Teaching HCI Engineering: Four Case Studies

Sybille Caffieau[1 [0000−0003−4222−2018]], José C. Campos[2 [0000−0001−9163−580X]], Célia Martinie[3 [0000−0001−7907−3170]], Laurence Nigay[1 [0000−0002−4854−626X]], Philippe Palanque[3 [0000−0002−5381−971X]], and Lucio Davide Spano[4 [0000−0001−7106−0463]]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France
{Sybille.Caffieau, Laurence.Nigay@univ-grenoble-alpes.fr}
[2] University of Minho & HASLab/INESC TEC
jose.campos@di.uminho.pt
[3] Université Paul Sabatier-Toulouse III, IRIT
{martinie, palanque}@irit.fr
[4] University of Cagliari, Dept. of Mathematics and Computer Science, Via Ospedale 72, 09124, Cagliari, Italy
davide.spano@unica.it

Abstract. The paper presents the work carried out at the HCI Engineering Education workshop, organised by IFIP working groups 2.7/13.4 and 13.1. It describes four case studies used in Human-Computer Interaction Engineering courses. We propose a common framework for presenting the case studies and describe the four case studies in detail. We then draw conclusions on the differences between the presented case studies that highlight the diversity and multidisciplinary aspects to be taught in a Human-Computer Interaction Engineering course. As future work, we plan to create a repository of case studies as a resource for teachers.

Keywords: Teaching HCI engineering · User-Centered Design · Software Engineering

1 Introduction

Engineering interactive systems is a multidisciplinary activity positioned at the intersection of Human-Computer Interaction (HCI), software engineering, usability engineering, interaction design, visual design and other disciplines. The field of Human-Computer Interaction Engineering (HCI-E) is concerned with providing methods, techniques and tools for the systematic and effective design, testing, evaluation and deployment of interactive systems in a wide range of application domains. This field, thus, requires an understanding of both HCI and Software Engineering topics as highlighted by the ACM/IEEE-CS Software Engineering [11] and ACM SIGCHI Human-Computer Interaction [5] curricula.

There are many challenges in teaching HCI [7]: keeping up to date courses and curricula because technologies and methods evolve fast, ensuring the involvement of students with users because computer science students are familiar with various interactive technologies and may not grasp the fact that other types

Author's version of the paper published in Sense, Feel, Design, volume 13198 of Lecture Notes in Computer Science, pages 195-210. Springer. 2021. The final version is available at Springer via: https://doi.org/10.1007/978-3-030-98388-8_18.

of users may struggle with interactive systems. Whereas these challenges do not target teaching HCI-E specifically, they also fully concern HCI-E and a curricula in HCI-E should contain pedagogical materials that help to take them into account. Beyond these field related challenges, Aberg [1] highlighted potential problems of existing curricula in HCI for computer science students and pointed out the need to propose to students pedagogical activities which make them "*create something that works*", "*focus on technology and issues that they feel related to*", and which provide them with "*a sense of realism, with projects or assignments connected to real and ongoing projects*".

In this paper, we discuss the teaching of HCI Engineering. More specifically, we present four case studies used as part of courses on teaching HCI-E. The case studies satisfy different pedagogical goals, including designing or developing interactive systems. We first present the common framework for presenting the case studies, then each of the four case studies, and at last, we discuss how they address the main challenges and needs of teaching HCI-E.

2 Presentation Framework

To present the case studies, we propose a framework made of five categories:

- An overall description as an identity card of the case study including title, type of case study, the studied type of interaction, the available resources, and a brief description of the case study;
- A description of the students and their pre-requisite levels in the disciplines involved in HCI-E, including HCI and Software Engineering (SE);
- A description of the pedagogical objectives of the case study;
- A description of the pedagogical management of the case study, including the tools used and the initial materials provided;
- A description of the expected outcomes and their evaluation.

3 Case study 1: Kart Racing Game

3.1 Identity Card

- **Title:** Kart Racing Game
- **Type:** Project
- **Application domain:** Game
- **Interaction techniques:** WIMP and post-WIMP
- **Brief description:** The case study is based on an existing open-source kart racing game SuperTuxKart (`https://supertuxkart.net/`). The goal is to design input controllers to drive the kart in the kart racing game. The set of input commands includes: turn right, turn left, slow down, speed up, backtrack, back view and standard view. By starting with an existing game and a set of input commands, the goal is to focus only on the input interface and not on the other parts of the interactive system, including the game engine and the output interface.

3.2 Targeted students and pre-requisites

The Kart Racing Game case study is used in a Master course in Computer Science. But this case study could be used for teaching HCI to UX designers or ergonomists. The pre-requisite in HCI for this case study includes knowledge on how to design the software of an interactive system in particular event-based principles and automata. There is no pre-requisite in Software Engineering since it is not required to develop an interactive system.

3.3 Objectives

The main objective is to apply a User-Centered Design (UCD) approach. To guide students, several methods, concepts and tools that support the UCD steps are taught in the course related to this case study. The case study is a practical application of the fundamental HCI methods and concepts to follow the UCD principle, that are necessary to any practitioner involved in the design of useful and usable interactive systems.

3.4 Pedagogical steps/monitoring, initial materials and tools

In order to help students to focus on UCD principles, 1) the functional core is provided, and the focus is only on the input user interface 2) the provided elements to start the UCD process are realistic representations of starting points of industrial projects.

During a 6-week period, groups of 4/5 students design and sketch/develop the game controller of SuperTuxKart karting races[5]. The set of commands and the output device (a screen) are fixed. As starting materials, we also provide realistic answers of three potential users to the question "Do you ever play video games such as kart racing?" (Fig. 1). These answers are fictive ones but allow students to define persona and realistic contexts of use. They are defined to present contexts of use with characteristics (Table 1). Such starting materials are unusual for the students (in computer science).

Based on these initial materials, the pedagogical steps that are discussed each week with the teacher include:

- Analysis of the answers to define persona. A canvas for describing persona is provided (Fig. 2). This canvas is adapted from a framework of the Marketing domain to define buyer-persona. From this analysis, student groups choose to target one or two user profiles.
- Analysis of the answers to define different contexts of use and the requirements they imply.
- Iterative design of different mock-ups or functional prototypes (for instance wire-frame prototypes using Balsamiq `https://balsamiq.com/`). The various solutions are discussed with a simple notation as QOC (Questions, Options, and Criteria) [8] for analysing the designed solutions in relation to the context requirements. Fig. 3 presents examples of designed solutions.

[5] `https://ihm2019.afihm.org/#challenge.html`

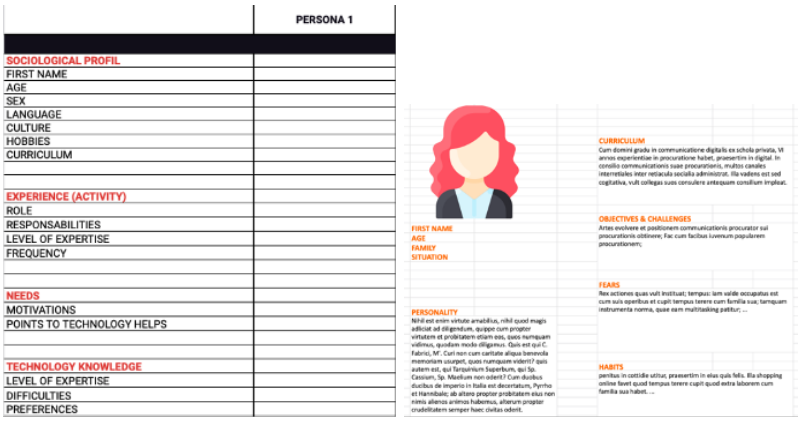
A1	Yes, but not often. I started because one of my granddaughters, Laura, came to our house during a recent holiday with her video game console. Usually when she comes, we go to the park but this time she was a little sick so we opted to stay inside most of the time. So, she spent a lot of time playing with her console. Of course, when I was a kid, I didn't have that, so I wanted to see how it works and what she is doing with it. I am curious. My granddaughter wanted to try but I didn't understand the game she was playing. They're heroes now, you see ... I don't know them. But Laura had a kart game and she said to me "OK grandpa, you know how to drive, so you have no excuse" and I couldn't say no: my granddaughter challenged me. I played and... I lost. I lost because this thing is not the same as driving. But now when Laura comes, she always takes her console to play with me. She is happy because for once she is teaching me something so she is proud.
A2	Yes, very often. I've always played on consoles or on the phone. About the car races... we mostly do it with my roommates because it is quick and everyone likes it. And in fact, we use it to plan household tasks. The apartment quickly gets dirty if we don't clean the house but nobody wants to do it so we take turns but hey ... sometimes it doesn't work. So we use the kart racing game to challenge ourselves and the one who loses has a pledge. It can be something other than cleaning, but often that's it. We even defined rules. The pledge must be known beforehand, the person who is challenged has the right to refuse the race but if s/he accepts it, s/he is the one who chooses the race and the vehicles... It's more fun than bickering all the time.
A3	Yes, but not often. Me, I like it but Mom, she doesn't like that I play too long so she doesn't let me play often. I am not allowed when there is school... and I am not allowed to play in my room, I have to be in the living room. The other day, my big brother looked after me and since I didn't bother him, he let me play, but we cannot tell Mom. When I have the right, Mom tells me how long on the clock and I have to stop when it's time. I do it because otherwise I have no right to play at all.

Fig. 1. Answers of fictive users, used as the starting point for designing the controller of the karting race game.

Table 1. Characteristics of the contexts that are used by the teacher to define the fictive answers.

	User profile	Motivation/Goal	Playing Mode	Enviornment
A1	Grandfather (novice, adult, 65 y.o) Granddaughter (regular player, child, 9 y.o)	Challenge the other, to spend time together	2 players	Living room, No dedicated space, isolated from other inhabitants
A2	Young roommate (expert player, young adult, 20y.o)	Plan household tasks	>2 players (several)	Shared living room
A3	Son (novice player, child, 7y.o)	Pleasure, reward	1 player	Shared living room (with parental time control)

– Evaluation based on heuristics that are specific for games [9] are applied to the designed solutions (at least one of the designed solutions).
– The last required step is to perform a simplified usability study. The pedagogical objective is to learn that the evaluation of an interactive system implies to perform a user study. They need 1) to identify a usability question to evaluate (e.g., user satisfaction, efficiency), 2) to define the data to be collected in order to answer the usability question (such as number of errors while performing a task, questionnaires e.g., SUS [2], AttrakDiff [6]), and 3) to define the instructions for the participants. From these specifications, they perform the test with at least three users, and they try to conclude on the usability of their designed solution.



	PERSONA 1
SOCIOLOGICAL PROFIL	
FIRST NAME	
AGE	
SEX	
LANGUAGE	
CULTURE	
HOBBIES	
CURRICULUM	
EXPERIENCE (ACTIVITY)	
ROLE	
RESPONSABILITIES	
LEVEL OF EXPERTISE	
FREQUENCY	
NEEDS	
MOTIVATIONS	
POINTS TO TECHNOLOGY HELPS	
TECHNOLOGY KNOWLEDGE	
LEVEL OF EXPERTISE	
DIFFICULTIES	
PREFERENCES	

Fig. 2. Canvas to help students define persona.

a) b) c)

Fig. 3. Examples of designed solutions a) for a novice grandfather, b) for an expert roommate student, c) for a 7-year-old child.

3.5 Expected outputs and Evaluation

The expected output is the design of a game controller of SuperTuxKart kart racing game that matches the requirements extracted from at least one of the three initial answers. The most important results of the project are the justifications provided to show that the designed controllers match the initial responses, the users' profiles and the contexts for using the game. The evaluation grid is set to focus on the presentation and argumentation of the designed solution. Intermediate productions are not evaluated. Students provide a video and possibly a text document to present their game controller and explain their design choices.

The 10-point evaluation grid includes:

- 5 points for the presentation of the interactive solution
  - 3 points for the presentation of the solution in its context of use
  - 1 point for the presentation of the interaction (dynamic specification)
  - 1 point for the presentation of the elements of controls, their positions and their shapes
- 5 points for explanation of design choices
  - 4 points to explain the choices made according to the context of use
  - 1 point to explain the limitations of the currently designed solution.

4 Case Study 2: Fantasy Soccer

- **Title:** Fantasy Soccer
- **Type:** Project
- **Application domain:** Sport, Leisure, Game
- **Interaction techniques:** WIMP (mobile)
- **Brief description:** The Fantasy Soccer application is a peer-to-peer variant of the usual game with top-league soccer players. The idea is to replicate such a game for lower leagues: the players create their fantasy team, including players from real teams. For each league turn, they will get points according to the grade (0 to 10) obtained by each player they put in the field, with some modifiers for special events (scoring a goal, providing an assist and so on). Differently from the usual game, the grading is not taken from newspapers or dedicated websites, but the application users assign scores in a peer-to-peer manner, attending the league games.

4.1 Targeted students and pre-requisites

The Fantasy Soccer app project is used in a Bachelor Course in Computer Science. However, at least for the interface design, the case study may be relevant also in other courses focused on UX design. Given that it is a final project assignment for assessing an introductory HCI course, the pre-requisites include both a basic knowledge of the HCI design principles and, for implementing the app, proficiency in Object-Oriented and event-based Graphical User Interface programming.

4.2 Objectives

The main objective is to practice the HCI principles discussed in the introductory course and apply a UCD approach. In particular, the students will go into the prototyping phase at different levels of fidelity (low and high). They will also perform a small user study to assess the overall usability of the proposed solution. Since it is a group project, the objective is also to develop the students' team-working skills, including the management of possible conflicting ideas in the design and implementation process.

4.3 Pedagogical steps/monitoring, initial materials and tools

The students should focus on applying the UCD process and implementing the prototypes. Therefore, the other aspects of the applications should be only drafted. For instance, they can avoid implementing a proper account sub-system or include method stubs returning hard-coded values for avoiding using databases. Since these are techniques we often use in the prototyping phase, but they require some experience for avoiding getting stuck into less relevant details, we provide the students with:

- Templates for the delivery of all the assessed material.
- Sample projects developed by the teachers, including all the required deliverables, with explanations regarding the implementation.
- Mentoring throughout the entire course and during the implementation of the high-fidelity prototype.

The steps for completing the project are the following:

- During the first half of the course, the students know each other during class and lab lessons. We ask them to define groups of 2 to 4 students. After the process finishes, we create random groups of the same size for those people that did not express any preference.
- During the lab lessons in the second part of the course, they must complete design exercises in the lab. This includes familiarising with design and prototyping techniques (scenarios, personas, sketching etc.). Students can work together and discuss ideas and problems with teachers. This leads to an iterative design of the application interface.
- The last lab lesson includes a discussion of the main principles for designing the evaluation. The students are provided with samples evaluation design and collected data from different types of applications. They learn how to define an evaluation goal, prioritise and select the correct metrics and/or questionnaires for collecting meaningful data (e.g., SUS [2]). Their task is to define the goal, the material, the questionnaire, and the metrics for evaluating the Fantasy Soccer application.
- At the end of the course, they must send an intermediate deliverable that the teachers will assess. It includes the specification of the requirements, the personas and scenarios, the discussion of a low fidelity prototype for the application. They receive feedback on such deliverable, which they must consider for the final implementation of the application.

- After that, they start with the implementation of a high-fidelity prototype. They must focus on the interface-related development aspects. The application back-end should include only stubs. Students are free to request a meeting with the teachers whenever they would like to.
- When the implementation is completed, the students must conduct a small usability study, including one or two metrics and about 10 people. They must identify which changes or improvements would be possible in the next iteration.
- Finally, at the end of the development, they must present the implementation results to the teachers. The discussion includes an introductory PowerPoint presentation, a demo of the high-fidelity prototype, the presentation of the evaluation results, and a question-and-answer session.

4.4 Expected outputs and Evaluation

The expected output is the design and implementation of a mobile application supporting the team management and the players in the peer-to-peer grading for each game. The expected outcomes are basically two: an intermediate deliverable including the low-fidelity prototype and a final deliverable including the high-fidelity prototype. The evaluation focuses on the application of basic HCI design principles and on the ability of the group to justify the design choices.

The evaluation grid for the project assigns 19 points out of the 30 available for the entire course, distributed as follows:

- 8 points for the low-fidelity prototype deliverable
  - 1 point for the identification of the requirements
  - 3 points for the identification of the scenarios and personas
  - 4 points for the development of the low fidelity prototype
- 11 points for the high-fidelity prototype
  - 5 points of the explanation of the design choices
  - 4 points for the high-fidelity prototype interface
  - 2 points for the usability evaluation

5 Case Study 3: Home Finder

5.1 Identity card

- **Title:** Home finder
- **Type:** Exercise
- **Application domain:** databases
- **Interaction technique:** WIMP
- **Brief description:** The Home Finder is an interactive application allowing the editing and the visualisation of a real estate database. It is composed of two main windows. The first one enables the user to edit the database. The second one enables the user to filter real estate according to different criteria (surface area, distance from the workplace, etc.) and to view the details of a selected real estate.

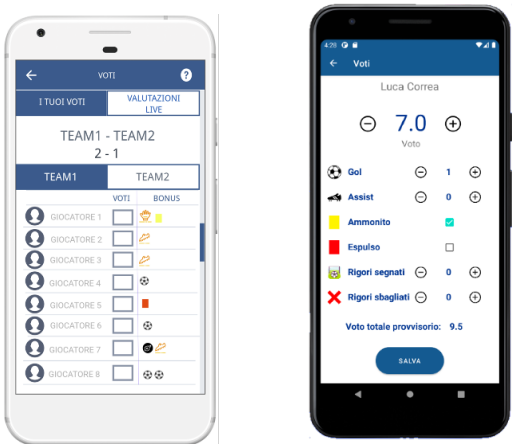


Fig. 4. A Sample low-fidelity prototype for the live scoring interface (by Eligio Cabras) and the high fidelity prototype for the same interface (by the group Urlo del sIUM: Alessandro Pruner, Alessio Piriottu, Eligio Cabras and Marco Mulas).

5.2 Targeted students and prerequisites

The main target type of student is a person enrolled in a Master in Computer Science which has been tuned to address interactive critical system aspects [4], and following the learning unit named "Interactive Systems Software Engineering". The main prerequisite is to have basic knowledge about databases, as well as about the Java Swing graphical toolkit for the programming of user interfaces.

5.3 Objectives

The main objective is to make the students understand that most of the code is dedicated to UI and that to program functions for users to add/modify/remove data is complicated. In addition, this exercise also aims to highlight that:

- the design of the UI can be relatively independent from the functional core users' tasks have to be identified,
- multiple interfaces are possible for a given task,
- some design solutions are more usable than others.

5.4 Pedagogical steps/monitoring, initial materials and tools

This exercise is the last exercise in the pedagogical progression of the learning unit "Interactive Systems Software Engineering" and tackles the programming of a user interface from a behavioral specification of a UI. The students have previously learned to read and to build a behavioral specification of interactive systems behavior. They also have previously learned a method to program a user

interface from a specification composed of a layout picture for the presentation part of the user interface and of a textual description for the behavioral part of the user interface. This method is composed of 5 steps:

1. Identify all possible events
2. Identify all possible actions
3. Build the automaton to describe the behavior of the UI
4. From the automaton, produce the state/event matrix
5. Program source code for event handlers

The students have already applied this method on very simple exercises and then on exercises with an increasing level of difficulty (increasing number of states and of events to be managed). For this exercise, the students have already produced the automaton and state/event matrix. We provide the students with the following statement:

"*You will conceive an interactive application allowing the editing and the visualization of the real estate database. The entries from the database can be manipulated via the editing interface of (Fig. 5) which behaves as specified in the automaton produced and validated during supervised work session 3 (at a previous stage of the learning unit). This interface allows adding, modifying or deleting entries in the database. This information can be visualized via the visualization interface in Fig. 6 which behaves as specified in the automaton produced and validated during supervised work session 3 (at a previous stage of the learning unit). This interface allows you to filter real estate according to different criteria (surface area, distance from the workplace, etc.) and to view the details of a selected real estate (Visual Information Seeking Mantra by Shneiderman [10]: Overview first, zoom and filter, then details-on-demand).*"

The students have to focus on implementing the user interfaces to be compliant with the specifications and on organizing the software using the Seeheim software architecture.

They have to use the Java Swing graphical toolkit and the NetBeans Integrated Development Environment.

5.5 Expected output and evaluation

The expected output is a software archive containing the code source of the program of the Home Finder user interface, as well as an executable version of the Home Finder. The evaluation focuses on the consistency between the specification automaton and the program, as well as on the consistency between the software architecture and the Seeheim architecture. The evaluation grid for the exercise is as follows:

- Programming method
  - Compliance with the automaton and state/event matrix: 5 points
  - Software project preparation in the NetBeans IDE: 1 point
  - Compliance of the UI layout with the specification: 1 point

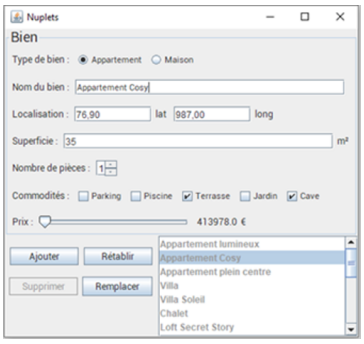


Fig. 5. Database editing user interface of the Home Finder

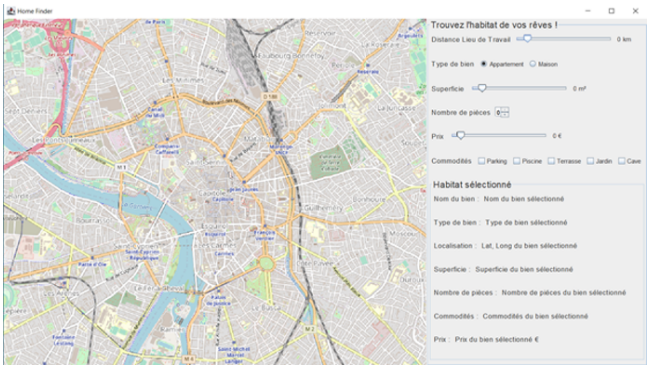


Fig. 6. Visualisation user interface of the Home Finder

  - One to one mapping between a UI component and an event handler: 1 point
  - One to one mapping between a column of the state/event matrix and the event handlers in the code: 1 point
- Coding rules
  - Java coding rules followed: 2 points
  - Legibility (explicit naming of variables, methods and functions...) and correct usage of the IDE refactoring commands: 2 points
  - Software architecture compliant with the Seeheim architecture: 2 points
- Functioning
  - The UI runs as specified: 5 points

6 Case study 4: Electronic prescription system

6.1 Identity Card

- **Title:** Electronic prescription system

- **Type:** Project
- **Application domain:** Information system, health
- **Interaction techniques:** WIMP
- **Brief description:** The Project consists in developing a system to support doctors in prescribing medicines. The project might be seen as part of a larger medical information system but, for the purpose of the project, only the act of prescribing medication is considered. One interesting aspect of the project are the constraints that are imposed on prescriptions: each prescription can only have up to three different medical products, and psychotropic medicines cannot be mixed with other substances in the same prescription. This can potentially create a distance between the doctor's goal of prescribing a treatment, and how the medical products needed for that treatment must be organized into prescriptions.

6.2 Targeted students and prerequisites

The electronic prescription system project was first used in a third-year course on object-oriented analysis and design (OOAD). The students attending the course have taken several programming courses (functional, imperative, object oriented), as well as courses on algorithms, program synthesis, among others, but no course on Human-Computer Interaction. During the course, students must design and implement a software system. Experience shows that a) most students reach the course with a self-centered view of software development, and b) lack of knowledge about user interface design and development creates barriers for the successful design of even the business logic layer (and it undermines the step from requirements to software architecture). The electronic prescription system project has shown to be useful in raising students' awareness of user-centered concerns in the engineering of interactive computing systems.

The pre-requisite for the project is proficiency in object-oriented programming (although the project can also be framed in the context of web programming). The project runs in parallel with the course, in which students learn OOAD (resorting to UML [3]). Basic notions of HCI and user interface prototyping (for example, the MVC pattern) are also introduced (4h).

6.3 Objectives

As stated above, from an HCI Engineering perspective, one objective of the project is to raise students' awareness of user-centered concerns. From that perspective, the focus is on taking into consideration user requirements and designing a user interface that addresses those requirements. Students will capture functional requirements in a use case model and later prototype a user interface to answer those requirements.

Use cases are analyzed in terms of how well they support the users in achieving their goals. Ideally, students will realize that the burden of organizing a list of medical products into valid prescriptions can be moved from the doctor to the

system. Whether doctors will feel comfortable with that loss of control, however, needs to be validated.

At a later stage in the project, the students must design and implement a system that answers the identified requirements. This is done by first defining the API needed at the business logic layer (starting from the use case descriptions and user interface prototype), and then designing an appropriate architecture and implementing it. One goal, here, is to realize the impact that different user interface designs will have on the architecture of the system. For example, depending on the strategy used to validate/generate prescriptions, different APIs and supporting architectures will make sense at the business logic layer.

6.4 Pedagogical steps/monitoring, initial materials and tools

The students self-organize in groups of 3 to 5 to carry out the project. The project is executed during the semester as the topics are worked on in class. Given its length the project is divided into three main steps: requirements, design, and implementation. Each stage is awarded a weight in the final grade (typically 30/30/40).

At the start of the project, students are provided with a copy of the Portuguese electronic medical prescription decree law, access to a database containing information about human medicines[6], and a set of scenarios describing how doctors prescribe medicines. Working from those scenarios, functional requirements are captured in a use case model. A user interface prototype is then produced to address usability concerns. The Pencil tool[7] is introduced, but students are free to choose the prototyping tool of their choice.

During the design phase, students define the control logic for the user interface, the architecture of the system and behavioral models for the business logic. Modelling is done using UML diagrams. Students are encouraged to start by using pen and paper and later, once stable models are reached, a modelling tool (currently Visual Paradigm). In the final phase the system is implemented.

Mentoring is provided throughout the semester on students' request. One aspect that requires attention is managing the complexity of the proposed solution. The goal is to present the project as something realistic, that they could be developing professionally. Typically, students will propose systems that they are not able to fully implement in the timespan of the project. Managing and prioritizing requirements is also a skill the project aims to promote. This is challenging for students and guidance must be provided. One approach that has been attempted is to reduce the set of requirements to consider from phase to phase. However, this creates problems when the proposed solutions diverge on how requirements are handled. Additionally, it can cause frustration as students feel that they are not being allowed to pursue their initial idea for the system.

[6] Infomed - Infarmed's medicinal products database. http://extranet.infarmed.pt/INFOMED-fo/index.xhtml (accessed on 20/10/2021).

[7] https://pencil.evolus.vn/ (accessed on 20/10/2021)

An alternative approach is to define a minimum set of scenarios that the system should support.

After each stage, students must hand in a report describing what they have achieved. General feedback on the first two reports is given in class, commenting on the main positives and negatives of the submitted materials, as a whole. The final delivery must be defended by each group at the last week of the semester, and individual feedback is then provided. The defense includes a discussion of the final report and a demonstration of the system.

6.5 Expected outputs and Evaluation

The expected outputs are the models mentioned above and the corresponding system implementation. More specifically, students should hand in: a use case model and a user interface prototype, from the first phase of the project; UML architectural and behavioural models for the system to be implemented; and the actual implementation and related deployment models. The project contributes 9 points out of 20 to the final grade of the course. The other 11 points are distributed by a final exam (9 points) and a continuous evaluation component (2 points). Projects are graded based on an evaluation of the quality of requirements analysis, system design and system implementation:

- 2.7 points for the requirements analysis
- 2.7 points for the system's design
- 2.6 points for the implementation
- 1.0 point for report

Final project marks, however, are attributed on a per student basis. Each student's contribution to the project is evaluated using a peer assessment approach. Three times during the length of the project, students distribute a fixed number of points between the members of their group (themselves included) in a number of criteria that are discussed and agreed upon at the start of the project. For each criteria, the points to be distributed are a multiple of the group size so, in an ideal situation, all members of the group receive the same number of points. Each student's final mark in the project is the result of combining the project and peer assessment marks. Peer assessment is supported by the TeamMates tool[8].

7 Discussion

The four case studies illustrate the diversity of types of projects and goals that reflect the multidisciplinary (i.e., Human-Computer Interaction, Software Engineering) aspects to be taught in a Human-Computer Interaction Engineering course.

Except for Case Study 3 on the Home Finder, which is used as a common framework for a set of exercises, the other case studies define projects made by

[8] `https://teammatesv4.appspot.com/` (accessed 10/11/2021).

groups of students during several months as part of a project-based learning strategy. Group work promotes discussion and the development of analysis capabilities. One potential issue is guaranteeing that all students contribute to the project in a balanced manner. Case study 4 addresses this through peer assessment within the groups. This both promotes responsibility and self-assessment during the running of the project and helps discriminate the students and award fairer grades.

The four case studies are integral parts of courses taught in Computer Science curricula in different Universities in Europe. The targeted students are computer scientists. The first case study, which does not require the software development of the designed solution, could be taught to non-computer scientists, for instance user experience (UX) designers.

To compare the four case studies, we position them according to the steps of the design and development of a software system: analysis, UI design, software specification, development, verification and validation.

- Case study 1 on the Kart Racing Game focuses on analysis and UI design.
- Case study 3 on the Home Finder is dedicated to software development and compliance with software specification (UI behaviour).
- Case study 2 on Fantasy Soccer and Case study 4 on Electronic Prescription System cover all the steps. Nevertheless, for Case Study 4, the project starts by considering the entire system to be designed and then focuses on a subpart for its development due to time constraints.

These differences in the coverage of the design and development steps of a software system are also reflected in the depth of the concepts and methods applied for each step. While Case studies 1 and 3 go deeper for a subset of steps with the corresponding assessment methods, Case studies 2 and 4 cover all the steps without going into detail on each step. Additionally, while Case study 2 aims for students to practice practice the HCI principles and UCD approach approach taught in an HCI course, Case study 4 is framed in the context of a OOAD course, and its goal is to raise students awareness to UCD, even if the approach is not fully explored in the course.

When we compare the case studies against main identified HCI-E teaching challenges and needs:

- **Fast evolution of technologies and methods:** all of the case studies focus on teaching main HCI principles and targets learners who are new to the domain, then the issue of the evolution of methods does not impact them much. Case study 1 and 2 rely on recent technologies (innovative interaction techniques with gaming consoles, smartphones) whereas case studies 3 and 4 target legacy systems (desktop computers, medical devices). From an engineering point of view, both are interesting and important as future professionals have also to be able to develop interactive applications for legacy interactive systems.
- **Students' involvement with users:** all of the case studies clearly highlight the importance of the users to the students, especially by providing them

with a set of the potential issues in not taking into account user needs and in not applying user centered design principles. They then all may foster the students' involvement with users.
- **Create something that works:** case studies 2, 3, and 4 include software programming activities and enable students to concretely run their produced user interface, whereas case study 1 stops once medium fidelity prototypes are produced.
- **Use tech they feel related to:** case study 1 and 2 may foster more motivation amongst students as they concern technologies they were born with and certainly use daily for most of them.
- **Real life systems:** all of the case studies deal with real life systems.

As future work, we plan to create a repository of case studies as an educational resource for teachers. In particular the repository can be enriched by case studies provided by the members of the IFIP WG2.7/13.4 (http://ui-engineering.org/) on "User Interface Engineering". Such a repository could support teachers to leverage resources and to address the main challenges in teaching HCI-E.

References

1. Aberg, J.: Challenges with teaching HCI early to computer students. In: Proc. of ITiCSE '10. pp. 3–7. ACM, New York, NY, USA (2010)
2. Brooke, J.: Sus: A "quick and dirty" usability scale. In: Jordan, P., Thomas, B., Weerdmeester, B., McClelland, I. (eds.) Usability Evaluation in Industry, vol. 189, pp. 4–7. Taylor & Francis, London (1996)
3. Fowler, M.: UML distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional, 3 edn. (2004)
4. Galindo, M., Martinie, C., Palanque, P., Winckler, M., Forbrig, P.: Tuning an hci curriculum for master students to address interactive critical systems aspects. In: Kurosu, M. (ed.) Human-Computer Interaction. Human-Centred Design Approaches, Methods, Tools, and Environments. pp. 51–60. Springer, Berlin, Heidelberg (2013)
5. Hewett, T.T., Baecker, R., Card, S., Carey, T., Gasen, J., Mantei, M., Perlman, G., Strong, G., Verplank, W.: ACM SIGCHI curricula for human-computer interaction. Tech. rep., ACM (1992)
6. Lallemand, C., Koenig, V., Gronier, G., Martin, R.: Création et validation d'une version française du questionnaire attrakdiff pour l'évaluation de l'expérience utilisateur des systèmes interactifs. European Review of Applied Psychology **65**(5), 239–252 (2015)
7. Lazar, J., Preece, J., Gasen, J., Winograd, T.: New Issues in Teaching HCI: Pinning a Tail on a Moving Donkey. In: CHI '02 Extended Abstracts. p. 696–697. CHI EA '02, ACM, New York, NY, USA (2002)
8. MacLean, A., Young, R.M., Bellotti, V.M., Moran, T.P.: Questions, options, and criteria: Elements of design space analysis. Human–computer interaction **6**(3-4), 201–250 (1991)
9. Schaffer, N.: Heuristics for usability in games (white paper). https://gamesqa.files.wordpress.com/2008/03/heuristics_noahschafferwhitepaper.pdf (April 2007), accessed: 2021-11-12

10. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: The craft of information visualization, pp. 364–371. Elsevier (2003)
11. The Joint Task Force on Computing Curricula: Software Engineering 2014: Curriculum guidelines for undergraduate degree programs in software engineering. Tech. rep., ACM & IEEE-Computer Society, New York, NY, USA (2015)