interactive Systems. However, not every school teaches HCI the same way. We can characterize most curricula in two broad categories. One, which we call *task-oriented,* hinges course delivery on task analysis and identifying user needs. Another, which we call *prototype-oriented,* focuses on the design of the interface proper. Its curricular structure focuses on exploring the design space through a series of prototypes. While we feel that both approaches have their advantages and disadvantages, we want to provide our students with the best of both worlds. In this section we show how we have combined the strong user focus of the first approach with the progressive refinement approach favoured by the second. To this end, we first tell students how to identify potential users and tasks that those users may want to perform on the interactive system being designed. Then, we lead them through the principles and basic guidelines required to design and develop creative solutions with usable user interfaces. Finally, we train them on evaluating interfaces at different phases of development, by applying the most suitable evaluation technique at each stage. We do this by combining recitation classes, laboratory work and group projects so that student work flows continuously in lockstep with subject matter.

In the next subsections we describe in detail the subjects of our theoretical and laboratory classes and explain how they are synchronized with the development of the course project.

### 2.1 Theoretical Classes

Our theoretical classes are organized around seven main chapters, which we believe are important to give students a good basis for their future work as interface

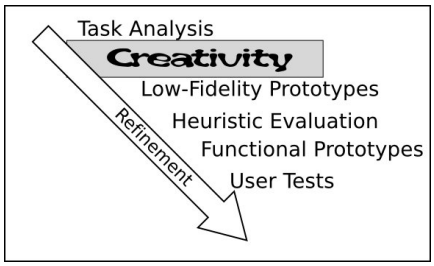


**Fig. 2**. The creativity gap in Iterative Refinement

designers, interface engineers or usability evaluators. These areas are organized in seven large study sections as shown in Table 1. Below we describe each section in more detail, highlighting its contribution to the goals of our HCI course.

| Unit | Subject Matter |
|---|---|
| I | Introduction to User-Centered Design |
| II | Know the Users and Their Tasks |
| III | Interactive Systems Design |
| IV | Evaluation |
| V | Documentation, Help and Interaction Devices |
| VI | Web Pages |
| VII | Toolkits |

Table 1: Course Organization

**Introduction to User Centered Design**
In this chapter, besides introducing the course and all administrative things, we try to give students an overview of the overall process of designing an interactive system, creating bridges to upcoming classes. During these classes we also try to provide students with a more comprehensive view of Interactive Systems (Appliances, Car consoles, etc), beyond desktop interfaces with windows and buttons. Additionally, we provide them an introduction to Usability Engineering and ISO Certification.

**Know the Users and Their Tasks**
One of the most important aspects to developing an interactive system is to know who will use our system and for what. To that end we teach our students the different methods to perform task analysis, from observation to questionnaires. Later on, during the development of their project, they will have the possibility of applying these techniques to collect information about users and tasks, and answer eleven questions on task analysis [4] (Who will use the system? What tasks do users perform now? What new functionalities do they want in the future? How do users learn to perform the task? Etc.). By performing task analysis and answering these questions, students obtain a clear idea of present user requirements and collect important information relevant to the next step of their project assignments. Since we are developing interfaces for people, it is important to know how the human information processing system works. The key idea is to show to students that humans are limited in their capacity to process information, and that they must take that into account while designing user interfaces.

**Conceptual Models**
In this chapter we teach how to go from user requirements (collected during task analysis) to the development of the prototype or prototypes. We start this section by studying the conceptual model, where students learn to create solutions for user needs independent of the intended devices or interaction styles. The conceptual model includes metaphors, to create analogies of real world entities and activities, and concepts that the system will expose to users. Such concepts can either be objects, attributes or actions that can be performed on objects. Other components from include relationships between concepts and the mapping between model

concepts and entities arising out of the adopted metaphor. Indeed, using metaphors forces students to seek existing solutions that can be adapted to their novel interface. By using good metaphors, students understand that their systems become easier to learn and to use, since users quickly associate new concepts provided by the system to those taken from the metaphor. For example, if we use the phone call cabinet metaphor for a drinking machine system, we can say that "buying a drink is like making a phone call. First we insert the money and then we select the desired drink (phone number)". Using conceptual models to bridge the gap from task Analysis to Interaction Design is a relatively new addition to the curriculum. Indeed, we have added Conceptual Models over the last three years after noticing that students experienced considerable difficulties in mapping user requirements to low-fidelity prototypes. This is because low-fidelity prototypes already entail many design commitments and embody sufficient decisions that students feel "locked" to a given screen layout and interface organization that they do not attempt to further explore the design space. To overcome this significant barrier to creativity, we have gradually introduced Conceptual Modeling into the design cycle, following the inspirational writing by Johnson [5]. Additionally, we are guiding the students through design space exploration by using scenarios [9]. In our revised setting, students flesh out user requirements into problem scenarios, which help them weave user requirements and specifications from task analysis into coherent stories that help them both explain and communicate the most important features their design has to provide. From problem scenarios they evolve into metaphors, concepts, relationships and activity scenarios to detail the main components of the interface but *before* committing to any detailed aspects of the interface while keeping the design at a very abstract level. We believe that this design discipline helps students flesh out the main components and structure of their approaches *before* committing to any details, be it screen or interaction design. In this way, creativity is fostered and better designs may emerge, since students are "forced" to think through their designs against the user requirements before crystallizing solutions into prototypes as was the case in the past.

**Interactive Systems Design**

After finishing studying the conceptual model, students go into the next step of interface development, which is to find solutions for the user interface that satisfies the conceptual model. Only on this phase of the process we ask students to worry about the type and "look & feel" of the user interface. To that end, we present to them the different interaction styles, going from the basic command line, menus, direct manipulation, to the more recent and futuristic interaction styles, such as, augmented reality, wearable computing or tangible interfaces. Additionally, we teach the more important guidelines about screen design, namely, spatial layout, font types, use of colors, alignment, etc. Finally, to conclude this chapter, we explain how to create low-fidelity prototypes (LFP), as a fast, simple and cheap way to develop prototypes to show to final users. We also stimulate students to create storyboards, as a mechanism to explain how tasks are performed on their systems. We would like to mention that along these years of teaching the HCI course with this methodology, students produced very creative and interesting LFP. We would also like to highlight that in this chapter we teach students to create activity scenarios (making part of the

conceptual model) and interaction scenarios, as a complement to LFP and storyboards. We believe it is important to teach the conceptual model, the interaction styles, the screen design and prototyping, as unique module, because this way students understand that the same conceptual model (and correspondent activity scenario) can lead to different interface solutions, LFP and interaction scenarios.

**Evaluation**

After creating prototypes, the next step is to learn how to evaluate them. To that end, we teach three types of usability evaluation: Evaluation by usability experts (heuristic evaluation), Predictive evaluation; and Evaluation with users. We start by teaching students to become usability experts. They learn Nielsen's heuristics [12], we give a set of interfaces that respect and violate these heuristics and finally, we practice an example in class. Our goal is that by the end of the course, and after performing six heuristic evaluations to their colleagues' projects (in the laboratory classes), students are usability experts. We believe that by practicing heuristic evaluation in laboratory classes, students not only learn the Nielsen's heuristics, but also apply the evaluation to practical cases. Besides the heuristic evaluation, we also teach predictive evaluation using GOMS, CCT and KLM. Finally, students learn how to perform usability tests with users, how to write a protocol for the tests and how to summarize and analyze the collected data using the correct statistic methods.

Documentation, Help and Interaction Devices

In this chapter we teach students to write documentation for interactive systems (tutorials, user manual, reference manual, quick reference manual), as well as to develop interactive and contextual help. One of the things that we highlight is that manuals and help must teach users on how to perform tasks with the system and not describe menus or options. To conclude this chapter of the course, we talk about different input and output devices, emphasizing to students that the design of the interface is very dependent of it. An interface for a PDA (with a limited resolution) will have to satisfy some constraints that an interface for large displays will not have, and vice-versa.

**Web Pages**

Until this chapter we have been teaching design and development of user interfaces in a general sense. However, in the last years, the majority of created interfaces are web pages. So, we decided to dedicate some classes to this particular type of user interfaces. First we show students the main differences between designing "ordinary" interfaces and interfaces for the web. Then, and taking into account that anyone, independently of its knowledge and education about user interfaces, creates web pages, we discuss the "Original Top Ten Mistakes in Web Design" [10], and the most recent "Top Ten Mistakes in Web Design" [11]. These way students can compare current web design problems with original ones. Another subject very important in web design is design patterns. We teach some of the more relevant patterns, such as, the rules to create a good Home Page, e-commerce and the shopping cart. Finally, we talk about personalization of web sites, standardization, accessibility, cascading style sheets and HTML and CSS validating software.

**Toolkits**

During the development of their prototype, students create functioning "simulators" of the final interactive system, where the interface is the most important thing. However, if the prototype were supposed to evolve into a real product, the tools used to create the prototypes (Flash, HTML, Javascript, Visual Basic, etc.) might not prove to be the more appropriate choices. To overcome this, we dedicate the last chapter of our HCI course to the study of UI software architecture and Toolkits. The main goal of this section is not to teach the particulars of a given toolkit, but rather to discuss fundamentals, such as the event model, windowing system and program interaction, and callbacks. We conclude this chapter by teaching the MVC model, in order to illustrate a programming architecture that separates the semantic of the application from visualization and control.

## 2.2 Laboratory Classes

One of the major goals of the course is to teach students a user-centered interface design methodology. While theoretical classes lay the knowledge foundations required to accomplish that goal, we feel it is important for students to actually use that methodology in the development of an interface and, thus, learn by doing. This posed an interesting problem: given the iterative nature of user-centered design practices, and the different stages it comprises, it would not be effective to simply require students to design an interface and check the result at the end. Indeed, it is the usage of the methodology itself that concerns us, rather than the final result, as we try to impart skills that can be used at later times, in the student's professional lives, in whatever interface design challenges they might face.

The only way for us to ensure that the appropriate design methodology is being used, and that students receive timely and relevant feedback is to closely follow the entire design process. To that end we create the course's laboratory classes, synchronized with the theoretical recitations. Usually, any given subject is used in the laboratory two weeks after it has been taught in a theoretical class. This gives time for students to assimilate that subject and resolve any doubts they might have regarding it. The order in which the different subject matters are considered (described in the previous section) mimics the order in which the different interface design stages should take place, allowing each laboratory class to focus on a specific stage. We typically have eight groups of three students on each laboratory class.

At the beginning of the semester, each student group is given a project assignment (as described in the next section). The project, consisting on the design of an interface, will be developed throughout the semester by students. Each laboratory class has a set of goals to be attained. These are known beforehand by students, at least a week before class, and directly reflect an interface design stage. Namely, there are classes for:

1. creating task analysis questionnaires
2. presenting the task analysis' results and a conceptual model for the interface
3. heuristic evaluation (HE) of a low-fidelity prototype
4. presenting the results of the HE

5. heuristic evaluation of a first functional prototype
6. heuristic evaluation of a second functional prototype
7. presenting the results of the HE of the second functional prototype
8. presenting results from usability tests with users

As can readily be seen from the above list, each class closely follows an iteration of the user-centered design cycle. As it would be unfeasible to develop the entire prototype in the classroom, the classes focus more on the presentation of results rather than on actual development. This allows us to provide instant feedback about their work, and correct any problems that might arise. Also, as students present their results to the entire class, they benefit from a discussion with their colleagues in which the instructor acts as moderator. This exposes them to alternative ways to solve the same problems, requires them to stand behind their choices and adequately justify them, and allows them to see other problems that might arise, so that they may avoid those pitfalls in the future.

The description of each class gives students not only the goals of what is to be accomplished in that class, but also what should be prepared beforehand. A list of work to be done and deliverables is provided with that description, on a weekly basis. Doing so has a major advantage: it imposes a constant work pace, so that the project is created in a timely and ordered fashion. Also, as each stage of the design cycle must be presented in a different class, it prevents students from skipping stages and cutting corners, enforcing the use of the appropriate design methodology. Finally, as laboratory and theoretical classes are synchronized, students will not tackle problems they are not yet ready to solve.

Students are graded in each class, based on their performance in the classroom, on the work they have prepared beforehand, and on the deliverables produced. This evaluation is accompanied by comments given by the instructor so that students may know what they could have done better. In some cases, when it is deemed reasonable both in terms of work involved and timings, students are allowed to correct the major flaws in their work, to improve their grades and to have a chance to apply the instructor recommendations.

Aside from the classes we mentioned above, there are three others, not directly related with the development of the course's project. The first two classes of the semester consist of an informal evaluation of two web sites by students and the presentation of their findings. We felt this is necessary as at that stage most students lack an awareness of interface problems (ours is an introductory HCI course). This evaluation and ensuing presentation and discussion helps motivate students and gives them an overall idea of what a properly designed interface should be like. The other class not directly involved with the project occurs before students have to present a conceptual model for the interface of their projects. We found that conceptual models are hard to grasp, as they require an abstraction power most students don't possess or seldom exercise. Thus, we spend an entire class guiding them through the construction of a conceptual model for a sample interface. This is done collaboratively. After a short exposition about conceptual models (complementing what was taught in theoretical classes) all students are asked to provide their opinions about the conceptual model that is being created. The instructor facilitates

the exchange of ideas between students, and provides comments about their suggestions. Gradually, a conceptual model emerges. As all students are involved in their creation and directly face the problems and questions involved in it, they gain insights that allow them to, after the class, properly develop conceptual models for their own projects.

## 3 Course Project

The course project plays a very important role in the course structure. It allows students to apply the knowledge acquired in the theoretical classes to a concrete scenario as close as possible to what they will find in the future. We usually propose eight different assignments which are presented at the beginning of the semester and randomly distributed for each group.

According to our experience, the assignments should only be a couple of paragraphs describing the general goals of the project. Students should look at the problem cleanly without being guided to a particular solution. During task analysis and the design phases (of the iterative development cycle) they should gather as much information as possible about users and their tasks, and they should find solutions for the problems encountered. Indeed, shorter assignment descriptions encourage students to be more creative in the way they explore possible solutions.

We found out that students are motivated if we use a commercial-looking language in the assignments stating for example that “company x wants to hire your team to create a new product y”. Most seem engaged by the prospect of doing something similar to what they would do if they work in a real company. We try to emphasize this aspect during both development and evaluation phases.

It is also very important to have several different assignments. Usually laboratory classes have eight groups and we try not to have two groups doing the same assignment. We found that by using different assignments for all groups in the same laboratory session, students tend to focus more on their work and less on the neighbors’ work, which has lead to better and more creative solutions. Moreover, since each group evaluates other group projects during the laboratory classes (these are mainly heuristic evaluations performed as part of the evaluation phase of the iterative development cycle), we observed that student groups perform better as evaluators and tend to find more usability problems when they evaluate project assignments which are different from theirs.

As we mentioned before, student projects are developed throughout the semester and most of the laboratory classes include checkpoints to assess the various steps of the project. In the next sections, we provide an overview of the methodology, some examples of projects developed as coursework and comments and remarks elicited from students as well as our informal assessment based on these remarks and observations of student performance.

### 3.1 Methodology

The main objective of the course project is to allow students to experience the iterative development cycle. We want them to practice all the phases in the cycle and to learn they should go through the cycle several times to achieve good results.

Students start with the creation of a task analysis questionnaire which is discussed in the first laboratory class dedicated to the project. Then, they use the questionnaire to enquire target users in order to get information and compile the task analysis' results.

After task analysis, they move to the design phase. They create a conceptual model of the interface with metaphors, concepts and activity scenarios. Both the task analysis and the conceptual model are presented in the laboratory class where they receive feedback from both their colleagues and the teacher. At this stage, we clearly highlight the importance of having a good conceptual model as a baseline for the prototyping phase.

After design, they go through prototyping. Students start by designing three alternative low fidelity prototypes for the conceptual model created before. From this set of prototypes, they choose one to evolve for the functional prototype. However, most of the times, students incorporate solutions from the other two prototypes in the selected one. By doing this, the quality and creativity of the resulting prototype increases relatively to the first version. Additionally, students do storyboards and interaction scenarios, for the selected prototype, bringing all these elements to the next laboratory class where they are evaluated.

In the evaluation phase, each group does a heuristic evaluation (HE) of the low-fidelity prototype of two other groups and gets his prototype evaluated by two groups also. During the process, students not only learn and practice HE as they get useful information to make their prototypes better in the next iteration of the cycle. The results of the HE are presented and discussed in the next laboratory class and it completes the first rotation of the iteration cycle.

Students are then encouraged to go through task analysis and design phases again, and to revise their conceptual models to reflect the results of the HE. Then, they do the first functional prototype which is evaluated in the next laboratory class. In this second rotation, they consolidate what they have learned before and they get a second evaluation done by an expert in interfaces.

In the third rotation, they repeat the process again and they create the second functional prototype (a revised version of the first after getting the results of HE). The second functional prototype is evaluated once more between groups of the same laboratory class.

In the fourth and final rotation, students present their final prototypes (a revised version of the second after getting the results of HE) and they do a final evaluation with users. These usability tests and their results are presented in the last laboratory class.

Students not only learn better experiencing the iterative development design, as they get a solid methodology to use in the future. All the deliverables produced along the iterative process are compiled into a group webpage on weekly basis and at the end we can get a very rich overview of the entire process.

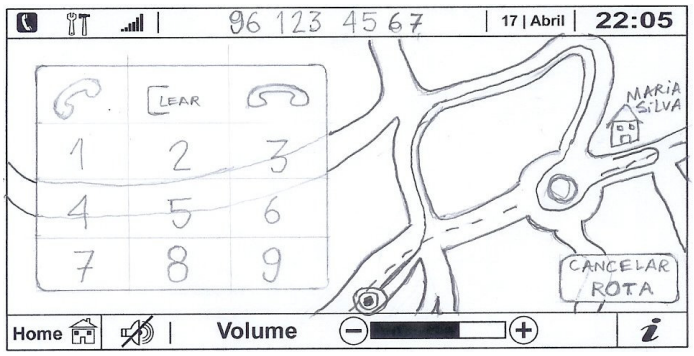


**Fig. 3** – Low fidelity prototype for a car intelligent system.

A final note pertains to regular presentations which students have to deliver in the laboratory sessions as part of their work. These contribute to create important soft skills such as how to make a presentation and how to present a project in development to colleagues and faculty and students regard them very positively.

### 3.2 Examples

In this section, we present some of the best student projects developed during the most recent course (at the time of this writing) – Spring 2006/2007.

The example from Figure 3 illustrates a low-fidelity prototype of an interface for an intelligent system that will integrate the different types of information accessed by drivers in a car, such as, car check-up, traffic information, calendar, phone calls, radio, GPS, etc. The resulting system should allow the monitoring of information and also the integration between the different functionalities, like for instance, show the way to the person the driver is calling, or alert the user if the car is running out of oil and give him the location, the path and the phone number of the nearest garage. The prototype illustrated in Figure 4, succeeded very well in integrating those functionalities, was very creative and presented a good screen design and layout.


**Fig. 4.** Intelligent car system prototype.

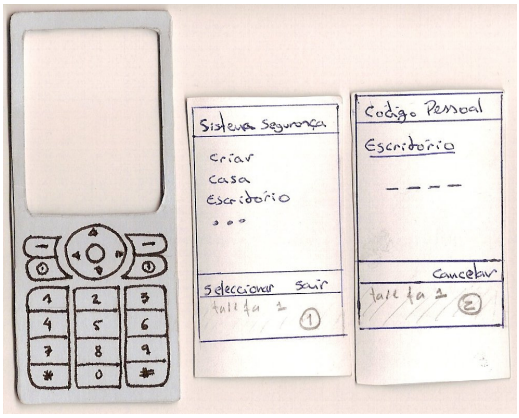


**Fig. 5** – Low fidelity prototype for the home security system.

Another example taken from student work is a system to control the security systems of an house or office, from a cell phone. The future system will provide access to all the cameras in the house, the execution of actions when an intruse is detected or the definition and activation of different profiles according to the time of the day and/or the day of the week. Figure 5 shows a low fidelity prototype of this assigment, where we can see the design of the device and two screens of the application. The main challenge of this project was the small display available to present the information to users. In the prototype presented in Figure 6, students were able to achieve this chalenge by creating an interface with a clean screen design and very good navegation through the execution of the different tasks.


**Fig. 6.** Security system prototype.

Finally, the last example presented here is a kiosk for tourist information about a city, which can provide information about monuments, restaurants, cultural events and transportation to go to those places. Moreover, it will be able to suggest a trip

plan to visit the city according to user's constraints and requests, namely time and important places to visit. The big challenge in this project was the smooth integration between the information about the city, tourist relevant items and the access to services relate to them, such as, tickets for events, tickets for transportation, city maps, etc. Figure 7 presents one of the prototypes that clearly achieve these goals by providing a system with a good screen design a good navigational and integration mechanism, which makes the performance of tasks very easy for users.

### 3.3 Student Comments

At the end of the course we collected feedback from students about the methodology that they were "forced" to follow during the development of the project. The majority of them complained about the quantity of work required from them week after week. However, all of them agreed that without this strict schedule and method the quality of their projects would be considerably worse and probably would not satisfy users' needs. Students also highlighted the involvement of the final users during task analysis and during the final usability tests. Finally, they understand the need for several iterations in the development of an interface, because they saw the positive evolution of their prototypes during the semester. An informal comparison with previous years' projects highlighted more creative and less uniform approaches to problems, and a more formal analysis showed that the average grade of the final prototype increased from 70% to 75%. We believe that the combination of the conceptual model design with the exploration of alternatives during the creation of the low fidelity prototype, positively influence the final quality of the prototypes.

## 4. Conclusions

We have presented our approach to teaching an introductory HCI course within a five year degree setting. While many challenges remain, it can be argued, that Conceptual Design enables a smoother transition from gathering user requirements to prototype user interfaces. Indeed, focusing on concepts rather than on screen- and interaction-design details allows them to better explore project alternatives without making early commitments to both interaction styles and screen layout. This approach includes carefully synchronizing main course components, from recitation, to Laboratory and Student Projects. We are happy that student response to the recent changes in curricular content has been highly positive. Additionally, we noticed that our introducing conceptual design in the course curriculum was accompanied by marked improvements in the general quality, usability and creativity of student projects. Further, informal evaluations, project quality surveys and assessment by students show that the curricular structure presented here had a positive influence on both learner attitude and performance. Of course, the ideal balance remains an ever elusive target. We plan to further improve the syllabus to address emerging interaction techniques and guidelines, introducing mobile devices and ubiquitous computing to replace the current emphasis on Web development [2]. Finally, we plan

to conduct a more rigorous assessment on how the teaching approach and curricular structure impact student performance, with special emphasis on project quality.


**Fig. 7.** Tourist kiosk prototype.

## References

1. A. Dix, J. Finlay, G. D. Abowd and R. Beale, *Human-Computer Interaction*, 3rd ed. Prentice Hall, 2004.
2. D. K. van Duyne, J. A. Landay and J. I. Hong, *The Design of Sites*, Addison-Wesley, Boston, MA, 2002.
3. J. Preece, I. Rogers and H. Sharp, *Interaction Design: beyond human-computer interaction,* John Wiley & Sons, 2002
4. C. Lewis and J. Rieman, *Task-Centered User Interface Design: A Practical Introduction*, downloaded from ftp://ftp.cs.colorado.edu/pub/distribs/clewis/HCI-Design-Book/, 1994
5. J. Johnson and A. Henderson, *Conceptual Models: Begin by Designing What to Design*, Interactions, Vol. 9 (1), pp. 25-32, ACM Press, January 2002.
6. M. Rettig, *Prototyping for Tiny Fingers*, Communications of the ACM, Vol. 37 (4), pp.21-27, ACM Press, April 1994.
7. A. Marcus and E. Chan, *Designing the PDA of the Future,* Interactions, Vol. 9 (1), pp. 34-44, ACM Press, January 2002.
8. J. Nielsen, *Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier*, Academic Press, 1994.
9. M. B. Rosson and J. M. Carroll, *Usability engineering: Scenario-Based Development of Human-Computer Interaction*, San Francisco, Morgan-Kaufmann 2002.
10. J. Nielsen, *Original Top Ten Mistakes in Web Design*, Jakob Nielsen's Alertbox, May 1996. (cited 2007-10-20), http://www.useit.com/alertbox/9605a.html
11. J. Nielsen, *Top Ten Mistakes in Web Design*, Jakob Nielsen's Alertbox, February 2007. (cited 2007-10-20), http://www.useit.com/alertbox/9605.html
12. J. Nielsen and R. Molich. *Heuristic evaluation of user interfaces*. Conference on Human Factors in Computing Systems (CHI'90), pp. 249-256. ACM Press, 1990.