Browser based room booking system

Project:	Bachelor project
University:	ITU 2011
Advisor:	Peter Sestoft

Kristian Klarskov Marquardsen	kkma@itu.dk	220387-1611
Frederik A. Wordenskjold Noerregaard	fawn@itu.dk	140987-1727

Contents

Chapter 1

Background

1.1 Introduction

In this project we have designed a user-friendly room booking experience for the IT University[1], and provided the concept for a solution to the complex task of assigning rooms to courses. The current process of assigning rooms to both students as well as courses, is both tedious, inefficient and without proper support. At the moment, it is hard to get an overview of how rooms are used and how efficient the current allocation is. This makes it hard for the staff to exploit the full capacity of the university, which is of high importance to expecially ITU, because of the rather limited number of rooms and auditoriums.

We want to construct a program that centralizes this process, and aids the persons responsible as much as possible. Such a system would make the day to day task of booking a room for group work, seen from the perspective of the students, much more flexible. The harder to grasp tasks such as completing the room assignments for a whole semester should be supported by the system, to make it easier for the employees to make the right decisions. As it should be possible to book a room at home, on the way to school or on site, we have implemented the system as a web-based solution. As our main focus is to create a solid user interface, parts of the application will not be implemented. With this in mind, we choose lift as our webframework, as it provides many features relevant to our product out of the box.

[1] IT University will henceforth be refered to as 'ITU'

1.2 Current situation

As of writing, room management at the ITU is split between the Study Administration and the Facilities Management. Within those departments there is another division of labour which all in all adds to the confusion of handling the limited rooms efficiently.

1.2.1 Facilities Management

The information desk, controlled by FM, is currently in charge of the day-to-day use of the meeting rooms. When a student or teacher wishes to reserve or use a room, the FM employee will refer to a binder with a paper-overview of current status, and if they succesfully find an available room a form is filled out, photocopied and archived.

Aside from this, another part of FM is in charge of assigning offices and workspaces to teachers, teaching assistants and researchers. This is done through a locally stored Microsoft Access database, and is handled primarily by one person.

1.2.2 Study Administration

In regards to room management, the Study Administrations biggest challenge is the planning of a semester. First step is collect data from the course database to see which courses will be held in the coming semester. When a list of courses have been compiled, a system called Tabulex[2] is used to create a draft schedule so no overlaps of teachers and courses required by several tracks[3]. The output from Tabulex is finally used to manually create the room assignments for a standard week of the semester.

The other function of the Study Administration is to assign rooms required by exams, for both the examnation itself and for the prepation-time required by some exams. Unfortunately, not all courses have ended by the time exams begin, so apart from rooms being occupied the problem of privacy and noise presents itself.

TODO:**Perhaps reformatting and elobaration**

[2] http://www.tabulex.dk/

[3] 'Track' is the term used to describe a semesters worth of courses in a specific direction, eg. the 'Software Engineering' direction of the 'Software Development and Technology' MSc education

1.3 Scope

Our approach to designing this system, has been primarily focused on usability and all that entails. As described in 1.1, the current situation holds multiple challenges. Tedious and de-centralized management of a single resource is both time consuming and highly error-prone.
We want to implement a system which fulfills two main tasks:

- 1. Helps the student to find a place to work
- 2. Aids the staff in the allocation of rooms

A large part of fulfilling these tasks is the creation of an interface, which by itself maintains a high level of support. Both of the above tasks can often be solved in multiple ways. A student query the database for a room with specific requirements, might lower his requirements if there is no results. On the other hand, he might decide to postphone the activity and try again when the specific room is available.
In comparison, the room allocation problem can contain not only a few solutions (which a program might be able to suggest), but hundreds of ways to allocate courses. When something is impossible to solve, the user has to be able to tell the computer what to prioritize, or which solution to use. A well thought out application succeeds in presenting this information to the user, and gives a proper overview so the best decisions can be taken.
Therefore, the far most important and interesting aspect of our application is the userinterface, in which we have choosen to put our focus. To exemplify a solid solution to this problem, and to keep the interface focused, we have narrowed our vision to the most used area: The day-to-day booking, and the semester planning (hence the tasks mentioned above).

The simple booking have been succesfully prototyped, whereas the semester planning have only been designed and developed theoretically, and not implemented. Doing the full scale implementation of the semester planning would be a possibility, but it is not crucial to prove our point of data presentation and usability. We have therefore chosen to perform the analytical work, and leave out large parts of the actual implementation.

1.4 Target audience

Our application is implemented solely for ITU. It would have been possible to implement a general solution capable of adapting to many instutions, and

while this is our general line of thought, a good UI needs to contain various elements specific to the domain in which it is used. This includes a map of the university for easy navigation and localization of rooms, and many application specific options. It turns out that many aspects of such a system differs from domain to domain (see **??**), which is hard to represent in a 100% generic solution.

Our target audience can thus be divided into two very specific, but different groups:

- Student body at ITU
- Administration personnel at ITU

Student body

The student body covers ages 18 - 40+ and with very different backgrounds. Based on the nature of the university, it is fair to assume that most students have atleast basic familiarity with IT and computer use in general.

Administration personnel

The computer proficiency of the administration can range from expert to novice, and no clear-cut distinctions can be made. We have, however, observed that the majority of the people our system would affect are using tools with some similarities, and are therefore assuming their proficiency to be above novice users.

Chapter 2

Analysis

When unifying the functions of several departments and optimizing the overall process at the same time, a multitude of non-triviel problems will need answering. Through this chapter we will go through the discussions made for and against our choices.

2.1 Domain analysis

2.2 Requirements

Since the ITU does not currently have a full system, we have put up initial requirements for such a system. As described in the target audience **??**, the system will be used by a wide variety of people, and there will be virtually no way to train them, thus putting a very high bar for user-friendliness. We have met this problem head-on by writing up all the tasks our system must support, along with quality requirements describing to what level our users must have succesfull uses. [1]

2.2.1 Non-functional requirements

Preparing for the creation of mockups following various usability tests, it is nessesary to set up a reference point to what we actually want to measure. This helps us to keep the following mockups and usability tests focused.
In a perfect world, all of the 5 usability requirements [1] would be met, though this is rarely desired. This does not emulate a clear focus of what should be prioritized in the application. Here follows a rundown of the most important usability factors.

- **Task efficiency**
  Since the task of finding a room often will be done on the fly, on the way to school, or right before a meeting takes place, it is important that the application is responsive and gives the user a proper response to a booking query. It should give the user a good overview to support his decision, and suggest alternatives if his query cannot be met. In general, we have failed if the user finds it more effective to book a room the old fashioned way, by asking the Facility Management.

- **Understandability**
  The domain analysis suggested that the staff responsible for doing the semester planning had a large need to double-check that the important tasks performed by the system was done correctly. **??**. In general, they had very little confidence in the system. Therefore, to avoid resistance towards change, it is important that the system communicates clearly with the user, during normal use, and expecially in error prone situations to avoid confusion.

- **Ease of learning**
  The set of people using our system is large and will change rapidly. It is therefore important that the system can be used "out of the box". If not, people will simply not use it, as it takes to much of their already limited time to get familiar with. Also, there is a close relation between ease of learning, and the effectiveness of the support the system offers. If the user does not understand how to use the system, they will indeed not be able to benefit from the support layer.

- **Subjective satisfaction**
  Some might argue that the subjective satisfaction is not very relevant in our case, because we're developing a tool for daily support and not a program that in any way should be amusing or entertaining.
  However experts believe [**?**] that users are more willing to accept the program and trust its efficiency if they find it visually appealing. Apple is a good example of using design and pleasing aesthetics as major selling points. The iPad used this to brand itself, thus creating a need that in reality was not present.

2.2.2 Functional requirements

To establish a foundation for the requirements of our system, we have considered the functionality our system must contain. These functional require-

ments only contains goals, and not *how* they must be met, as we will adjust the way they take place based on usability factors.

The system must...

1. distinguish users from eachother
2. store data and information persistently
3. support the user in performing the tasks specified in section 2.3

In regard to the first item, all students and faculty at the ITU already have unique logins, and it would be natural to reuse those for our system. Being a webbased system, it comes as no surprise that we wish the data to be stored persistantly so different computers at different times, without the loss of bookings or the like. Naturally, the system has no use without actually fulfilling the tasks it has been designed to, hence the listing as a functional requirement.

Our functional requirements are fairly scarce compared to some systems, but this stems from the fact that the system, from an end-user perspective, is pretty simple. Our vision is simply to enforce this perspective, and provide the *expected* functionality, without hindrance of high complexity.

2.2.3 Quality Requirements

Based on the task list above, we can finally put forth the list of quality requirements we wish to meet in able to consider the level of usability acceptable.

1. Without instructions, 90% of novice users must be able to complete T1.1 to T1.4 without variants.
2. Without instructions, 80% of novice users must be able to complete T1.1 to T1.4 including variants.
3. With a maximum of 30 minutes instructions, 90% of administration personel must be able to complete T2.1 to T2.5.
4. Without instructions, 50% administration personel must be able to complete T2.1 to T2.5.

2.3 Tasks

The following tasks are the actions able to be performed within our system. The full list including subtasks covering all scenarios can be found in appendix TODO:**ref til full task list**

1. **Day-to-day booking**

T1.1 Book room. May or may not be a specific room or a specific time/day

T1.2 Check availability of room. May or may not be today.

T1.3 Cancel booking. May or may not be before the booked timeslot.

T1.4 Extend booking. May or may not be before the booked timeslot.

2. **School Administration**

T2.1 Create course. May or may not have full information available at the point of creation.

T2.2 Create track.

T2.3 Edit course.

T2.4 Edit track.

T2.5 Plan semester. A full solution may or may not be possible

Chapter 3

User interface design

TODO:**general introduction to this section, and why it warrents a whole chapter**

3.1 Use cases

3.2 Mockups

3.3 Usability tests

3.4 Design process

To our project, the design of the graphical user interface is one of the most important things. Instead of focusing on hiding the difficulty of the problem by obscuring it with automization and huge calculations, we seek to present suggestions and options for solutions in such a manner that a human being can lay the final touch. It has become apparant through our tests and interviews that no solution granted by a computer alone will be sufficient - there will always be conflicts needing a (to a computer) not obvious resolution, and even in the simplest of cases where no conflicts occur, human polish and/or acceptance is key.

We've undergone many iterations of each and every screen in the system, both those implemented in the prototype and those only on mockup stage. What follows is a tour through the process of the design.

3.4.1 Trimming the fat

The design of a page begins with deciding what function it should serve in our system, eg. a page to book a room for 2 hours. With that constantly in mind, we attempt to find all the possible things that particular function COULD include. Some of them are extremely basic and obvious, eg. it needs to display which room you are booking, others are more complicated, as for example the ability to check whether this room is available on a regular basis.

Through the process of 'overthinking' each page, we came up with smarter solutions to cover several different items on the could-list, or found out we needed a new page to handle all the "advanced" settings as an example. Once we had settled on the functionality of the page, we began piecing the puzzle together.

3.4.2 'Instant booking' design

For a full history of our mockups, please refer to

TODO: **indsæt billede af første mockup til book rum**

As seen above, the lower right panel was thought to be in charge of all interaction, trying to give the user the impression that they never left home, and to make sure no-one got lost. Our first usability tests quickly revealed that we had been mistaken. All the test subjects felt overwhelmed by the amount of options, and due to all the clutter did not spot the vital functions.

TODO: **indsæt billede af næste mockup til book rum & billede af weekoverview mockup**

This time around, we tried making it as basic as possible and simply presenting the user with a dialog (not a pop-up, just an overlay) for selecting how many hours from now the room should be booked. The flaw with this was that noone was able to figure out the availability of a room in the near future. This is when we realized that the best successes we've had with room booking was the first draft of the week-overview screen. All the test subjects used the 'matrix' overview both creatively and effectively, so we decided to try and utilize that success to the maximum by redirecting even a simple booking task to the same screen. The only exception we have made to this is when using the "book now"-functionality, which will simply book a room for 4 hours and notify the user which one they have been assigned, with the ability to either accept or reject.

TODO: **indsæt Billede af færdig book skærm**

The final design to book a room, as shown above, grants availability overview of 7 days, starting with current, and supports both dragging of the

mouse to select a timespan, and clicks[1] as we had experienced both during the course of our tests. To assist the user in figuring out how it works, we naturally have highlighting of both the day and time when mouse is drawn across the matrix.

Søren Lauesens —- first law of usability states that a heuristic evaluation only has a 50

3.4.3 Layout

Being a website poses some extra considerations - namely those of browsing habits. Two examples we will refer to, and assume basic familiarity with, in this section are wikipedia.org and google.com TODO: **indsæt små billeder af wikipedia og google med F-pattern eyetrack**

3.5 Proposed design

TODO:**Explain and illustrate the proposed design**

[1]Click for start time, click again on end time

Chapter 4

Implementation

4.1 Choice of platform

Choosing a webbased solution over a desktop application was both a choice of interest and of usability. Accessibility is a crucial factor in usability, and by allowing end-users to access the system instead of having to go through the bottleneck of a single department, the workload is both distributed and handled immediately, as opposed to being dependant on an employee from said department to perform the whole process associated with locating and booking a suitable and available room.
The drawbacks to TODO:**finish this sentence**

TODO:**How to implement, discussion of for and against our choices of platform, language, etc etc. Possibly split this in to several sections**

4.2 Challenges

TODO:**What challenges must our user interface overcome?**

4.2.1 Concurrent room selection

The handling of concurrent users is always an issue in webbased services, and more specifically in our case it presents itself when multiple users wants to book a room at the same time. No solutions are perfect, and we've tried discovering the pros and cons of the solutions we considered.

Possible solutions

These solutions are based on the the case where users **X** and **Y** are searching for rooms at the same time. They could obviously be extrapolated to cases where even more users are interacting.

*Make all rooms that fits **X**, unavaliable for **Y** while **X** is deciding on the which room to use.*

- Cons:
  - If all rooms fit the search of **X**, no rooms will be available for **Y**
- Pros:
  - Easy to implement.
  - Extremely unlikely to encounter dead ends.

*The screen which shows avaliable rooms, of user **Y** should update when user **X** makes a booking, to make sure **Y** doesnt hit a dead end.*

- Cons:
  - Increased number of requests to the server.
  - Dead ends possible.
  - Ultimately, it is a tradeoff between probability of dead ends vs. server request-rate.
- Pros:
  - High usability and responsiveness, resulting in a pleasant experience for the user if no dead ends are hit.
  - More flexible; allows several users to have many choices at once.

4.3 Datastructures

TODO:**Introduction to this section**

4.3.1 Datamodel

TODO:**Explain our database design and datamodel**

4.3.2 Algorithms

TODO:**Outline the important algorithms in our program, and why they are awesome**

4.4 Server-Client relationship

TODO:**Explain the design; what we do, and what lift handles for us.**

4.5 Lift

TODO:**This section probably doesn't belong here, but it needs to be somewhere.**

Chapter 5

Evaluation

5.1 Perspective

TODO:**Take a step back and look at our work as a part to a whole. Tabulex, FM/Study admin etc.**

5.2 Possible improvements

TODO:**If this was a commercial product, what would we change/add/fix etc.**

5.3 Expansions

TODO:**Consider if this section is needed. Perhaps it could fit into the above. If not, we could talk about ridding the need for tabulex, centralizing everything in our system and so forth.**

5.4 Reflection

TODO:**Reflect on the process.. hurp derp**

Chapter 6

Conclusion

TODO:***drumroll* Round off the paper in an elegant, humble, and yet awesomely awesome way.**

Bibliography

[1] Soren Lauesen. *User Interface Design; A software Engineering Perspective*. Pearson Education Ltd., Essex, 2005.