AJAX from the users point of view

The development and evaluation of a Rich Internet Application using asynchronous client-server communication

By

Karoline Driveklepp and Øyvind Fanebust

Department of Information Science and Media Studies
University of Bergen, Norway

Spring 2008

Contents

List of Figures

Acknowledgments

During the course of this master thesis we had been privileged with motivation and support from the following people. First of all we would like to thank our supervisor Dag Elgesem for providing excellent guidance, good ideas and a consistently positive attitude towards the study. Secondly we would like to thank the people who made the experiment possible: Intermedia for lending us an office, research equipment and software, Randi at the Department of Information Science and Media Studies for trusting us with a video camera, and of course the experiment subjects for their participation. We would also like to thank our friend Stine for proof reading the final draft. Last but not least we would like to thank our friends and family for support throughout the study. We would never have finished writing this thesis without their friendly nagging.

Bergen, July 2008

Karoline Driveklepp and Øyvind Fanebust

Introduction

> *"[...] Web interaction designers can't help but feel a little envious of our colleagues who create desktop software. Desktop applications have a richness and responsiveness that has seemed out of reach on the Web."* (Garrett, 2005)

Much research has been done in the field of web application development. During the last years there has also been research targeting so called "Rich Internet Applications". This research has mainly focused on the challenges in developing these web applications. An area that, in our opinion, has been largely ignored by researchers is how Rich Internet Applications are perceived by its users.

According to Duhl (2003) the Internet is now the default platform for developing applications, but the Internet technologies and interaction model reveal several limitations when having to deal with more complex applications. This leads to a user experience that is far from perfect, leaving users frustrated and confused. The Internet has not, according to Duhl, lived up to its expectations.

A current trend in web applications is the move from traditional web applications, with little use of interactivity, to Rich Internet Applications. One technology that enables web developers to create rich interfaces is commonly known as "Ajax". The term "Ajax" was coined by Jesse James Garrett and describes what he calls "a new approach to web development"(Garrett, 2005). By combining several existing technologies such as (X)HTML, CSS, the Document Object Model, JavaScript and asynchronous data retrieval, it is possible to create web applications where the entire web page does not have to be updated between each request sent to the server.

This master thesis focuses on Ajax-based web applications. One of the main reasons why we were interested in doing research on Ajax applications was the improved user experience they offered compared to other web applications. Ajax-based applications like Google Gmail did in our opinion represent a paradigm shift in how we use Internet applications. Web applications could finally compete with desktop applications in terms of responsiveness and usability while offering the possibility of being accessed from anywhere at any time. We were not the only ones who were impressed however; The Ajax driven Google Gmail was named second best product of the year in PC Worlds "The 100 best products of 2005" (PCW, 2005) only beaten by the web browser Firefox. Ajax applications from Google were also featured on this list in 2006 (Stafford, 2006) and 2007 (Dahl, 2007) with the map application Google Earth and the document editing suite Google Apps. We personally feel that rich Internet applications in some cases are now almost as usable as or even more usable than desktop applications.

Another reason for choosing this area of study was our interest for the field of Human Computer Interaction. As mentioned earlier we feel that the user perspective in rich web application research has been overlooked. Amongst the research that actually looks into on the users perspective, the focus has been mostly on web *pages* not web *applications*.

A final reason for deciding to study Ajax was our wish to create an application using this new and exciting technology and additionally to conduct a development project using various development practices. We chose to develop a modeling tool for creating RIS models. RIS is a technique for modeling the work- and information flows in processes. The main reason for choosing this type of application was that the relatively limited functionality of such an application seemed like a manageable project.

Research focus and research question

The focus of this study is on how the use of asynchronous communication affect the users of the web applications, specifically considering the expectations the users have towards the application. The purpose of the research was to study the use of Ajax in web applications, and try to say something about whether Ajax affects the way the user interacts with a web application. More specifically we wanted to see if the use of Ajax in a web application could influence the way users employ certain conventions in the application.

The research question is "*what kinds of conventions does the use of asynchronous communication in web applications lead the user to expect?*"

As the use of RIA-technologies, such as Ajax, make web applications look and behave more like desktop applications we decided that an interesting research project would be to study whether users of Rich Internet Applications employ conventions that are common in desktop applications. Based on this the following hypothesis emerged:

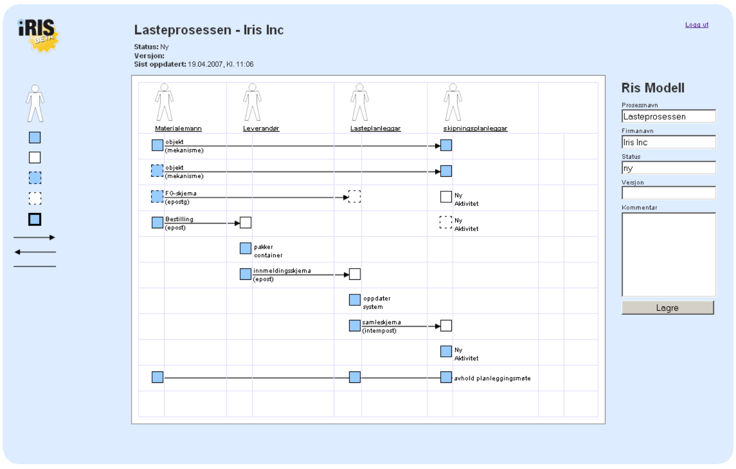
> **H0:** *Ajax influences the users to try to use common desktop conventions in the application as if it were a traditional desktop ap-*

plication.

To test the hypothesis we first needed an overview of the different relevant conventions, both desktop conventions and web application conventions. As there did not seem to exist such a general overview, we decided to gather and categorize this information ourselves. The collection of categories made up a taxonomy of user interface- and interaction elements common in desktop applications. For each category in the taxonomy we examined whether and how it was used on the web. This developed into an analytical framework we could use for evaluating interaction in a web application.

In addition to the framework we needed a web application to perform research on. To test the effects of asynchronous communication in the application, we needed two prototypes of the same application, where one used traditional synchronous communication and one used asynchronous communication. Comparing the conventions used in the two applications could give us a hint on what conventions Ajax can lead to. We decided to create our own web application (see fig. 1), in order to ensure that we had two prototypes that were identical, except for the type of client server communication used.

To be able to compare the users interactions with the two prototypes we carried out an experiment. One group of users tested the Ajax-based prototype while another group tested the prototype using traditional synchronous communication. Comparing the two groups regarding their interaction with the application could provide us with insight as to whether users expected to be able to use common desktop conventions.



Lasteprosessen - Iris Inc

Status: Ny
Versjon:
Sist oppdatert: 19.04.2007, Kl. 11:06

Ris Modell

Prosessnavn: Lasteprosessen
Firmanavn: Iris Inc
Status: ny
Versjon:
Kommentar:

Figure 1: The web-based modeling tool iRIS

Outline

The *Theory*-chapter presents some important concepts in the field of HCI. It also defines traditional desktop applications providing an historical context, and describes traditional web applications including technologies important to that field. We finish off the chapter with an overview of Rich Internet Applications and Ajax.

The *Development*-chapter starts by presenting the web application we developed, with a detailed description including screenshots. The chapter also describes the frameworks used in the development both server side and client side. We then describe the techniques used during the development including agile methods. Finally we present some lessons we learned creating the application.

In the *Research Design*-chapter we first describe the experimental design chosen for this study. We then present the subjects and the settings of the experiment. The data collection methods are then laid out including observation and verbal reporting methods. We then go through the research process before finishing off the chapter by describing the methods used during analysis.

In the *Analytical Framework*-chapter we describe the framework created for the study. We explain how and why it was developed. The framework itself is laid out in detail. Finally we present a resulting checklist used in the experiment and analysis.

The *Analysis & Results* chapter categorizes and presents the findings using the analytical framework and the checklist. Findings are illustrated with transcriptions from the experiment and screenshots of the application.

The *Discussion*-chapter tries to relate the results of the experiment to the research question. The implications of the study are discussed using the categories in the framework. The quality of the study is then discussed highlighting strengths and weaknesses of both the experiment methods and the analysis methods chosen. We finish off with a general discussion where we try to draw conclusions in relation to the hypothesis.

In the chapter titled *Research Contributions* we list the research contributions this thesis has made as well as recommendations for developing asynchronous web applications that were drawn from the results of the study.

The final chapter *Future Work* we recommend further research related to this study.

Theory

In this chapter we present theoretical aspects of the thesis. We start by presenting some relevant concepts from the field of Human Computer Interaction. After that we present important characteristics of desktop applications, web applications and Rich Internet Applications respectively that are relevant to our work.

Human Computer Interaction

This thesis focuses on how people interact with a computer system, which puts it in the field of Human Computer Interaction (HCI). HCI is a field that concerns itself with "the design, evaluation and implementation of interactive computer systems", and related topics (Heath and Hindmarsh, 2002).

One important term in the field of HCI is the user interface. A user interface is the part of an application that is presented to a user. There are several types of user interfaces, including text-based command line interfaces and graphical user interfaces. Graphical user interfaces may also be divided into sub categories. In this thesis we are primarily concerned with the differences between graphical user interfaces on the desktop and on the web.

Another aspect of HCI that we are particularly concerned with in this thesis is direct manipulation. Direct manipulation is a form of interaction where one uses a pointing device to directly manipulate objects on the screen (Myers, 1998). According to Myers the concept was first implemented in Ivan Sutherlands 1963 PHD thesis where he presented a program called "Sketchpad". Direct manipulation is now widely used for many different purposes, but until recently it has not been as widely applied on the web as in desktop applications.

Another term from HCI that we use is "user expectations". In this thesis, user expectations refer to how the user expects a computer system to behave, including which elements and functionality a user expects in a graphical user interface.

With the term "convention" we refer to elements and functionality commonly associated with a particular type of user interface. In the framework chapter, we list common desktop conventions and contrast these to similar conventions on the web.

By “rich interaction” on the web we mean advanced features and functionality previously only common in traditional desktop applications, such as direct manipulation.

The term “usability” is defined by Nielsen (2003) as a quality attribute describing ease-of-use in a user interface. Usability consists, according to Nielsen, of five categories. *Learnability* refers to how easy it is to use a design for a first time user. *Efficiency* defines how quickly one can perform tasks after learning the design. *Memorability* says something about how easy it is to use a design again after a period of not using it. *Errors* refer to how many and how severe errors users make and how easy is it to recover from them. *Satisfaction* concerns whether the design is pleasant to use.

To help developers create consistent user interfaces, some software vendors published design guidelines called ”style-guides”. Style-guides can be related to a special operating system or a special GUI toolkit. We have studied and compared guidelines for both Sun’s Java and Apple’s Mac OS X in this thesis to find out which features seem to be used across different operating systems. The web does not have as many established guidelines in regards to application development as desktop applications. We have used a mixture of the HTML-specification (Raggett et al., 1997), Jacob Nielsen’s heuristics (Nielsen, 2005) as well as our own experiences to contrast desktop elements to web elements.

Traditional Desktop Applications

With the term traditional desktop applications we refer to applications with graphical user interfaces running on windowing systems using among other things direct manipulation for interaction.

One of the first public demonstrations of a windowing system was done by Douglas Engelbart and his Augmentation of Human Intellect project at Stanford Research Institute in 1968. In his televised appearance he demonstrated a windowing system that among other things hosted a direct manipulation text editor (Lohr, 2002). Engelbart’s ideas were later polished and implemented, first by researchers at Xerox’s PARC and later by Apple and Microsoft. Today operating systems with graphical user interfaces dominate in most fields of computing and they come in both commercial and open source varieties.

In the years that have passed since it was first conceived, the graphical user interface has matured and been standardized. Several user interface elements and interaction styles are now common across most modern window systems.

Users are accustomed to relatively instantaneous feedback and rich interaction.

Traditional Web Applications

The World Wide Web was created in 1990 by Sir Tim Berners-Lee at The European Organization for Nuclear Research (CERN). It's primary purpose was to help transporting documents and information, not to create applications, and therefore did not have the level of interactivity of traditional desktop applications. The web was "a platform not for building rich interactive software, but rather for sharing the kinds of textual and illustrative information that dominated the late age of print" (Noda and Helwig, 2005, p. 2).

Two technologies that were important for the rise of the Internet were the Hypertext Transfer Protocol (HTTP) and the Hypertext Markup Language (HTML).

The HTTP protocol is a communication protocol that has been in use since the 1990's to publish and retrieve HTML pages (Fielding et al., 1999). The HTTP protocol is stateless, meaning that each request is treated independently of other requests. It is the de facto standard for communication on the web (Fielding et al., 1999), and has been updated several times during it's lifetime. The HTTP/0.9 protocol, released in 1990, was "a simple protocol for raw data transfer across the Internet" (Fielding et al., 1999). The later versions, HTTP/1.0 and HTTP/1.1 added support for messaging in other formats and new web technologies.

The HTML language was based on the SGML markup language. HTML is used to describe the structure of text and was originally meant to be used to deliver simple documents over the Internet (Berners-Lee and Connolly, 1993). The first official draft of HTML was published in 1993 and it has been continuously revised since then.

Even though the web was developed for the simple purpose of delivering documents statelessly across a network, it was clear that it had potential to do far more than that. The web was soon used to create simple applications, the limits compared to traditional desktop applications were however many. Two key difficulties when implementing a web application was the statelessness of the HTTP protocol and the lack of a way to style HTML documents.

A solution to this problem of statelessness came in the form of so-called "cookies". Cookies are small packets of text that are sent from the web server in the HTTP header. A cookie contains information that can be used to identify the current user, and an expiration time. When the web browser makes a new request it returns the cookie information it received from the server. This enables a form of statefullness over HTTP (Kristol and Montulli, 1997, 2000).

The first HTML specification did not contain any methods for changing the appearance of a web page. Cascading Style Sheets were introduced in 1994 by Håkon Wium Lie and Bert Bos and made it possible to specify fonts, colors and layout for HTML pages. The uptake was slow, the first implementation was made in 1996 for Internet Explorer 3.0, soon followed by Netscape 4 (Lie and Bos, 1997). Most modern browsers now support almost all of the CSS specification.

Application development on the web required a way to process data from the client. The common gateway interface (CGI) enables a web server to execute a program that can process client data. It can then return the processed data and present it to the user, usually in the form of HTML (Gundavaram, 1996).

With HTML cookies and CGI it was possible to create applications on the web. These applications could be styled with CSS to get a nicer user interface but the web still lacked in the area of interactivity and asynchronous communication.

The W3C continues to develop the HTML specification. As this thesis is written the HTML 5 specification is still being drafted. This specification addresses the area of web applications, which the W3C admits has not been adequately addressed until now.

The term ”traditional web application”, will in this thesis mean a web application that consists mainly of HTML elements like hyperlinks and forms with input fields and buttons for the user interface.

Rich Internet Applications

Desktop applications offer a richness and a responsiveness that so far has seemed to be impossible to implement on the Web. Applications being developed are becoming increasingly complex and the technologies behind the Web are starting to show its limits both considering interactivity and usability aspects. The simplicity and limitations of the Web has led to a gap between the experience desktop and web applications can provide their users (Garrett, 2005).

Rich Internet Applications (RIA) are applications that are said to bridge the gap between desktop and web interfaces (Bozzon et al., 2006). RIA are Web-based applications providing more sophisticated interfaces than what are commonly encountered on the web. There exist technologies that require browser plug-ins to create what may be considered RIA, but in this thesis we mainly focus on JavaScript-based RIA.

The RIA technologies enable more complex applications featuring responsive user interfaces and interactive capabilities enhancing the user experience (Paulson, 2005, Noda and Helwig, 2005). With RIA one can introduce direct manipulation, such as drag and drop, to enhance user experience.

A key component of some Rich Internet Applications is the technology commonly called JavaScript. The client side scripting language is one of the most widely used solutions to the problem of introducing interactivity to the web. JavaScript was first introduced and deployed in Netscape 2 in 1995 (Niederst, 1998).

In order to manipulate the HTML page, JavaScript needed a way to represent it conceptually. The W3C introduced the Document Object Model (DOM) as a way to represent a document in object form and to replace browser specific implementations that were often referred to as DHTML (Wood et al., 2000).

RIA utilizes client-side rendering engines, making the applications able to cache data in the client. The interaction and presentation layers in RIA are moved from the server to the client (Bozzon et al., 2006). These applications are therefore less dependent on the server; Rich Internet Applications may therefore not require as many trips to the server as traditional Web-applications. When data needs to be submitted to the server, however, most JavaScript based RIA require the whole page to be reloaded. For applications requiring frequent communication with the server, this can be very limiting.

Ajax

In 2005 Jesse James Garrett wrote an article that gained interest from web developers all over the world. In his article Garrett describes Ajax; a collection of technologies for developing interactive and responsive web sites. Garrett writes that Ajax, which is short for Asynchronous JavaScript + XML, represents a fundamental shift in web development, solving the problem of full page reload by using asynchronous communication.

Developers use Ajax technologies to build Rich Internet Applications that, compared to traditional web applications, provide a higher degree of interactivity and performance as well as responsive user interfaces. Ajax applications can implement functionality that previously has been available only in desktop software. A key aspect of Ajax applications is that they look and act like desktop applications (Paulson, 2005).

One example of where Ajax can be applied to enhance the user experience, by making it more responsive, can for instance be an online map application. In a classical web-based map application it would be possible to navigate a map by dragging, but using the classical web interaction model the entire page would have to be reloaded each time a user tried to navigate. Google Maps[1] is an online map application utilizing Ajax to load parts of the map every time

[1]http://maps.google.com

the user “drags”. This does, in our opinion, greatly enhance the experience of browsing an online map.

Some say developers building Web applications are ”going back to the future”, since all of the Ajax technologies has been around for a while. Most of the Ajax technologies were actually developed during the 1990s (Paulson, 2005). As described in Garrett’s article (2005), Ajax technologies consist of (X)HTML and Cascading Style Sheets (CSS) for standard presentation, Document Object Model (DOM) for dynamic display and interaction, XML and XSLT for data interchange and manipulation, XMLHttpRequest for asynchronous data retrieval, and JavaScript to bind all of the technologies together.

XML is a SGML based markup language developed on behalf of the W3C. In contrast to HTML, which describes documents, XML is a more general-purpose language used to structure information. It is designed to, among other things, “be straightforwardly usable over the Internet” (Bray et al., 1997). The ”X” in Ajax stands for XML, which is a very popular exchange format. XSLT is a language for transforming XML documents into other XML documents or other kinds of documents. It could, for example, be used to transfer an XML-document received from the server to HTML.

Of all these technologies the element that has attracted most attention is the XMLHttpRequest object. It allows for asynchronous data interchange between the web browser and the server. The word ’asynchronous’ comes from the nature of its HTTP request/response processes. Asynchronous calls enable a client and server to talk to each other without reloading pages. What this means in practice is that you no longer have to update the page every time you communicate with the server, leading to improved performance and interactivity and a more responsive user interface (Paulson, 2005). The XMLHttpRequest object combined with (X)HTML, CSS, DOM, XML and XSLT makes it possible to build real RIA using only standard web technologies. Ajax works across multiple platforms, without the need for special plug-ins.

An Ajax application incorporates what Garrett (2005) calls an Ajax engine (see fig 2 on the facing page). This engine is loaded when the application is started and it both renders the user interface and is responsible for the communication between the client and the server.

The engine consists of JavaScript event listeners that sense, among other things, whether a field is updated, a button is pushed or an item is dropped on another item. It then asynchronously sends a request to the server with the new information. While the server is processing the request the client continues to receive input. When the engine receives the answer from the server, it updates just the parts of the page that needs to be updated. The Ajax engine uses asynchronous communication (see fig. 3 on page 20) with the server so that

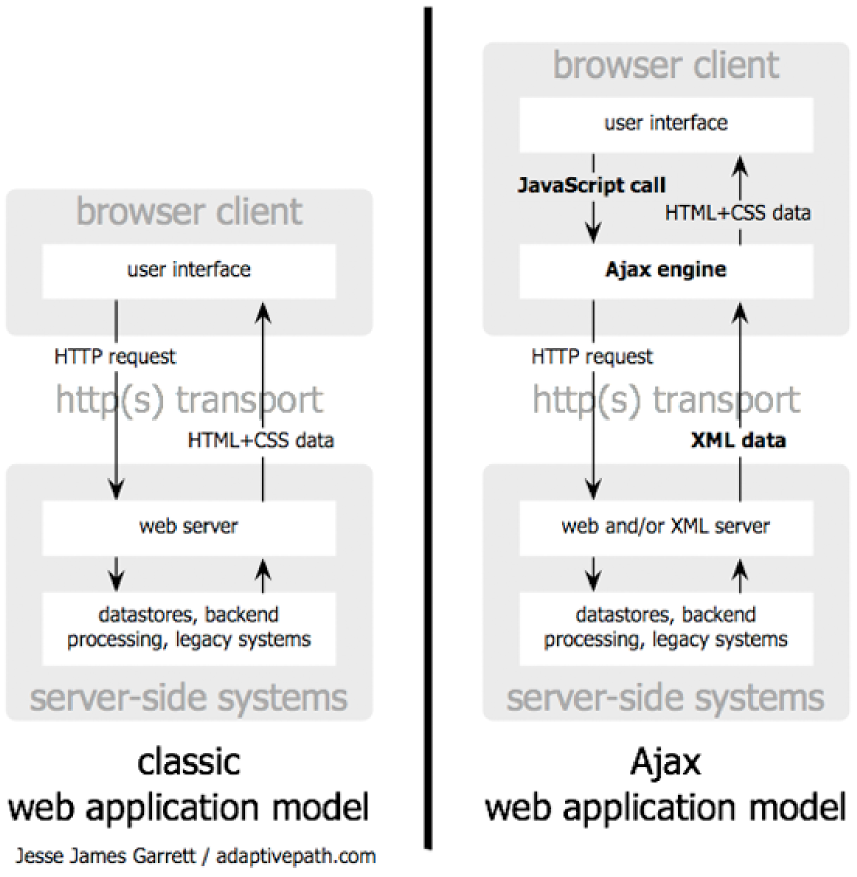


Figure 2: Classic vs. Ajax web application model (Garrett, 2005)

the user is never waiting for the servers actions. This is in contrast with the traditional web applications use of synchronous communication (Garrett, 2005).

Summary

In this chapter we have described some HCI topics central to this thesis. We have described traditional desktop applications as well as traditional web applications with the components that they are made up of. We have also described the advent of Rich Internet Applications (RIA) and last but not least; Ajax. In the next chapter we describe the development of our own Ajax-based RIA.

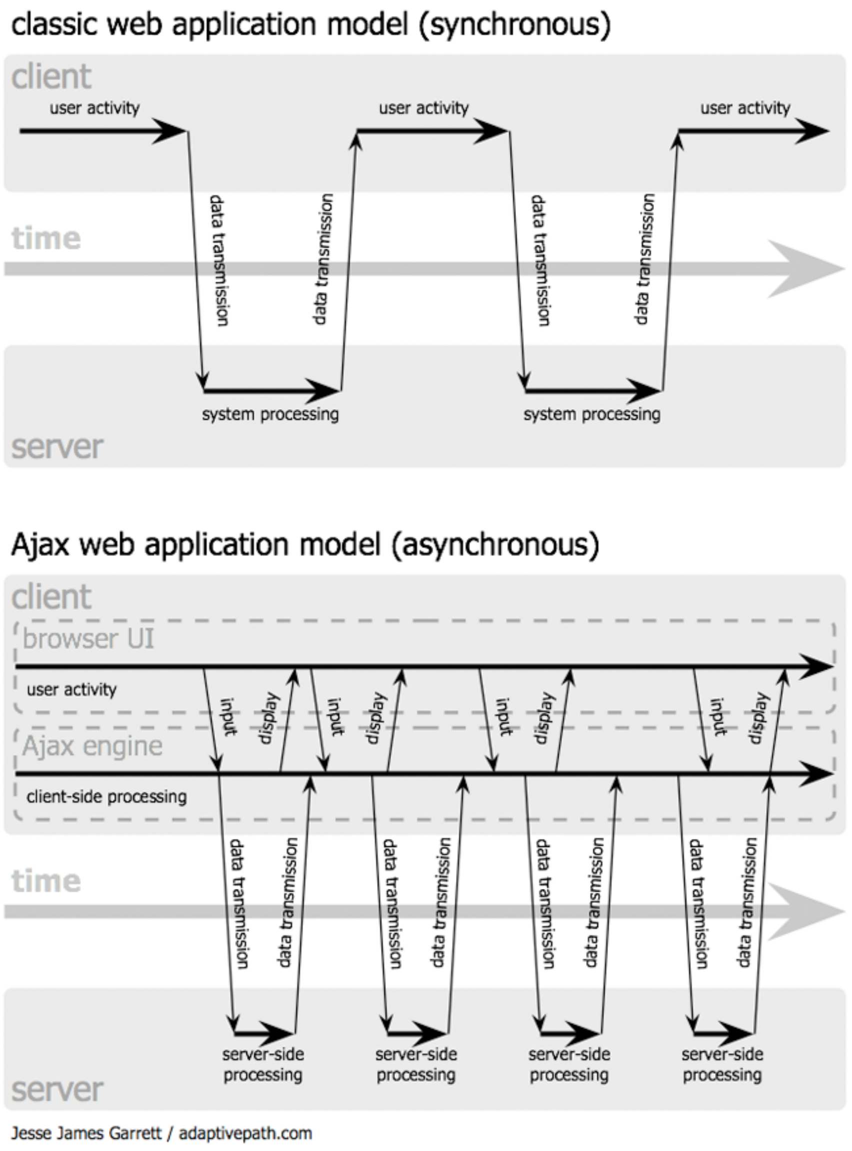


Figure 3: Synchronous vs. asynchronous communication (Garrett, 2005)

Development

In this chapter we describe how we developed our web application. We begin by describing the application and some of its limitations, as well as the differences between the two prototypes. After that we present the development frameworks and techniques used during the implementation phase. We finish off the chapter by presenting what we have learned during the development.

The application

iRIS is a Rich Internet Application that can be used to create graphical models called RIS models. RIS stands for “roller i samarbeid” or “roles in collaboration”, and is a technique for making models of the work- and information flows of processes. The purpose of the RIS model is to emphasize the roles that are involved in a process, the tasks each role perform and the relations between the roles (Iden, 2005).

We chose to make our web application a graphical modeling tool for several reasons. As we wanted to make a Rich Internet Application, it was important to us that the application had typical desktop functionality. Being a RIA, our application should have elements typical of both web and desktop application. Making graphical models is something we think is a very typical task for desktop applications, but not for web applications. The drag and drop technique is very common in graphical modeling tools, and with the emergence of Ajax and related technologies it became possible to implement this technique in web applications as well. This means we had a chance to make a graphical modeling tool on web very similar to those on desktop.

Considering that the purpose of this thesis is to study the consequences of using asynchronous server-client communication making the web application a modeling tool and implementing drag and drop would be an advantage. This because drag and drop requires a lot of communication between the server and the client.

The RIS modeling technique is relatively simple. When performing the experiments the research subjects would not have to be experts in order to understand how the application works and how to use it. The subjects would not be caught up in trying to understand the technique, but most likely focusing solely on performing their tasks.

The final reason for choosing this kind of application is that there are not many modeling tools on the web, and as far as we know no RIS modeling tools. Developing a unique application made the project more exciting and increased our motivation.

iRIS is located online at `http://oyvind.textdriven.com`.

The RIS modeling technique

In order to clearly understand our application and the exercises that the subjects of the experiment performed, it may be useful to understand the RIS modeling language and the different RIS objects within it. The RIS objects that can be used for making a process flow in our application are the role-object, the activity-object, the collaboration-object and the delivery-object (see fig. 4 on the facing page).

A role is represented by a human shape and a role name, and is placed at the top row of the model. An activity is represented by a square box. There are five types of activities each one with its own style. A collaboration is an activity where two or more roles collaborate in performing the activity. It is represented by two activities connected by a line. A delivery element represents a delivery from one role to another by an arrow between two activities. On the arrow you can write what object is delivered and how it is delivered (delivery mechanism).

The RIS model example in figure 5 illustrates the use of some of the elements, representing a part of an invoice process. In this model there are three roles identified; a supplier, an accountant and a certifier. The supplier delivers an invoice to the accountant by mail. The accountant then performs the manual activity of registering the invoice into a computer system. After this there is collaboration between the accountant and the certifier which consists of controlling the invoice. If the invoice does not pass this control it can be sent back to the supplier by mail. Another example of a RIS model can be viewed in the Appendix and represents the process of loading a supply-ship.

Limitations of the application

The RIS modeling technique offers two sets of symbols that represents the work- and information flows of a process. The first set is called RIS Basic, the second RIS Advanced. The RIS Basic symbols consist of the primary symbols

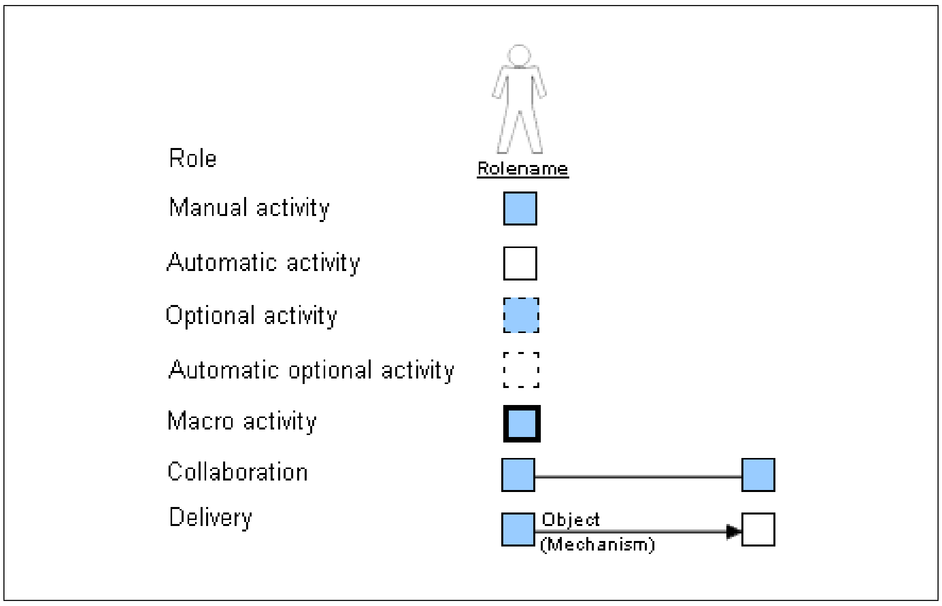


Figure 4: The RIS elements found in our application

needed for making a complete graphic process model. RIS advanced consists of additional symbols to use when the process has special features (Iden, 2005).

The application implements none of the RIS Advanced symbols, and only some of the RIS Basic symbols. We did not consider it necessary to implement all of the symbols, as we only needed an application offering enough symbols to be able to make simple RIS models. The application implements the role, the activity, the collaboration and the delivery symbols.

The application enforces only the simplest rules regarding these symbols, like for instance the rules saying that a role can have several activities and a delivery can only have one sender and one receiver. Other more advanced rules are not implemented, as the RIS modeling itself is not the purpose of this study.

We also decided that the application would only be tested in Firefox. Different web-browsers have varying support for technologies like JavaScript and CSS. We did not want to spend time making the application work in other browsers such as Internet Explorer as this too was outside the purpose of this study.

Application overview

The application consists of the modeling tool itself and a user account module where you can log into your own account and access and create your own models. The start page of the application (see fig. 6 on the next page) presents a way to create a new account or to log in if you already have an account. If you choose to create an account you will be redirected to the account page and have to fill

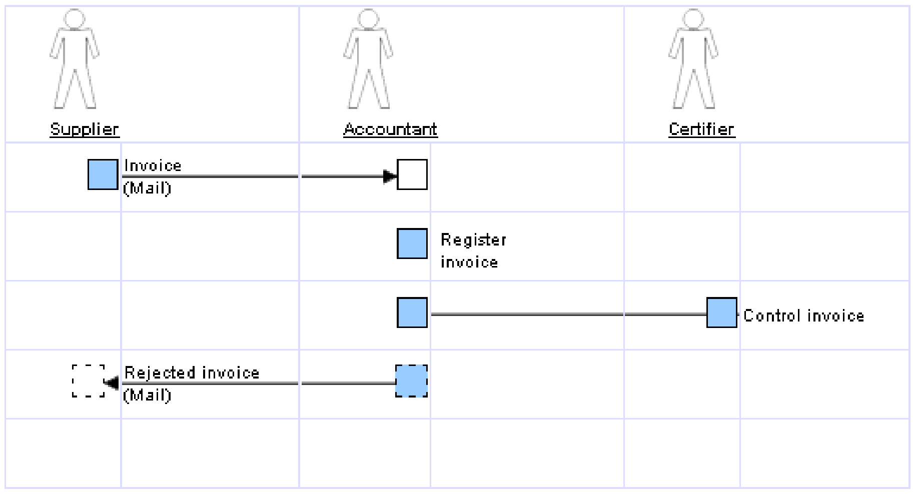
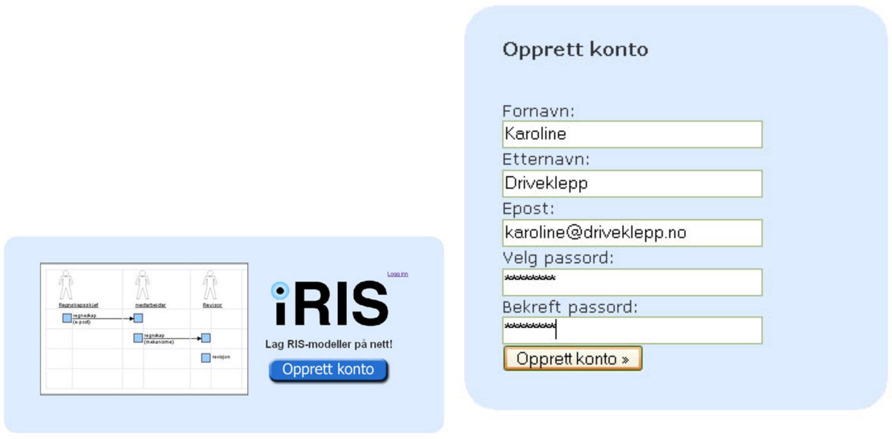


Figure 5: RIS model example

out a form with name, email and a password.



Opprett konto

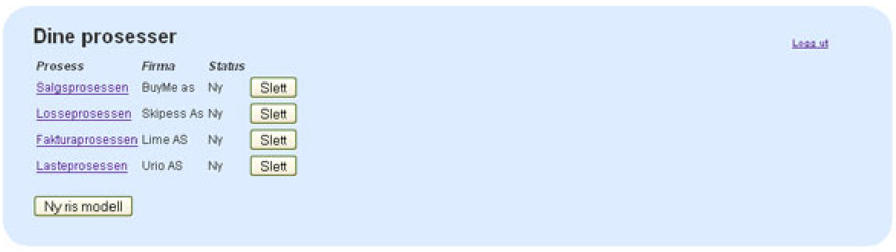
Fornavn: Karoline
Etternavn: Driveklepp
Epost: karoline@driveklepp.no
Velg passord: ********
Bekreft passord: ********

Opprett konto »

Figure 6: *The start page and the account page of the application*

The first thing that meets you when you are logged in is the overview page listing all your previously made RIS models (see fig. 7 on the facing page). You are presented with the process names, company names and statuses of the models. You can click on the process name to see and manipulate the model, or you can click the button labeled "Slett" to delete it. You can create a new model by clicking the button labeled "Ny ris model". When you are done modeling and need to log out you can use the log out URL placed in the upper right corner.

The main page (see fig. 8 on page 26) of the application consists of the modeling tool itself. You can manipulate an existing model or create a new one.


Dine prosesser

Logg ut

Prosess	Firma	Status	
Salgsprosessen	BuyMe as	Ny	Slett
Losseprosessen	Skipess As	Ny	Slett
Fakturaprosessen	Lime AS	Ny	Slett
Lasteprosessen	Urio AS	Ny	Slett

Ny ris modell

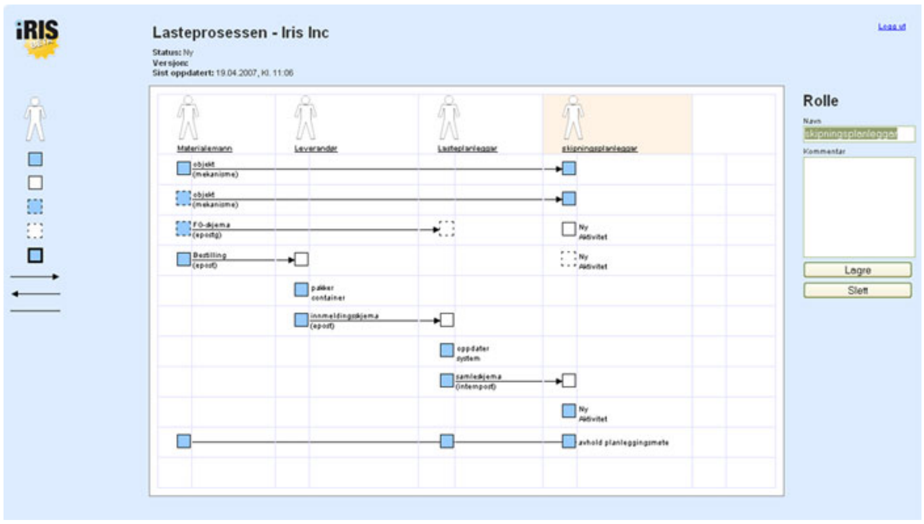
Figure 7: *The overview page, when a user is logged in.*

The main page consists of four components; The RIS modeling area, the RIS model information, the Object pane and the Editing pane.

The Object pane holds all of the RIS symbols available in the application. These symbols are draggable, and can be dragged into the RIS modeling area to create elements in the model. The RIS modeling area is where the RIS model is created, and consists of a table of cells that can hold elements. The top row of the table consists of cells that can hold Role elements. These cells span two columns each. The column on the left can hold activity elements and the column on the right can hold arrows for deliveries or collaboration elements. When you hover (hold mouse pointer over) one of the elements in the Object Pane, the cells in the RIS modeling area that can hold that specific element changes its color to indicate where the element can be placed. In addition when you drag an element and hover one of those cells, the cell changes to another color to indicate in what cell the element would be placed if you drop it. For example, if you hover the Role element, every empty top-row cell will change its color to blue to indicate where you can put a new Role. If you drag the Role element and hover one of the empty cells it changes its color from blue to orange in order to indicate that this is where the element will be placed if you decide to drop it. The line and arrow elements can only be placed in between activities as they are part of deliveries or collaborations, and colors indicate where theses elements can be placed.

Once an element is placed in a cell you can manipulate it further using the Editing pane. Clicking an element in the modeling area causes the Editing pane to appear, and the color of the cell with the element is changed to indicate that the element is selected and can now be manipulated. With the Editing pane you can rename the selected element, add a comment or delete it. Deleting it would make it disappear from the RIS modeling area. If you were to delete a Role element all of the activities, deliveries and collaborations connected to that Role would also be deleted. Deleting an activity would only remove that activity from the modeling area, and deleting a collaboration or delivery would only delete the lines or arrow and leave the activities on the modeling area. In addition to RIS elements the Editing pane can be used for updating the RIS model information, which is presented by the RIS model information component. This component presents the name and company of the process, the status and version number of the model, and time of last update.

While working in the application creating the RIS models you do not have to worry about saving the models. The database is continuously updated with the changes, meaning that as the model is being built all the changes to the model are saved instantly.



Lasteprosessen - Iris Inc

Status: Ny
Versjon:
Sist oppdatert: 19.04.2007, Kl. 11:06

Rolle
Navn: skipningsplanlegger
Kommentar:

Figure 8: *The main page of the application, where the RIS models are created and manipulated*

The two prototypes - difference and consequences

We created two prototypes of the iRIS application. These two prototypes have only one difference. One prototype uses Ajax for client-server communication, and the other uses traditional synchronous client-server communication. Since the infrastructure of the web does not hide the server calls from the user, using different types of client-server communication led the prototypes to behave slightly different from the users perspective as well. Using Ajax hides the server calls from the user. As a consequence of strictly implementing asynchronous communication instead of synchronous communication in one prototype, this application prototype lost some of its feedback. In the non Ajax-based prototype, the interface is locked whenever server calls are made and the mouse pointer shows an hourglass. In the Ajax-based prototype, the interface is not locked during server calls and no hourglass symbols appear. As a consequence the user does not know when server calls are being made. This is discussed further in the Discussion chapter.

From this we also can draw another difference between the prototypes. Since the Ajax prototype does not lock the application whenever server calls are being made, the users of the Ajax prototype can work continuously without pauses between server calls.

Development frameworks

We spent a great deal of time looking at the different frameworks available for web development before we started implementing the application. We found that we wanted both a server side framework and a client side framework or library.

Server side framework

As we were going to develop a web application we wanted a good starting point as well as a way to organize the code. This led us to look for a web application framework. After evaluating several different frameworks, the search led us to a framework that has grown in popularity the last few years: Ruby on Rails (RoR). RoR is built on the object oriented Ruby programming language. It is a web development framework based on the Model View Controller (MVC) software architecture pattern.

The MVC pattern divides the application code into three object types: models, views and controllers (Thomas et al., 2006). The model component takes care of maintaining the applications data. It also enforces rules on the data, making sure that they do not become invalid. The view is responsible for maintaining the user interface that the end user of the application uses to access the data. Controllers tie the application together by reacting to outside events, querying the model for relevant data and presenting it to the end user via the view (Thomas et al., 2006). RoR also has built in object relational mapping (ORM) via a component called Active Record. Instead of having to interact directly with a relational database, RoR converts results from the database into native ruby objects that can be used directly in the code.

Client side framework

Building a GUI using HTML and JavaScript can be a daunting task. For this reason we felt the need for a JavaScript software library to help performing complex tasks like for instance drag and drop and asynchronous communication. We originally tried implementing the application using the JavaScript library

“Scriptaculous” [2], which was available through RoR, but we found out that it was not suited for our purposes. After evaluating several other frameworks, we settled on MooTools[3], which was more appropriate for our purposes. This object-oriented JavaScript framework is both small and modular. One of the main reasons we settled on MooTools was that it supplied methods to work with JavaScript in an object-oriented manner making us able to create classes, constructors and employ class inheritance.

Development techniques

To make the development as smooth as possible we used the following techniques and work methods.

We developed the application on local machines using the RoR framework’s built in web server. The application was not deployed to the production server until we were ready to conduct the experiment. To try to make the experiment as realistic as possible in regard to network latency we used a web host located in the USA for the experiment itself. All testing during the development phase, however, was done locally on the development machines.

The version control system ”Subversion” was used to keep track of code changes. Every time we had finished a feature we checked the change into the version control system. This made us able to go back to previous versions if something went wrong.

While implementing the application we utilized a technique called “pair programming”. Pair programming (Cockburn and Williams, 2001) is a development technique where two developers work together on one computer. One of the developers writes code, while the other one watches and makes suggestions. The developers periodically switch places. Reasons for using pair programming include immediate code review, knowledge sharing and simultaneously keeping both a high level and low level view of the code.

When writing the server side code, we utilized the programming technique ”test first”. Using this technique the developers write a so-called unit test for each method they write. The test is written to ensure that methods work as intended and are written before the actual method it is supposed to test is implemented. Testing first also, in our opinion, improves code quality since you have to think carefully through what the method is suppose to do before implementing it.

When writing client side code we did not use unit testing and therefore had

[2]http://script.aculo.us
[3]http://mootools.net/

to do quite a bit of debugging. In this work the Firefox-extension Firebug [4] helped us a great deal, by letting us find out what was going on at runtime.

After the first phase of the development the application was tested using the usability evaluation technique "heuristic evaluation". Heuristic evaluation is used to uncover usability problems in software. The evaluators are given a list of heuristics, which are recognized usability principles *(Nielsen, 2005),* and use them to evaluate the application. We used three students with some experience conducting usability tests to evaluate iRIS. The main reason for conducting a heuristic evaluation was to identify and remove usability issues to reduce their effects on the experiment.

The development phase

Developing iRIS was the most time consuming task of our research. The main development period lasted from September 2006 to January 2007.

Before we could start developing the application, we needed to know what functionality the application should have. We began by talking to the developer of a desktop RIS modeling tool called MRIS. He showed us his application, which gave us some ideas regarding functionality and user interface. We also spoke to a consultant who uses RIS modeling in his daily work. He had several suggestions that were added to the feature list. The original feature list contained much more functionality than what we finally decided to implement, such as full RIS Advanced support and collaborative functionality. We prioritized adding only what we deemed necessary for conducting the research.

We started creating a "quick and dirty" prototype of the main RIS modeling page, to see what possibilities we had (see fig. 9). The first few days of development we spent setting up the development environment. First we had to install and configure Ruby on Rails and the database engine MySQL. After that we created a subversion repository to keep track of changes. A blog was also created to keep track of the development process and to help us keep focus. In the blog we summarized the work that had been done every day, added TODO lists and useful URLs.

After the development environment was set up we started creating a database model based on RIS. An ER-diagram of the database model can be viewed in the appendix (see fig. 43). We then used RoR to generate model classes for each table in the database. A unit test was created for each model class as we started implementing validation functionality. We tried using the test first methodology as much as possible during this period. After the models were relatively solid we started implementing controllers to add, edit and remove RIS elements.

[4] http://www.getfirebug.com/

When the first controllers were in place we started working with the main task of the development, which was the GUI. This was created using JavaScript, HTML and CSS. The prototype was created so that it easily could be modified to work both synchronously and asynchronously. When the development was done, the result was two prototypes of iRIS, one using traditional synchronous communication, the other an Ajax based prototype using asynchronous communication.

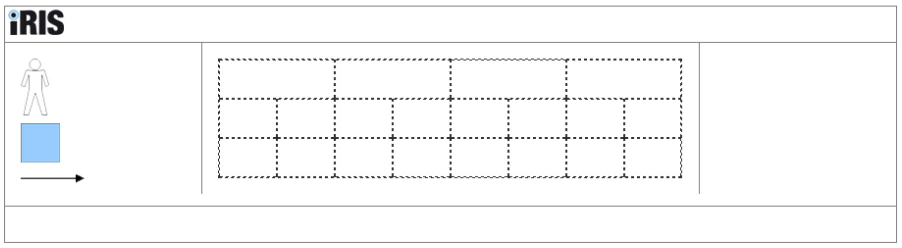


Figure 9: The first prototype of IRIS created September 6. 2006

Lessons learned

Finding and choosing a JavaScript framework was more difficult than anticipated due to the abundance of frameworks and libraries of varying richness and quality. The state of especially JavaScript frameworks has improved a great deal since we started developing our application. If we had started the development now we would probably have chosen a higher level GUI toolkit like YUI [5] or written the entire application using the Java-based Google Web Toolkit[6].

As mentioned earlier, we used Ruby and JavaScript for developing the application. These were both languages we had little or no experience with making the development phase last longer than anticipated. Since both Ruby and JavaScript were relatively unfamiliar to both of us, pair programming helped us share our limited knowledge of the programming languages. In our opinion, pair programming also resulted in higher quality code.

Although the use of JavaScript libraries reduced browser compatibility problems, we came to the conclusion that we would only test the application in Firefox. The reason for this was that debugging JavaScript in multiple browsers and create workarounds for each browser quirk, would be too time consuming.

[5]http://developer.yahoo.com/yui/
[6]http://code.google.com/webtoolkit/

Summary

In this chapter we have given an overview of the application and explained what RIS is and how the application can be used for creating RIS models. We have described how the application was built presenting both the development frameworks and techniques, and also the process of the development phase. Our experiences developing this application have resulted in some recommendations presented in the "Research contributions and recommendations" section in the Discussion chapter on page 90. The following chapter will describe the research design and methods, and additionally the methods for analysis.

Research Design

In this study we wanted to observe users interacting with our application and to find out whether users facing the two different prototypes behaved differently from each other. A good design for studying causality, the relationship between cause and effect, is the experimental design. This is the design we have chosen for this research,

In the experiment conducted for this study we invited ten participants to join. These were split into two groups testing one prototype each. Individual experiment sessions gave us the chance to study one participant at the time. In addition all sessions were recorded both on videotape and with screen-capture software to ensure they could be observed several times. The goal of the experiment was to compare the two groups interaction with the prototype, and discover differences and similarities between the groups.

In this chapter we first discuss the choice of research methods for this study. Secondly we present the experimental design and our choices within this design. The subjects and settings of the research are explained next, followed by the data collection methods. Finally we explain the research process and the analysis methods and process.

Research Methods

When conducting research one has to carefully consider which research methods to use. There are two general directions of research methods; qualitative and quantitative methods.

A main difference between qualitative and quantitative research methods is, according to Ringdal (2001), that quantitative methods focuses on measurement and results in numerical data, while qualitative methods gathers information to provide an insight into a phenomenon, resulting in textual data. Quantity is defined by Bell (2005) as the amount of something, and quality as the "what, how, when and where of things" (Bell, 2005, p. 2).

The purpose of a quantitative study is to convert gathered data into measurable units in order to be able to perform statistical operations on it (Dalland, 1993). Quantitative methods gather information of a large amount of subjects in a limited focus area. Qualitative methods, however, focus on depth and detail. Their purpose is to give a deeper understanding of the studied phenomenon resulting from large amounts of detailed information gathered from a small number of units (Patton, 2002).

> “Qualitative research refers to the meanings, concepts, definitions, characteristics, metaphors, symbols and descriptions of things“ (Bell, 2005, p. 2).

Qualitative studies are typically carried out in a natural setting, without the researcher manipulating the studied phenomenon. This would allow the phenomenon to unfold naturally in contrast to in a a controlled setting (Patton, 2002). Sasse (1991) notes that field studies would be ideal also in the field of HCI, but are virtually never used because of resource requirements and problems related to organizational issues.

Since it would be too difficult and time consuming to conduct research in a natural environment, we have conducted a controlled experiment in an “unnatural“ setting. The application has very limited functionality and would be almost useless in a professional setting. The purpose of the research was to analyse trends related to the emergence of Rich Internet Applications and the user expectations they invoke. We therefore wanted to gather numerical data making the collected data quantitative. According to Holliday (2002), one needs to delve deep into the subjects qualities to understand human behavior. It would therefore not be sufficient to rely on quantitative data only. We needed the qualitative aspects to ensure the analysis of the numerical data was correct. This was especially important in our experiment because having a small sample makes the extraneous variables influence the result to a larger degree. By analyzing the results with qualitative aspects we also increased the internal validity. This could also have been achieved by using a substantially larger sample, lessening the impact of the extraneous variables. Still with a larger sample one could not completely rule out the extraneous variables. The influence of an extraneous variable could be stronger than the impact of the independent variable. Using qualitative methods could help us locate these hidden connections. This is one of the reasons why we chose the combination of qualitative and quantitative approach. Such an approach is referred to by Patton (2002) as methodological triangulation.

Experimental design

An experiment is a research method in which one directly manipulates some factor or factors, hold every other factor constant, and observe the results of the manipulation (Goodwin, 2005). Experiments allow for conclusions about cause and effect to be drawn, which according to Ringdal (2001) make the experimental research design the best design for studying causality. In our study the factor being manipulated, the 'cause' or the independent variable, was the communication between the iRIS application and the application server. The 'effect', also called the dependent variable or the measured factor, was the participants' requests for and use of common traditional desktop conventions in the iRIS application (see fig. 10).

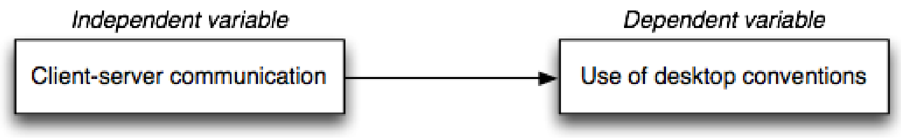


Figure 10: Experiment variables

In the classic experiment one separates the participants in two groups, an experiment group and a control group (C.Cozby, 2007). Ideally, the participants in one group are identical to the participants in the other group, except that the ones in the experiment group are exposed to an experimental treatment that the ones in the control group are not. To assess the effect on the independent variable, the dependent variable is measured twice for each group. One measurement is conducted prior to the experimental treatment, the pretest, and another after the treatment, the posttest. The experimental effect can then be measured by the comparison between the two groups in their possible difference between the pretest and the posttest.

In our experiment the independent variable was the client-server communication of the developed application. The independent variable was manipulated by using two different prototypes of the application, one with the asynchronous communication and one with synchronous communication to the server. The experiment group was exposed to the experimental treatment, interacting with the Ajax-based prototype. The control group interacted with a prototype equal to the Ajax prototype, only using traditional synchronous communication instead of asynchronous communication.

As an experimental design we chose what is called "posttest-only design" (C.Cozby, 2007). This is a design where, unlike the classical experiment, the experiment- and control group is only tested after the experimental treatment (see fig. 11 on the facing page). The experimental effect is measured by the difference in the two groups after or during the experimental treatment.

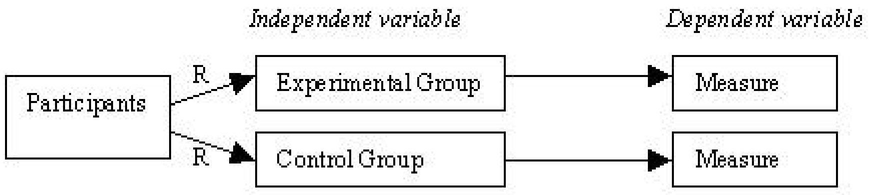


Figure 11: Posttest-only design. The “R” stands for randomization (C.Cozby, 2007, p. 151)

We chose posttest-only design for several reasons. First of all we think that a pretest could have affected the way the participants used the application in the posttest. The interesting aspect in this study is what the participants expect of the application. If they already have used the application in a pretest, they would know how it works and would most likely use it accordingly in the posttest. Another disadvantage of using a pretest is that the participants would be given a chance to figure out the hypothesis, and might have reacted differently than they would have without the pretest and without knowing the purpose of our study. This would further lead to difficulties in generalization of the result to people who have not had any pretest. Pretesting the groups would also have been too resource demanding as it would have doubled the experiment period and the data material to be analyzed.

As an experimental group design we chose what is called ”Independent group design” (C.Cozby, 2007). This is a design where each participant belongs to only one of the groups. We chose to use this design for some of the same reasons as why we did not want a pretest. Participants using the application in the experiment should not have used it before, as that most likely would have influenced how they would behave and what they would have expected from our application. A further discussion on the design choices follows in the discussion chapter.

Subject of research

Sampling strategy

To select subjects, or participants, for the research we used the non-probability sampling strategy called convenience sampling. A convenience sample is also called a accidental or availability sample, as it consists of subjects from those easily accessible (Berg, 2004). The great advantages of this strategy is that one can obtain a sample without spending a lot time and money (C.Cozby, 2007). As the research was carried out to study variable relationships and not to accurately estimate some values of the population, we concluded that this sampling strategy would be sufficient. By inviting fellow students as participants we quickly got a subject sample for our research. We chose to use fellow students because they were close at hand and because we figured they would be more willing to participate than others considering our shared interest in the special field. Additionally, by inviting only master students to join, the sample can be seen as being a relatively homogeneous group. This we think could have been important to the experiment, especially since we did not have a large sample. Because of the small sample, the randomization would not work as well as in larger samples, but by using somewhat similar subjects we hoped to reduce this problem. Randomization is discussed further in the discussion chapter.

Our sampling frame was the master students belonging to the department of information- and media-science at UIB. To select our research subject we sent an e-mail to all the people in the sampling frame. The e-mail contained a short description of the experiment and an assurance that anyone could participate, as there were no demands for any particular skills or knowledge. The e-mail said nothing about our motivation for the experiment and especially nothing of the research question or about the existence of two different prototypes of the application. This was important to the experiment, because, as discussed earlier, having participants knowing the research question could ruin the experiment.

Sampling Size

For the experiment we only had ten subjects. Due to the projects limits, this is the number of subjects we thought to be many but still manageable. A larger sample would probably have made both the experiment phase and the analysis phase too time-consuming. According to Patton (Patton, 2002), the sample size depends on “what you want to find out, why you want to find out, how the findings will be used, and what resources (including time) you have for the study” (Patton, 2002, p. 244). In this study we are trying to find indications on whether or not users expects certain things or behave in a certain way when interacting with a web application using asynchronous communication. Therefore we believe that we do not need a very large sample. If we could, we would have used more subjects, but considering the time limits for the projects ten subjects is a good compromise. In our opinion ten participants would be sufficient to get an indication on the types of expectations and behaviors we are to study.

Settings

The experiment sessions were carried out in a regular office environment. The office was located in the same building as the research subjects had their reading rooms, so the subjects did not have to walk very far to participate. The technical equipment consisted of a test machine, which was a laptop with screen-capture software installed, and a video camera. The screen-capture software 'Camtasia' was used for recording on-screen activity, and the video camera on a tripod was placed to record the keyboard and mouse unit of the test machine. In addition, for audio recordings, one microphone was attached to the test computer and another attached to the video camera. We were both present in the lab with the participant to directly observe the interaction. This gave us the chance to ask questions or prompt the participant along the interaction, and to keep an eye on the recordings. It also gave the participant a chance to ask questions relevant to the exercises. We felt it would be better for the participant if one of us kept in the background and said as little as possible. In that way the participant only had to relate to one experimenter instead of two. This led to two roles: One experimenter, seated next to the participant, was in charge of talking to the participant and directly observe the interaction. The other experimenter was in charge of the experimental settings and equipment, staying in the background. For each session we switched roles. A sketch of the experiment environment is shown in figure 12.

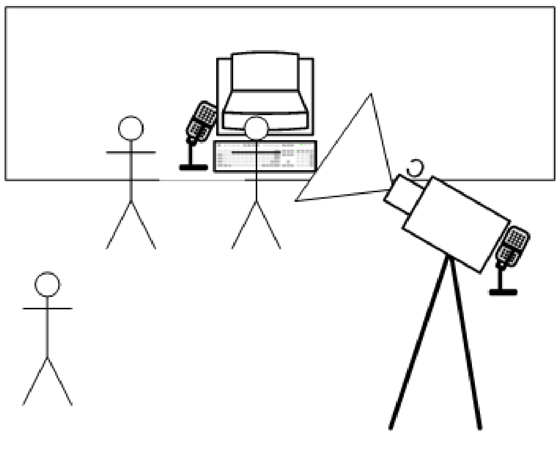


Figure 12: Sketch of the experiment environment

Data Collection

In this section we describe the methods used for collecting the data to be analyzed. We first describe general observation and then screen-capture, video- and audio-recording which is a part of our observation. We finish off by describing the technique called “think aloud”.

Observation

We needed a method for eliciting the users’ behaviors and expectations of our application. The best method seemed for us to be to observe the participants in action. As Sasse (1991) points out, the users cognitive models cannot be observed directly. One can only, by for instance studying the interaction between user and system, try to infer the user’s model. Another method for eliciting the user’s behavior and expectations could be interviewing. Our choice fell on observation because interviews does not necessarily reflect what happens, they only reveal how people perceive what happens (Bell, 2005). Observation on the other hand reveals whether people do and behave in the way they claim to. According to Frankfort-Nachmias and Nachmias (1996) the directness of observation enables behavior to be studied as it occurs. This is a big advantage of observation.

Although observation reveals what people being observed actually do, Bell (2005) points out that observation depends on the way the observers perceive the actions and utterances of the subjects. To be sure the interpretation was as correct as possible, the observation was done not only in the form of direct observation of the interaction but also in the form of video- and audio recordings as well as screen-capturing. This made it possible to observe one experiment session over and over, until we were satisfied with our interpretations.

Another reason for choosing the observation method is that while observing we got a chance to ask questions or prompt the users during the experiment. Questions like ”why are you pushing that button?” or ”What do you expect should happen now?” could be asked. Additionally by being present we could ask the participant to do a certain task or skip a task that would be unnecessary for the specific subject to do. When participants were uncertain whether to do a specific action or not we could say: ”Why don’t you try?”, instead of the participant ending up not doing it.

Bell (2005) separates structured and unstructured observation. In unstructured observation you do not necessarily know what the focus of the observation is before your study. Rather, the focus would emerge after spending some time observing (Bell, 2005). In structured observation on the contrary, Bell says you have already identified the objects and aspects of behavior to study and you have a hypothesis ready for testing. In structured observation the focus is decided on beforehand. Our observation can be said to be of the structured type as we had a specific focus and knew what aspects of behavior we were to observe. We had a checklist consisting of expected actions for use during observation (see fig. 20 on page 60). Still this was not a yes/no answer type of experiment, and we could not exclude the possibility of discovering other aspects than initially anticipated. As there might have been checklist items that we had failed to map, one could argue that the focus was not completely set, making the observation "semi structured".

The observation method is also separated into "open" and "hidden" observation. The choice here fell on an "open" observation, where the persons being observed are aware that they are being watched (Dalland, 1993). This was a natural choice as we wanted to be able to observe the participant in their interaction with the application and being able to ask the participants questions during the experiment. Asking questions during and not after the experiment we believe gave us more honest answers as the participants might not have remembered all details of their interaction at the end. Additionally asking at the end could have made them think twice about their answers, trying to figure out what we would like to be answered rather than expressing what they really felt.

During an experiment, even a trained observer would not be able to keep track of every little detail. We therefore wanted to, as mentioned earlier, record the experiment so that it could be observed several times. To capture the participant's interaction with the application and to later analyze the actions and utterances, we used screen-capturing software in addition to a video camera recording the keyboard and mouse. There was also a need for audio recordings to capture the verbal reports, which could give us an indication of what the participant thought.

Screen-capture The screen-capture tool was useful for capturing and later analyzing the actions the participant performed in the application. This tool provided us with information such as mouse movements, button clicks and menu selection, and it supported us in understanding the chronological thought-process of the participant. The screen-capture tool only captured what was happening in the application, on-screen actions, and excluded the surrounding context. To capture the activity off the screen as well we needed to do an audio- and video based (and also a direct) observation in addition. The participants' verbalizations and behavior were also important aspects of the interaction.

Video recording One of the big advantages of recording the experiment on video was that the experiment could be replayed, over and over again. Recordings gives the opportunity to, for instance, watch one specific scene several times, or to play a scene in slow or accelerated motion. The behavior of our participants could thus be studied in detail and more easily and precisely interpreted and categorized. As Heath and Hindsmarch (2002) put it:

> “They [videotapes] allow us for example to track the emergence of gesture, to determine where people are looking and what they are looking at, and to recover the ways in which they orient to and handle objects and artifacts.” *(Heath and Hindmarsh, 2002, page 8)*

Given the opportunity to study the details, the chances of discovering the more hidden patterns or phenomenon in the behavior increase. Hence the analysis of video based observation could be more complete than with standard observations (Erickson, 1992).

Using video cameras past experiments can be reconstructed closely up to its original form (Mehan, 1979). It would be just approximately the original form because, as Jordan & Henderson (1995) writes, all transformations are less rich than the original event, there will always be some loss of information. Still video recording is the kind of data collection method that loses the least data from the original event. Since the videotapes are almost identical to the original form, a high degree of inter-subjectivity would be preserved. The videotapes are kept in its raw form; it is not interpreted information. Other researchers could perform measurements on the same material. One could therefore say that videotapes preserve a high inter-subjectivity (Hellevik, 1991) in research. According to Jordan and Henderson (1995), the decisions made by the video camera operator and the characteristics of the technology are primarily the sources for the video recordings not being identical to the original form.

One drawback with video recording is that the persons being observed can be influenced by the presence of the camera. The person could get nervous and behave or react differently than he or she normally would do, and that could produce data that are not depicting reality. Luckily people accustom themselves to the camera surprisingly quickly, and that especially if there is no one operating it (Jordan and Henderson, 1995). Additionally the tape itself can provide evidence that the camera influenced the participant, for instance revealing the participant always turning its face away from the camera, or regularly looking into the camera, and changes in behavior could indicate that the participant’s awareness of the camera fades out as time passes (Jordan and Henderson, 1995).

Audio recording To capture the participants’ verbalization of their thought process, audio recordings were needed. Audio is a very important supplement to video and especially important to our project because of the central role of the think-aloud technique. Having recorded the participants' verbal reports gives some advantages compared to other kinds of qualitative data. Audiotapes can be replayed many times and one could transcribe the utterances for analysis. For our purposes, pauses and prompting could be just as important than what is uttered. All these characteristics are preserved using recordings.

Think-aloud / Verbal reports

In qualitative methods it is highly important that the subjects are given the chance to express themselves with their own words, as that is mainly how the researchers accesses the phenomenon to be studied (Gentikow, 2005). With this in mind and considering the fact that users expectations and thoughts cannot be directly observed, we wanted to use some method of verbal reporting. Verbal reporting means that users express their opinions and thoughts on a given task. Someren et el. (1994) speak of several different methods for verbal reports. "Retrospection" is when the user is questioned on his or her thought process after solving the task at hand. "Introspection" is when the user is to speak his thoughts at intermediate points, which the user himself chooses. With "questions and prompting" the user can be asked questions or prompted during the task solving, and with "dialogue observation" the user is in dialogue with another person, perhaps in collaboration or in a learning process. Finally there is the "think-aloud" method where the user is simply asked to verbalize what he or she is thinking while solving a task.

In our experiment we chose to use the think-aloud method to get the participants to verbalize their thoughts and conclusions while interacting with the application. We chose this method because of some advantages compared to the other verbal reporting methods. The think-aloud method does not interrupt the user's thought process in the way that other methods, like for instance questions during the task, would do. Furthermore the think-aloud method does not leave room for "memory errors" (van Someren et al., 1994) like with the retrospection method where the user have to remember the thought process after the task is done (van Someren et al., 1994). In addition we would avoid having the participants interpret his or her thoughts because, as Someren (1994) says, the verbalization is done almost automatically and the participant would render the thoughts just as they arrive. The think-aloud method allows the participants to use their own language, making it easier for the participant to express their thoughts. According to Someren (1994) the think-aloud method would in just a few minutes become routine to the participant, and he or she would be so consciously focused on doing the task that the participant would not be reflecting on what he or she is doing and saying. The data would therefore be direct; there would be no delays that could allow memory errors or interpretation. Even though questions or prompting would cause disturbance in the participants' task performance and lead to memory errors or interpretation, this was sometimes needed to get the participants going. Even though Someren says thinking aloud would become a routine, one cannot expect all participants to be able to manage the think-aloud method perfectly. Additionally we sometimes needed to ask more detailed questions to really understand why the participant acted as he or she did. Sometimes we needed to specifically ask the participant to express verbally what she or he expected, or to reason about the application's behavior.

The Research Process

Before the actual experiment started, we conducted a pretest of the actual experiment to see whether the research methods were fitted or if adjustments to the research design were needed. According to Cozby (2007), pretesting would also reveal whether the participant understood the assignments. In addition, pretesting gave us an opportunity to test the technical equipment, and to get familiar with the experimenter role. Note that the pretest was done using a subject not participating in the actual experiment.

In the actual experiment we performed ten sessions, with only one participant per session. Before the experiment session started we explained to the participant what was going to happen, describing the think-aloud method and the assignments the participant was to perform. For instance we explained that the video camera was used only to record the keyboard and mouse unit, and that a screen-capture software would record what happened in the application. We hoped this would make the participant feel more comfortable with the situation.

The participants were given a set of exercises to perform using iRIS, and they were asked to verbalize their thought process while performing the tasks. If the participant was unfamiliar with RIS modeling we gave a short introduction on RIS models and elements. These introductions were comprehensive enough for the participants to understand the assignments. The assignment was split into two exercises. In the first exercise the participant was presented with a RIS model, which they were to copy using the iRIS application. In the second exercise the participant was given a sheet describing how to change the model they just created. The exercises can be viewed in the Appendix.

After the participants had completed the assignments, we asked them a few questions. First of all we asked them questions regarding the application hoping to catch things they had thought during the interaction but not yet expressed. For instance, we asked them if they had any comments on the application, and whether they thought the assignment went well. Additionally we asked them whether they had used any Rich Internet Applications before and what kind, whether they had tried the drag and drop technique on the Internet before, and what kind of modeling tools they had used previously. The questions were asked at the end of the experiment session to avoid affecting the participant in the interaction. During the experiment session the participant were less likely to guess the purpose of the experiment since the questions were asked afterwards. As discussed earlier in this chapter, participants knowing the purpose of the study might behave differently then when not knowing the purpose.

Analysis Methods

Once the experimental phase was done, the data material was prepared for the analysis phase. This section describes how the data material was handled and the methods used for analyzing the collected data material.

Research data handling

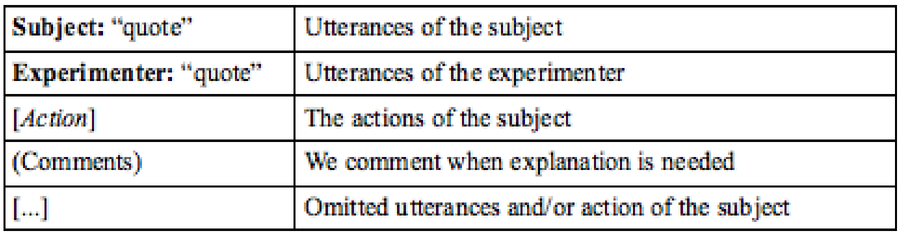
As described in the section on research methods, all the experiment sessions were recorded with a digital video camera, a screen-capturing software and microphones. After the experiments, the video recordings were transferred to a computer using the video editing software ”iMovie” and saved using the Quicktime file format with maximum resolution. The screen-captures were recoded using Camtasia studio and saved in the AVI file format. Both the video- and screen-capture recordings were then transferred to an external hard-drive for storage. Audio was recorded together with both the video and the screen-capture. This was done in order to allow easier synchronization between the two recordings, and to have a backup in case one of the audio sources was lost. The primary audio source was the audio recorded with the screen-capture.

In order to organize the data material, as well as keeping the research subjects anonymous, they were given a number in the range one to five and a letter referring to the group. The subjects in the experiment group are referred to as E1 through E5, and the subjects in the control group as C1 through C5.

Transcription

In the analysis, when trying to understand why the subjects acted as they did, their utterances would play an important part. The subjects' verbal reports would give us some valuable hints on what the subjects were thinking and thereby hopefully explaining their actions. In order to analyze the reasons of actions more carefully we transcribed both the subjects' utterances and their actions. In the next chapter we introduce a checklist of actions we were to look for in the subjects interaction. During the viewing of the recordings we transcribed the utterances of the subjects whenever they would perform any of the actions listed in this checklist. With the transcription we had a step-by-step explanation of what the subject did and thought, and this made it easier to discuss and analyze the events.

Some of the transcriptions can be found in the analysis and results chapter were they act as examples of results. The transcriptions are formatted to ease reading and understanding. The syntax of the transcriptions as well as an example of usage is shown in figure 13 and figure 14. Utterances of the subject is prefixed with “Subject:” and utterances of the experimenter is prefixed with “Experimenter:”. All actions are described inside brackets and with italic font style. Text inside parentheses is our added comments that are suppose to help understand the example. This can be explanations of program functionality or our interpretations of the subjects’ utterances. Three dots inside brackets indicate utterances or actions that are omitted because of its irrelevance to the example.



Subject: “quote”	Utterances of the subject
Experimenter: “quote”	Utterances of the experimenter
[*Action*]	The actions of the subject
(Comments)	We comment when explanation is needed
[...]	Omitted utterances and/or action of the subject

Figure 13: Syntax of the transcriptions

Subject: ”But, you can’t..no..drag it over”
[Tries to drag the activity-object]
Subject: ”So far it hasn’t worked at least”
[Clicks the activity-object]
Subject: ”But can I do this?”
[Pushes the delete button on the keyboard] (this removes the text in the name field of the ’Editing pane’)
Subject: ”No. You cannot use delete to...” *[pauses]*
Experimenter: ”No...”
Subject: ”That is a typical shortcut which...” *[pauses]*
Experimenter: ”Yes...”
Subject: ”...many uses to.. surely in these context... contexts”

Figure 14: *Transcription example from analysis and result chapter.*

The progress of the analysis

For each of the subjects we had one screen-capture recording showing on-screen actions, and one video recording showing the hand movements of the subject. During the analysis we wanted to watch both recordings at the same time, to get a complete overview of the activity of the subject. The on-screen actions and the keyboard and mouse actions are closely linked. Watching the two videos at the same time was therefore essential for the interpretation of the actions.

The analysis was conducted using a laptop computer with an extra screen. The laptop was connected to the external hard-drive containing the videos. The screen was placed right next to the laptop and the video recording was played on this screen. The screen-capture recording was the main source for our analysis, and the video recording was mostly used for support. Our eyes were mainly focused on the screen-capture recording, glancing over at the other screen as needed.

Watching the videos together, as two observers, reduced the chances of overlooking something important. In front of us we had a checklist, resulting from the analytical framework, containing expected actions (see fig. 20 on page 60). If the subject performed a listed action we wrote down the elapsed time of the video, in order to go back and take a closer look. We did this to keep the natural flow of the videos, and to be able to watch them continuously without constantly having to pause, rewind and stop the videos. As we took a second and closer look at the relevant actions, we transcribed the behavior into detail. Interesting points that were expressed verbally were also transcribed.

The result of the analysis is presented in the analysis and results chapter.

Summary

In this chapter we have presented the research design as well as research and analysis methods for this study. We have explained our experiment and how it was conducted by having an experiment group testing the Ajax-based prototype and a control group testing the application with traditional synchronous communication. The experiment sessions were both videotaped and recorded by screen-capture software. Open and unstructured observation was used in addition to the think-aloud method as a form of verbal reporting. For analysis we transcribed the parts of the interaction we found interesting, which would also be the behavior found in the checklist. The checklist is described in the following chapter where we present the analytical framework developed for capturing and categorizing findings.

Framework

In this chapter we present the analytical framework that we developed for the purpose of capturing and categorizing the findings of the experiment. The framework consists of categories describing common ways for humans to interact with computers, and of common elements used in such interaction. Despite the large interest in the field of human computer interaction and the interest of usability and interaction elements, such a framework does not seem to exist and we therefore decided to create one ourselves. In this chapter we first describe why and how the framework was built. We then present the taxonomy category by category.

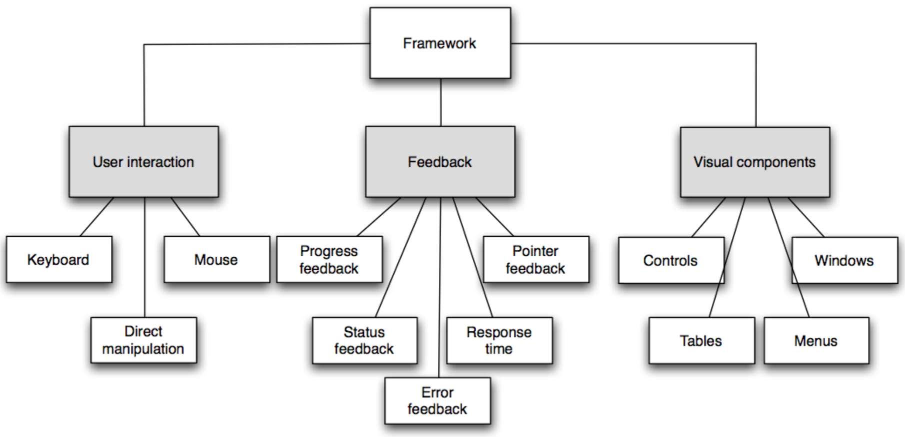


Figure 15: Overview of the categories

The use of the framework

Before we could start the research on user interaction and conventions, we needed to gather information about the differences in common functionality and common elements between traditional desktop applications and traditional web applications. To get an overview of this information we decided to organize it into a taxonomy and use this as a basis for an analytical framework.

The framework presents elements and functions that are common in traditional desktop applications, contrasting them to those found in traditional web applications. It is important to note that the framework is developed with our specific application in mind, including only elements relevant for evaluating iRIS. In addition to providing a general overview, the framework was used as a basis for a checklist for use in the analysis phase and for categorizing findings.

There are several reasons for mapping traditional web applications and not Rich Internet Applications. First of all RIA is a relatively new concept, and we have not found any style-guides for this type of application. As there do not seem to exist many established conventions, it would be difficult to map RIA elements and functions. Most importantly, studying traditional web applications would probably make the influence of Ajax more obvious than if we were to study RIA.

How the framework was developed

As a foundation for mapping the elements and functions common to desktop applications, we studied two well known style-guides, the ”Apple Human Interface Guidelines” (Apple Computer, 2006) and the ”Java Look and Feel Design guidelines” (Sun Microsystems, 2001). These two guidelines are both well structured and credible, and their use of categories made them easy to read and to navigate.

From these two style-guides we drew a sample of elements and functions that appeared in both, and that at the same time were relevant to our application and organized this into a taxonomy. We then studied the W3C HTML 4.0 specification (Raggett et al., 1997) and the CSS level 2 Specification (Bos et al., 1998) to get an impression of which elements exist in traditional web applications. As these sources do not give as complete an overview of the elements and functions in traditional web applications that we needed, we supplemented with other sources such as articles and books and also our own experiences as advanced Internet users. Based on the taxonomy we wrote a definition for each category and explained how the category works in practice in desktop applications and on the web.

The framework was created with our application in mind, but we believe that with few modifications it could be used to perform similar experiments on other applications.

Taxonomy

In the following sections we present the categories resulting from the merging of style-guides and the traditional web application resources. We start by describing common means of user interaction, after which we describe the category feedback. Finally we categorize different visual components. Figure 42 in the appendix shows the categories hierarchically structured. Each main category in the taxonomy is illustrated by a sub-hierarchy detailing that category.

User interaction

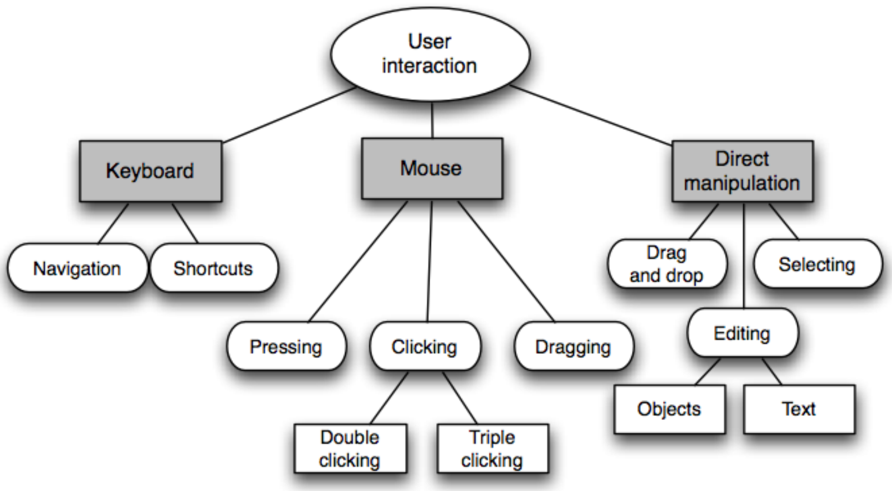


Figure 16: User interaction categories

Mouse

The purpose of the mouse is to move the on-screen cursor, or pointer. This is done by moving the mouse without pressing any of the mouse buttons (Apple Computer, 2006). The pointer on the display copies the movement of the mouse.

Clicking Clicking means pressing a mouse button and then releasing it. This selects or activates what is located beneath the pointer [Sun 2001]. If the click is supposed to trigger an action, the action takes place when the mouse button is released (Apple Computer, 2006).

Double clicking Double clicking means clicking twice, where the first click is immediately followed by a second click (Apple Computer, 2006). The purpose of this action is to select and open an object, or to select a larger unit (Sun Microsystems, 2001).

Triple clicking Triple clicking means clicking three times in a row. According to Apple some desktop applications support this type of clicking.

> ”For example, in a text-oriented application, the first click sets an insertion point, the second click selects the whole word containing the insertion point, and the third click might select the whole sentence or paragraph.” (Apple Computer, 2006, p. 273).

Triple clicking is in our experience common only for selecting larger chunks of text in desktop applications.

Pressing Pressing means holding the mouse button down while the mouse pointer stays put. This should not cause any other effect than what clicking does (Apple Computer, 2006). Pressing is a part of the actions selecting and dragging.

Dragging Dragging means moving the mouse after pressing the mouse button but before releasing it. This is used for selecting multiple objects, for moving objects or for choosing items from a contextual menu or a drop-down menu (Sun Microsystems, 2001). Dragging is also a central part of drag and drop.

Mouse interaction in traditional web applications The mouse pointer itself has the same function on web- as in desktop applications, to be able to move around on the screen. There seem, however, to be some differences in usage.

In our experience, single clicking works similarly on the web as in desktop applications, but single clicking on web can generally not be used for selecting. Clicks are on web commonly used for activating hyperlinks or buttons. Wroblewski (2001) writes that single clicking is a part of the established convention of the WWW, and web users are accustomed to single-clicks while interacting. Additionally, Shubin (1998) relates double-click interfaces to desktop applications while relating single-click interfaces to browsers. In our experience, double clicking on the web is used only when selecting text, not for manipulating objects like in desktop applications. Triple clicking, pressing and dragging also seem to only be commonly used when selecting text, both on the web and in desktop applications. Dragging is a part of the technique drag and drop. Drag and drop is, as we discuss later, not a feature commonly used in traditional web applications.

Keyboard

In desktop applications the primary use of the keyboard is for entering text, but it is also quite commonly used for navigating. The keyboard should work as an alternative to the mouse (Apple Computer, 2006).

Navigation In desktop applications, keyboard navigation means moving the keyboard focus from one interface element to another using the keyboard. The keyboard focus, or 'input focus', is where the next keyboard input will appear (Sun Microsystems, 2001).

Shortcuts In desktop applications, keyboard shortcuts provides a way to rapidly activate menu items using keystroke combinations, and are usually a composition of a modifier key followed by a character key (Sun Microsystems, 2001).

Keyboard interaction in traditional web applications In our experience keyboard navigation on the web consists mostly of using the arrow keys for scrolling the web page and the 'tab' key to jump between input fields in forms and in some cases between hyperlinks. Using the keyboard to invoke shortcuts seems to be uncommon in traditional web applications in contrast to desktop applications. The exception is shortcuts defined by the browser. Silver (2006) goes as far as saying that in web applications no keyboard shortcuts exist as an alternative to using the mouse.

Note that we have not taken into consideration that some people with disabilities use built in browser functionality to navigate the web using keyboard and specialized software.

Direct Manipulation

Selecting Selecting an object in desktop applications is done in order to perform an operation on it. Immediate visual feedback is used for signaling what has been selected (Apple Computer, 2006). Selection of one object can be done by clicking it. Selection of a range of objects can be done by dragging or by 'shift-clicking'. Shift-clicking means holding down the Shift key when clicking (Apple Computer, 2006).

In our experience, selection of single and multiple objects are closely related. If selection is possible, selection of multiple objects is usually possible as well.

Editing

Editing objects In our experience desktop applications often provide ways to edit objects, like changing the color or shape of the object, by performing operations on it directly.

Editing Text To insert text in a desktop application one clicks where the text should be placed before typing on the keyboard. To delete text the user presses the 'delete' or the 'backspace' key on the keyboard (Apple Computer, 2006).

> “A text field is a rectangular area that displays a single line of text. A text field can be editable or non-editable” (Sun Microsystems, 2001, ch.11).

Drag and Drop Drag and drop is a technique for dragging an element and dropping it in a suitable place (Apple Computer, 2006). This technique is used in desktop applications for moving, copying and linking objects, and is, according to Sun (2001) a “[...]convenient and intuitive way to perform many tasks using direct manipulation” (Sun Microsystems, 2001, ch. 6).

Direct manipulation in traditional web applications According to Zhu (2001) selection and direct manipulation of an object is something that is hard to implement in traditional web applications due to its limited interaction model. Traditional web applications provide no means for directly editing objects (Zhu, 2001). This is consistent with our own experience. Selection of an object is not something we consider common on the web. Selecting a range of objects seems even less common. Object editing on the web seems to be uncommon except when interacting with form elements. Editing text on the web is in our experience only common within input or text area fields, other text elements can be marked and copied but not edited. Drag and drop is also in our experience not common in traditional web applications.

Feedback

Feedback is an important aspect of both desktop applications and web applications. According to the Java Look and Feel Design Guidelines ”Users interact more smoothly with your application if you keep them informed about the application’s state” (Sun Microsystems, 2001, ch. 6). Feedback plays a role in interaction and is an important factor for the users perceptions of an application. We therefore wanted to gather information on the differences in feedback between desktop and web applications.

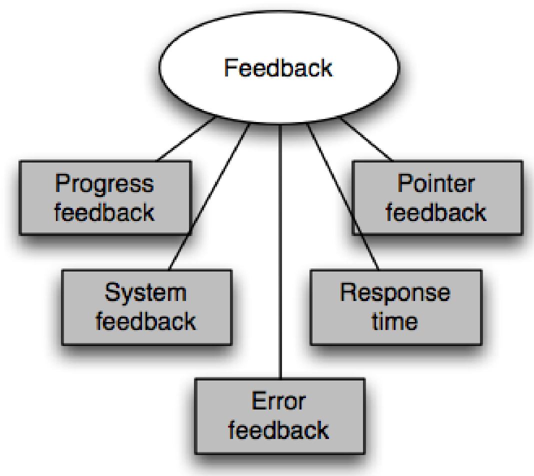


Figure 17: Feedback categories

Progress feedback

Progress indicators in desktop applications keep the users of the application informed about the application state (Sun Microsystems, 2001). These indicators come in different forms from a simple indication that something is happening, to more of an estimation of how long an action will take to perform. The user can be presented with that estimate, for instance in the form of a progress indication bar. In traditional web applications the browser provides feedback that a new page is loading, for example by displaying a progress indicator at the status bar or another form of activity indicator. Progress indicators on the web are in our experience only displayed when a page is reloaded. The page reload may in our opinion also be thought of as a kind of progress indicator. In desktop applications a progress indicator can also show the progress of individual objects, not just for entire pages like what is common on the web.

Pointer feedback

The mouse pointer can change its appearance, its cursor, according to its positioning and context. The shape of the cursor is used for providing feedback to the users about specific functionality or the general state of the program's execution (Apple Computer, 2006).

> ”The pointer can assume a variety of shapes. For instance, in a text-editing application, the pointer might assume an I-beam shape to indicate where the insertion point will be if the user presses the mouse button” (Sun Microsystems, 2001, ch. 6).

Both desktop applications and web browsers give feedback through the mouse cursor. Even though they seem to use the same standard cursor, the meaning of the cursor seems to vary.

The cursors that appear in both specifications mostly have the same usage definitions, but in some cases they are dissimilar. The text cursor and the pointer/hand cursor have a slightly different usage on the web compared to desktop applications. Here are the usage definitions for the text cursor:

> ”Selecting or inserting text ” (Sun Microsystems, 2001, ch. 6).
> ”Indicates text that may be selected. Often rendered as an I-bar” (Bos et al., 1998, p. 271).

In this example we see that in desktop applications the text cursor indicates that the text beneath the pointer can be selected, or that text can be inserted. On the web, however, the text cursors (except in input fields and text areas) symbolize selection only. In other words, text cursors in web applications usually do not imply that you will be able to insert text like it does in desktop applications.

Error feedback

Error dialog boxes are a common way to alert the user that an error has occurred. A dialog box describing the problem, sometimes with an error code, is displayed. Good error messages come, according to Apple (2006), with information on how the user can rectify the problem.

On the web there are at least two types of error messages; *system errors* and *user errors*. When the browser cannot connect to a server, due to for example network or server errors, a page explaining the error is displayed. A common example of this is that when a web server cannot find the page the user has requested; it displays the so-called ”404 error page”. Another common form of error reporting is displayed when a user has submitted a HTML form with errors in it. The form is in most cases redisplayed and an error message is displayed in plain text, usually in a distinct color or in a boldface font. Error dialog boxes are in our experience not that common on the web.

System feedback

According to Jacob Nielsen’s heuristics for user interface design, visibility of system status is an important factor in keeping the user informed about what is happening (Nielsen, 2005). Desktop applications have established ways to keep the user informed of the system’s state. To indicate that a document is saved or not saved, many applications use for instance the title bar to inform the user of this. In our experience this is not common on the web. This may be because information is usually only saved when the information the user has submitted is sent to the web server.

Response time

In our experience a desktop application will in most cases give the users immediate feedback on their actions. Web applications, however, are limited by two factors that make immediate feedback hard or impossible; network latency and full page reload.

Network latency is a problem that is hard to solve. Due to the web applications communication with the server, there will always be a delay. Desktop applications are in most cases not dependent on communication with a server commonly storing their data on a local hard drive.

Full-page reload in traditional web applications adds to the problem of network latency since a whole page of information will have to be sent from the server on each request. The full-page reload is not a problem while browsing documents, but in web applications the latency caused by this becomes a serious issue that is not present in desktop applications.

Visual Components

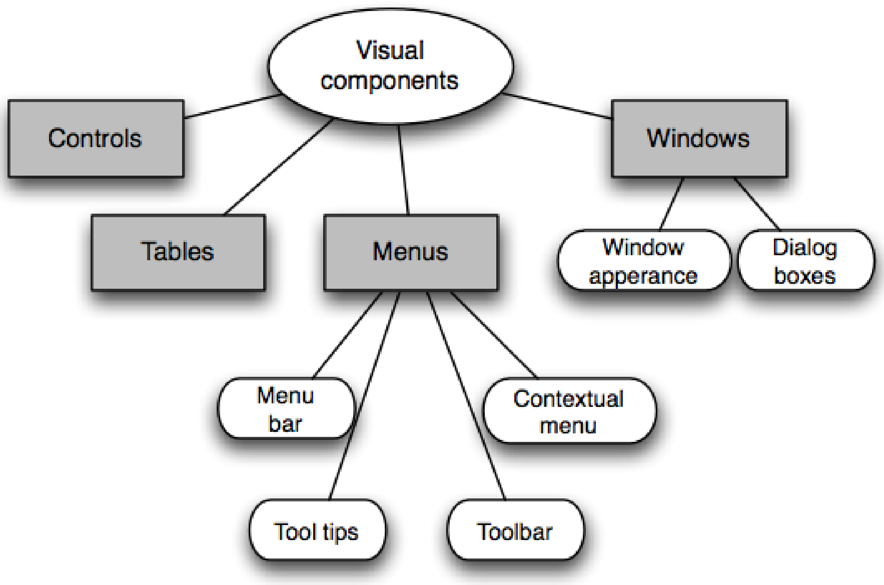


Figure 18: Visual component categories

Menus

A menu is a user interface element that displays a list of menu items that the user can choose amongst. The items, or options, can be commands, attributes or states (Apple Computer, 2006). Menus are typically grouped together and displayed in a way so that users do not need to memorize all of the available options (Sun Microsystems, 2001). According to Apple (2006), menus are specifically used for function seeking.

Menu bar One common element in desktop applications is the menu bar. A menu bar is a menu that appears across the top of the main screen, and consists of a row of pull-down menus (Apple Computer, 2006). People use menu bars to quickly find options they are looking for (Apple Computer, 2006).

Toolbar Toolbars are also quite common in desktop applications. Sun (2001) has this definition of a toolbar.

> ”A toolbar is a collection of frequently used commands or options that appear as a row of toolbar buttons. Toolbars normally appear horizontally beneath a primary window’s menu bar, but they can be dragged anywhere in the window or into their own window. Toolbars typically contain buttons, but you can provide other components (such as text fields and combo boxes) as well” (Sun Microsystems, 2001, ch. 9).

Contextual menu Contextual menus are a commonly used desktop feature that is brought up by pressing the right mouse button. In desktop applications, if there is an object underneath the mouse when right clicking, the contextual menu provides access to often-used commands associated with this object (Apple Computer, 2006). If there is no object underneath, more general commands would be presented by the contextual menu.

Tool tips A tool tip is a small rectangle of text that provides a short description about an interface element. This description is meant to help the user understand the meaning of this interface element. For the tool-tip to be visible one has to hover the area or the component with the mouse (Apple Computer, 2006, Sun Microsystems, 2001).

Menus in traditional web applications Menus in the forms described in Sun (2001) and Apple (2006) do not seem to be as common in traditional web applications as in desktop applications. According to Zhu (2001), a menu bar is something that is hard to implement on the web. In addition to this we believe users may be confused, as there is usually already a menu bar in the browser.

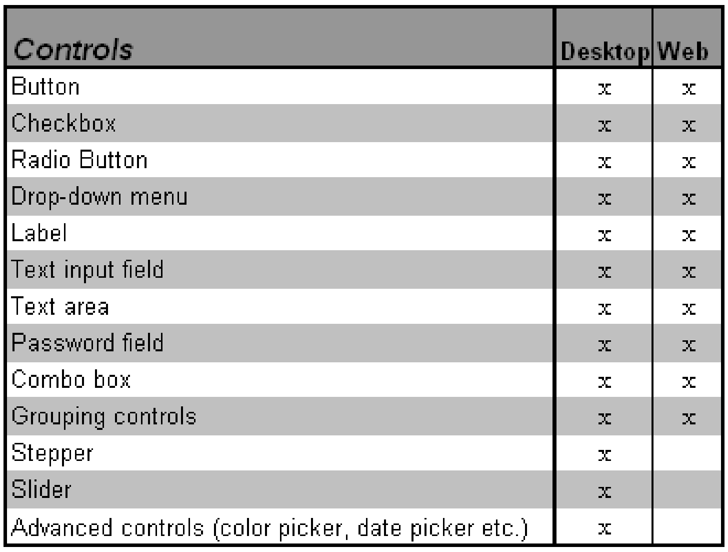
Toolbars are in our experience not common on the web. Toolbars inhabit certain features that do not seem to be common on the web. One of these is the ability to be dragged around the screen (Sun Microsystems, 2001). Most browsers have toolbars; a second toolbar could possibly be confusing to users.

According to Zhu (2001), a contextual menu is hard to implement in traditional web applications. In our experience the contextual menus that exist in the traditional web applications are related to the browser and not the actual application. This may also possibly contribute to confusion on the part of users.

The use of tool-tips is possible in web applications. The title attributes of some of the HTML elements are often displayed as tool-tips. In our experience, however, this is not a feature that is consistent between different browsers.

Controls

Controls in desktop applications are ”interface elements users can manipulate to perform an action, select an option, or set a value” (Sun Microsystems, 2001, ch. 10). Both Sun (2001) and Apple (2006) list a number of common controls. We tried to merge these, and we made a list of the most common desktop controls. To find out whether or not these controls where used in traditional web applications we studied the HTML specification (Raggett et al., 1997). Figure 19 lists all these controls indicating whether or not they are common on desktop or on the web.



Controls	Desktop	Web
Button	x	x
Checkbox	x	x
Radio Button	x	x
Drop-down menu	x	x
Label	x	x
Text input field	x	x
Text area	x	x
Password field	x	x
Combo box	x	x
Grouping controls	x	x
Stepper	x	
Slider	x	
Advanced controls (color picker, date picker etc.)	x	

Figure 19: Common user interface elements

As one can see in figure 19 on the preceding page, the only controls listed that do not seem to be commonly used on web is 'Steppers' and 'Sliders' as well as 'Advanced Controls' like color pickers or date pickers.

Tables

A table is a common desktop feature that helps to organize information and the relationships between the pieces of information (Sun Microsystems, 2001). A table consists of a series of columns and rows, and each field in the table is called a 'cell'. A cell can be selected and the user can edit its content if the table supports editing (Sun Microsystems, 2001).

Tables also exits in HTML, but are much more limited regarding direct manipulation. In our experience the HTML table is a static element that cannot be manipulated.

Windows

A window is a user interface element used in desktop applications that provides a frame for the users view and interaction with the applications (Apple Computer, 2006). Inside the window there are objects that are used for performing actions or provide information about the actions (Sun Microsystems, 2001). Apple (2006) distinguish between several different types of windows. 'Document windows' are windows containing file-based data, presenting a view of the created and stored content. 'Application windows' are not document-based but act as the main window of applications. 'Utility windows' present tools and controls for the user to work with on an open document. These utility windows float above other types of windows. 'Dialogs' and 'Alerts' are windows that prompt the user and require a response.

Window appearance According to Apple (2006), every document, application, and utility window should at least have a title bar, so the window can be moved, and a close button to provide users with a consistent way to dismiss the window. To close windows there are clearly marked exits, like a Quit or Exit command on the File menu (Shubin and Meehan, 1997).

Dialog Boxes A dialog is a window designed to elicit a response from the user. A dialog can be modal or modeless (Sun Microsystems, 2001). According to Sun (2001), a modal dialog box *"[...]prevents users from interacting with the application until the dialog box is dismissed"*. A modeless dialog box, however, *"[...]does not prevent users from interacting with the application they are in or with any other application." (Sun Microsystems, 2001, ch. 8)*

Windows in traditional web applications According to Silver (2006), using multiple windows in web applications can be problematic. One of the problems of using windows in web applications is, as pointed out by Wroblewski (2001), that *"unfamiliar representations of windows may not be perceived as windows, but rather as HTML tables or other visual elements" (Wroblewski and Rantanen, 2001, p. 3).* The users might not understand that the element is a window and that it can be manipulated. Another problem is that web applications usually do not have any exit paths, there are no way to stop 'running' the applications without terminating the browser and maybe losing the work in progress because of the lack of offers to save (Zhu, 2001, Shubin and Meehan, 1997). When it comes to dialog boxes they are not very common in traditional web applications, according to our experience.

Checklist

Our dependent variable, the usage of the iRIS application as if it was an desktop application, is a somewhat abstract concept, and we therefore needed an operational definition so that it could be studied empirically. We have defined the abstract concept of the dependent variable into concrete terms, based on our framework. This resulted in a checklist that consists of all the behaviors and expectations we were to look for in the participant's interaction with the application. These behaviors and expectations are in the framework associated with conventions for desktop applications. This means that if we can check an item in the list for a user, it could imply that this participant is using a desktop convention. At the end of the analysis we would have a list with all the items for all the participants and an overview of whether or not they are using desktop conventions. The checklist functioned as a good base for noting what we needed to take a closer look at, and to get a quick overview of the participant's behaviors and expectations. The reason for taking a closer look at the items was was to ensure that the behaviors were indeed caused by asynchronous communication and not by any extraneous variables. The checklist is divided into the same three top-level categories as in the framework. This was done in otder to enhance the understanding of where in the framework the checklist item is found. The checklist should reflect the biggest differences between desktop and web applications defined in the framework. Figure 20 on page 60 presents the checklist in its entirety.

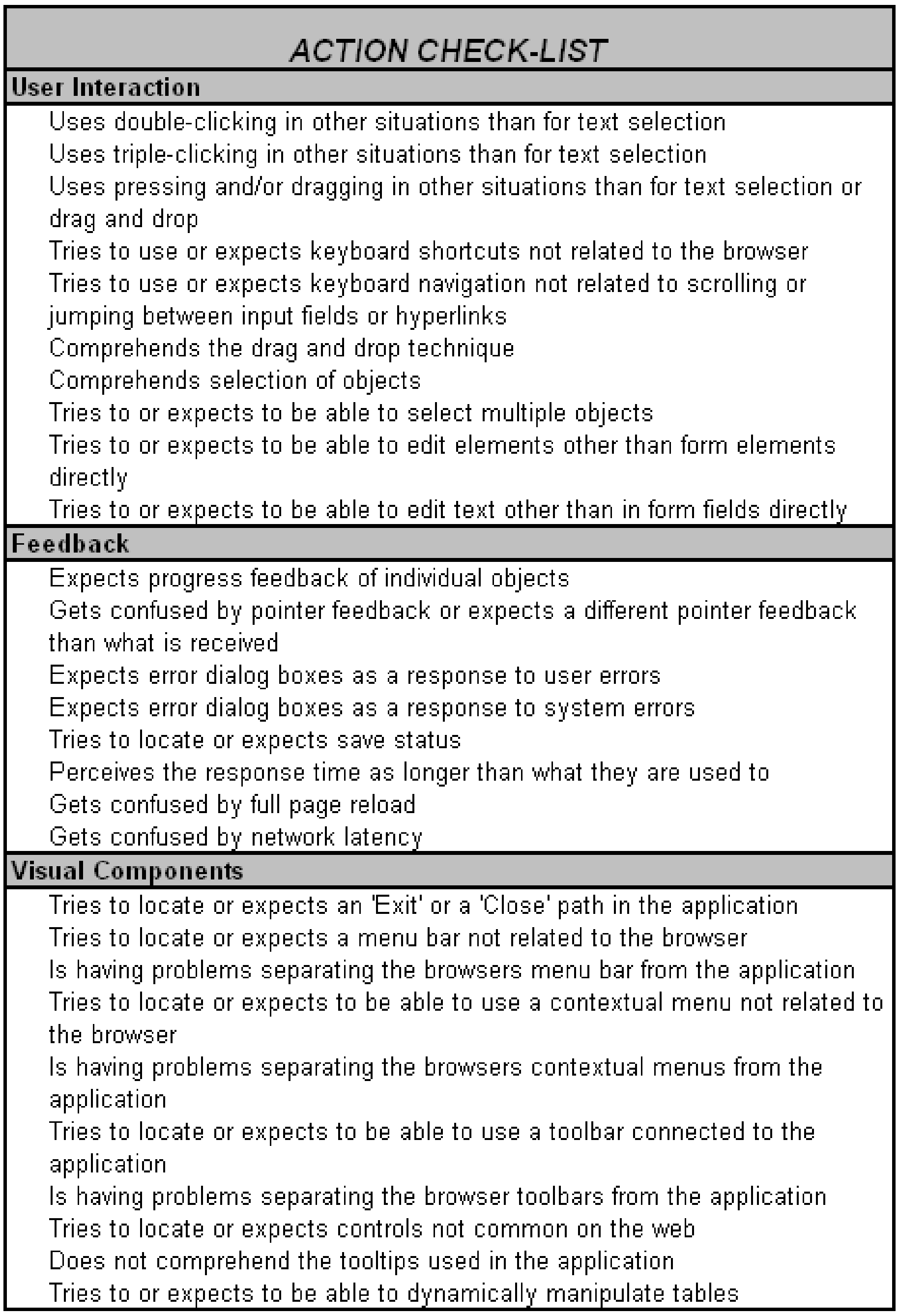
Summary

In this chapter we have attempted to map the main differences between the desktop and the web.

- Desktop applications seem to have greater use of the keyboard for navigating and more use of keyboard shortcuts.
- Desktop applications utilize double clicking for more than text selection, something that seems less common on the web.
- Direct manipulation seems much more common in desktop applications than on the web.
- Desktop applications seem to use menus and toolbars to a much greater extent than what is common on the web.
- Desktop applications seem to provide more feedback on specific objects than what is common on the web.
- Mouse pointers may have different meanings on desktop and on the web.
- Desktop applications typically show a "save status". This is less common on web, as information is usually only saved during a page reload.
- Error dialog boxes are common on desktop but are not much used on web.

Although many of the features we have mapped are not used in our application, we still feel that they are relevant as part of the framework, as the subjects in the experiment may express that they expected some of these elements to be part of the application.

In the next chapter we use the framework to categorize the findings from the experiment.



ACTION CHECK-LIST
User Interaction
Uses double-clicking in other situations than for text selection
Uses triple-clicking in other situations than for text selection
Uses pressing and/or dragging in other situations than for text selection or drag and drop
Tries to use or expects keyboard shortcuts not related to the browser
Tries to use or expects keyboard navigation not related to scrolling or jumping between input fields or hyperlinks
Comprehends the drag and drop technique
Comprehends selection of objects
Tries to or expects to be able to select multiple objects
Tries to or expects to be able to edit elements other than form elements directly
Tries to or expects to be able to edit text other than in form fields directly
Feedback
Expects progress feedback of individual objects
Gets confused by pointer feedback or expects a different pointer feedback than what is received
Expects error dialog boxes as a response to user errors
Expects error dialog boxes as a response to system errors
Tries to locate or expects save status
Perceives the response time as longer than what they are used to
Gets confused by full page reload
Gets confused by network latency
Visual Components
Tries to locate or expects an 'Exit' or a 'Close' path in the application
Tries to locate or expects a menu bar not related to the browser
Is having problems separating the browsers menu bar from the application
Tries to locate or expects to be able to use a contextual menu not related to the browser
Is having problems separating the browsers contextual menus from the application
Tries to locate or expects to be able to use a toolbar connected to the application
Is having problems separating the browser toolbars from the application
Tries to locate or expects controls not common on the web
Does not comprehend the tooltips used in the application
Tries to or expects to be able to dynamically manipulate tables

Figure 20: Checklist of user actions

Analysis and Results

In this chapter we present the findings of the experiment. The findings are illustrated by transcriptions of the subjects' actions and utterances, and by screenshots from the subjects' experiment sessions. We finish up the chapter with a summary containing the framework checklist filled out for all of the experiment participants.

Findings

The findings are presented according to the framework taxonomy, starting with user interaction, then feedback and finally visual components (see fig. 42 on page 96 in the Appendix). All of the transcribed examples follow the same syntax as defined in the research design chapter (see fig. 13 on page 44). The experiment participants are referred to as E1-E5 for experiment group subjects and C1-C5 for control group subjects.

User Interaction

Mouse

Considering the fact that iRIS is a web application, two of the research subjects used the mouse in a way that was, according to the framework, not consistent with traditional web application conventions. These observations were all related to double clicking.

Both research subjects E3 and E5 double-clicked RIS objects in the application. These subjects demonstrated an extensive use of double clicking. Figure 21 is an example where subject E3 is double clicking several objects.

In this example the subject seems convinced that he needs to double-click the RIS object in order to trigger the Editing pane. Subject E3 double-clicks all RIS objects during the creation of the first half of the first assignment, even though only single clicking is necessary. After double clicking for some time he started single clicking instead. Subject E5 mainly single-clicked to trigger

[Drags a RIS object onto the modeling area] [Double-clicks the object] [Enters the name of the object in the Editing pane] [Clicks the save button in the Editing pane] [Drags another RIS object into the modeling area] [Double-clicks the object and enters the name in the pane]

Figure 21: *Subject E3 is double clicking*

the Editing pane, but occasionally he double-clicked instead. Additionally he double-clicked when trying to insert Role objects (see figure 27 on page 66).

Only subjects in the experiment group tried double clicking RIS objects, the subjects in the control group used single clicking only.

Pressing and dragging was not used for anything other than text selection, except as a part of the selection and the drag and drop techniques. As for triple clicking we did not observe any use other than for text selection.

Keyboard

The keyboard usage in the application was almost entirely concentrated around textual input to the attributes of the RIS objects and the RIS model information, none of the subjects tried to use the keyboard for navigation. One of the research subjects, C5, seemed to expect to be able to use keyboard shortcuts and tried to use it to operate on a RIS object. In figure 22 the subject tries to move an object, but since that is not possible he tries to delete it instead, using the delete button on the keyboard.

Subject: ”But, you can't..no..drag it over”
[Tries to drag the activity-object into the RIS modeling area]
Subject: ”So far it hasn't worked at least”
[Clicks the activity-object]
Subject: ”But can I do this?”
[Pushes the delete button on the keyboard]
Subject: ”No. You cannot use delete to...?” *[pauses]*
Experimenter: ”No...”
Subject: ”That is a typical shortcut which...” *[pauses]*
Experimenter: ”Yes...”
Subject: ”...many uses to.. surely in these context... contexts”

Figure 22: *Subject C5 is trying to delete a object using the keyboard*

No other subjects tried using keyboard shortcuts for manipulating the RIS model.

Direct Manipulation

Most of the research subjects seemed satisfied with the ways they could manipulate RIS objects, but some wanted to be able to use more direct manipulation techniques.

Selecting All of the research subjects seemed to comprehend selection of an object as a way be to edit its attributes. A few went a little further and tried to select a range of objects at once. They seemed to expect to be able to operate on multiple objects simultaneously, which is not possible in iRIS. Using the mouse they marked several objects at once, and then tried to perform an operation like for instance delete. Subject E5 demonstrated this type of behavior while subject C5 expressed that he expected to be able to do it. In figure 23 E5 has just realized that he has forgotten to insert a RIS object, and that the object was supposed to go almost at the top of the model.

Editing

Editing objects One of our research subjects wanted to resize the name of the RIS object. Figure 24 is an example where subject E3 expresses his wish to change the layout of the RIS objects name.

Editing text Four of the research subjects tried to or seemed to expect to be able to edit the RIS objects' names by typing directly into the field where the name appears. Two of the subjects in the experiment group and two subjects in the control group tried to perform or seemed to expect in place text editing. These subjects were E1, E2, C2 and C5. Figure 25 shows an example where E1 is trying to mark the RIS objects name and type the name directly using the keyboard.

Drag and Drop All of the research subjects seemed to quickly comprehend the drag and drop technique. They located the Object pane, and they understood that the RIS modeling area was where they should drop the RIS objects. Additionally, all subjects tried to move RIS objects on the modeling area by dragging them. They did this for instance when they had dropped an RIS object into the wrong cell, or when the assignment involved moving a RIS object.

Subject: ”Aah, and there too” (Realizes that he is missing an RIS object near the top of the model)
Subject: ”What to do now then? I have forgotten things in the middle of the model, so now maybe I’ll have to do everything all over again”
[Uses mouse pointer to point to the area where the missing RIS object should be placed]
Subject: ”Unless I can move it all then?” *[Tries to move one of the RIS objects by dragging it]*
Subject: ”That doesn’t work...[I will] try to mark them all”
[Tries to mark all the RIS objects he wants to move by using the mouse.] [Clicks the delete button in the ’Editing pane’]
Subject: ”I’ll just have to delete them all then...That doesn’t work... then I’ll have to delete all of them manually”

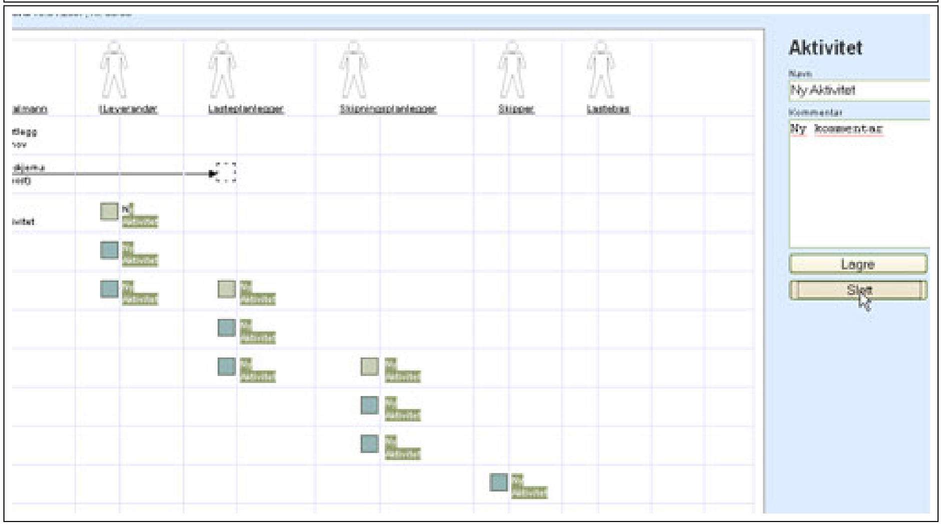


Figure 23: *Subject E5 is trying to select a range of objects and delete them.*

Feedback

Progress feedback

Three of the subjects had problems related to progress feedback. Figure 26 and figure 27 are two examples where subjects get confused by the lack of feedback.

In figure 26 on page 66, when E1 did not get immediate feedback on his action, he seemed to get confused, probably thinking that he had done something wrong. When the object did appear, he complained that the application was slow. In figure 27 on page 66, E5 also seemed to have problems due to the lack of feedback during a delay. In this case it led him to think that he had to double click an object in order for it to appear in the modeling area.

Some of the subjects seemed to get confused due to the lack of a progress indicator on individual objects. In figure 28 subject E3 explicitly expresses a

[Writes the name of the RIS object]
Subject: ”But when you place this text here, isn’t it possible to stretch it out? Or will it automatically be placed down? (onto the next cell) ..you get it..”
Experimenter: ”It places itself automatically, as it wants to”
Subject: ”So I cannot adjust it, so if I want to stretch it out?”
Experimenter: ”No, you can’t do that”
Subject: ”Okay”

Figure 24: *Subject E3 is asking about the possibility of resizing objects.*

[Drags a RIS object onto the modeling area] [Clicks the RIS object which triggers the Editing pane] [Clicks ’Backspace’ and then ’Enter’ on the keyboard] (the name is not removed) *[Uses the mouse pointer to mark the RIS objects name in the Modeling area] [Writes the name of the RIS object with the keyboard]*
Subject: ”It looked like I could edit the text of the object, but not.. *[inaudible]*”

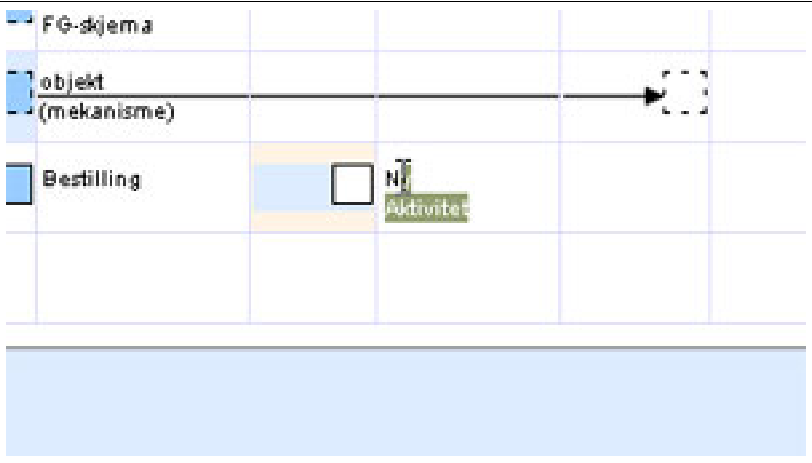


Figure 25: *Subject E1 seems to expect in place text editing*

request for feedback on his actions.

Pointer feedback

The cursor that showed when the mouse pointer hovered certain RIS objects seemed to confuse two of the research subjects, E2 and C5. In figure 29 on page 67 and figure 30 on page 68, E2 and C5 explain that they think the text cursor symbols means that the text is directly editable. As mentioned in the framework, in web applications the text cursor symbol does not normally mean that the text can be edited directly.

In figure 30 on page 68 the subject explains that the cursor appearing as he hovers a RIS object leads him to think that the text can be manipulated directly. He also suggests a different cursor when the mouse actions made the Editing pane appear. Figure 25 also shows a good example of confusion caused

[Drags the first role object onto the modeling area] (It does not appear immediately)
Subject: ”No.”
[The role object appears]
Subject: ”Yes. Slow!”

Figure 26: *Subject E1 seems confused by the lack of feedback.*

[Drags a role-object into the first cell] [Waits for a few seconds] [The role-object does not appear]
Subject: **”**[I] thought perhaps it would appear here when I dragged it over.”
[Drags another role object into the same cell] (The role object does not appear) *[Double clicks a new role-object on the menu on the left]* (The object then appears after the delay)
Subject: ”Double click, yes.”
[Clicks the role object] [Writes in role name] [Clicks save] [Double clicks a new role-object on the left]
Subject: ”[I will] add these roles first.”
[Double clicks role object in the Object pane several times but no objects appear in the RIS modeling area]
Subject: ”I don’t fully understand what I should do here...”
[Double clicks role object]
Subject: ”Apparently not... Perhaps I should drag it after all...”
[Drags role object to second cell]
Subject: ”Yes.”

Figure 27: *Subject E5 seems to be tricked into thinking that double clicking a role in the object pane will make it appear in the model.*

by the mouse pointer symbol

Error feedback

Some of the subjects in the experiment group had problems due to the lack of error feedback in the application. We have separated the finding into two categories: User errors, where the participant makes a mistake, and system errors, where the network connection disappears or the server goes down.

User errors Two of the subjects, E2 and E5, tried to put an object in a cell where there was already an object. The combination of network latency and no visible feedback seems to have caused the subject to try to drop an object where another object is already placed but not yet visible. Figure 31 on page 68 is an example where E2 seems to be confused.

One of the subjects, C1, tried dropping a delivery-object into an invalid cell. In figure 32, C1 has problems creating a delivery object, as he does not know whether it is caused by delay or a user error.

Subject: ”It’s nice to get some feedback on stuff you have done, because [if you do not] then you are very unsure: Did something happen here or not?”

Figure 28: *Subject E3 wants feedback on his actions.*

[Holds mouse pointer over an activity object]
Subject: ”Perhaps a different... a different icon when you hover it... It’s not intuitive that it is possible to push it... Like when it becomes like that I imagine that I should be able to mark the text.”
[...]
Subject: ”I would like to, for example, have a ’link finger’.”

Figure 29: *E2 is not satisfied with the cursor symbol.*

Neither subjects in the experiment group nor subjects in the control group received any feedback when they committed any of these mistakes, yet the subjects in the experiment group committed mistakes much more frequently. Additionally the subjects in the experiment group, unlike the subjects in the control group, kept making the exact same mistakes over and over.

System errors Another area where the lack of error feedback seemed to confuse the subjects in the experiment group was when there was a network error or a problem with the server. The RIS objects dropped in the modeling area would simply not show up, seemingly making the subject wonder if the problem was due to a delay or a mistake they had made. When an error occurred the subjects in the control group received an explanatory error page, making them realize that there was a problem. The subjects in the experiment group received no such notice, as a consequence of using asynchronous communication.

In figure 33 on page 69 the subject does not get feedback that the network is down, and seems to wonder if the application is just slow.

System feedback

Several of the subjects, E1, E2 and E3 in the experiment group were uncertain whether or not their work was saved. In figure 35 on page 70 one of the subjects seemed to be afraid of losing his work and wanted feedback on whether or not it was saved.

In figure 36 the same subject, E1, even asked explicitly for a save button.

The question of whether or not the work was saved seemed to be no issue for any of the subjects in the control group. Having to save your work is, as discussed in the Framework chapter something that is more common in desktop applications than in traditional web applications.

[Clicks role-object] [Hovers the object for a while]
Subject: ”I’m thinking that a... finger [symbol] like that... that appears. Instead of an ordinary cursor... Then I would have understood it... I at least would have understood it at once, that then it is an area you can click, then perhaps you would get more...” *[pauses]*
Experimenter: ”Ok...”
Subject: ”...information. Instead of when you use this... I don’t know what I should call this cursor...”
Experimenter: ”The text-cursor?”
Subject: ”Text-cursor, yes.”
Experimenter: ”Yes.”
Subject: ”Then I think that it means that you can drag over and change...”
Experimenter: ”Directly?”
Subject: ”...directly there. Not that a new window appears. And what I normally think of then, is that pointer thingy.”
Experimenter: ”The finger -pointer?”
Subject: ”Yes.”
Experimenter: ”Yes.”
Subject: ”That either a new window appears or... Or like it is here, that at least a new dialog... No not dialog... Boxes where you can... do changes and stuff. But it was easy to understand once I got there.”

Figure 30: *Subject C5 seems to be confused by the cursor.*

[Drags an activity-object into a available cell] [Drags another activity onto the RIS modeling area and drops it in the same cell before the first activity-object has appeared] (The first activity-object then appears) *[The subject pauses]*
Subject: ”Hmmm...”
[Tries to drag the activity to the cell underneath] [The subject pauses again] [He drags another activity-object into the cell underneath the first one]
Subject: ”Did I hit myself there? Yes.”

Figure 31: *Subject E2 seems to be confused by the delay.*

Response time

There seemed to be a difference in perceived efficiency between the experiment- and control group. The participants in the control group generally seemed more patient than the subjects in the experiment group. After dragging an object into the modeling area they leaned back and waited for the page reload, seemingly content with the fact that they could only perform one action at the time in the application. Some of the subjects even started whistling, leaning back and chatting while waiting for the update. One of the subjects in the control group told us that he thought that creating a model using the application was quite fast (see fig. 37 on page 71).

The subjects in the experiment group, however, seemed more impatient. They seemed to expect instant feedback on their actions. One of the subjects in

Subject: ”Now we’ll have an arrow.”
[Drags an arrow to a cell on the right of the activity the delivery starts from.]
Subject: ”[I will] put it there... When does it stop? I’ll just put it there first”
[Drops arrow] [Waits for a second]
Subject: ”That didn’t work.”

Figure 32: *Subject C1 tries to perform an invalid action*

[Adds a new role-object] [Writes role name] [Clicks save button] [Waits a few seconds for the role-object to update] (It does not update) *[Drags a new role-object onto the RIS modeling area] (The new role-object does not appear) [Hovers the role he tried to change the name of]* (The name has still not been updated)
Subject: ”Is it just me or is it a bit slow right now? Or was it me that didn’t...”
[Clicks role-object] [Notices that the name is correctly updated in the Editing pane]
Subject: ”The name of the role didn’t appear (in the model).”
[Clicks save button]
Subject: ”Probably just slow.”
Experimenter: ”I wonder if the network is a bit...”
Subject: ”I think it has registered it at least...”
[Experimenter clicks the ’refresh’ button] (An error message appears. The connection to the server seems to be down.)

Figure 33: *Subject E4 is experiencing a lack of system error feedback.*

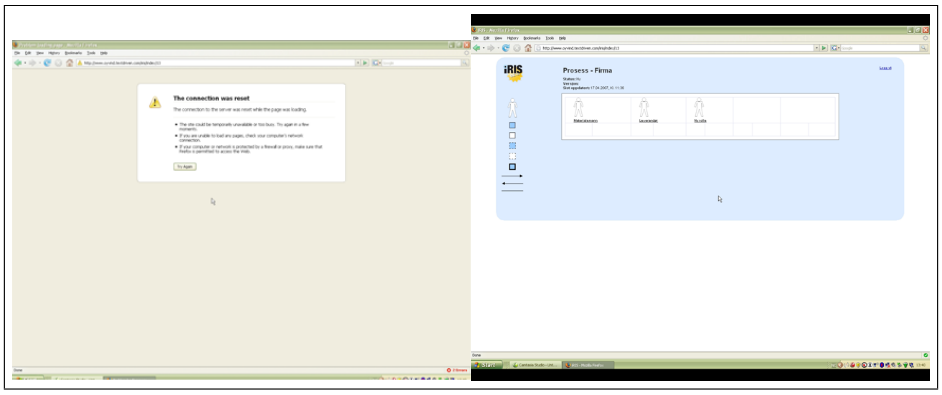
the experiment group described the application as slow (see fig. 26 on page 66). In contrast to the subjects in the control group, who dragged one object at the time, the subjects in the experiment group would rapidly drag several objects into the modeling area, and would sometimes get confused when the objects did not appear immediately. In some cases, they started clicking wildly when RIS object did not appear in the modeling area immediately.

Visual Components

Menu

None of the research subjects seemed to be looking for any menus, except for research subjects E1 and E2 who seemed to be expecting a contextual menu in some situations. In figure 38 on page 71 subject E2 tries to find a way to delete an RIS object using the right-click menu.

Two of ten subjects used the contextual menu, six of the other subjects also right-clicked, but only accidentally. All these ignored or clicked away the menu.


The connection was reset

The connection to the server was reset while the page was loading.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Figure 34: Server error

Subject: "By the way, I think the system gives too little feedback that things are saved and kept and stuff... Compared to, like, 'gmail' or 'Google Docs' like... it says that it's saved and stuff then, so... And I'm a bit nervous that everything I'm going to do will be lost."

Figure 35: *Subject E1 is experiencing a lack of save status.*

Controls, Tables and Windows

None of the subjects were expressing any expectation of controls that are normally only related to desktop applications, like a 'stepper', a 'slider' or any other advanced controls. There were also no findings regarding tables.

The only finding in these categories was when subject E1 asked for a way to close the application (see figure 36). This is, according to the framework, something that is most common in desktop applications.

Summary

In this chapter we have presented the findings from our experiment and categorized these according to our framework. These observations are summarized in figure 39 on page 72. Figure 40 on page 73 summarizes the answers given by the subjects at the end of each experiment session. In the next chapter we will discuss the findings presented in this chapter.

(The subject has just completed the first assignment)
Experimenter: ”Do you have any comments?”
Subject: ”I would like a save button! I would like a save button or something so that I can get out of here, so I'm very uncertain what I'm supposed to do now.”

Figure 36: *Subject E1 is requesting a save button.*

Experimenter: ”Did you think it creating a model went ok?”
Subject: ”Yes, I think it went fast.”

Figure 37: *Subject C4 thinks that creating the model went fast*

[Drags a RIS object onto the modeling area] [accidentally right-clicks]
Subject: ”I accidentally pressed a button.”
Experimenter: ”Did you lose it (the RIS object)”
Subject: ”Yes”
[Clicks away the right-click menu which had appeared] [Tries to drag the dropped RIS object to where it actually was going to be dropped] (This does not work)
Experimenter: ”Now you have to delete it”
Subject: ”Yes, now I have to delete it”
Experimenter: ”Let me know if you need any help”
[Right-clicks the RIS object] [Chooses ”properties” from the contextual menu] [Looks at the ”properties” pop-up and then closes it] [Clicks the RIS object] [Finds the delete button in the Editing pane]
Subject: ”Aaah”
Experimenter: ”that was where it was hidden”
Subject: *[Laughs]* ”I've seen that button every time I opened (the pane)”

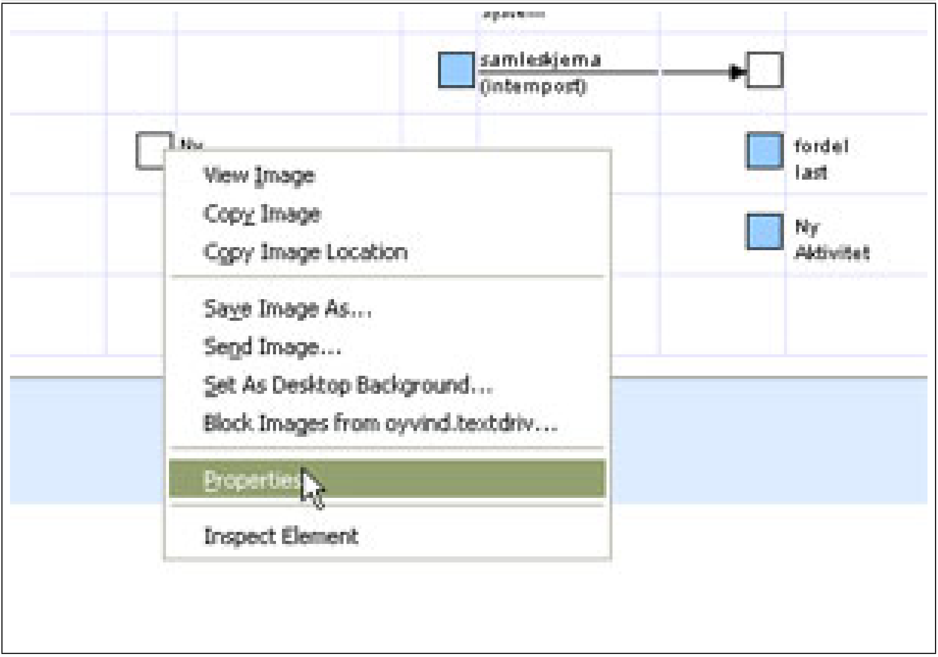
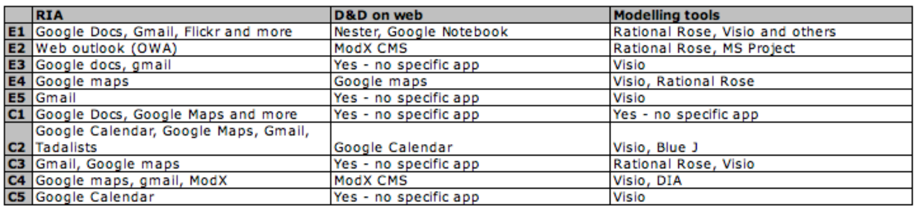


Figure 38: *Subject tries to use a contextual menu*

ACTION CHECK-LIST	E1	E2	E3	E4	E5	C1	C2	C3	C4	C5
User Interaction										
Uses double-clicking in other situations than for text selection			X		X					
Uses triple-clicking in other situations than for text selection										
Uses pressing and/or dragging in other situations than for text selection or drag and drop										
Tries to use or expects keyboard shortcuts not related to the browser										X
Tries to use or expects keyboard navigation not related to scrolling or jumping between input fields or hyperlinks										
Comprehends the drag and drop technique	X	X	X	X	X	X	X	X	X	X
Comprehends selection of objects	X	X	X	X	X	X	X	X	X	X
Tries to or expects to be able to select multiple objects					X					X
Tries to or expects to be able to edit elements other than form elements directly			X							
Tries to or expects to be able to edit text other than in form fields directly	X	X					X			X
Feedback										
Expects progress feedback of individual objects										
Gets confused by pointer feedback or expects a different pointer feedback than what is received	X	X								X
Expects error dialog boxes as a response to user errors										
Expects error dialog boxes as a response to system errors										
Tries to locate or expects save status	X	X	X							
Perceives the response time as longer than what they are used to	X									
Gets confused by full page reload										
Gets confused by network latency	X	X	X	X	X					
Visual Components										
Tries to locate or expects an 'Exit' or a 'Close' path in the application	X									
Tries to locate or expects a menu bar not related to the browser										
Is having problems separating the browsers menu bar from the application										
Tries to locate or expects to be able to use a contextual menu not related to the browser	X	X								
Tries to locate or expects to be able to use a toolbar connected to the application										
Tries to locate or expects controls not common on the web										
Does not comprehend the tooltips used in the application										
Tries to or expects to be able to dynamically manipulate tables										

Figure 39: Checklist with results from the experiment



	RIA	D&D on web	Modelling tools
E1	Google Docs, Gmail, Flickr and more	Nester, Google Notebook	Rational Rose, Visio and others
E2	Web outlook (OWA)	ModX CMS	Rational Rose, MS Project
E3	Google docs, gmail	Yes - no specific app	Visio
E4	Google maps	Google maps	Visio, Rational Rose
E5	Gmail	Yes - no specific app	Visio
C1	Google Docs, Google Maps and more	Yes - no specific app	Yes - no specific app
C2	Google Calendar, Google Maps, Gmail, Tadalists	Google Calendar	Visio, Blue J
C3	Gmail, Google maps	Yes - no specific app	Rational Rose, Visio
C4	Google maps, gmail, ModX	ModX CMS	Visio, DIA
C5	Google Calendar	Yes - no specific app	Visio

Figure 40: Subjects answers to experiment questions

Discussion

In this chapter we will discuss the findings captured during the analysis and their implications in regard to the research question. In addition we discuss the quality of the study and highlight its strengths and weaknesses.

Implications of the study

In the following sections we will discuss the findings from the analysis, raise questions and discuss theories. The sub sections are structured according to the framework taxonomy (see fig. 42 on page 96). In this chapter the main focus is on categories where there were direct findings.

User interaction

Mouse

In the area of user interaction we measured a difference between the experiment group and the control group regarding double clicking. Two of the subjects using the Ajax version of the application were double clicking, and these subjects double-clicked extensively. Of the subjects using the non-Ajax version no one were double clicking. As suggested in the Framework chapter, double clicking is not something one would normally do while using a traditional web application other than when selecting text. In web application, double clicking would not normally trigger any different behavior than single clicking. The question then becomes; why did some of the subjects in the experiment group double-click? Our findings suggest that there may be a correlation between the use of double clicking and the independent variable; client server communication. The use of Ajax seems to have caused them to behave in a way that is, according to the framework, considered to be a desktop convention. Might there, however, be other explanations for why the subjects were double clicking so extensively? One other possible explanation relates to the consequences of network latency. As suggested and seen in an example in the analysis (see fig. 27 on page 66) subjects may be confused due to network latency and the lack of immediate feedback in the Ajax version. In this example the subject drags an object, which is the correct thing to do, but due to network latency the result of the triggered action is not visible immediately. The object coincidentally becomes visible just as the subject has double-clicked, confusing the subject. In the experiment we also observed that sometimes the user got frustrated due to the network latency and were clicking wildly, attempting to trigger an action.

An odd observation that we would like to mention was that both the subjects who were double clicking, switched to single clicking after a while. It would be interesting to know why they suddenly changed their behaviour. If they thought double clicking was the correct action why did they switch to single clicking? One theory may be that that they accidentally discovered that a single click was enough, or that they suddenly remembered they were working on a web application where double clicking is less common.

Other than the use of double clicking we found no differences between the experiment and the control group considering mouse actions. Ajax seems therefore not to have any influence on any other mouse related actions.

Keyboard

We did not observe subjects in either group trying to use keyboard navigation. None of the subjects expressed any expectation for keyboard navigation in the application. Therefore keyboard navigation does not seem to be related to the independent variable.

Neither in the category of keyboard shortcuts did we find any significant difference between the subjects in the two groups. Only one research subject tried to use keyboard shortcuts. The subject, belonging to the control group, pushed the delete button on the keyboard trying to delete an object. Using the keyboard to trigger shortcuts is, as suggested in the framework chapter, not common in traditional web applications.

Our findings suggest no correlation between the use of the keyboard and the independent variable.

Direct manipulation

Selection We measured no difference between the experiment group and the control group considering the action of selection. All subjects seemed to comprehend the selection of objects, so there seems to be no relation between selection and the independent variable. The fact that subjects had to use selection even before they were exposed to Ajax, as they would have to select an object to trigger an action, makes any correlation even less likely.

Selection of multiple objects and operating on several objects at the same time is not something one would normally do in traditional web applications, but is a much used feature in desktop applications. Still two subjects, one from each group, seemed to be expecting this feature in the application. As we measured no difference between the two groups, there seem to be no correlation selection of multiple objects and the independent variable. Still, we wonder why did these two subjects expect this feature. The answer might be related to the fact that they were already exposed to single object selection. We gave them the opportunity to select objects, although this is not common on web. This might have triggered them to believe they could select multiple objects at once, as these features are often closely related in desktop applications.

Editing We found no significant differences between the experiment and the control group considering editing objects. There were no observations in this category except for one subject wanting to resize the name-labels of the objects. To be able to resize elements is not a common feature in traditional web applications. An important aspect to this example is that the subject explicitly told us that he expected the application to act like Microsoft Visio, and not surprisingly, in Visio you can resize elements. This leaves us to believe that there is probably no correlation between editing objects and the independent variable.

Neither in the area of direct text editing did we measure any differences between the experiment and the control group. Four of the subjects did try to edit text directly, two in the experiment group and two in the control group. Asynchronous communication therefore does not seem to have caused of this behavior. As the framework chapter suggests, direct text editing is not a common feature in traditional web applications, but very common in traditional desktop applications. Why did four of our subjects still expect this feature? One theory might be what we have called the “drag and drop effect”; giving them the opportunity to use the drag and drop technique might have given the subjects expectations of other advanced features, like for instance direct manipulation, right click menus and keyboard shortcuts.

An even more likely theory is that of ambiguous pointer feedback. As we discussed in the framework chapter, the pointer symbol visible when hovering text in a web application is the same symbol as hovering text in a desktop application, but the meaning of the symbol is different. As we saw in the examples in the analysis (see fig. 29 on page 67 and fig. 30 on page 68) this symbol seemed to confuse the subjects, leading them to think the text could be edited directly.

Drag and drop The drag and drop technique was something that all of the subjects comprehended immediately. No one seemed to have any problems using this technique. In other words we found no difference between the experiment and the control group considering the drag and drop technique, neither does there seem to be any correlation to the independent variable.

Even though the drag and drop technique is not something that is commonly used in traditional web application, we do not find it surprising that everyone comprehended this technique. The way the application is built; its layout similar to Visio and other types of modeling tools, might have given hints that the drag and drop technique was to be used. The objects in the Object pane are styled in a way that when you hover them they get larger, and the pointer symbol changes to indicate that the object can be dragged. The RIS modeling area starts out with empty "cells" and when you hover an object in the Object pane the cells that can receive the kind of object you are hovering are colored blue to indicate that you can drop the object there. We think that the symbol and the color indications makes it is more obvious to the subject that the drag and drop technique is to be used.

Feedback

Progress feedback

The area of progress feedback was one area where the difference between the Ajax version and the non-Ajax version of the application was quite noticeable.

As one can see in figure 26 on page 66 and in figure 27 on page 66, subjects in the experiment group got confused by the lack of progress feedback. Additionally, one of the other subjects in the experiment group explicitly asked for more progress feedback on his actions. None of the subjects in the control group seemed to have any problems in regard to progress feedback.

This leads us to conclude that there seems to be a difference between the experiment group and the control group regarding progress feedback. It may suggest that asynchronous communication leads the user to think of the application as more of a desktop application, thereby strengthening our hypothesis. There are however other possible explanations for this behavior.

A web browser usually displays a progress indicator when information is being transferred synchronously, but not when asynchronous communication is applied. The full-page reload that occurs in synchronous communication could perhaps also be thought of as a kind of progress feedback. In other words the subjects of the control group received standard browser feedback while the subject of the experiment group received no feedback at all. This was not a conscious design choice on our part, but an accidental consequence of applying asynchronous communication. The web browser does not display any built in progress indicators when asynchronous requests are being made.

Pointer feedback

Confusion regarding pointer feedback was observed in both the experiment and the control group. Two of the subjects in the experiment group and one subject in the control group thought that the text-cursor that appeared when hovering a RIS object meant that they could edit the text-attribute directly. Two of the subjects suggested the use of a pointer/hand cursor to be more appropriate to signal that the RIS objects were clickable.

As discussed in the Framework chapter, the text-cursor can inhabit two separate meanings in desktop applications. It can either symbolize that text can be selected or that text can be edited. In traditional web applications the text cursor when displayed outside form fields simply means that text can be selected. This may suggest that the subjects thought of the application as a desktop application.

There does not, however, seem to be any correlation between asynchronous communication and expectations of mouse pointer symbols in the application as confusion regarding mouse pointer symbols seems to appear both in the experiment group and control group.

Error feedback

In addition to having problems with progress feedback, some of the subjects in the experiment group had problems related to error feedback as well. These subjects made more mistakes, and also seemed to have more trouble realizing that they had made a mistake.

Placing a RIS object in an invalid position was something subjects in both the experiment group and the control group did, but the participants of the control group seemed to more easily realize that they had made a mistake. In addition, the subjects in the experiment group seemed to make the same mistakes over and over.

As mentioned in the Framework chapter error messages as response to user errors in desktop applications usually comes in the form of dialog boxes, while web applications usually display a message on the page itself. Although iRIS did not display any error messages in either the Ajax version or the non-Ajax version there seemed to be a difference between the two groups. One reason for this may be that, as a result of using asynchronous communication, the subjects in the experiment group lacked the status indicator and page refresh that the subjects in the control group got. These subjects could therefore try to update an element while it was processing. The subjects of the control group had no possibility of working while data was being sent to the server and did not have the possibility of making this mistake. This may have caused the subjects of the control group to more easily understand when they had made a mistake. It may also have caused these subjects to more easily learn the correct way of placing the RIS objects in the model.

One other area of error feedback where we measured a difference between the two groups was regarding how they handled system errors. The subjects of the experiment group that were affected by system errors would continue working, unaware of the error. When for instance the server went down, participants in the control group were presented with an error message from the browser. For the subjects in the experiment group however, a network error would produce no direct error message, the RIS objects would simply not appear or be updated. Although there was a difference between the experiment group and the control group in this area we do not think it has any relevance to the research question, but is an example of things one should take into account when designing Ajax-based applications.

As was the case of status feedback, none of these differences in error feedback between the Ajax version and the non-Ajax version were conscious design choices. The use of asynchronous communication simply removed the progress feedback and the system feedback, normally displayed by the browser on page reload.

System feedback

In the system feedback category we measured a difference between the two groups related to save status. Three of the subjects in the experiment group expressed concern as to whether or not their work was saved. One of the subjects even asked explicitly for a save button. None of the subjects in the control group expressed any concern whether their work was saved or not. Having to save your work is, as mentioned in the Framework chapter, something that is quite common in desktop applications, but not very common in traditional web applications. All this may suggest that there is a correlation between save status and the independent variable.

It is, however, important to be aware that there are other possible explanations. For instance, the application prototype used by the control group did not utilize asynchronous communication. The web page would therefore be reloaded whenever information had to be sent to the server. This may have led the subjects to understand that their work had indeed been saved.

Response time

There were differences between the two groups considering the perceived response time. Even thought the Ajax version was in practice "faster" than the non-Ajax version as there was no need for full-page reload, the subjects in the experiment group seemed generally more impatient than the subjects in the control group. While waiting for the page to reload, the control group subjects seemed very patient. They were chatting, humming, whistling or leaning back in the chair with their arms crossed. The subject in the experiment group got confused when RIS objects did not appear in the modeling area immediately. They had the possibility to do several things at the time, something the control group subjects could not. The question is: Why did the experiment group's subjects seem more impatient than the control group subjects?

Response time is one area where there seems to be a correlation to the independent variable. The use of Ajax seems to cause the experiment subjects to perceive the application as slow. One reason may be that the subjects expect our application to be as responsive as a traditional desktop application. Another reason for the confusion may be the lack of feedback in the Ajax version, making the subjects unaware that the system is processing their input.

Visual components

Menu

Intentional use of a contextual menu was only found amongst the subjects in the experiment group. One of the subjects that used the contextual menu purposely chose a menu item from the menu. This might imply that the subject had a problem separating browser functionality from the application functionality; he might have thought that the menu item had a relation to the RIS object he was focusing on. Contextual menus are not, as discussed in the chapter framework, a feature you would usually find in traditional web applications, but a much used feature in traditional desktop applications. The fact that two of the subjects in the experiment group but none of the subjects in the control group tried using a contextual menu may suggest a correlation between use of contextual menus and the independent variable.

We did not observe any other interesting aspects related to the menu category other than the contextual menu. None of the research subjects seem to be expecting or trying to locate a menu bar or a tool bar. Neither did anyone seem to expect tool tips on any interface element, or even notice the tool-tips we had added to the RIS elements. This seems to correspond with what the framework suggests.

Controls, Tables and Windows

Under the categories of controls, tables and windows there were few interesting findings. None of the subjects seemed to expect any of the controls defined in the framework as not common on web. This may be because there was no areas where such controls were necessary or natural. None of the subjects seem to expect any more dynamic behavior of the tables. In the windows category one of the subjects in the experiment group requested a way to exit the application. It seems that he wanted an “Exit” or “Close” function in the application, although this is not something that is common in traditional web applications. There might be a correlation to the independent variable, but it is not a strong indication. Other theories include the “drag and drop effect” mentioned under “direct manipulation”.

Asynchronous communication seems to have had no effect on these categories, the subjects seems to behave and expect the same things whether the application uses asynchronous or synchronous communication.

Quality of the study

In this section we begin by defining some terms for measuring quality of a research project. We then discuss strengths and weaknesses of different aspects of the study, including the research design, the analytical framework, the prototypes, the experiment and the analysis phase.

Internal and external validity

To help ensure that the experimental effect is actually caused by the manipulation of the independent variable and not other factors, one needs to have what is called experimental control. This means that all the extraneous variables are kept constant (C.Cozby, 2007). Internal validity of the experiment refers to the possibility of ensuring causality. The stronger the experimental control, the higher the internal validity. To be able to ensure high internal validity, three things must occur according to Cozby (2007). First there must be temporal precedence; the cause must precede the effect. Second there must be covariation of the cause and effect; change in one follows change in the other. Finally, alternative explanations must be eliminated. An important issue is also the external validity of the experiment. Stronger experimental control leads to higher internal validity, but at the same time it leads to a lower external validity. External validity is defined by Cozby (C.Cozby, 2007, p. 86) as ”[...] the extent to which the results can be generalized to other populations and settings..” This means that when the experimental control gets stronger, the experiment itself gets more unrealistic and the finding less generalizable.

Reliability

Bell (2005) defines reliability as "[...] the extent to which a test or procedure produces similar results under constant conditions on all occasions" (Bell, 2005, p. 117). If an item is unreliable then it also lacks validity, a reliable item is however not necessarily also valid.

This study has in our opinion a high degree of reliability. The experiments were videotaped, meaning that the data material could be analyzed by others. If someone should want to repeat the entire experiment, both the prototypes and the framework and checklist can be used again with different subjects.

Research design

The data collection methods

Preceding the actual experiment, we carried out a pretest on order to validate that the experimental design and the technical equipment was good enough. This turned out to be a very smart move, as we discovered that using the default built-in video format 'CAMREC' made it difficult for us to save the screen-capture recording session. The test session was probably too long for the software to handle, and the software crashed when we were trying to save the recording of the session. Switching to 'Avi' format for the screen-capture video solved this problem, and the software seemed to be much better at handling long sessions when using this format, as it has a higher degree of compression. The pretesting also made us realize that the positioning of the video camera was not optimal. We did not see the keyboard and mouse as well as we had hoped. We also realized, due to the loss of the screen-capture session recording, that the video camera should be positioned recording the screen as well as the keyboard and mouse movements. The recordings from the video camera would then also function as backup if the screen-capture recording for some reason should be lost.

Other than losing the test persons screen-capture session the screen-capture software worked well. Even in a compressed format the recordings had high quality, and we had no problems perceiving the actions on the screen when watching the recordings. The recording from the video camera on the other hand did not look that good. Many of the videos were too dark and had various color disturbances, which made it difficult to catch all the details. This may be due to our lack of video camera expertise or technicalities related to the camera equipment. This did not matter that much, however, as the recordings from the camera was only used as support. The screen capture recordings were to be used as a primary source. Audio was recorded both by the video camera and by the screen-capture software. This backup turned out to be a good idea as one of the audio recordings failed during one of the experiment sessions.

The think-aloud method generally worked well, although there were variations regarding how well the subjects comprehended this technique. Some of the subjects talked almost non-stop while others kept quiet for most of the time. When the subjects were quiet for an extended period we tried to ask them questions like: ”why did you do that?” or ”what are you thinking of?” to help them along. This would sometimes work and other times they just answered the question and then went on doing the tasks quietly.

One problem with the think aloud method is that we cannot be sure that subjects verbalize all their thoughts. The subjects may leave out important parts of their thought-process. In addition we cannot be sure that what they were uttering reflected what they actually were thinking. However; this would be a problem using any verbal reporting method.

Another problem with the verbalization was the fact that the variations amongst the subjects' ability to think-aloud left us a bit uncertain on how to act. We did not want to treat any the subjects differently, yet some of them seemed to need more questions and prompting than others. Some of the subjects occasionally asked us questions during the task solving. This may therefore have led us to treat the subjects in an differently, even though we tried to be careful about what we said. We discuss this further under “Experimental bias”.

The observation during and following the experiment generally went well. It was of great value to be able to watch the video recordings several times. By using only direct observation we would have missed out on a lot of interesting findings. We carried out all of the experiment session in the course of a few days. The first time we watched the video recordings we discovered more than what we first had observed during the actual experiment session, and during the second viewing we noticed even more.

Experimental design

The experimental design chosen for this study is the posttest-only design, and as group design we chose independent group design. In our opinion, these choices leave us with a strong experimental control. By using two separate groups and performing measurements only after the experimental treatment, we reduced the influence of factors that could threaten the experiment's validity. Frankfort-Nachmias and Nachmias (1996) speak of eight of these factors; selection, history, maturation, experimental mortality, instrumentation, testing, regression artifact and interactions with selection.

The selection factor could threaten the validity if the experiment- and control group are assembled using different sampling strategies making the groups less equivalent. The history factor describes the events in the period of the experiment which could provide alternative explanations to the experimental effect. Maturation is a term that involves biological, social or psychological processes that could affect the participants during the experiment. Experimental mortality is when participants drop out of the experiment. Instrumentation are changes in the measuring instrument between the pretest and the posttest. The testing factor involves the possibility for the testing process itself to affect the phenomenon being measured. The regression artifact is when individuals have been chosen to be on the experiment group due to their results in the pretesting.

By choosing the posttest-only design we avoid some of the threat of selection due to the randomization of the participants. Further by excluding the pretest, the testing and the instrumentation factors become irrelevant threats. The remaining factors are controlled due to the exposure of the same external events on both groups and the same maturation process (Frankfort-Nachmias and Nachmias, 1996). Additionally we avoided different types of ”order-effects” (C.Cozby, 2007) having participants only belonging to one group each and also having the groups measured only once.

In order to minimize the influence that the subjects' individual characteristics could have on the results, we used randomization when dividing subjects into groups. If a variable cannot be held constant one can try to control it by making sure the effects of it is random. The achievement of randomization is that the extraneous variables would be just as likely to affect the experiment group as the control group. Whether or not randomization really minimized the influence of the subjects' characteristics is still an open question, because the subject sample was small, with only ten participants. Randomization would not be that effective on small groups because the individual characteristics would still be able to shine through in such a small sample.

In addition, the sampling strategy and the types of subjects gathered could have influenced the result of the experiment. Inviting only students from our department resulted in a subject sample consisting of a relatively homogeneous group. We could argue to some degree that the subjects were all experts on the domain of web applications (see the subjects answers to the experiment questions in fig. 40 on page 73). Using other subjects with different backgrounds, for instance beginners in this domain, could have given us different results.

Framework

The analytical framework is essential to the analysis of the findings. Although it is based on two well-known style-guides, and is tailored for the needs of our application, we cannot exclude it from being on some level incomplete. We could have missed elements that might have affected the results. Nevertheless, we find it solid enough for our needs, and we find it unlikely to have excluded elements central to the domain. It is in our opinion highly unlikely that any missed elements would have affected the results to a large degree.

The category of Visual Components did not contain a lot of findings. The reason for this may be that this category contained many elements that were not present in our application. The lack of results may be due to the difficulties of observing expectations of something that is not there. Few subjects uttered requests for different sorts of visual components. Some of the elements under this category would perhaps not be natural in our application. One could therefore argue that this category could be unnecessary.

The prototypes

The application itself; how it is built and the choices of functionality could also have influenced the result. As a consequence of using Ajax, some browser-feedback such as full-page reload and the hourglass symbol was removed. This could have affected the way the subjects perceived the application. Is is possible that replacing the removed browser feedback would have yielded different results. Additionally, performing the experiment on an application with different characteristics could also cause the results to be different.

Experimental phase

Experimental Bias

To accomplish experimental control an important factor is to treat all the participants in the experiment the same way. The only difference between the groups should be the manipulated variable. We tried to make sure that both groups were given the same information and the same treatment by us, the experimenters. An important issue when operating with an experimental design is the experimenter bias. Experimenter bias is behavior of the experimenter that unintentionally influences the participants (Frankfort-Nachmias and Nachmias, 1996). The experimenter might, without knowing it, communicate expectations to the participant. This could for instance be the experimenter smiling or nodding more while interacting with either the experiment or the control group, or emphasizing certain words while giving instructions to one group but not the other. Using multiple experimenters and using automated data collection is suggested to minimize the experimenter bias (Frankfort-Nachmias and Nachmias, 1996). This is one of the reasons why we chose to both be present in each experiment session and why we were taping the experiment. It is also the reason for using the think-aloud method and not asking more questions or giving more comments than necessary during the experiment. We did not want to manipulate our research subjects.

Even though we wanted all of the subjects to have as equal experimental sessions as possibly, we could not be sure the equipment behaved in the same way. Technical difficulties happened during a few of the experiment sessions, which made them differ in certain ways. During one of the sessions the server went down, interrupting the session and hence the subject who had to wait for the server to come back up before continuing the exercise. Still we do not think this had much influence on the subjects' interaction with the application.

Participant bias

Participant bias might occur when the participants know they are in an experiment situation and believe that certain ways of behavior are expected (Frankfort-Nachmias and Nachmias, 1996). The participants might try to guess the purpose of the study, and then respond according to what they assume is expected or what they assume is not expected, depending on whether they would like the experiment to ”succeed” or not, and you could end up with what is called the Hawthorne effect. Subjects are influenced by the fact that they know they are part of an experiment, and their response may be related to this fact rather then related to the experimental treatment. We tried to minimize this type of bias by keeping the purpose of our study hidden from our research subject. Nor did we tell them about there being two different prototypes of our application. Additionally, by using the think-aloud method instead of asking questions, we made the guessing even harder for the subjects as they got no hints on what aspects of their behavior or thinking-aloud we were interested in.

Even though we were able to keep the purpose of the study a secret, we can not be sure we avoided the Hawthorne effect. The participants were aware of being a part of an experiment. The settings and the equipment used could also have made them nervous. Several of the subjects were clearly influenced by the fact that they were a part of an experiment, and knowing that they were being filmed. Hands shaking was an obvious indication that the subjects were a bit nervous about the situation. One subject turning around in the middle of the session, talking directly into the camera, is another example of the subjects' awareness. We believe that being aware of the experiment could have made the subjects less adventurous, and more focused on doing the exercise perfectly and fast. A person testing the application outside of an experiment would perhaps have "played" more with the user interface, and tested it a bit more than what subjects aware that they were watched would do.

Analysis

The most uncertain factor in the analysis is that it is based on our interpretations. We tried to interpret the subjects' behaviors and utterances in order to understand the interaction between the subject and the application, but we cannot be certain that our interpretations are correct. In addition we cannot be certain that the utterances of the subjects really reflects their thoughts. It is a difficult task trying to elicit the expectations and thoughts of another human being. Still we had the advantage of being two interpreters, making us more certain that we understood the behaviors of the subjects.

Another factor is the transcriptions of the actions and utterances. The subjects were using their Norwegian dialects and their own words and phrases, which were transcribed to English. We cannot be certain that no subtleties were lost during translation. All these factors may or may not have affected the result of this study.

Conclusion

In this chapter we have examined the results of the analysis, category by category, and reviewed the quality of the study, pointing out strengths and weaknesses of different aspects of the study. While we in some areas measured no differences between the two groups regarding their interaction with the application, we have gathered several findings that seem to support the hypothesis. In the categories of double clicking, progress feedback, system feedback, response time and contextual menus there were indications of desktop conventions caused by Ajax. This conclusion is based on the fact that there was a pronounced difference between the two groups considering these categories. The hypothesis *"Ajax influences the users to try to use common desktop conventions in the application as if it were a traditional desktop application"* seems to be strengthened in these cases. There might be other possible explanations as we have seen in this chapter, both considering other theories discussed and the concern around the small subject sample, and the experimental control. Still the use of Ajax seems to be the most plausible explanation and we feel that this is an area worth taking a closer look at.

In addition to the use of desktop conventions caused by Ajax, we observed use of desktop conventions that seemed to have other causes. This usage was found in both the experiment group and the control group making it seem unrelated to the use of Ajax. There are, in our opinion, several possible explanations for this behavior. The "drag and drop effect" mentioned earlier in this chapter is one possible explanation. By introducing the drag and drop technique, the subjects may have been led to expect other advanced desktop features. Another possible explanation might be that the subjects were all expert users with knowledge of RIA and modeling tools. This may have led them to have certain expectations for the application, comparing it to familiar applications. The user interface of iRIS is quite similar to for example Microsoft Viso with a object pane on the left and a modeling pane on the right. One of the subjects even said jokingly that iRIS was a "rip-off" of Visio. This similarity may have lead both subjects in the experiment group and the control group to expect this kind of functionality.

Although we measured differences between the two groups in the interaction, we were nevertheless surprised that the differences were so few. The subjects seemed to be affected by participating in an experiment. As we discuss in the section on participant bias on page 86, this may have led them to focus on their tasks at hand, and not behave as they would if they encountered the application in "the real world".

An interesting observation that we would like to highlight is the fact that the use of Ajax, although technically making the application faster, may actually have caused the subjects to perceive the application as slower than the non-Ajax counterpart. This seems to be caused by the unintentional removal of visual feedback on the users action when replacing synchronous communication with asynchronous communication. Replacing the missing feedback may be one of the most important design considerations when creating an Ajax application.

We were somewhat surprised that the subjects seemed to have relatively little difficulties using the application, since applications incorporating direct manipulation are not very common on the web. The subjects seemed to understand the application relatively quickly. This may be caused by the fact that all the subjects were relatively advanced computer users, and is it possible that beginners could have behaved differently. Another reason may be that people are more adaptable and open to new conventions than we originally thought.

When creating the research question and the analytical framework, we separated user expectations into two distinct groups; web conventions and desktop conventions. This divide may not be as strong as first anticipated. It may be that our view of user expectations was wrong, and that this divide was not so absolute. Something that we think is more likely, is that the advent of Rich Internet Application have caused mixed conventions. The framework was based upon traditional web applications but pure traditional web applications are becoming less and less common as most applications employ some interactive aspects. As RIA become even more common, the divide between web and desktop application may eventually disappear entirely.

Research Contributions

Throughout the course of this project we have gained insight into different aspects of the field of HCI in general and development of Ajax applications in particular. In this chapter we highlight the contributions resulting from this project and present recommendations for developers creating Ajax applications.

The main contribution of our research are the results from the experiment concerning how asynchronous communication in a web application affects the way people interact with the application. The results from the experiment seemed to strengthen our hypothesis on whether the use of Ajax causes behavior normally associated with desktop applications. Although these results seemed to support the hypothesis, we were surprised that the use of asynchronous communication did not have a greater impact on the interaction. We have discussed whether the use of certain interaction styles could in fact have had a greater impact than the use of asynchronous communication.

In addition to the direct results of the experiment this project has produced another valuable contribution. The analytical framework developed to capture and categorize user behavior and expectations could potentially be used to perform similar research in the field of Rich Internet Applications.

The final contribution of the research project is a set of recommendations for developing Ajax-based web applications. These recommendations are based on the experiences and insight we have gained during the course of the research project.

Recommendations

We have discovered some aspects regarding user experience that one needs to consider when using Ajax in web applications. We find that these aspects, which all concern feedback, are important to take into consideration to avoid confused and frustrated users.

One of the most valuable lessons we learned was the importance of providing status feedback to the user. The use of asynchronous communication deactivates the existing activity indicators in the browser. As lack of feedback seems to be very confusing to users, we recommend that feedback indicators be added to replace this functionality.

The error feedback, which is automatically displayed in the browser, such as notifications of server errors, is also deactivated when asynchronous communication is used. This leads to users being unaware of the error that has occurred. We therefore recommend that Ajax-based web applications should warn the user when something unforeseen happens.

In our experience users are very concerned as to whether or not their work is saved. This is not immediately obvious in web applications without page reload, as the refresh of the page seems to assure the user that the information is indeed saved. We recommend that asynchronous web applications incorporate a visible save status.

Future work

During our research we have identified potential projects that could extend our research as well as several other topics that we think deserves further investigation. These topics could in our opinion become interesting research projects in their own right.

One of the aspects of our research design that may be thought of as a limitation, is the fact that our subject sample was relatively small and homogeneous, making the results hard to generalize. It could therefore be valuable to conduct a project with a more extensive and varied sample to see if this would produce a different result. Another aspect of our project that also may have reduced the external validity is that the project is limited to the evaluation of one specific application. It could therefore be interesting to conduct the same experiment on a different application.

One of the most puzzling discoveries we made during the experiment was related to the perceived efficiency of the application. Although the non-Ajax version of our application was technically slower than the Ajax version, this version seemed to be perceived as more efficient than the Ajax version. A potentially interesting project could be to investigate perceived efficiency in Ajax-based versus traditional web applications.

Much of the desktop like behavior observed in the interaction with the application seems not to be caused by the use of asynchronous communication. We have suggested that this may have been caused by the fact that the application is very dependent on drag and drop. We have theorized that using some techniques from the realm of desktop applications in a web application might lead the user to perceive the application as "desktop like" and look for more such desktop elements or functionality. We feel that this subject deserves a research project of its own.

A more general trend that we have witnessed is the merging of web applications and desktop applications. We have seen that web applications using technologies like Ajax have started incorporating features that are commonly associated with desktop applications. At the same time it seems that desktop applications are also being influenced by web applications, incorporating web pages in their interfaces and using an interaction model that is similar to that of the web. This area could become an important subject of research in the near future. As we have shown in this research it is important to take the users perspective into consideration when faced with evolving technologies resulting in a different interaction model.

Appendix

Assignments

Assignment 1

- Open the web browser (Firefox)
- Go to URL: http://oyvind.textdriven.com
- Create an account ith username and password. (email does not have to be a real address)
- Create a new RIS model
- Add RIS-model information: Process name: ”Lasteprosessen”, Company: ”Iris inc.”, Version: ”1”
- Create modell as in figure 1 on the sheet

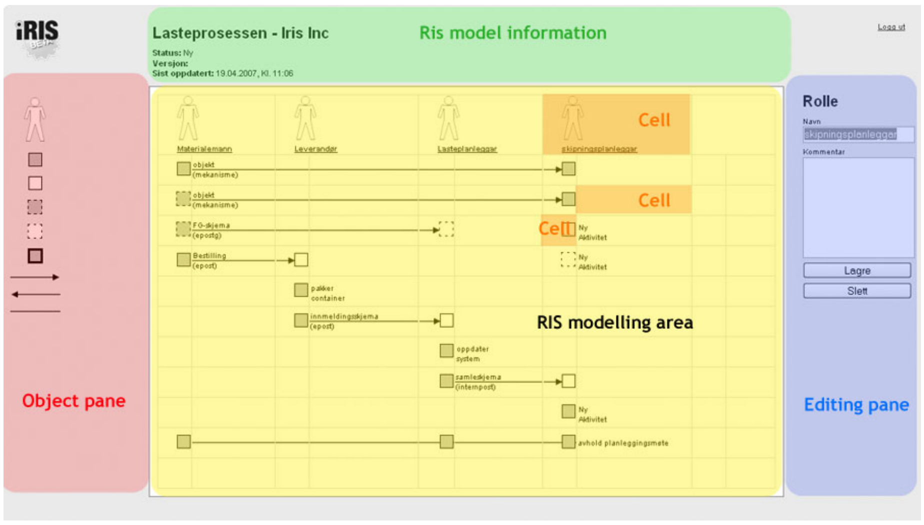
Assignment 2

- Change the name of the role ”Materialemann” to ”Materialesjef”
- Change the name of the role ”Skipper” to ”Kaptein”
- Change the mechanism ”samleskjema” and ”fordelte båtar” from ”intern-post” to ”e-post”
- The ”FG-skjema” that the role ”Materialesjef” delivers to the role ”Lasteplan-leggar”, should the ”Materialesjef” now deliver to the ”Skipningsplanleggar”
- Delete the role ”Lastebas”
- Create a macro activity at the bottom of the column of the role ”Lasteplan-legger” with the name ”organiserer lasting”.
- Change the status of the model to ”ferdig”.

Questions

1. Have you ever used a Rich Internet Application (RIA)?
   (a) Which RIA's have you used?
2. Have you ever used drag and drop on the web?
3. Have you ever used a modelling tool?
   (a) Which modelling tools are you familiar with?

Screenshots


Lasteprosessen - Iris Inc
Status: Ny
Versjon:
Sist oppdatert: 19.04.2007, Kl. 11:06

Rolle
Navn: skipningsplanleggar
Kommentar:

Figure 41: The main page consists of four components; RIS model information, RIS modelling area, Object pane and Editing pane.

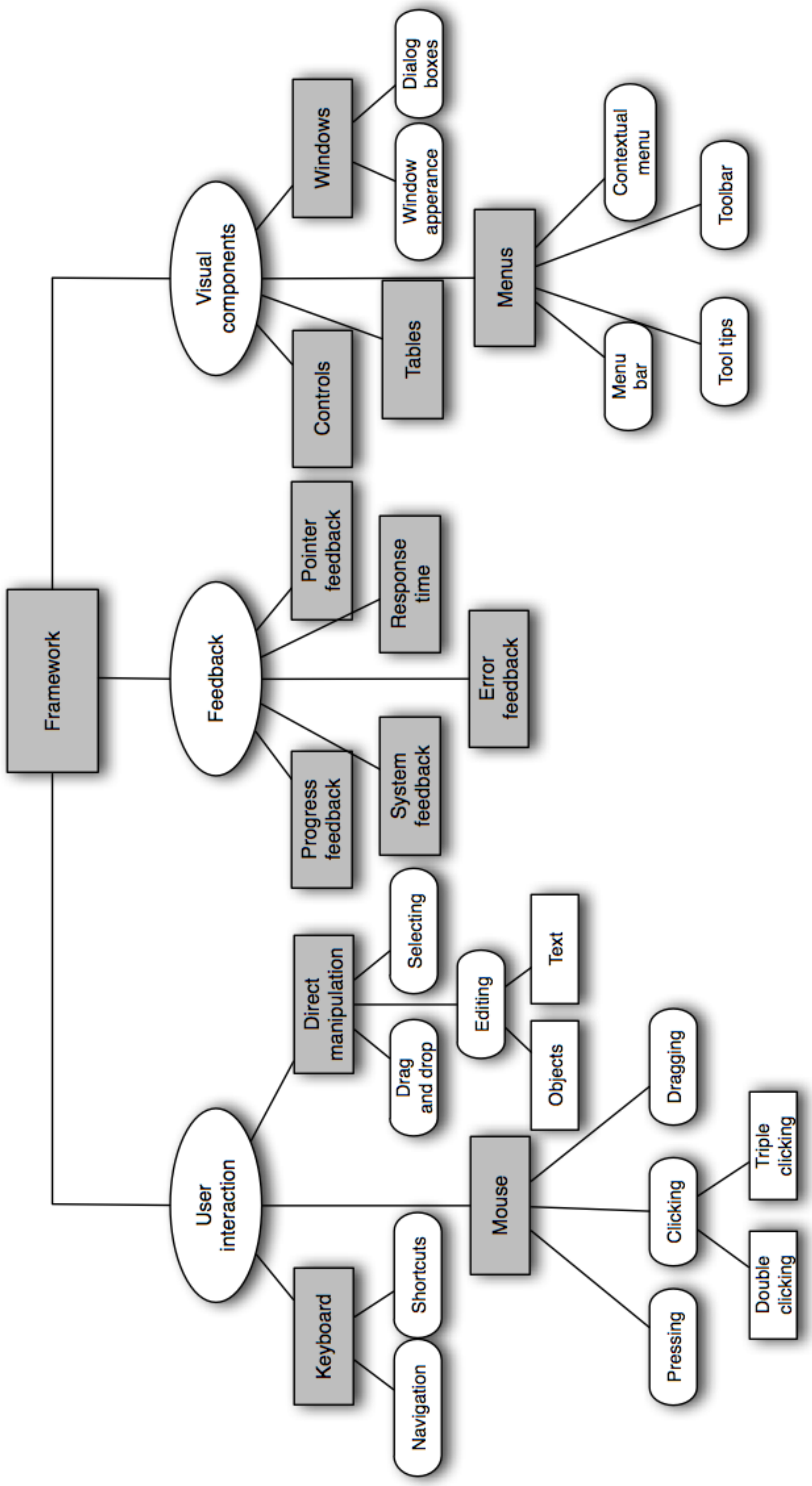


Figure 42: Hierarchical representation of the framework

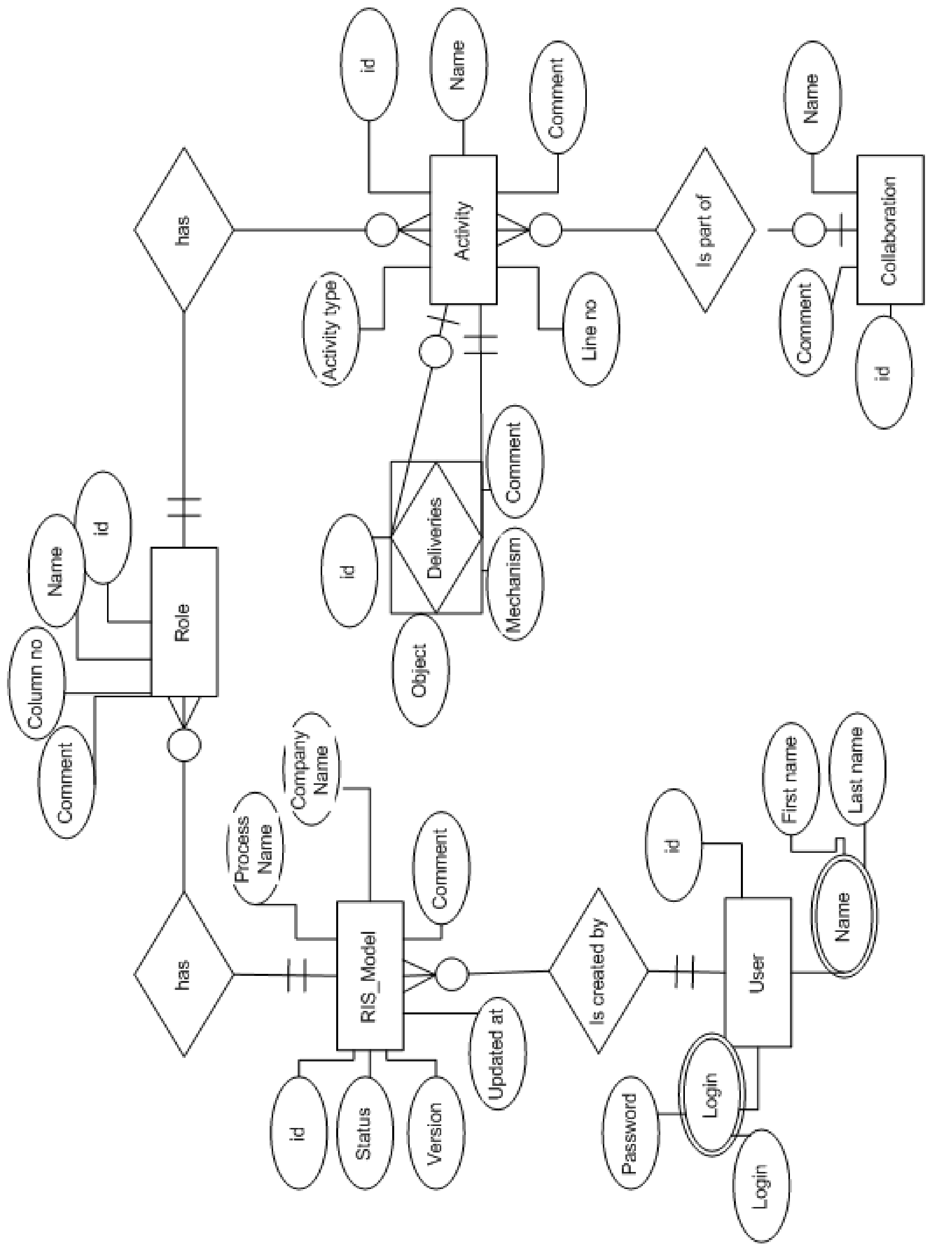


Figure 43: ER diagram of the iRIS DB

Bibliography

The 100 best products of 2005, 2005. URL `http://www.pcworld.com/article/id,120763-page,12/article.html`.

I. Apple Computer. Apple human interface guidelines. Technical report, Apple Computer Inc., 2006.

J. Bell. *Doing your research project.* Open University Press, 2005.

B. L. Berg. *Qualitative research methods for the social sciences.* Pearson Education, Inc., 5 edition, 2004.

T. Berners-Lee and D. Connolly. Hypertext markup language: A representation of textual information and metainformation for retrieval and interchange, 1993. URL `citeseer.ist.psu.edu/berners-lee93hypertext.html`.

B. Bos, H. Lie, C. Lilley, and I. Jacobs. Cascading style sheets, level 2 css2 specification. w3c recommendation. `http://www.w3.org/TR/REC-CSS2`, 1998. URL `citeseer.ist.psu.edu/bos98cascading.html`.

A. Bozzon, S. Comai, P. Fraternali, and G. T. Carughi. Capturing ria concepts in a web modeling language. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 907–908, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-323-9. doi: http://doi.acm.org/10.1145/1135777.1135938.

T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible markup language. *World Wide Web J.*, 2(4):29–66, 1997. ISSN 1085-2301.

P. C.Cozby. *Methods in Behavioural Research.* McGraw Hill, ninth edition, 2007.

A. Cockburn and L. Williams. The costs and benefits of pair programming. pages 223–243, 2001. URL `http://class.ee.iastate.edu/berleant/home/Courses/SoftwareEngineering/CprE486fall2002/XPSardinia.pdf`.

E. Dahl. The 100 best products of 2007. *PC World*, May 2007. URL `http://www.pcworld.com/article/id,131935-page,1/article.html`.

O. Dalland. *Metode og oppgaveskriving for studenter.* universitetsforlaget, 1993.

J. Duhl. Rich internet applications. Technical report, IDC, 2003.

F. Erickson. Ethnographic microanalysis of interaction. *The handbook of qualitative research in education*, pages 201–226, 1992.

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Rfc 2616, hypertext transfer protocol – http/1.1. ftp://ftp.isi.edu/in-notes/rfc2616.txt, 1999. URL `http://www.rfc.net/rfc2616.html`.

C. Frankfort-Nachmias and D. Nachmias. *Research methods in the social sciences.* St' Martins Press, Inc., fifth edition, 1996.

J. J. Garrett. Ajax: A new approach to web applications, 2005. URL `http://adaptivepath.com/publications/essays/archives/000385.php`.

B. Gentikow. *Hvordan utforsker man medieerfaringer? Kvalitativ metode.* IJ-forlaget, 2005.

C. Goodwin. *Research in psychology. Methods and design.* John Wiley & Sons, Inc, 2005.

S. Gundavaram. *CGI programming on the World Wide Web.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1996. ISBN 1-56592-168-2.

C. Heath and J. Hindmarsh. *Analysing Interaction: Video, ethnography and situated conduct*, pages 99–121. Sage, 2002.

O. Hellevik. *Forskningsmetode i sosiologi og statsvitenskap.* Universitetsforlaget, 1991.

A. Holliday. *Doing and Writing Qualitative Research.* Sage Publications Ltd, 2002.

J. Iden. *Prosessutvikling - handbok i modellering og analyse av prosesser.* Tapir akademisk forlag, 2005.

B. Jordan and A. Henderson. Interaction analysis: Foundations and practice. *The Journal of the Learning Sciences*, 4:39–103, 1995.

D. Kristol and L. Montulli. Http state management mechanism, 1997. URL `citeseer.ist.psu.edu/kristol98http.html`.

D. Kristol and L. Montulli. Http state management mechanism, 2000.

H. W. Lie and B. Bos. *Cascading style sheets: designing for the Web.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997. ISBN 0-201-41998-X.

S. Lohr. *Go to: Superheroes of Software Programming from Fortran to the Internet Age and Beyond.* Profile Books Ltd, 2002.

H. Mehan. Learning lessons: Social organization for the classroom. 1979.

B. Myers. A brief history of human-computer interaction technology. *interactions*, 5(2):44–54, 1998. URL `ï\T1\guillemotright\T1\textquestiondownhttp://www.cs.cmu.edu/~amulet/papers/uihistory.tr.html`.

J. Niederst. *Web Design in a Nutshell: A Desktop Quick Reference.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1998. ISBN 1565925157.

J. Nielsen. Usability 101: Definition and fundamentals - what, why, how (jakob nielsen's alertbox). 2003. URL `http://www.useit.com/alertbox/20030825.html`.

J. Nielsen. Ten usability heuristics. 2005. URL `http://www.useit.com/papers/heuristic/heuristic_list.html`.

T. Noda and S. Helwig. Rich internet applications technical comparison and case studies of ajax, flash and java based ria. Technical report, UW E-Business Consortium, 2005.

M. Q. Patton. *Qualitative Research & Evaluation Methods.* SAGE, 2002.

L. Paulson. Building rich web applications with ajax. *Computer*, 38(10):14–17, Oct. 2005. doi: 10.1109/MC.2005.330.

D. Raggett, A. L. Hors, and I. Jacobs. Html 4.0 specification, 1997. URL `citeseer.ist.psu.edu/raggett97html.html`.

K. Ringdal. *Enhet og mangfold. samfunnsvitenskaplig forskning og kvantitativ metode.* Fagbokforlaget, 2001.

M. A. Sasse. How to t(r)ap user's mental models. In *Selected papers of the 8th Interdisciplinary Workshop on Informatics and Psychology*, pages 59–79, Amsterdam, The Netherlands, The Netherlands, 1991. North-Holland Publishing Co. ISBN 0-444-88602-8.

H. Shubin and M. M. Meehan. Navigation in web applications. *interactions*, 4 (6):13–17, 1997. ISSN 1072-5520. doi: http://doi.acm.org/10.1145/267505.267508.

H. Shubin and R. Perkins. Web navigation: resolving conflicts between the desktop and the web. In *CHI '98: CHI 98 conference summary on Human factors in computing systems*, page 209, New York, NY, USA, 1998. ACM Press. ISBN 1-58113-028-7. doi: http://doi.acm.org/10.1145/286498.286694.

M. S. Silver. Browser-based applications: popular but flawed? *Information Systems and E-Business Management*, 4(4):361–393, oct 2006. URL `http://www.springerlink.com/content/e2h33g1447367px8/`.

A. Stafford. The 100 best products of 2006. *PC World*, 2006. URL `http://www.pcworld.com/article/id,125706-page,13/article.html`.

I. Sun Microsystems. *Java Look & Feel Design Guidelines*. Sun Microsystems, Inc, Boston, MA, USA, 2001. ISBN 0201615851.

D. Thomas, D. Hansson, L. Breedt, M. Clark, J. D. Davidson, J. Gehtland, and A. Schwarz. *Agile Web Development with Rails*. Pragmatic Bookshelf, 1st edition, 2006. ISBN 0977616630.

M. W. van Someren, Y. F. Barnard, and J. A. Sandberg. *THE THINK ALOUD METHOD A practical guide to modelling cognitive processes*. Academic Press,, 1994.

L. Wood, A. L. Hors, V. Apparao, L. Cable, M. Champion, M. davis, J. Kesselman, P. L. Hegaret, T. Pixley, J. Robie, P. Sharpe, and C. Wilson. Document object model (dom) level 2 specification, 2000. URL `citeseer.ist.psu.edu/wood00document.html`.

L. Wroblewski and E. M. Rantanen. Design considerations for web-based applications. 2001.

W. Zhu. Designing and evaluating a web-based collaboration application: A case study. *Usability Evaluation and Interface Design: Cognitive Engineering, Intelligent Agents, and Virtual Reality*, 1:838–842, 2001.