UNIVERSITY OF OSLO
Department of Informatics

Social Navigation on the Social Web

Unobtrusive Prototyping of Activity Streams in Established Spaces

Master thesis

Eivind Uggedal

August 2008

SOCIAL NAVIGATION *on the* SOCIAL WEB

Unobtrusive Prototyping of
Activity Streams in Established Spaces

Eivind Uggedal

August 2008

Submitted in partial fulfillment of the requirements
for the degree of Master of Science

to the

Department of Informatics
Faculty of Mathematics and Natural Sciences
University of Oslo

This thesis was typeset using the LaTeX typesetting system originally developed by Leslie Lamport, based on TeX created by Donald Knuth.

The body text is set 12/14.5pt on a 26pc measure with Minion Pro designed by Robert Slimbach.
This neohumanistic font was first issued by Adobe Systems in 1989 and have since been revised.
Other fonts include Sans and Typewriter from Donald Knuth's Computer Modern family.

Typographical decisions were based on the recommendations given in *The Elements of Typographic Style* by **?**.

The use of sidenotes instead of footnotes and figures spanning both the textblock and fore-edge margin was inspired by *Beautiful Evidence* by **?**.

The guidelines found in *The Visual Display of Quantitative Information* by **?** were followed when creating diagrams and tables.
Colors used in diagrams and figures were inspired by the *Summer Fields* color scheme found at
http://www.colourlovers.com/palette/399372

ABSTRACT

Social navigation usage on the Social Web were studied by conducting content analyzes to see how prevalent such navigation is now compared to the Web's earlier years. The common characteristic of the types of social navigation we found in these sites were the reliance on peers for the information used in the navigation process. We therefore built on existing definitions of social navigation an provided our own definition which emphasized the essentialness of peers in the web site one is navigating in.

We found activity streams – chronological listings of what all the individuals one is particularly interested in have recently been doing on a web site – to be an interesting and seemingly useful technique of social navigation. A prototype of activity streams were built on top of the Urørt web site to test the usefulness of such a social navigation technique in a real world two-group experiment with a pre-post setup. The experiment results were somewhat inconclusive, partly because of high non-accomplishment rates and some ambiguous results. The high non-accomplishment rates seemed to have a strong connection to the technical prototype plattform we used as participants had a hard time trying to install the necessary software.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This is a master thesis of 60 credits[1] in the field of Informatics. It was written for the Design of Information Systems research group at the Department of Informatics, Faculty of Mathematics and Natural Sciences, University of Oslo.

1. Prescribed to one year of full time study.

Asbjørn Følstad at SINTEF was of great help in building the research design used for our real world experiment. He did also accept our work as part of the RECORD project where we were able to obtain founding. Morten Skogly at NRK Urørt was helpful with providing information about their web site and recruiting participants to our experiment.

Andreas Dieberger, Peter Brusilovsky, and Robert Mertens was helpful with giving us permission to freely use illustrations from their published articles.

Last but not least my supervisor, Gisle Hannemyr, deserves credit for his guidance and thoughtful input during the majority of the master thesis process. Thanks to Tone Bratteteig for helping me out when Gisle was unavailable.

Eivind Uggedal
Oslo, Norway
August 2008

1 INTRODUCTION

The web has come a long way since its inception when it functioned as a global interconnected system for document sharing amongst researchers (?, p. 82). We've seen the coming of an increasingly more social web as "the digital domain has seen a significant growth in the scale and richness of on-line communities" (?, p. 44). There have been an increase from 18% to 45% in blog usage by the general public[1] in an 18 month period from 2005 to 2007. It has been argued that web citizens' familiarity with blogging laid the groundwork for the explosion we are seeing in user participation in web communities (?, p. 20; ?, paragraph 2.2).

1. Represented with a total of 6,545 respondents to a survey conducted in Canada, France, Germany, Japan, the United Kingdom, and the United States by ?, ch. 1, p. 2.

At the same time advances in hardware and web development tools have made it easier and cheaper to create new web sites. We're now seeing an abundance of new offerings in this field. It has been argued that many of the concepts this modern web brings are evolutionary instead of revolutionary (?, p. 45). ?, p. 17 also witnesses a continuous evolution, but with exploratory innovations as he notes that most technological changes are incremental.[2] ?, p. 18 have noticed this trend:

2. ? also believes innovation in computer science is incremental: "I firmly believe that computer science advances by thousands of people solving small problems, which go together and create a massive edifice. Every year that goes by, hardly anything is done that appears to be a milestone worthy of mass attention; yet after five or ten years pass, the whole field has changed significantly".

When we consider a hot, buzz-worthy Web site of the new Internet evolution [...] they are at the same time incredibly innovative and yet – not.

What we're experiencing today with the World Wide Web and social/collaborative software systems was envisioned several decades ago by ? and ?.

During the initial studies of our research we frequented many of these modern web sites. Our impression is that this area of the web infamously coined *Web 2.0* – an increment in version opposed to the age when the Web was in its infancy – is bringing interesting innovations. While they might not be groundbreaking, we justify a closer look at them in this thesis.

1.1 FOCUS

This thesis have a focus on navigational problems and only those which are of a social type.[3] Navigation in context of computer systems is essentially a metaphor based on how people find their way in the physical world. So just as a compass and map can be crucial in your ability to find a cabin deep in the woods during a hike – reliable and efficient naviga-

3. Take a look at § 2.4 (p. 13) to learn more about navigational systems with social characteristics.

tional systems on the web is of uttermost importance when you're trying to locate a certain electronic object containing valuable information.

In addition to only focusing on *social navigation* we're only concerned which such types of navigation on the Web. On the Web we're using hyperlinks (**?**, p. 90) to provide users with navigational choices. We're only focusing on the use of such hyperlinks within web browsers and not navigation support in auxiliary tools as email clients, instant messaging clients, and so on. Our focus is further refined by targeting our research only on what happens inside various web pages. This means that other navigation forms supported by the browser itself or third party extensions or plugins is outside of our scope, as detailed in § 2.2.1 (p. 8).

While we're aware that search is an important part of peoples every day navigational behavior we've introduced additional confinements and decided to only concentrate on browsing behavior (see § 2.2.1 (p. 8) for details).

When studying various web pages it became apparent that some use of social navigation mechanisms implies pretty large privacy concerns. By mining users' previous actions specific user profiles can be generated. One can then represent very sensitive characteristics of individuals such as sexual orientation, political status, and religious beliefs. We feel this subject area of social navigation in relation to privacy warrants a master thesis on its own. Discussion of privacy concerns have therefore been excluded from our research so that we can look more closely at the navigational characteristics of social navigation.

1.2 MOTIVATION

Social navigation are as we'll see in § 2.4 (p. 13) a well defined term within the academic community. During our literature review we collected to the best of our abilities all academic articles where social navigation was discussed. Our approach was to use keyword search and citation search in the databases listed in Table 2.1 (p. 8).[4] Table 1.1 shows the metrics of articles we found about social navigation in context of the modern web as captured by the Web 2.0 term (social network sites, folksonomies, and wikis) and other areas of computer science (classic web, general user interfaces, security, and so on).

	Articles
Modern Web	5
Other	21

Table 1.1: Social navigation in academia, by content. When collecting these statistics we encountered similar articles by the same authors discussing the same problems and systems. In such circumstances the collection of two or more similar articles was counted as one.

4. For more about our literature collection method, see § 2.1 (p. 7).

Our current area of Web 2.0 in relation to navigational problems have in our view (based on our literature findings) little coverage in academia. **?** notes that "'internet time' now runs at at a clock speed several orders of magnitude faster than that of academic research". We described earlier the growth we're seeing of web sites with social aspects and we believe that some of these provide for novel examples of social navigation. It would therefore be interesting to look at some of the state-of-the-art social web sites and look at what contributions they have made to the field of social navigation.

1.3 OBJECTIVE

We'll first try to give an overview of the disparate field of social navigation as found in academic literature. Here we'll look at what social navigation is, different characteristics of social navigation, and finally provide an overview of different types of social navigation.

Based on the concepts we introduce in this overview of social navigation we'll collect examples of social navigational implementations in the wild and analyzing them. In doing this we hope to give a clearer view of how social navigation is used in the Social Web. As we are unaware of any established technique for conducting such a study on real world navigation systems we create our own method as we go – fine tuning it as we learn from our experiences.

We try to improve an existing web site by implementing a navigational prototype using the knowledge we gained from collecting social navigation examples from real world web sites. The navigational technique we decided on implementing is a so called *activity stream*.[5]

5. See § 3.2.2 (p. 39) and § 4.2.2 (p. 47) for more information about this particular social navigation technique.

The Norwegian Broadcasting Corporation's joint TV, radio, and internet project Urørt – a site where artists upload their demos and get valuable playtime on radio and TV if their products are judged to be of sufficient quality – was the candidate for implementing a navigational prototype. Our focus was on the Urørt web community[6] where users can interact in a social manner, listen to other people's songs, and upload their own creations.

6. Available at http://nrk.no/urort.

We decided to build our application in an unobtrusive manner on top of the web site Urørt offered.[7] Based on our prototypical implementation of an activity stream for Urørt we'll provide a discussion of the technical feasibility of such an approach.

7. The rationale for such a decision can be found in § 4.1 (p. 45).

With our technical solution in place we were able to test how it performed in practice by conducting an empirical study with real world users. The insights into activity streams as a social navigation technique will be shared, as well as our experience with providing our technical prototype solution to real world users.

1.4 CONTRIBUTIONS

Contributions from our research on social navigation is threefold:

1 Informing navigational design by giving a structured overview of various social navigational schemes used in academia and the real world.
2 Exemplifying transparent prototyping methods by sharing experiences with creating an unobtrusive shell of navigational designs on top of an existing web site.
3 Applicability of a activity streams as a particular social navigation technique by discussing findings from an experiment of its real world usage.

1.5 OUTLINE

This thesis is composed of two parts:

1. *Social Navigation on the Social Web*. In this part we first give you background information about social navigation before we analyze social navigation in two modern social web sites.
2. *Unobtrusive Prototype of Social Navigation*. The second part starts with an account of how we created an unobtrusive social navigation prototype before we go through an empirical study of the prototype implementation.

After these two parts we conclude our work and give pointers to future work in Chapter 6 (p. 99).

PART I

SOCIAL NAVIGATION ON THE SOCIAL WEB

INTRODUCING SOCIAL NAVIGATION

2

After we've descried how we collected our secondary literature for which we've based this chapter, we'll briefly discuss navigation and sociality – both in general terms and relating to the web. Then we'll concentrate on these two topics together by looking at scholarly research where social navigation is used consciously as a concept. By this we mean the research where either social navigation is defined, redefined or problems relating to the concept is discussed with a basis in such definitions. We'll give an overview of social navigation and its concepts before diving in to the various forms of social navigation that researchers have implemented or proposed. In this latter section of applications of social navigation we'll also include related examples from the real world where appropriate. Finally, at the end of this chapter we'll briefly see if social navigation can be valuable for navigation in web sites.

2.1 LITERATURE SEARCH

Before the literature search was conducted we did some preliminary thinking about (i) the focus of our topic to get more precise results, and (ii) what literature databases would yield sufficient and accurate findings. Based on these concerns we settled on the literature indexes laid out in Table 2.1 (p. 8) and used the following keywords[1] for search:

1. With varying use of modifiers (i.e. AND) or quotations to find exact phrases

- *social navigation* is the concept of our main topic.
- *collaborative filtering* is often used to realize our main topic.
- *recommender system* can be an application of our main topic.
- *tagging* can be related to our topic depending on use.

In addition to keyword based search we also conducted citation searches on the articles that in our opinion seemed to be the most important in the field. The articles that we found relevant during our literature search phase was collected and studied. During this process we eliminated articles by the same authors where similar topics and implementations were discussed and focused on either the most recent or the most representative article.

Type	
Full-text	ACM Digital Library
Bibliography	The Collection of Computer Science Bibliographies
Reference	Inspec Online
Bibliography	HCI Bibliography

Table 2.1: Literature databases used for search

2.2 NAVIGATION

Navigation was traditionally associated with controlling a vessel at sea to a given destination.[2] Since then it's been used to describe behavior related to safely finding ones way whether one is driving a car, flying a plane, or walking on foot. Maps (a graphical representation of the medium one are navigating in) and compass (a tool for connecting graphical maps to the physical world) are often used as aids in this wayfinding.

2. *Navigate* is in fact derived from the two Latin words *navis* meaning "ship" and *agere* meaning "to drive" (**?**, p. 756).

When used in context of computer systems navigation is essentially a metaphor of our usage of the word in our physical world. Through computer systems we present users with a conceptual space in which they can navigate (**?**, p. 189). Today we normally present such a space as a GUI.[3]

3. GUI is short for graphical user interface. Our notion of a GUI was pioneered by **?** and his Sketchpad system.

2.2.1 *Navigation on the Web*

The Web is based on the ideas of *hypertext* – a term coined by **?**, p. 86. The essential part of hypertext are *hyperlinks* (**?**, p. 90) which enables navigation between distinct documents. While **?** was clearly inspired by the work of **?** it has been argued (**?**) that many of the features of hypertext was envisioned by Paul Otlet in his *Traité de documentation* of 1934.

Navigation is important on the Web. Without a way to efficiently and safely navigate one is in danger of becoming lost. This problem was evident even before the Web was invented as **?**, p. 38 describes:

Hypertext offers more degrees of freedom, more dimensions in which one can move, and hence a greater potential for the user to become lost or disoriented.

? studied the navigational support provided by the Web's first browsers: (i) loading of a page by entering its location, (ii) loading a bookmarked page, (iii) loading a page by using a hyperlink on the current page, (iv) recall previously visited pages with forward and backward buttons, (v) recall a previously visited page by locating it in a history list, and (vi) reloading the current page. While modern web browsers support more forms of navigation[4] than the earliest applications we're not concerned with those here. We're only interested in the navigation which are conducted within the main browser window (where web pages are

4. These early browsers' history lists were not remembered between sessions. In addition we're seeing browsers as Flock (available at http://flock.com) with new methods of navigation integrated. There is also an abundance of plugins and extensions for the main stream browsers which enable new possibilities for navigation.

rendered) enabled by following hyperlinks. We can therefore define navigation on the Web for our purposes as:

The behavior of clicking on a hyperlink in a web page.

Following hyperlinks is today the most used navigation method on the Web (**?**, p. 10). **?** gives us a description of the physiology of such navigation which illuminates the thought process of the navigator:

A typical user, faced with a typical, freshly loaded Web page – her eyes bouncing around the page – takes in all the options available. Maybe she scrubs the pointer over a few navigation elements. Then, finally, she's poised to click. In that moment, as her pointer hovers over the link and finger hovers over the mouse button, she has a picture in her mind of what is on the other end of that link.

More specifically, we're either using a strategy of *browsing* or *searching* when we're navigating the Web through hyperlinks. **?**, p. 71 describes the characteristics of browsing:

Browsing is an exploratory, information-seeking strategy that depends on serendipity. It is especially appropriate for ill-defined problems and for exploring new task domains.

Serendipity – "the art of making an unsought finding" (**?**, p. 631) – is what makes browsing effective as a navigation method in some situations. Search as a navigation method does not rely on serendipity as much as browsing since you have a clearer idea about what you're navigating towards.

Today we often use search engines – either local to a particular web page or global for all web pages – when navigating the Web with a search oriented mind set. **?**, pp. 53–54 distinguishes between browsing by navigating with hyperlinks and searching by using a search engine. We've taken this distinction and decided, as stated earlier, to only focus on navigation through hyperlink usage. We'll therefore not look at search with search engines as a form of navigation in our thesis.

2.3 SOCIALITY

If one looks up the adjective "social" in the *Oxford English Dictionary, second edition* **?**, p. 905, vol. 15 it's defined as "capable of being associated or united *to* others". Discussion about explicit social matters is left for scholars of the social sciences. We've therefore briefly introduced the term and are more concerned with situations where it relates to computer systems. More specifically we're going to look at sociality on the Web.

2.3.1 The Social Web

Sociality has become an integral part of our modern age version of the Web. We called this generation of the web for *Web 2.0*[5] in our introductory chapter. When **?** introduced the term he emphasized the characteristics of interaction, community, and openness. But different people give Web 2.0 various meanings and there is no established definition as **?**, p. 15 have experienced:

5. Web 2.0 was first used as the name of a conference arranged by O'Reilly Media. The "2.0" part of the conference name was then used to signify the revival of interest in the web after the dot-com bubble in the early 21st century (**?**). Later the founder of O'Reilly Media, Tim O'Reilly, defined the term as the characteristics of the web sites that survived the dot-com bubble and the web sites he deemed to be the best newcomers to the field (**?**).

Pinning down Web 2.0 is like trying to scoop up water with your hands. You can't really hold onto all of it, but after most of the water runs through your fingers, there's still something left.

Some have synonymized Web 2.0 with the various types of systems on the Web which have been popular in the recent years. Examples of such systems are wikis, social network sites, folksonomies, mashups, blogs, and syndication (**?**, paragraphs 2.10–2.24; **?**, pp. 35–37). But Web 2.0 is not a class of systems (**?**, p. 28) even though these examples often live up to the aspirations of interaction, community, and openness embodied in Web 2.0.

We're sympathetic with the view of defining Web 2.0 more by the attitude it has for enabling user participation for all people (**?**, p. 101) as the Web is becoming democratized (**?**). This shift is concerned with both social and technical factors as certain technology had to be in place for building products that adheres to the principles of Web 2.0. We do however think it's beneficial to use some examples of systems when describing Web 2.0 as a term. We'll now look at several of these examples of Web 2.0 systems and the more general characteristics, both social and technical, of the Social Web.

6. AJAX is an acronym for Asynchronous JavaScript and XML and was introduced as a term in 2005 (**?**). It captures how modern applications on the Web uses JavaScript for retrieving data asynchronous with the `XMLHttpRequest` object found in most recent web browsers. XML was then exemplified as a possible data-interchange format for the asynchronous requests. It's not a technical term but describes how a suite of technologies can be used together to create interactive web pages. In addition to the technologies mentioned above one commonly use standardized markup and presentational languages for presenting information and JavaScript to not only fetch data, but enable behavior (**?**, p. 282).

Improved interaction

One can argue that the most important technological change related to interactiveness since the Web's infancy was when **?** introduced AJAX.[6] More elaborate interaction due to technological advances such as AJAX enables production of applications on the Web previously only viable to implement as desktop software (**?**, p. 101; **?**, p. 44). We're now able to create systems just like we've done on the desktop for 25 years, only in a different medium (**?**, p. 64). Interestingly, support for the core technical feature of AJAX was introduced in March 1999 when Microsoft Internet Explorer 5 was released (**?**). It would still take almost six years before such technologies saw such widespread use that a new term was warranted.

One possible reason for the lack of early developer uptake of this new technology could be the disparate field of browser implementations. Different browsers have variations in their interfaces for interacting with web documents through JavaScript.[7] It's quite hard to implement an ap-

7. For more about JavaScript, a programming language often used to implement behavior in web browsers, see § B.1.1 (p. 123).

plication when one have to write your code to handle several differences in browsers. The JavaScript web platform have been described as “a really hostile programming environment” (?). AJAX comes with a price. One have to be quite proficient in the intricacies of each browser to develop truly cross-browser applications. Thankfully frameworks that abstract away such tediousness have come to the rescue (?, p. 45). We believe part of the flourishing of AJAX technologies are due to frameworks' ability to make browser development friendly for the average programmer. At the moment of critical mass AJAX hit a tipping point and the usage and uptake changed dramatically similar to the way an epidemic spreads (?, pp. 8–12).

Social network sites

Community brings the social aspect to the Web. While social interaction on the Web is nothing new, greater availability for all citizens of the Web to take part in such interaction is.

? give their definition of a *social network site* as:

web-based services that allow individuals to (1) construct a public or semi-public profile within a bounded system, (2) articulate a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system.

The first site that adhered to **?**'s definition was arguably SixDegrees, which saw the day of light in 1997.[8] There had been web sites that implemented parts of this definition of social network sites, but SixDegrees was the first to include all necessary characteristics.

The three most influential social network sites until this point of time have been Friendster,[9] MySpace,[10] and Facebook[11] according to **?**. We feel this holds true if one have an American world view.[12] Friendster made some blunders by not listening to its users' wishes and are therefore not widely used today. MySpace was launched in 2003 hoping to attract unsatisfied Friendster users (?). They were successful in this endeavor in addition to attracting loads of music bands and later teenagers. Facebook seems to be the hot social network today[13] attracting users from many walks of life.

8. SixDegrees closed their operations in 2000. Its founder believes SixDegrees simply was ahead of its time (?).

9. Available at http://friendster.com.

10. Available at http://myspace.com.

11. Available at http://facebook.com. For the social navigational characteristics of Facebook, look at § 3.2.2 (p. 38).

12. Blink (http://blink.dagbladet.no) for instance was a popular social network in our home country, Norway, before Facebook became adopted by Norwegians.

13. Facebook is now the largest social network site in our home country, Norway. As of May 14, 2008 it had approximately 1,142,300 registered profiles from Norwegian users. If every profile represents a unique individual (highly unlikely) this would mean that almost a quarter of all Norwegians are registered at Facebook. Data was collected by starting to register an advertisement for all Norwegian members and capture the market reach statistics. Population data was gathered from Statistics Norway (http://ssb.no).

Mash-ups

Openness enables exchange of information between different parties so that new services on the Web easily can be created. This phenomenon, a *mash-up*,[14] occurs when information and/or functionality from separate web sites and services are brought together in a complementary way (?, p. 36).

14. The term mash-up is taken from the similar activity finding place within the music scene where artists combine the music from one song with the *a capella* from another song (?).

Mash-ups is most often created by utilizing one or more web APIs.[15] ?, p. 86 argues that web APIs enable innovation since they provide access to robust technologies and massive amounts of content – something no individual could create for himself. In addition these APIs lowers the barriers to entry since they often provide a sound and efficient interface to content. Before the days web APIs were commonplace developers would resort to scraping web sites for getting a hold of data. Even though such scraping still happens today, it seems that web APIs are proliferating.

15. API is short for Application Programming Interface and can be seen as a "set of calling conventions defining how a service is invoked through a software package" (?, p. 2).

Collective intelligence

The notion of *collective intelligence* is important for understanding the characteristics of our modern web. It's been argued that the sharing we're seeing in blogging, Wikipedia,[16] and mash-ups "could lead the way to a truly democratic network, where producers and consumers are one and the same" (?, p. 23). This change is however not only technological, it represents a fundamental mind shift (?, p. 206). Collective intelligence is not unique to the Internet but the communication facilities enabled by this relatively new technology have created new ways for widely dispersed people to work together (?). The result is a lower barrier to entry for taking part in a collaborative process where a shared intelligence emerges.

16. Wikipedia – the free content encyclopedia allowing submissions from everyone – can be found at http://en.wikipedia.org.

Collective intelligence is closely related to *wisdom of crowds* – a phenomena that describes the amount of information contained in a group's collective verdict. In many situations the crowd is able to hold a complete picture of the world in their collective brains (?, p. 11). The larger the crowd, the more accurate their answers will be.[17] A wise crowd is characterized by diversity of opinion, independence, decentralization, and aggregation (?, p. 10). ? argues that one have to design for selfishness to make collective intelligence work in a community. If an individual don't have self-interest in contributing knowledge, it will seldom happen. ? therefore sees collective intelligence as "selfish behavior aggregated for the common good".

17. Take for example the Google search engine which we've now grown accustomed to use in our daily search because of the accuracy of the results it provides. The underlying principle of Google's search algorithm called PageRank is that a page is rated of importance based on how many pages who link to that page and the importance of the pages that linked there (?, p. 109).

In the case of Wikipedia, ? found a sample of science articles to be comparable in accuracy to similar articles in *The New Encyclopædia Britannica*. While the quality of content in these two sources was similar, readability and structuring of content seemed to be better in the professionally edited encyclopedia. ? argues that while Wikipedia can be accurate it lacks personality and context. In his view it's important to know whom the author is and in what setting information is written.

? goes on to questioning the resurgence of collectivism on the Web, not just in Wikipedia. He thinks the reason for people's blindly usage of collectivism is happening since bad old ideas packaged in modern technology have an confusing ability to appear fresh. Just as individuals can be either stupid or intelligent he feels the collective can be both stupid in some cases and intelligent in others. Both individual and collective intelligence is important since these two forms seems to not be intelligent in the same settings.

? offers a set of conditions that have to be in place for enabling the collective to be smarter than the individual:

The collective is more likely to be smart when it isn't defining its own questions, when the goodness of an answer can be evaluated by a simple result (such as a single numeric value), and when the information system which informs the collective is filtered by a quality control mechanism that relies on individuals to a high degree.

Taking the advice of **?** we have to question the answers the collective gives us by providing structure and constraints and firstly rely on intelligent individuals.

2.4 SOCIAL NAVIGATION

Drawing on the previous explanation of navigation and definition of social, we can combine the two terms. Social navigation then means going from one point to another in a medium with other people.

Social navigation as a term was introduced in an article by **?** where they discussed three types of navigational mechanisms, spatial, semantic,[18] and social, which they argue can be separated even though there is evidence of situations where the different mechanisms are combined. In their description of the social type of navigation **?**, p. 1 coined the term *social navigation*:

18. *Oxford English Dictionary, second edition* defines the adjective *semantic* as "Relating to signification or meaning" (**?**, p. 939, vol. 14). Semantic navigation on the Web is navigation when one utilizes the semantic properties of hyperlinks and the semantic relationships amongst them.

When navigable information systems are extended to support collaborative activity, a third model of navigation arises. This is social *navigation. In social navigation, movement from one item to another is provoked as an artifact of the activity of another or a group of others.*

? exemplifies two cases where neither location (spatial) nor content (semantic) is used for exploration – the social model is used on its own. Based on these two experiences **?** argues that we possibly need to move away from spatial models of navigation and rather focus on designing explicitly with semantic and social navigational techniques.

? highlights an important aspect for making interaction on the Web smoother. With an "awareness of the presence of other users" (**?**, p. 812) one can give an indication of what parts of a web page that is of high demand and possibly identify the users accessing them. This means that one can move in the direction others are heading – one can follow the stream.

?, p. 39 include the properties of *personalization* and *dynamism* into their understanding of what social navigation is. Social navigation is not pre-planned, but grown dynamically in an organic fashion. This distinction is shown by the example of walking down a road in a city versus walking on a forest trail. Personalization means that the navigation advice is given to the receiver in a fashion that suits him. Related to dynamism is social navigation's temporal nature. **?**, p. 39 shows this with the analogy of a forest trail which will vanish if it's not used. This idea was envisioned for computer-like systems by **?** over half a decade ago in that "trails that are not frequently followed are prone to fade, items are not fully permanent" (**?**, p. 106).

? argues that while social navigation is plentiful in our everyday world it's not implied that it's a good idea to implement computer based systems with this perspective in mind. Instead of creating translations from our physical world to our virtual world they explain that one instead have to "make information spaces afford social interactions and accumulate social trails" (**?**, p. 377). With *social trails* the authors mean traces left in the system by past users guiding current users' navigational behavior.

? on the other hand argues that one can not rely on technological structures alone when using social navigation which "transforms a space on a computer network into a virtual place" (**?**, par. 50). During an ethnographic study the author examined social navigation in relation to the persistent structures found in the physical world during a distance education program. She found that these real world structures supported and afforded social navigation in virtual places.

2.4.1 *Definition*

The most detailed definition of social navigation to our knowledge is given by **?** in his Ph.D. thesis. To understand his definition we'll have to introduce his nomenclature for the actors in a social navigation process:

- *The navigator* is "the person seeking navigational advice" (**?**, p. 20).
- *The advice provider* is a "person or artificial agent providing navigational advice to a navigator" (**?**, p. 20).

Social navigation was then defined by **?**, p. 20 as:

... navigation that is conceptually understood as driven by the actions from one or more advice providers.

Firstly, **?** talks about navigation which is "conceptually understood" as driven by these advice providers. As long as the user believes his navigational choices are driven by advice providers it is social navigation. Secondly, the actions that the navigator is driven by need not be only direct advice from a single advice provider, but can also be aggregated of nature.

2.4.2 *Fundamental categorization*

We'll take a look at broader characteristics of social navigation before we'll continue with a discussion of several technical applications of social navigation found in secondary academic literature.

Active, direct, passive, & indirect social navigation

In his classic article **?** distinguishes between *active social navigation* and *passive social navigation*. Such a distinction is grounded in the nature of the exchange of information between the two parties involved in a social navigation process. These are the advice provider – the creator of navigation cues – and the navigator:

- *Active* social navigation finds place when a person either deliberately seeks out another and asks for a navigation advice or intentionally gives away such navigational advice.
- *Passive* social navigation happens when people make available navigational aids that later can be used by another person.

? groups navigation of a social type in *direct social navigation* and *indirect social navigation*:

- *Direct* social navigation occurs when "communication between navigator and advice provider is mutual and two-way" (**?**, p. 21).
- *Indirect* social navigation is where "communication between navigator and advice provider is non-mutual and in one direction" (**?**, p. 21).

Despite **?**'s more precise wording, active social navigation is similar to direct social navigation. Both are differentiated with passive social navigation which is similar to indirect social navigation. **?** characterizes the relationship between the navigator and advice provider. **?** on the other hand describes the nature of the communication between the two parties.

Explicit & implicit advice

Related to passive or indirect social navigation is the notion of *explicit feedback* and *implicit feedback*. These terms can be used for distinguishing how passive or indirect social navigation is provided by an advice provider. Collecting advice given by an advice provider explicitly means that the advice provider have to use conscious effort to make the advice available. Such an advice provider can for instance choose to share an interesting web site and does so by putting a hyperlink to it on his web page.

When advice is mediated implicitly the process for so doing are transparent and unobtrusive for the advice provider. Based on the work the advice provider would have done regardless of the social effects it conveys one can provide advice to future navigators. An example of such behavior is recording of browsing history that can be computationally evaluated to provide advice for others. We'll return to such recording of history in § 2.5.3 (p. 21).

Active and direct social navigation inherently make advice available by explicit feedback. Partaking in these direct methods of social navigation will always require conscious effort by the advice provider.

2.5 FORMS OF SOCIAL NAVIGATION

Social navigation have been applied in various forms described in academic literature. What follows is a review of these forms of social navigation sprinkled with examples from our modern web.

2.5.1 *Hyperlink sharing*

? is particularly concerned with making handling of URL[19] entities transparent for users both in the operating system and in various tools related to web browsers. Making URLs invisible to users will in his opinion enable more widespread use of social navigation through pointer sharing. Web browsers handles URLs embedded as hyperlinks transparently, so we're not going to elaborate on matters of URL handling in auxiliary systems here.

19. Uniform Resource Locator. URL was formerly an abbreviation of Universal Resource Locator.

Both **?** and **?** observed social navigation activity on the Web when hyperlinks were shared on web pages. Creators of these pages often had a list of pointers to other web pages. These were the pointers they deemed interesting enough to actually go through the trouble of creating such a listing for. By doing this they created an opportunity for navigation based on social factors.

While pointer pages still is in existence, it seems that the increasing use of blogs (see Chapter 1 (p. 1) for details) have resulted in a new form for sharing interesting web pages, which often is other blogs. So called *blogrolls* is a way for blog authors to list other blogs they are reading regularly. They thereby function "as a navigation tool for readers to find other authors with similar interests" (**?**, p. 3). An example of a blogroll can be seen in Figure 2.1.

Social bookmarking

A new phenomena appeared to the mainstream with the introduction of the del.icio.us[20] *social bookmarking* system. This web site made individual bookmark collections globally available, making it easy to discover what other people was taking notice of. Interestingly **?** created a system in the early days of the web with almost the same features as social bookmarking systems of today. This system, WebTagger, allowed individuals to store bookmarks that later could be retrieved by other people. The

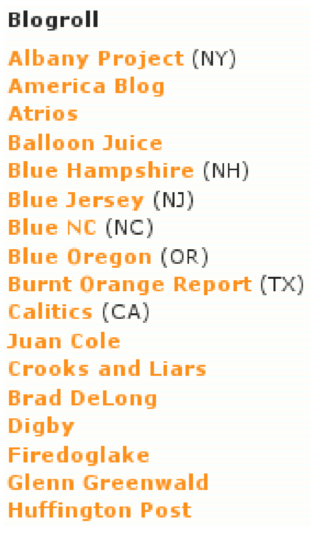

Blogroll
Albany Project (NY)
America Blog
Atrios
Balloon Juice
Blue Hampshire (NH)
Blue Jersey (NJ)
Blue NC (NC)
Blue Oregon (OR)
Burnt Orange Report (TX)
Calitics (CA)
Juan Cole
Crooks and Liars
Brad DeLong
Digby
Firedoglake
Glenn Greenwald
Huffington Post

Figure 2.1: Blogroll for Daily Kos, retrieved December 5, 2007, from http://www.dailykos.com/. Daily Kos is one of the most popular American collaborative political blogs where peoarchitecture of the system was based on a web proxy, which enabled controls for storing the location of a given web page to be present within the web page itself. Since the system was proxy-based, only users having enabled the proxy server in their browsers could take advantage of the shared bookmarks.

As **?**, p. 806 argues, the Web's growth – even at its modest size in 1997 compared to its staggering size over 10 years later – have implications on how easily it is to locate information. By creating pointer pages, and now socially shared bookmarking services, users are imposing a structure on the web. By navigating these kinds of interlinked hyperlink collections it's quite plausible that users are getting access to more highly related and higher quality information. Sharing a hyperlink, either on you web page or through a bookmarking service, requires a conscious effort. One would believe that people only choose to do so for information they find interesting.

2.5.2 *Tagging*

In addition to being a modern form of pointer pages, social bookmarking with del.icio.us introduced a new way to annotate all kinds of items.[21] By applying textual keywords to bookmarks – and later other types of content – users were able to browse such collections in new ways. These keywords have been popularized as *tags* and the act of applying them is called *tagging*.[22] Joshua Schachter, creator of del.icio.us, highlight tagging as its most essential feature – the feature that set it apart from the competition (**?**, p. 225). Tagging solves a recurring problem with using traditional folder or hierarchical categorization of items like bookmarks. In such a system an item can only go in one folder. With tags items can live in several categories at once (**?**, p. 93).

21. Like photos, articles, wine, books, videos, music, and so on.

22. Tagging was discovered by Joshua Schachter when he kept a plain text file with a list of all his web page bookmarks. He annotated these bookmarks by introducing single-worded labels prefixed with a number sign (#). He could then easily search his bookmarks file with these labels by prefixing searches with the number sign. Schachter later introduced tagging to the masses by creating the del.icio.us social bookmarking site (**?**, p. 92).

The WebTagger system by **?**, p. 1109 we described earlier had a novel approach to bookmark categorization:

The system provides a simple means of organizing and sharing bookmarks using a structure-neutral categorization scheme, rather than a hierarchical filing scheme. The neutrality of this bookmarking scheme allows users to concentrate on tagging URLs with the most appropriate categories to facilitate subsequent retrieval, rather than forcing users to select a single best location within a rigid hierarchical structure.

This description of the categorization scheme used in WebTagger very much resemble tagging as found in del.icio.us. **?** even describes the act of categorizing in this way as *tagging*. While Joshua Schachter may never have heard of WebTagger, it's evident that tagging was first used in the WebTagger system – the first social bookmarking service.

Folksonomy

Tagging enables a user driven taxonomy (classification) which is often called an *folksonomy* – a combination of the words *folk* and *taxonomy*. A folksonomy is a strictly bottom-up approach because of the lack of any predefined taxonomic structure. Folksonomies therefore rely on "shared and emergent social structures and behaviors, as well as related and linguistic structures of the user community" (**?**, p. 31). Since we're mainly interested in the navigational possibilities tagging can give us, we're leaving out a deeper discussion of the benefits and drawbacks of folksonomies.[23]

23. For those interested **?** and **?** gives a detailed account of the tag usage and structure in respectively del.icio.us and Flickr – a social image sharing web site we dig deeper into when we analyze its social navigation capabilities in § 3.2.1 (p. 35).

Tag sharing & scope

As we've described tagging is often a collaborative process. Some web pages for instance give suggestions for tags if the item you're annotating have been tagged by others previously. Based on our own usage of collaborative tagging system we seem to be more inclined to use some or all of these tags than to come up with our own. In other words, our vocabulary is influenced by the user community. **?**, p. 186 confirmed our personal observations when they found that the community influence affects the vocabulary of tags an individual uses. **?**, p. 355 conducted similar studies on a collaborative tagging service where the user interface did not display the tags other people had applied for a similar resource. They did not find any significant reuse of tags from other users and explained this discrepancy with the lack of visualization of other user's tags while tagging an item, as evident in other bookmarking services. This means that one can influence the vocabulary of users when they are applying tags by showing the vocabulary of other users.

In addition to being shown other people's tags when tagging one can also be given a list of the tags oneself have previously used. Under such circumstances **?**, p. 185 found that the probability of applying a previously used tag rose as the amount of tags the user had applied increased. **?**, p. 355 validated this phenomena by showing similar results from another collaborative tagging system.

Applying your own tags for a given resource makes sense if you're annotating a bookmark. You have your own representation of the bookmark given by the name you gave it and the tags you have chosen to apply. Since a bookmark is distinguished by a URL others can have other representations of the same resource. For other content items as photos in a photo sharing site it may make more sense to allow every user, not only the creator, to apply globally visible tags for this single item. There is then only one representation of this item and its tags. Tags need not be collaboratively created. If for instance one are tagging one's personal email messages it makes sense to keep such behavior private.

As we've seen folksonomies can be separated by their level of tag sharing: private systems, fully open systems, and systems with user con-

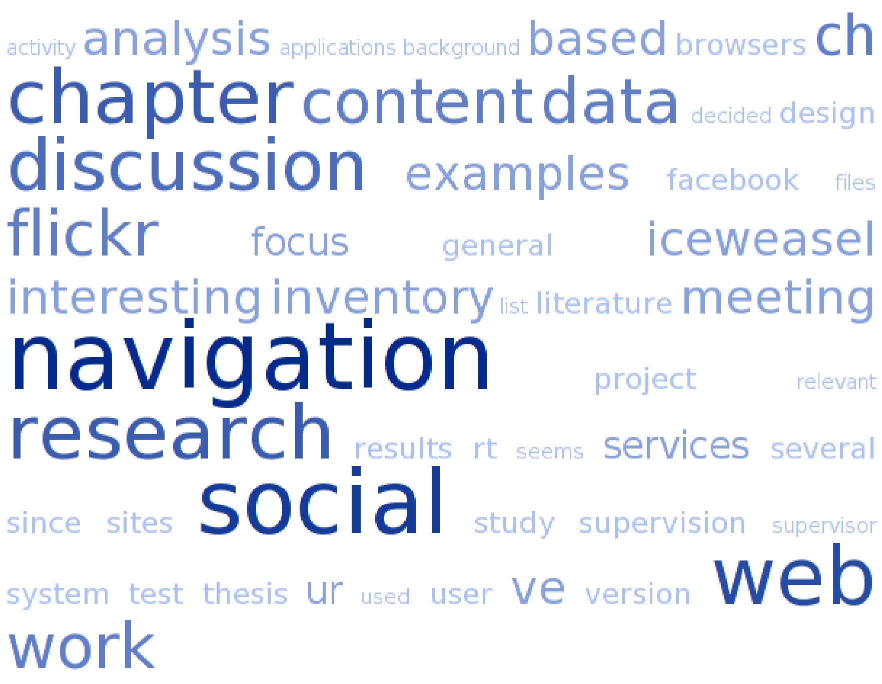

activity analysis applications background based browsers ch
chapter content data decided design
discussion examples facebook files
flickr focus general iceweasel
interesting inventory list literature meeting
navigation project relevant
research results rt seems services several
since sites social study supervision supervisor
system test thesis ur used user ve version web
work

Figure 2.2: Tag cloud for the authors private research journal located at http://journal.redflavor.com. The tag cloud uses both font size and color to distinguish between the frequency of usage for the different tags. Generated with TagCrowd (http://tagcrowd.com) on May 12, 2008.

trol over what gets shared. In addition one can categorize folksonomies based on tag scope: tags applied to an item globally or belonging to separate users.[24]

24. For more about different characteristics of folksonomies see ?, pp. 34–36 and their detailed account of such matters.

Annotating items seems to have benefits with regards to describing the items and using them for categorization. But how does this relate to navigation? By giving users a means to better describe various items it will hopefully be easier for others to use this information in navigation – they will hopefully more easily find the items or resources they are searching.

Tag Clouds

The seemingly most used way to display tags for navigation is by generating a so called *tag cloud*.[25] ?, p. 1 succinctly defines this visualization technique:

25. According to ? the first use of tag clouds was on Flickr for showing tags applied to photos. The idea of such visualization most likely came from ? in his display of search terms on his web site.

The cloud is a representation of the frequency-based relation of tags.

This means that a tag cloud is used to visualize how frequent various tags are applied to one or more objects. Frequency is usually portrayed by varying the font size based on usage. A highly utilized tag has a large font size while less used tags have smaller font sizes. There are usually several levels of font sizes in a tag cloud to visualize how popular tags are in relation to others. Sometimes colors is used in addition to font size to even better distinguish among the frequency of tag usage by showing the most used tags with a higher contrast color than less used tags. Lastly tags are most often listed alphabetically giving a visualization that in many ways resembles clouds of various sizes in the sky. Figure 2.2 (p. 19) shows an example of a tag cloud utilizing font sizes and color tones while Figure 3.4 (p. 36) shows an example of a tag cloud using only font sizes for distinguishing tags by frequency.

?, p. 996 conducted a study of how varying properties of tag clouds affect use. Unsurprisingly, font size had a large affect on how well a given tag was perceived. Layout had a minor, but noticeable effect on how well the users got an impression of the tag cloud. **?**, p. 1314 expanded on the findings by **?** and found tag size to be important in how fast information is found. During their study **?** also noticed that unalphabetized tag clouds were inferior to alphabetized tag clouds when finding information.

?, p. 18 studied whether tag clouds provided value for individuals seeking information through a folksonomy by making a user interface which supported both search by keyword and navigation by tag cloud. They found that a majority of users utilized the tag cloud when looking for information (**?**, pp. 22–23). When looking for non-specific information – when the users were merely browsing – this trend were even stronger. But for finding specific information using search by keyword required fewer queries than using tag clouds (**?**, p. 24). **?**. give merit to tag clouds as a navigational interface since it reduces the costs of query – clicking is faster than typing and scanning the tag cloud is faster than formulating a search query (**?**, p. 27). The most important finding to take away from this study is that tag clouds does not function well as the sole navigation mechanism for folksonomies. Complementary navigation, with for example search by keyword, needs to be in place for enabling efficient navigation.

Tags as social navigation

? gives an account of how they used collaborative tagging as a means for enabling social navigation in their Dogear social bookmarking system. When users were navigating bookmarks they most frequently browsed bookmarks for a given user. But browsing by a tag was not much less frequent, supporting evidence of the usefulness of folksonomies for enabling social navigation. In addition it was found that of all bookmarks clicked, 74% was of other user's bookmarks. Such a high ▮▮ usage of other users' bookmarks was interpreted as a sign of high degrees of social navigation within the system.

Problems with tagging

?, p. 1 argues that social tagging is ideal in situations where you have objects one can not easily perform keyword search on. If such objects for instance are composed of video content, tags can serve as an augmenter for performing keyword based searches as one can do in textual content. They leveraged tagging in this manner when creating a prototype of Wikipedia supporting video content – using tags as the principle navigation mechanism. By doing so **?**, p. 2 saw the need for bootstrapping the availability of tags so that users would be more inclined to create their own tags. Their solution was to algorithmically create tags based on the textual contents of Wikipedia. They hoped the existence of these tags would stimulate users to start tagging themselves. Just as social network sites have problems satisfying users before a sufficient amount enrol, folksonomies have initial pains when few annotations are available.

Tagging have further shortcomings. Tags could be misspelled, tags with the same name are not always homonymous, and tags with the same meaning does not always have the same name because of synonyms (**?**, p. 59).

In addition **?**, p. 943 argues that browsing tags by traditional methods with keyword search or tag clouds is inefficient when the set of tags are quite large. They implemented a system to mediate the synonymy and homonymy problems with tags in addition the the problems with browsing a large collection of tags. Their solution for tag ambiguity was to generate the semantic concept[26] of a tag and use that semantic meaning when the user was looking for resources through tag browsing (**?**, p. 946). As **?**, p. 95 argues, this problem with tag ambiguity does not really matter when the collection of annotated items becomes sufficiently large. One would only be concerned with such matters if one need to find every possible item that is associated with a concept.

The problem of browsing large scale tagging collections can be tackled by inferring a hierarchy[27] from the flat tag space (**?**, pp. 946–948).

26. Generating the semantic concept of a tag means to derive its meaning in a broader sense. Say for instance that a user is browsing for *movies*. An algorithm that generates the semantic meaning of *movies* could for instance map this to the concept of *movie*, where such tags as *movies*, *film*, and *flick* could be associated with the concept of *movie*.

27. One can tag objects by several levels of abstraction. One can for instance tag a movie with *movie* to identify what it is. Then one could use the tags *comedy*, *romanticcomedy*, and *norwegian* for describing the object's features. One could computationally derive a hierarchy from the varying levels of abstraction in such tags saying that *comedy* is the child of *movie* and *romanticcomedy* is the child of *comedy*.

We've seen that item annotation or tagging can be used to annotate items for describing the information they convey and thereby afford navigation. As we'll see in § 2.5.5 (p. 25) annotations can also be used for describing the quality, importance, or usefulness of an item and thereby potentially creating recommendations.

2.5.3 Interaction history trails

? contrasts the digital world of computers with our physical world with respect to the formers lack of history. In our traditional world we exploit such historical information traces "to guide our actions, to make choices, and to find things of importance or interest" (**?**, p. 270). It's argued that this apparent lack of history in computerized systems must be sorted out such that future users can take advantage of past users' historical traces left when they were working on solving problems similar to the current user's. A possible remedy for this problem on the Web is put forth in the authors' Footprints system – a navigational aid as an extension to normal web browsers. This navigational aid visualizes interaction history of past users, enabling current users to navigate this history. The interaction history consists of several navigation trails which are "coherent sequences of nodes followed by an individual" (?, p. 273).

The idea of such trails of navigation far preceded ? as they were envisioned by ? when he proposed the infamous theoretical computer-like system named the Memex.[28] ? describes a scenario where users are building trails explicitly, inserts comments if needed, and gives it a name. ? on the other hand implemented a system where trails were automatically collected using a set of heuristics to identify browsing behavior representing a coherent navigation trail. These characteristics makes Footprints a passive and indirect social navigation system.

28. The Memex was not envisioned as a computer system but as a mechanical system consisting of a set of controls hooked up to a microfilm reader and camera. It was theorized by ? to be a system for handling a person's entire collection of documents, books, and communication. It was important that a user would be able to access this information with great speed and flexibility. An integral part of enabling such efficient access was a user' and content providers' ability to introduce trails between information items. ?'s writing about trails inspired hypertext (?, p. 86) which in turn was the grand idea behind the World Wide Web (?, p. 49).

? wrote his essay before the invention of computer networks and he thinks of each Memex as a separate island. Sharing of trails is possible through an exportation and following importation process, making it an explicit action for its users. The Footprints system makes the social process of sharing trails implicit and transparent to its users – multiplayer is forced.

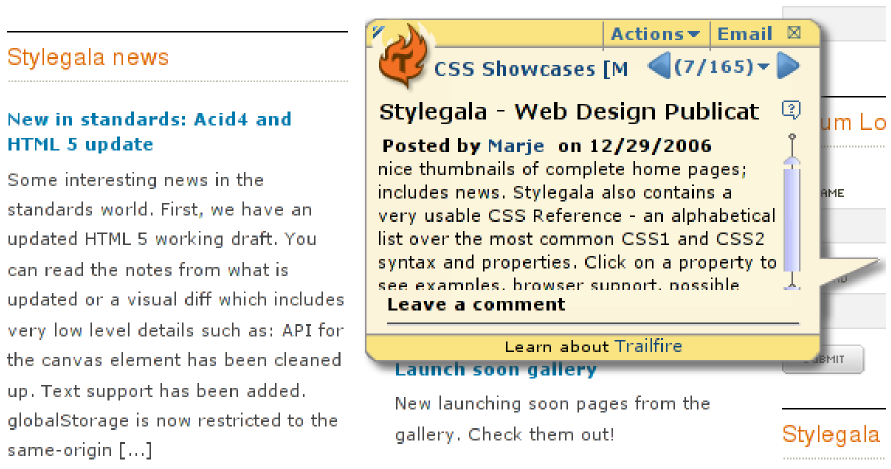
Controlled user studies by ? did partially falsify their pre-test hypothesis of Footprint's ability to let users find more relevant results in a more efficient manner during a specific browsing task. Users of the history-enriched system reported significant lower values of mean page count in their browsing tasks. No significant difference in the quality of the located information was found between users of a plain web browser and those with a browser enhanced with Footprints. They also found people experienced in the problem domain of the browsing task to a larger degree being able to take advantage of interaction history compared to novices. ? attributed this finding to experienced people's ability to have a clearer mental model of the information they was browsing.

Trailfire[29] is a modern incarnation of some of ?'s Memex ideas. By installing a browser extension users can create trails of web pages which are automatically shared with other users. The difference between Trailfire and Footprints is that trail creation in the former system is an explicit task while the latter system make this process implicit. Figure 2.3 shows one web page in a trail created with Trailfire. Each page in the trail gets an information box with navigation controls for moving through the trail, in addition to information about why this particular page was included in the trail.

29. Available at http://trailfire.com.

2.5.4 *Collaborative filtering*

Collaborative filtering "help people make choices based on the opinions of other people" (?, p. 175). These choices are often used for navigation. Since the collaborative process of creating opinions are of a social nature collaborative filters are a form of social navigation.


Stylegala news

New in standards: Acid4 and HTML 5 update

Some interesting news in the standards world. First, we have an updated HTML 5 working draft. You can read the notes from what is updated or a visual diff which includes very low level details such as: API for the canvas element has been cleaned up. Text support has been added. globalStorage is now restricted to the same-origin [...]

Actions ▾ | Email

CSS Showcases [M (7/165) ▾

Stylegala - Web Design Publicat

Posted by Marje on 12/29/2006

nice thumbnails of complete home pages; includes news. Stylegala also contains a very usable CSS Reference - an alphabetical list over the most common CSS1 and CSS2 syntax and properties. Click on a property to see examples, browser support, possible

Leave a comment

Learn about Trailfire

Launch soon gallery

New launching soon pages from the gallery. Check them out!

um Lo

Stylegala

Figure 2.3: Browsing a trail in Trailfire, retrieved May 22, 2008, from http://trailfire.com/Marje/marks/37759.

The origin of the concept comes from the Information Tapestry project where users could annotate electronic documents arriving in a continuous stream – typically email messages (?, p. 69). By installing a pre-existing filter, or creating one from scratch in a special query language, users were able to filter out the essential documents based both on explicit feedback through annotations and meta-data concerning the document (?, p. 62).

? expanded on the ideas put forth in the Tapestry project and created GroupLens, a system for collaboratively filtering Usenet postings. What was novel with their approach at the time was that they could "automatically determine how much weight to place on each evaluation, based on the degree of correlation between past opinions of the reader and evaluator" (?, 185). This made the filter more personalized and would arguably give more satisfying results. Such an approach gives a conceptual model where (i) the user enters ratings which constitutes a profile for that individual, (ii) the collaborative filtering system finds the individuals with similar profiles – neighbors, (iii) the ratings of the user's neighbors are aggregated to form recommendations (?, p. 243).

During further studies of the GroupLens system ?, p. 84 noticed some users' lack of incentive to rate content. Their solutions was to introduce implicit ratings in addition to the existing explicit ratings system already present.[30] The implicit ratings were collected by monitoring whether a user read an item, and for how long he kept reading it. Their initial studies showed positive results for implicit ratings in that they were nearly as accurate as explicit ratings. ?, p. 39 validated the results ? experienced when they tested explicit and implicit ratings against each other. ? found implicit ratings to be somewhat less accurate than explicit ratings, but suggested implicit ratings could be used successfully since they won't introduce the additional overhead explicit ratings embodies.

Tapestry and GroupLens lacks a direct connection between the advice giver and the navigator. Using the language of ? they are both

30. Implicit and explicit ratings reflect how the feedback is given by the advice provider. We looked at this separation in § 2.4.2 (p. 15).

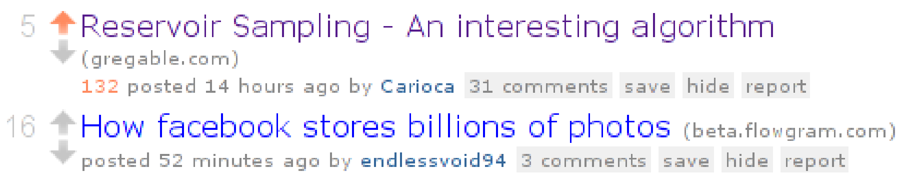

5 Reservoir Sampling - An interesting algorithm (gregable.com)
132 posted 14 hours ago by Carioca 31 comments save hide report
16 How facebook stores billions of photos (beta.flowgram.com)
posted 52 minutes ago by endlessvoid94 3 comments save hide report

Figure 2.4: Collaborative filtering at Reddit. Stories can be voted up and down by using the up and down arrows. The first entry has been voted up by the author (the up arrow is highlighted red) which gave the submission a point, resulting in 132 points total. Retrieved June 25, 2008, from http://reddit.com/r/programming.

passive collaborative filtering systems. **?** noticed how passive filtering systems required a critical mass of users to be useful. They therefore created an active collaborative filtering system. **?** found circumstances where one of the two types of collaborative filtering systems offered better solutions:

In "passive" collaborative filtering the system works better the higher the convergence of votes on the same set of documents. In contrast, the benefit of "active" collaborative filtering increases with the divergence of documents that are found.

?, pp. 242–243 gives compelling reasons for explaining to users how the collaborative filtering process works. Usage experiments showed explanations' ability to increase the acceptance of advice given by collaborative filtering systems. Whether explanations improved actual filtering performance was not certain – even though the authors strongly believed in this idea (**?**, p. 250).

Collaborative filtering is used on several modern web sites. One example is Reddit – a web site where people can submit links to everything they find interesting. Users of the community then votes these submissions up or down (explicit advice) as one can see in Figure 2.4. Submissions are then sorted based both on how many votes it have and how new it is. Users are not given personalized listings based on their voting history. Reddit is thus similar to Tapestry with regards to how the collaborative filtering works in that similarities amongst users is not inferred (as in GroupLens).

As Reddit recently became open source[31] we were able to take a peek into its collaborative filtering functionality. We did this to dissect how collaborative filtering algorithms are used in real applications. The source code of this algorithm can be found in Source Code Listing D.1 (p. 141). What follows is the algorithm used on Reddit in mathematical notation.

31. The source code of Reddit can be found at http://code.reddit.com.

Given the time the entry was posted t_{posted} and the time of 7:46:43 a.m. December 8, 2005 t_{cutoff}, we have t_s as their difference in seconds

$$t_s = t_{posted} - t_{cutoff}$$

and v_{sum} as the difference between the number of up votes v_{up} and the number of down votes v_{down}

$$v_{sum} = v_{up} - v_{down}$$

where the sign $s \in \{-1, 0, 1\}$

$$s = \begin{cases} 1 & \text{if } v_{sum} > 0 \\ 0 & \text{if } v_{sum} = 0 \\ -1 & \text{if } v_{sum} < 0 \end{cases}$$

and v_{max} as the maximal value, of the absolute value of v_{sum} and 1

$$v_{max} = \begin{cases} |v_{sum}| & \text{if } |v_{sum}| \geq 1 \\ 1 & \text{if } |v_{sum}| < 1 \end{cases}$$

we have the rating as a function $f(t_s, s, v_{max})$

$$f(t_s, s, v_{max}) = \log_{10} v_{max} + \frac{st_s}{45000}$$

The resulting rating is a real number which is used for numerically sorting entries. This is neither a computationally expensive algorithm nor a very complicated algorithm. In spite of this we feel it delivers satisfying results (as daily users of Reddit).

As we've seen, collaborative filtering is applied to generate recommendations for users. These recommendations can aid users in navigating towards content hopefully interesting for them.

2.5.5 *Recommender systems*

?, p. 56 clearly describes what a recommender system is:

It is often necessary to make choices without sufficient personal experience of the alternatives. [. . .] Recommender systems assist and augment this natural social process.

What is then the difference between a collaborative filtering system and a recommender system? Recommender systems is a broader concept than collaborative filtering. There are two ways a recommender system can generate recommendations for its users:

1 Using collaborative filtering where the verdicts of other users, which usually in some way are deemed to have similar taste as yourself, are used for recommendation – a social process.

2 Using a *content-based* approach where one is relying on the nature of potential items instead of other people's perception of them. Based on your previous usage of or reaction to items, similar items can then be recommended – an asocial process.

The distinction between a content-based and collaborative filtering approach lies in the way collaborative filtering "are based on human and not machine analysis of content" (**?**, p. 241). To put it another way the similarity of users are measured instead of the similarity of content. Because of this distinction, collaborative filtering systems and content-based systems have been called *user-based* and *item-based* recommendation systems respectively (**?**, p. 156).

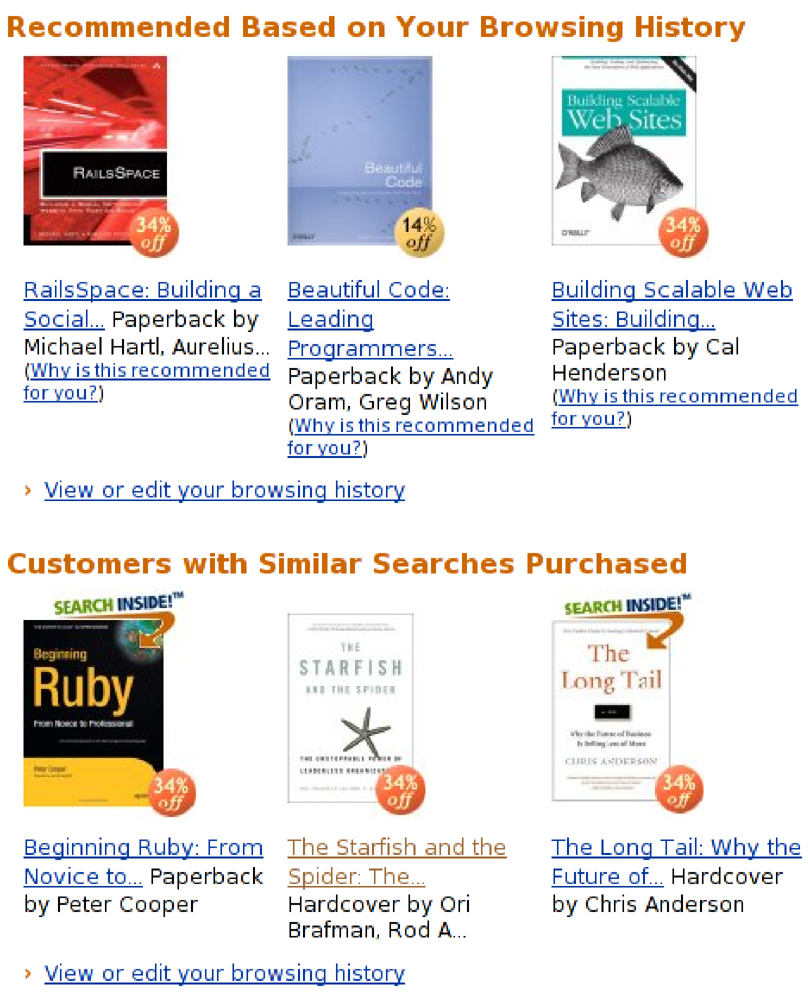
Recommender systems can be based on either one or both of these approaches (**?**, p. 241). **?**, p. 70 argues that a hybrid approach to recommendation is superior to an purely content or collaborative solution since it (i) enables one to use content-based recommendation for items not yet recommended by other individuals, (ii) enables one to use content-based recommendation when there are no other individuals with the same taste, (iii) when collaborative recommendations are present those can be used in favor of more imprecise content-based techniques, and (iv) collaborative recommendations can be generated even though users have not rated the same items by inferring such ratings through content similarity. Since we're concerned with socially constructed navigation possibilities we're not going to discuss content-based systems further in this thesis.

Recommendation from tags

One of the two forms of social navigation found in Knowledge Sea – a digital educational library – is recommendations by item annotation (**?**, p. 13). Users can leave their emphatic marks on content and thereby specify its usefulness. Questionnaires showed that a fair majority of users was agreeable to the use of such recommendations (**?**, p. 15). Analysis of server logs strengthened the impression **?**, p. 38 had by showing that:

Social navigation support and specifically annotation-based social navigation increases the chance of accessing a resource dramatically.

Advanced algorithms from the field of collaborative filtering and recommender systems have been used together with folksonomies consisting of collaborative tags (**?**, pp. 112–113). Preliminary studies have shown positive results when harnessing such social knowledge with filtering algorithms as opposed to traditional folksonomy representation (**?**, p. 114). Using a folksonomy for this purpose can hoverer interfere with an existing recommender system. **?**, p. 190 found that the introduction of tagging and tag display in their established movie recommender system interfered with some of the users' primary objective: finding in-


Recommended Based on Your Browsing History

RailsSpace: Building a Social... Paperback by Michael Hartl, Aurelius... (Why is this recommended for you?)

Beautiful Code: Leading Programmers... Paperback by Andy Oram, Greg Wilson (Why is this recommended for you?)

Building Scalable Web Sites: Building... Paperback by Cal Henderson (Why is this recommended for you?)

› View or edit your browsing history

Customers with Similar Searches Purchased

Beginning Ruby: From Novice to... Paperback by Peter Cooper

The Starfish and the Spider: The... Hardcover by Ori Brafman, Rod A...

The Long Tail: Why the Future of... Hardcover by Chris Anderson

› View or edit your browsing history

Figure 2.5: Item-based and user-based recommendation at Amazon, retrieved May 21, 2008, from http://amazon.com.

teresting movies. Note that this dislike of folksonomies was more likely to be present for users familiar with the old movie recommender system sans a folksonomy. New users, having not witnessed the recommender system without a folksonomy, seemed more acceptant towards tagging.

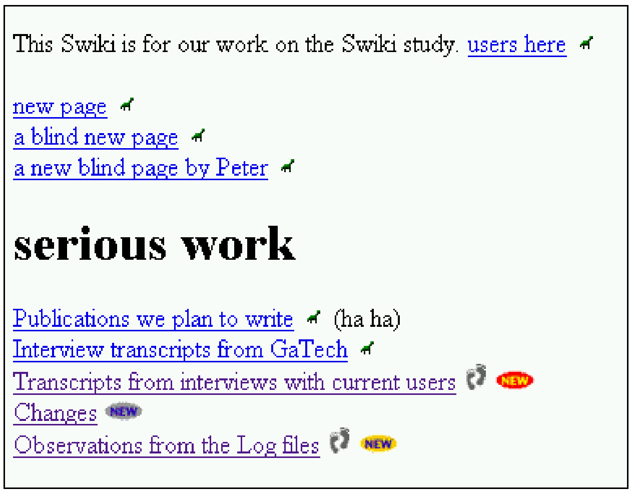
Commercial use

Recommender systems have not only been extensively used in research settings. Many of todays web sites uses some sort of recommender system to deliver a better experience to their users. The canonical example is Amazon's[32] usage of recommendations for book purchases. As seen in Figure 2.5 Amazon deploys both collaborative filtering (similar people's purchases determined by shared search history) and content-based recommendation (based on personal browsing history).

In 2006 Netflix[33] announced a contest where the first person to improve their existing recommender system by 10% would be eligible for a $1 million prize (?, p. 1). By offering such a large reward Netflix highlight how important accuracy in recommender systems can be for

32. An online store primarily carrying books at http://amazon.com.

33. An online movie rental company located at http://netflix.com.

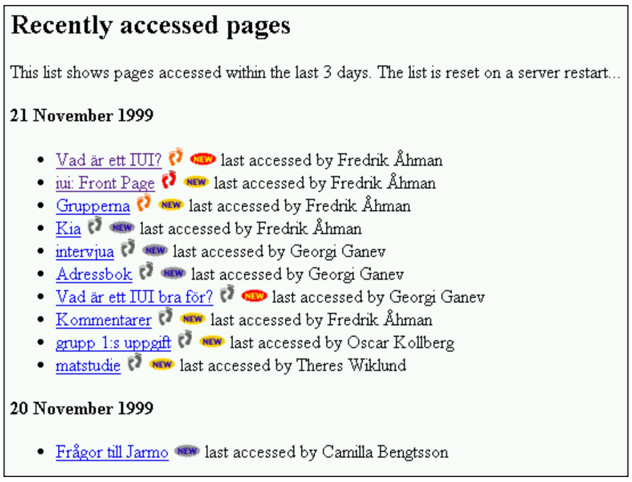

This Swiki is for our work on the Swiki study. users here

new page
a blind new page
a new blind page by Peter

serious work

Publications we plan to write (ha ha)
Interview transcripts from GaTech
Transcripts from interviews with current users NEW
Changes NEW
Observations from the Log files NEW

Figure 2.6: Contextual interaction history cues next to hyperlinks in CoWeb, retrieved January 25, 2008, from http://homepage.mac.com/juggle5/WORK/publications/SwikiWriteup.html.


Recently accessed pages

This list shows pages accessed within the last 3 days. The list is reset on a server restart...

21 November 1999

- Vad är ett IUI? NEW last accessed by Fredrik Åhman
- iui: Front Page NEW last accessed by Fredrik Åhman
- Grupperna NEW last accessed by Fredrik Åhman
- Kia NEW last accessed by Fredrik Åhman
- intervjua NEW last accessed by Georgi Ganev
- Adressbok NEW last accessed by Georgi Ganev
- Vad är ett IUI bra för? NEW last accessed by Georgi Ganev
- Kommentarer NEW last accessed by Fredrik Åhman
- grupp 1:s uppgift NEW last accessed by Oscar Kollberg
- matstudie NEW last accessed by Theres Wiklund

20 November 1999

- Frågor till Jarmo NEW last accessed by Camilla Bengtsson

Figure 2.7: Global list of interaction history cues in CoWeb, retrieved January 25, 2008, from http://homepage.mac.com/juggle5/WORK/publications/SwikiWriteup.html.

a company's earnings. As of this writing no team has yet claimed the prize, but the front runners from AT&T Labs Research currently have a 9.12% improvement (?). The secret to their recommender improvements was to employ a variety of collaborative filtering methods and content-based methods. This approach resulted in a situation where the different methods complemented each other (?, p. 4).

2.5.6 *Social texture*

We use *social texture* to describe socially constructed annotations or visualizations which may be used for navigation or in some form guide users in navigational choices.

Social texture ties in with the forms of social navigation we've recently discussed. Tagging, for instance, is a social texture. Tag clouds are especially good examples of social texture. The interaction history systems we've discussed uses forms of visualizations in close proximity to hyperlinks to convey their degree of usage. This is also a form of social texture.

The first usage of social texture in computer systems (to our knowledge) occurred when ? took advantage of *computational wear* – an analogy for the wear physical objects experience when being used. They modified a text editor to show both wear related to document edits and readings of documents. This wear was graphically visualized through the editor's scroll bar. The concept of edit and read wear has since been used on the Web in for instance the Footprints system (?).

? modified CoWeb – a collaborative Web space modelled after Ward Cunningham's famous Wiki – to include interaction history visualization hoping to make it a more social space, enabling social navigation. They visualized other users' access of different pages both by including a global

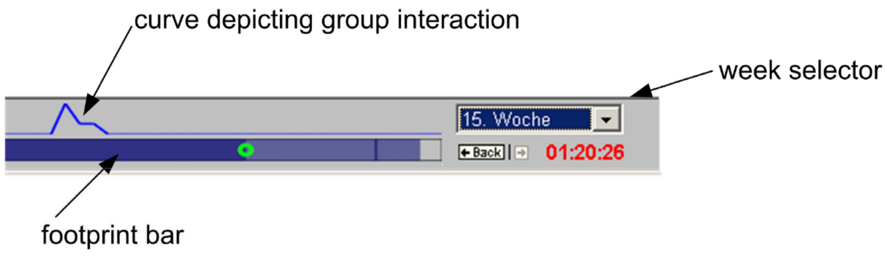


Figure 2.8: virtPresenter timeline (?, p. 43).

list of such behavior and contextual cues about access next to internal hyperlinks. It was inferred by ? that markers of interaction history increased the overall activity on the web page during a user study. They also learned that it's important to provide both global and contextual interaction history cues as seen in Figure 2.6 and Figure 2.7.

? modified a Wiki in even more elaborate ways with the aim of integrating several social navigational mechanisms. They used read-wear information for creating social texture in the Wiki both in-line pages, on a page level, and on a global level. ? took the approach of displaying read-wear in real time, thus making the system look like a populated space. To make such an approach useful the Wiki needs a certain amount of users present at all times. If it's not frequently trafficked it would probably be better to represent historical read-wear as done in CoWeb (?, p. 220).

virtPresenter is a hypermedia based lecture viewer where read-wear have been used to visualize a groups' interaction with continuous content (?). By following the traces other users have left current users can interpret what's the most sought after parts of a lecture. The visualization is implemented in ways similar to ? by showing graphs of usage in-line with a timeline selector. The result can be seen in Figure 2.8.

We discussed KnowledgeSea regarding its use of recommendations in § 2.5.5 (p. 25). The system also incorporates what the authors calls *traffic-based social navigation* (?, p. 12) – in other words history based visualizations in the form of read-wear. The access of different articles in the digital library are recorded and the degree of usage is then visualized in the form of different color tones where darker indicates a more popular resource. Such visualization is used consistently throughout the web site. A questionnaire revealed that in excess of 70% (?, p.15) of the users found such history based visualizations useful and appropriate.

2.6 IS SOCIAL NAVIGATION VALUABLE?

? performed a throughout evaluation of the Kalas[34] system where they sough out to answer two questions related to social navigation:

34. An online system for sharing and finding cooking recipes.

1 Will social navigation enable users to navigate more efficiently?

2 Do social navigation increase the perceived subjective quality of a navigation process?

Server logs were statistically mined and more in depth qualitative interviews were conducted. The results showed that people tended to move to the most populated part of the system and used recommendations for helping select which items to navigate. **?** also found that the subjects overall had a positive impression of the social features of the system. They seemed more interested in expressing themselves through such features than using information from others to help their navigation process.

Favorable results for the effectiveness of social navigation was observed during a simulation experiment conducted by **?**. In most circumstances social navigation had favorable results in efficiency contrasted with asocial navigation. It's important to note that social navigation decreased the effectiveness of navigation in some instances of their simulations (**?**, p. 365). Interestingly, it was discovered that social navigation was more beneficial in environments with high uncertainty[35] than environments with higher certainty – provided that the simulated agents could reach the social media (**?**, p. 368).

35. **?**, p. 363 says that the two sources of such uncertainty is "arising from the correctness of the information gained in any state, and the potential difficulty of reaching that state to obtain the information".

SOCIAL NAVIGATION ON FLICKR & FACEBOOK

3

In this chapter we'll look at the social navigation possibilities provided by two large web sites with social characteristics – Flickr and Facebook. Before we present the results of our investigation we'll describe the methodology we used for dissecting the navigational structures of Flickr and Facebook. We'll conclude this chapter, and the first part of our thesis, with a discussion of our larger findings of how social navigation is used in modern social web sites.

3.1 METHOD

The term *content analysis* is traditionally used to signify a qualitative research method used in the social sciences. **?**, p. 18 defines it as "a research technique for making replicable and valid inferences from texts (or other meaningful matter) to the contexts of their use". Even though such an analysis of the contents, meanings, or effects of communication messages also have been utilized on the Web (**?**) it does not seem very well suited for understanding navigational mechanisms.

We turn to content analysis as the more pragmatic practice conducted within the field of *information architecture*[1] hoping that it will help us get an better understanding of navigational structures. Content analysis is deployed as a technique by information architects for helping them generate a sound and well structured web site architecture. It's seen as a bottom-up process. In its essence a content analysis should identify the various relationships (or lack of correlation) between a web site's content items. It consists of two phases: (i) a collection of a representative sample of data and (ii) an analysis of this collected data (**?**, pp. 241–243).

1. Information architecture can be explained as (i) the structural design of shared information environments, (ii) the combination of organization, labeling, search, and navigation systems within web sites and intranets, (iii) the art and science of shaping information products and experiences to support usability and findability, and (iv) an emerging discipline and community of practice focused on bringing principles of design and architecture to the digital landscape (**?**, p. 4).

Information architects are concerned with the system's content and "need to move below the surface of the system interface to examine the system information itself" (**?**, p. 94). We on the other hand are actually concerned with the system interface and specifically its navigational structures. This creates a striking contradiction as we're not interested in content unless it can help or guide users during their navigation. We therefore had to adapt both our inventory and analysis process accordingly.

3.1.1 *Inventory*

Content inventory is a technique for collecting data from web sites in a structured manner. Its strength as a technique lies in its ability to truly inform us about a web site's content (**?**). The process of actually conducting a content inventory can be equally rewarding as the resulting documents (**?**).

Sampling

Content inventories are often tedious and time consuming to perform. **?**, p. 267 argues that every single bit of content needs to be determined while **?**, p. 241 believes a representative sample is sufficient.

The web sites that are interesting to look at in our research are vast and loaded with enormous amounts of user generated content. An all-inclusive approach to content gathering would simply be impossible in such situations. As a remedy to this we've decided to ignore certain parts of web sites in our content inventories since the scope of our research is limited to navigational constructs, and only those of a social nature.

Our experience is that social navigation and more static navigation are intermixed all over web sites. Often one have to use non-social forms of navigation before social navigational options appear. Thus we could not simply ignore navigational aims which were non-social in our content inventory phase. We did however eliminate the following parts of web sites under investigation:

1 *Administrative sections* where users can change their profiles or set their preferences – a private and asocial endeavor.
2 *Help pages* where FAQS, guides, and instructions are presented in a static manner.
3 *Legal information* including terms of service, privacy policies, and copyright notices.
4 *Content generation* facilities like uploading, categorizing, and editing photos, commenting, posting items, and so on.[2]
5 *Advertisements* from third party providers.

2. While there is no question about the usefulness of such content for providing social navigation possibilities we've found few examples where social navigation is used in the content generation phases itself. There is however a few exceptions, like applying tags (see § 2.5.2 (p. 18) for details).

In addition to eliminating certain form of web pages we synthesized abstract page representations by introducing variables. Take for example a typical social network site. There are from thousands to several millions of profile pages. In context of what navigational options these pages present to us they are all essential similar. So we could introduce a variable called `$user-name`[3] and thereby describe all potential profile pages as: "Profile of `$user-name`".

3. The variable notation with a dollar ($) prefix is inspired from variable usage in UNIX shell scripting (?, p. 88).

We would however have to make sure that the one page we used in our inventory to represent the abstract notion of a profile page was representative. To exemplify, say that a profile page included a stream of the 10 most recent actions your friends had conducted. If the user of our collected profile page had zero friends we would lack the navigational opportunities such a stream could give in our inventory. Therefore we used only pages which provided all possible forms of navigation as basis for abstraction.

Approach

We started out on the first page that was given us when entering the web site under investigation. From there we stepped through each page of the web site by following all navigational hyperlinks provided on individual pages. We did not however frequent a web site in its entirety, but bearing in mind our sampling constraints and abstractions we frequented the site to full coverage. We stopped browsing a particular page if it[4] had previously been inventoried. During the course of this browsing each page was noted down in a table with the following characteristics:

4. Either the exact same page or a page deemed to be the same by our variable driven abstraction method.

1 *Identifier* of numerical and hierarchical form where page with id 4.3.1 is the first child of a page with id 4.3, representing its place in the navigation structure. The first page was given an id of "0", its first descendant an id of "1", the first descendant's sibling an id of "2", the first descendant's first child and id of "1.1", and so on.
2 *Page title* as a description of what the page contains. Some of the web site's we surveyed had a slight ambiguity of title usage. In these cases we decided to collect the most representative sample.[5] If this resulted in unsatisfactory results we created a new title using the best of our abilities to make it as clear and descriptive as possible.
3 *Link name* is either the textual name or a description of the contents (i.e. a graphical representation) of the hyperlink that was utilized to navigate to this very page.
4 *Link location* as a description of the spatial position (for example: global navigation, content area, right sidebar) of the hyperlink used to navigate to this page.
5 *Page URL* as an identifier for the page we visited.

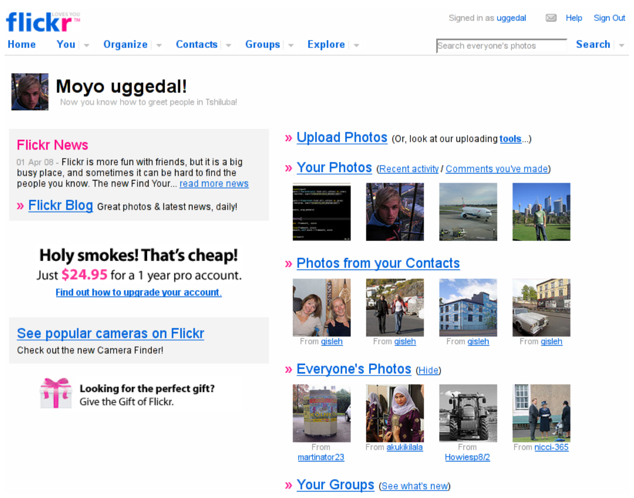
5. A choice between the `<title>` element in the `<head>` of the HTML document and the `<h1>` top level heading in-line the `<body>` portion of the document was made.

The result was a table representing a web site's various pages and the navigational relationships amongst them.

While we took note of the URL of each page we've decided not to display this information. We're not convinced of its usefulness in light of navigation and therefore for brevity's sake omitted them. We did however use them in our inventory process as a way to identify previously collected pages.

In a traditional content inventory other characteristics is usually collected.[6] We were only concerned with the navigational parts of web pages. We opted to only record what we found to be useful for this purpose. This lead to a situation where we were collecting more information about site structure than the attributes of a site's content.

6. For an example of more traditional collection methods see **?**, p. 269.



Moyo uggedal!
Now you know how to greet people in Tshiluba!

Flickr News
01 Apr 08 - Flickr is more fun with friends, but it is a big busy place, and sometimes it can be hard to find the people you know. The new Find Your... read more news

» Flickr Blog Great photos & latest news, daily!

Holy smokes! That's cheap!
Just $24.95 for a 1 year pro account.
Find out how to upgrade your account.

See popular cameras on Flickr
Check out the new Camera Finder!

Looking for the perfect gift?
Give the Gift of Flickr.

» Upload Photos (Or, look at our uploading tools...)

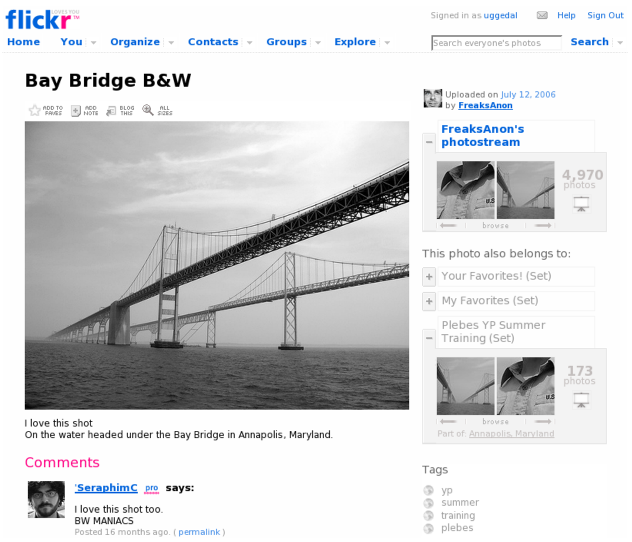
» Your Photos (Recent activity / Comments you've made)

» Photos from your Contacts
From gisleh · From gisleh · From gisleh · From gisleh

» Everyone's Photos (Hide)
From martinator23 · From akukikilala · From Howiesp8/2 · From nicci-365

» Your Groups (See what's new)

Figure 3.1: The welcome page of Flickr showing three different streams of photos taken both by yourself, your contacts, and the entire Flickr populace. Retrieved October 16, 2007, from http://flickr.com



Bay Bridge B&W

Uploaded on July 12, 2006 by FreaksAnon

FreaksAnon's photostream

I love this shot
On the water headed under the Bay Bridge in Annapolis, Maryland.

This photo also belongs to:
Your Favorites! (Set)
My Favorites (Set)
Plebes YP Summer Training (Set)
Part of: Annapolis, Maryland

Comments

'SeraphimC pro says:
I love this shot too.
BW MANIACS
Posted 16 months ago. (permalink)

Tags
yp
summer
training
plebes

Figure 3.2: A photo detail page on Flickr showing both comments and tags, retrieved October 26, 2007, from http://flickr.com/photos/benbengraves/187609810

3.1.2 *Analysis*

An analysis of the collected content follows after an inventory phase is completed. Typically information architects use content analysis for making decisions on what and how to improve an existing web site's content architecture. With such an aim they look for patterns and relationships when analyzing their content inventory. These patterns and relationships will then suggest groupings and connections amongst separate content items (?, p. 243).

Since our focus were dissimilar compared to that of most information architects' we've had to tailor the analysis process to best help us discover and understand patterns of social navigation in web sites. Analyzing content inventories for such means is as far as we know not conducted before. We were therefore exploring unknown waters and had to adapt our method as we went about with our analysis.

We started with our impressions from the content inventory[7] and based our discussion on the findings we regard most conspicuous in relation to social navigation. During the resulting discussion we referenced the relevant pages recorded in our content inventory by their identifiers.

7. As stated earlier the process of conducting a content inventory is not only beneficial just because of the resulting documentation one creates of a web site. People conducting content inventories tends to get deeply informed about a web sites content and structure after having exhaustively recorded large parts of it.

3.1.3 *Subjects*

Based on time considerations we decided to select two web sites for which we would carry out a content analysis for illuminating social navigation usage. We selected two web sites which we were somewhat familiar with and believed was using social navigation in novel ways. Based on these criteria we decided to study Flickr[8] and Facebook[9].

8. Available at http://flickr.com.

9. Available at http://facebook.com.

3.2 RESULTS

This section includes a survey and analysis of the most interesting data we've collected in our inventory phase of a content analysis of two well known web sites. The inventory results can be found in its whole in Appendix A (p. 105). When we're referencing to IDs we're using the identifier in the tables found at the specified pages.

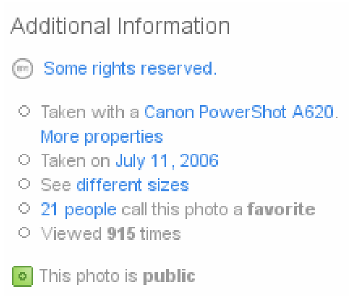
3.2.1 *Social navigation on Flickr*

Flickr is a photo sharing site which are known to be on the cutting edge when it comes to enabling new and innovating features in its domain. Flickr has a quite peculiar history as it started out as a massively multi player online game. An environment for photo sharing within the game was added in 2004 which quickly became more popular than the game itself. The focus of the company was shifted and their new photo sharing community was bought by Yahoo! Inc. in March 2005 (**?**, p. 257).

This subsequent analysis of Flickr was carried out as a registered user. One has to be registered for interacting with the site in such a way that one leaves persistent traces. The site has a open nature enabling anonymous access to the majority of content.

Thumbnails

Already on the welcome page (Figure 3.1) we're finding navigation links that are social of nature. Four thumbnails functions as sample of the most recently uploaded photos by other members of the community and four thumbnails samples the latest photos by your friends. One can either navigate straight to a detailed page for each particular photo by clicking on the respective thumbnail (ID 6, p. 110) or the profile of the uploader by clicking on their user name (ID 7, p. 110). Such thumbnails with minimal meta data (the uploader) are prevalent all over Flickr. Of the 120 pages we collected in our content inventory 26 of them ▮▁ contained thumbnails. Most of these thumbnails are giving users incentives to navigate using social means.[10] Which photos these thumbnails portray is dynamic. That is to say that other users' actions – uploading a photo, tagging a photo, taking a photo with a specific camera, collecting photos into sets, and adding photos to a certain group – all determine the navigational choices you as a user is presented with. A good example of passive and indirect social navigation with both implicit and explicit transfer of advice.


Additional Information

Some rights reserved.

- Taken with a Canon PowerShot A620. More properties
- Taken on July 11, 2006
- See different sizes
- 21 people call this photo a **favorite**
- Viewed **915** times

This photo is **public**

Figure 3.3: Photo meta-data on Flickr, retrieved October 28, 2007, from http://flickr.com/photos/benbengraves/187609810/

10. Apart from the few pages that only show a stream of your own thumbnails when you're browsing your own photos by various methods.

Meta-data

We arrive on a photo detail page (ID 1.1, p. 106) as in Figure 3.2 if we utilize one of these thumbnails for navigation. In addition to comments on the photo we find meta-data as in Figure 3.3. Meta-data include the date the photo was taken, the manufacturer and the model of the camera

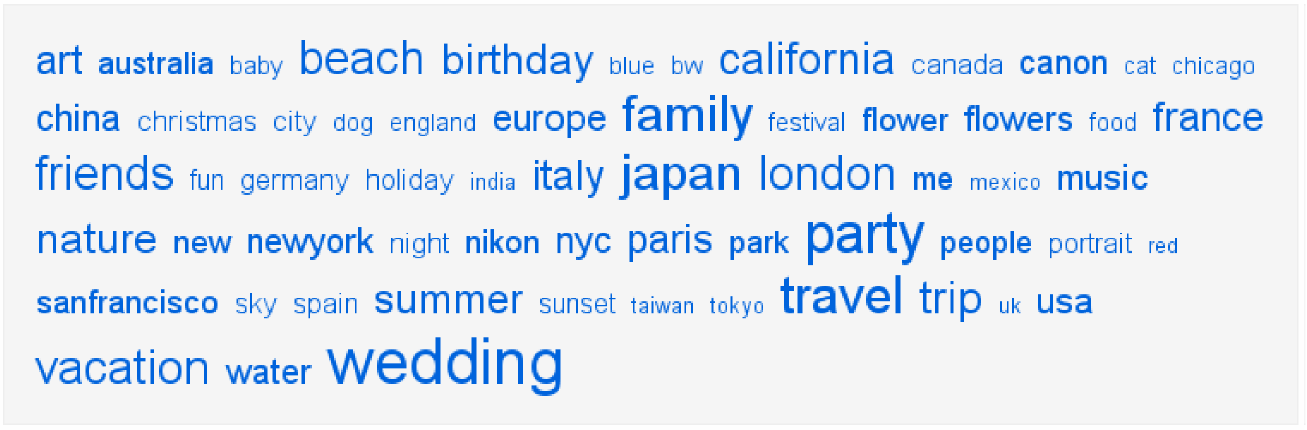

art australia baby beach birthday blue bw california canada canon cat chicago china christmas city dog england europe family festival flower flowers food france friends fun germany holiday india italy japan london me mexico music nature new newyork night nikon nyc paris park party people portrait red sanfrancisco sky spain summer sunset taiwan tokyo travel trip uk usa vacation water wedding

Figure 3.4: Tag cloud on Flickr, retrieved November 1, 2007, from http://flickr.com/explore

that was used which are all so called *Exif*[11] data. Flickr utilize this data by enabling navigation based both on the dates a picture was taken and by camera make and model – social navigation with implicit advice. Say you're trying to find a picture from your home town on a particularly beautiful summer day. By using date-of-picture-taking based navigation coupled with tags or geographical data (which both will be discussed shortly) you're probably increasing you chances of finding what you want. Camera make information could also be useful when looking at the quality of pictures taken with certain cameras before purchasing one yourself.

11. Short for Exchangeable Image File – a specification for an image file format used in digital cameras.

Folksonomy

Of most importance for Flickr, and indeed what makes Flickr a folksonomy, is its tagging abilities. Caterina Fake, co-founder of Flickr, explains its importance as "Tagging really revolutionized the way the product behaved" (**?**, p. 261). All registered user can label anyone's photos by applying such short descriptive tags. This collaborative process lay the ground work for other user's ability to easily browse photos by topic – a form of indirect and passive social navigation where advice are explicitly given. Figure 3.4 exemplifies how the user generated data through tagging (ID 5, p. 108) can be used as a navigational aid. A tag cloud is used to visualize the popularity (and thereby importance) of the individual tags. The larger the tag title, the more frequent the tag has been applied to photographs.

Tag clustering was released in the fall of 2005 (**?**) as a way to easier see the relationships between separate tags. For any given tag a cluster of three related tags is generated and displayed (ID 5.6.1.2, p. 109) to users when they are browsing as seen in Figure 3.5. Flickr algorithmically generates these listings based on what tags users tend to use together for labeling a photo.

Tagging is a very flexible approach only hindered by users' imagina-

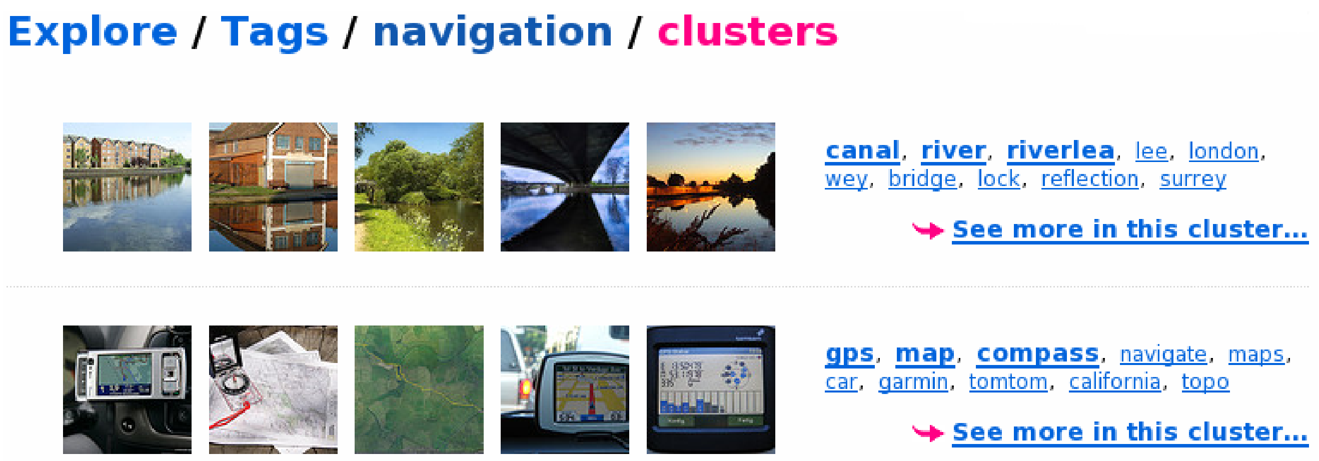

Explore / Tags / navigation / clusters
canal, river, riverlea, lee, london, wey, bridge, lock, reflection, surrey
See more in this cluster...
gps, map, compass, navigate, maps, car, garmin, tomtom, california, topo
See more in this cluster...

Figure 3.5: Tag cluster on Flickr for "navigation", retrieved November 19, 2007, from http://flickr.com/photos/tags/navigation/clusters/

tion. In the early days of Flickr there was no support for geographical data. Users soon found a remedy for this by tagging photos with the longitude and latitude of the place where they were taken. By using the same technology we're using in our prototype application (see § B.1.1 (p. 122) for details) they were able to integrate Google Maps[12] in Flickr, enabling user's to place their photos on a map and automatically generate geographical coordinate tags.[13]

12. Google Maps can be found at http://maps.google.com.

13. More info about *geotagging* in the early days of Flickr can be found in the remains of the Flickr Geotagging group, available at http://flickr.com/groups/geotagging/.

Geographical data

In late August 2006 Flickr introduced geotagging abilities (?) by integrating mapping aspects from Yahoo! Maps.[14] Users could now place their photos on a map to signify where they were captured without resorting to clever hacks of the standard tagging system.

14. Yahoo! Maps, a service similar to Google Maps, have its home at http://maps.yahoo.com.

Figure 3.6 (p. 38) shows how one of the authors photos are placed on a map (ID 1.1.6, p. 106). One can then cycle through the adjacent photos of other users that are interesting or recently published. What we see here is indirect social navigation where the advice is explicitly given. When a user places his photo geographically on a map he makes available navigational choices for other users who for some reason are interested in that particular geographical area. Such visualizations of geographic navigation clues are a good example of social texture.

Interestingness

The concept of *interestingness* was introduced by Flickr during the same time tag clustering was unveiled (?).

Interestingness is a rating of how interesting a photograph is deemed to be. Interestingness is based on how many views the photograph has, how many users who have favored the photograph, and how many comments the photograph has. Favoring have the highest weight, comments

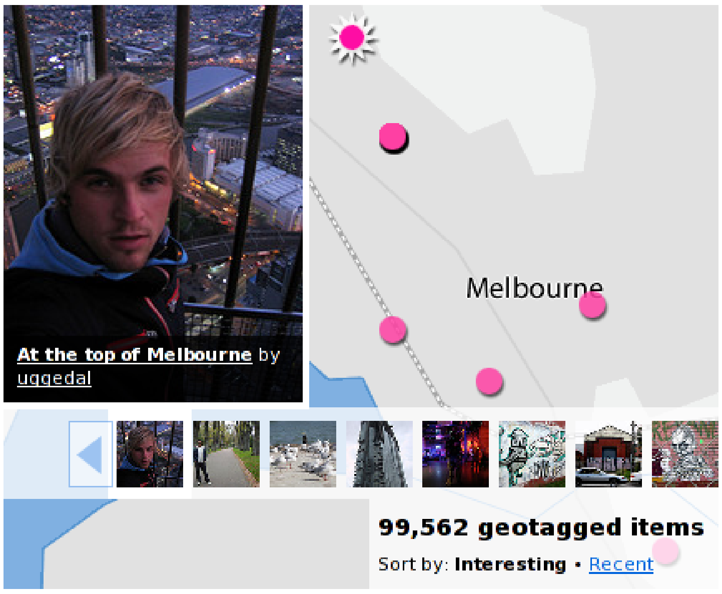

At the top of Melbourne by uggedal
Melbourne
99,562 geotagged items
Sort by: Interesting • Recent

Figure 3.6: Geotagging on Flickr, retrieved March 3, 2008, from http://flickr.com/photos/uggedal/261824261/map/?view=everyones

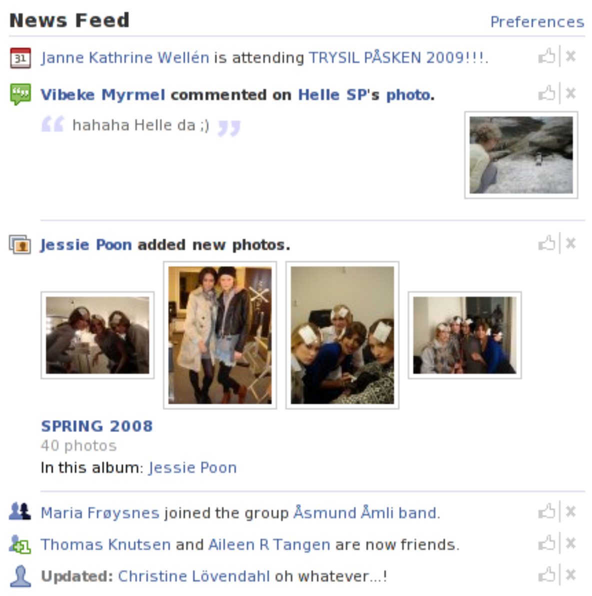
medium weight, and views the least weight in the algorithm that generates the interestingness rating (?). Users are not aware of the score of a particular photograph's interestingness, but the highest rated photographs are available through the "Explore" part of Flickr (ID 5.3, p. 108; ID 5.4, p. 108).

The interestingness system is a great example of passive and indirect social navigation. Based on the behavior of other users the more interesting photos are made more visible in Flickr's interface. Flickr can be seen as a collaborative filtering system which uses implicit and explicit ratings. When a user leaves a comment on a photo the aim is likely to voice his reactions to the picture, and not vote the picture down or up. Page views are also implicit of nature while favoring are explicitly given by users.

Interestingly Flickr uses collaborative filtering without personalization of it's recommendations and rather gives the whole population similar recommendations. The use of interestingness seem to have worked well for Flickr as users are trying hard to make their photos receive higher interestingness scores. As of this writing a patent on interestingness is pending (?).

3.2.2 *Social navigation on Facebook*

Facebook is a social network site which started as a service only available for Harvard students in February of 2004. Within the same month Facebook was opened up for students at several other universities in the US. More universities and colleges were supported before Facebook opened up it doors for high-school students in September of 2005 (?). Anyone – student or not – was allowed access in September of 2006 (?).


News Feed
Preferences

Janne Kathrine Wellén is attending TRYSIL PÅSKEN 2009!!!.

Vibeke Myrmel commented on Helle SP's photo.

“hahaha Helle da ;)”

Jessie Poon added new photos.

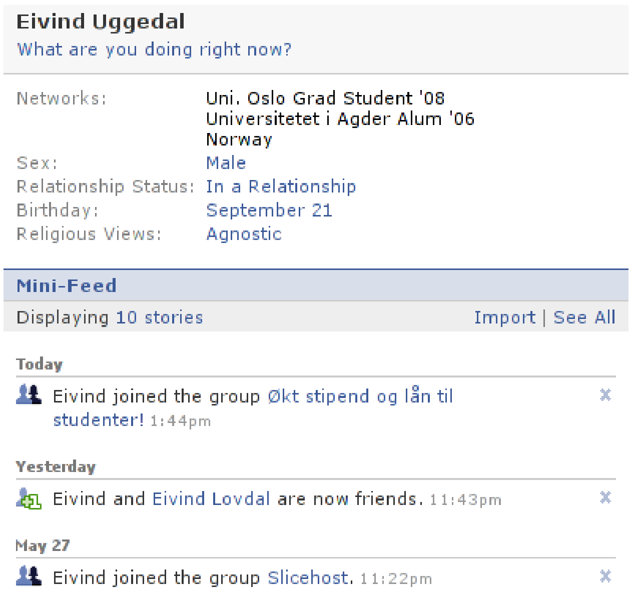
SPRING 2008
40 photos
In this album: Jessie Poon

Maria Frøysnes joined the group Åsmund Åmli band.

Thomas Knutsen and Aileen R Tangen are now friends.

Updated: Christine Lövendahl oh whatever...!

Figure 3.7: Author's news feed on Facebook showing events, comments, photos, groups, persons, and status messages. Retrieved March 26, 2008, from http://www.facebook.com/home.php


Eivind Uggedal
What are you doing right now?

Networks:	Uni. Oslo Grad Student '08 Universitetet i Agder Alum '06 Norway
Sex:	Male
Relationship Status:	In a Relationship
Birthday:	September 21
Religious Views:	Agnostic

Mini-Feed
Displaying 10 stories Import | See All

Today

Eivind joined the group Økt stipend og lån til studenter! 1:44pm

Yesterday

Eivind and Eivind Lovdal are now friends. 11:43pm

May 27

Eivind joined the group Slicehost. 11:22pm

Figure 3.8: Author's profile page on Facebook, retrieved May 30, 2008, from http://www.facebook.com/profile.php?id=903795175

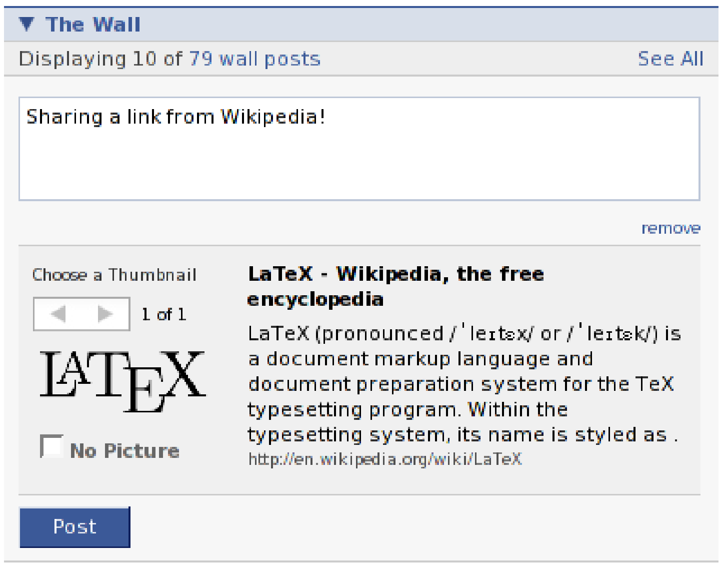
Facebook has seen an enormous growth and are today the largest social network in some countries (see § 2.3.1 (p. 11) for details).

The following analysis was carried out as an authenticated user of Facebook. Most of Facebook's content is only available for registered users. Each user's privacy settings also regulates how openly available content is.

News feed

The landing page when you log in to Facebook after having configured your account is the "News Feed" (ID 0, p. 112). The news feed shows the recent activities your friends have conducted. This eliminates the tedious process of checking every profile page of your friends to keep on top of what they're up to. The feed shows a list of items, each representing a type of activity. As shown in Figure 3.7 (p. 39) these activities are distinguished by an icon and presented chronologically. The data presented in the news feed are an aggregation of every friend's "Mini-Feed" located on their profile page (ID 1, p. 112) as seen in Figure 3.8 (p. 39).

The news feed enables social navigation – all navigational choices the feed provides through representations of activity are constructed as a by-product of other people's actions. When you select to attend an event, join a group, post a photo, change your relationship settings, befriend a person, post a comment on a wall or photo, or update your profile information your action is added to the feed. The navigation presented by the feed is therefore indirect and passive social navigation provided by implicit feedback – providing no additional overhead for the advice


▼ The Wall

Displaying 10 of 79 wall posts See All

Sharing a link from Wikipedia!

remove

Choose a Thumbnail

1 of 1

No Picture

LaTeX - Wikipedia, the free encyclopedia

LaTeX (pronounced /ˈleɪtɛx/ or /ˈleɪtɛk/) is a document markup language and document preparation system for the TeX typesetting program. Within the typesetting system, its name is styled as .

http://en.wikipedia.org/wiki/LaTeX

Post

Figure 3.9: Sharing of hyperlinks on Facebook, retrieved June 2, 2008, from http://www.facebook.com/profile.php?id=903795175

provider.

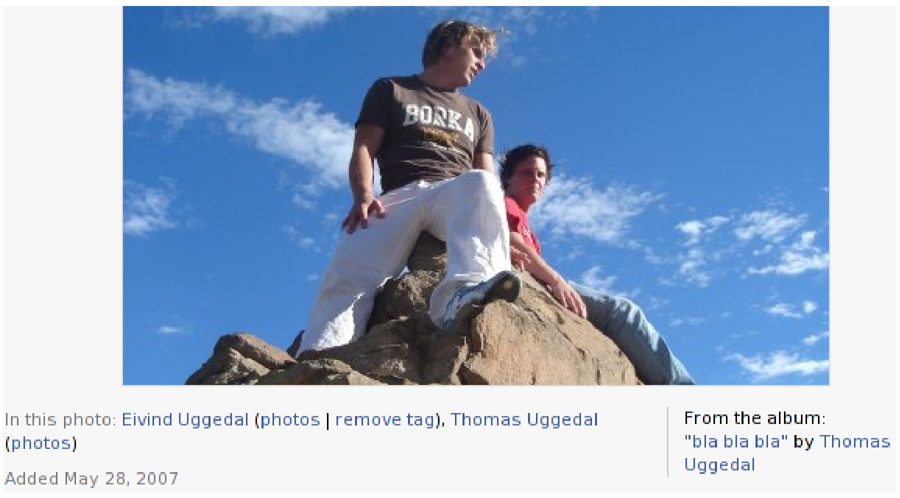
Hyperlink sharing

Facebook have an interesting feature that makes hyperlink sharing easier for its users. When one are posting a comment on a the wall of a person (ID 1.19.1, p. 118), group (ID 1.1.3.1.1.9, p. 113), or event (ID 1.1.3.2.1.9, p. 114) one can provide a URL. By providing such a hyperlink to a third party web page Facebook extracts an excerpt of the linked page with optional graphics as seen in Figure 3.9. Since the recipient of a message with a hyperlink can respond to the advice provider we're witnessing active and direct social navigation. Such improved URL handling was envisioned by **?**, p. 811.

Photo tagging

Photos on Facebook can be tagged (ID 1.19.1, p. 118) like those on Flickr. But as you can see in Figure 3.10 one are tagging with people as identifiers on Facebook instead of keywords as on Flickr. We are therefore not witnessing a folksonomy on Facebook. Photo tagging on Facebook can be used as a means of navigating photos of particular individuals or navigating towards people based on photos with various taggings of individuals.

Photo tagging can therefore be seen as passive and indirect social navigation where advice is given explicitly. Navigational advice is made available for future users when they stumble upon a photo with person tags, person tags in news feeds, or navigates photos by a certain tagged person.


In this photo: Eivind Uggedal (photos | remove tag), Thomas Uggedal (photos)

Added May 28, 2007

From the album: "bla bla bla" by Thomas Uggedal

Figure 3.10: Photo tagging on Facebook, retrieved June 2, 2008, from http://www.facebook.com/photo.php?pid=177290&id=579356186

3.3 DISCUSSION

Through our study of Flickr and Facebook we've gotten an impression of how social navigation is used in modern web sites. We'll discuss our most interesting findings in the next sections.

3.3.1 *No explicit design for social navigation*

It does not seem like the designers of social web pages design for social navigation explicitly. There seem to be a trend of designing for social interaction. As we've seen in § 2.3.1 (p. 11) social network sites have become very popular amongst web citizens.

An implicit by-product of such a design approach seems to be the creation of several forms of social navigation constructs. We base this observation on our studies of two large social web sites in addition to cursory observations from other social web sites.

3.3.2 *Social navigation have become mainstream*

We were able to locate navigational mechanisms which could be considered social navigation in all the web sites we studied. While our view of the social web surely is incomplete, we take this as a sign for an increased use of social navigation. Social navigation seems to have become mainstream.

The reasons for this increased usage can be many. We think the most dominant factor is the high focus on creating web sites where social interaction is supported. As described earlier, social navigation is then often implicitly created when one designs web sites with such a focus.

3.3.3 *Social navigation advice is given by peers*

An overlying theme of the forms of social navigation we found in the wild were that it was created by equal peers. The constructs for enabling social navigation are created by the web site creators. But the data that enables navigational choices of a social nature are created by the community.

We therefore purpose that navigational advice have to be given by peers to be considered social navigation, whether indirect or direct, explicit or implicit. In other words navigational advice have to be given by individuals on the same vertical level as yourself. This means that navigational advice given by web editors and web designers – people vertically superior to yourself – can not be considered true social navigation.

Based on this idea we propose our own definition of social navigation on the Social Web. We base our definition on the nomenclature by **?**, p. 20 which we've used so far in this thesis:

- *An navigator* is an individual who is looking for navigational opportunities on the Web.
- *An advice provider* is one or many peers positioned on the same vertical level as the navigator within the web site where advice are given.

With the actors of the navigation process defined we can define social navigation on the Social Web as:

Social navigation is navigation in the Web by a navigator, using information given explicit or implicit by an advice provider.

3.4 GENERALIZABILITY AND VALIDITY

Since Flickr as of this writing is the 39th most popular site on the web and Facebook is placed 8th,[15] one could argue that their usage of social navigation is representative for what a sizeable portion of web citizens have used. On the other hand, two web sites is like a drop of water in the ocean compared to the Web's over 170 million web sites (**?**). The claims that we've made regarding the use of social navigation in modern social web sites are based on these two web sites alone. One should therefore take these claims as indication of how social navigation can be used on the Social Web, and not as generalizations of how every social web site functions.

15. According to Alexa's traffic rankings as of July 18, 2008. Retrieved from http://www.alexa.com/site/ds/top_sites?ts_mode=global.

PART II

UNOBTRUSIVE PROTOTYPING OF SOCIAL NAVIGATION

4 IMPLEMENTATION OF AN UNOBTRUSIVE PROTOTYPE

During our research on social navigation we came in contact with SINTEF[1] and their RECORD research project. RECORD is a research project which "aims to provide knowledge and methodologies to improve development of online community products and services" (?). One of the partners of this project is NRK[2] with their Urørt web site. After some coordination meetings with the RECORD project it became clear that Urørt and its web site would be an excellent candidate for trying out social navigation techniques.

As we'll see in the next section, it's possible to build applications on top of existing web sites by creating unobtrusive implementations. Following the background information on building applications on top of established web sites we give an account of what kind of navigation system we wanted to build for Urørt, go on to describe why we decided on such navigational designs, and conclude with an explanation of the deeper technical decisions we had to make. Appendix B (p. 121), describes what kind of third party software we used for realizing the implementation details we describe in this chapter.

1. SINTEF – headquartered in Norway – is the largest independent research organization in Scandinavia.

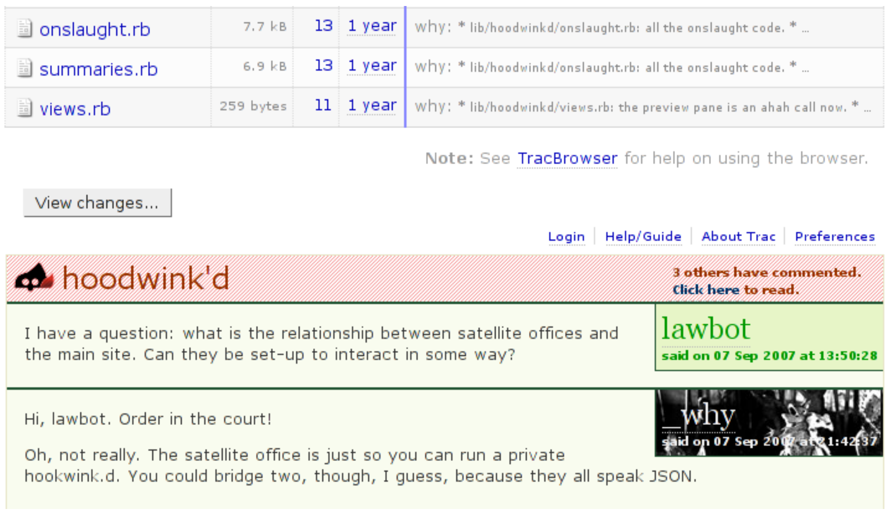
2. The Norwegian Broadcasting Corporation.

4.1 BUILDING ON TOP OF THE WEB

Going in and making changes to an existing web site can be both an daunting and time consuming task. First one have to establish a trustworthy relationship with the creators of such a site so they can be certain you're not introducing bugs in their production software. Secondly, grasping the code base, third party libraries, and development tools of such a software project demands a lot of upfront effort before any real development work can begin. This goes against the prototypical process we intended to use while experimenting with Urørt.

Even though we've had an ongoing dialog with the developers of Urørt, we decided to create our prototype as a layer on top of their site. By using an extension for a leading open source[3] web browser we were able to create a script which made changes to the way Urørt were presented to users who were participating in our study. Such an approach would hopefully result in a transparent experience for our end users, as long as they had taken the necessary steps to set up our script.

3. The Firefox web browser. Available at http://firefox.com.



onslaught.rb	7.7 kB	13	1 year	why: * lib/hoodwinkd/onslaught.rb: all the onslaught code. * ...
summaries.rb	6.9 kB	13	1 year	why: * lib/hoodwinkd/onslaught.rb: all the onslaught code. * ...
views.rb	259 bytes	11	1 year	why: * lib/hoodwinkd/views.rb: the preview pane is an ahah call now. * ...

Note: See TracBrowser for help on using the browser.

I have a question: what is the relationship between satellite offices and the main site. Can they be set-up to interact in some way?

lawbot
said on 07 Sep 2007 at 13:50:28

Hi, lawbot. Order in the court!

Oh, not really. The satellite office is just so you can run a private hookwink.d. You could bridge two, though, I guess, because they all speak JSON.

_why
said on 07 Sep 2007 at 21:42:37

Figure 4.1: Comments on the repository browser of Hoodwink.d's source code, retrieved March 7, 2008, from http://code.whytheluckystiff.net/hoodwinkd

4.1.1 *Inspiration*

The idea of creating additional features for a web site in this manner came from the author's involvement in an underground community based around the Ruby programming language. Hoodwink.d[4] is a service that lets members post comments on all kinds of web sites. These comments are only visible to the members of the community.

4. Hoodwink.d's starting point for new users can be seen at http://hoodwinkd.hobix.com.

The really interesting part of Hoodwink.d is the features it enables on top of the Web. One can create a comment visible only to the community's members on any web page that is supported. This support is not dependant on the creator of the web site. The users of Hoodwink.d need to record some information of the web site (where the comments should be placed) to make it supported. Figure 4.1 shows an example of how these comments are displayed on a web page. This page does not natively support comments. Using Hoodink.d for such means is a very cheap (time wise) option compared to implementing commenting in the web site itself. They are perceived as being part of the web page, even though they are inserted right after the page is fully loaded.

4.2 DESIGN

With design, we mean how the application we're making is presented to users. For discussion of how the software is designed or architected internally, take a look at § 4.4 (p. 54).

4.2.1 *Philosophy*

Our philosophy when creating a navigational design have been similar to that practiced by **?**, chapter 3 in aviation design:

And now, having spoken of the men born of the pilot's craft, I shall say something about the tool with which they work, the airplane. Have you ever looked at a modern airplane? Have you followed from year to year the evolution of its lines? Have you ever thought, not only about the airplane, but about whatever man builds, that all of man's industrial efforts, all his computations and calculations, all the nights spent over working draughts and blueprints, invariably culminate in the production of a thing whose sole and guiding principle is the ultimate principle of simplicity?

It is as if there were a natural law which ordained that to achieve this end, to refine the curve of a piece of furniture, or a ship's keel, or the fuselage of an airplane, until gradually it partakes of the elementary purity of the curve of a human breast or shoulder, there must be the experimentation of several generations of craftsmen. In anything at all, perfection is finally attained not when there is no longer anything to add, but when there is no longer anything to take away, when a body has been stripped down to its nakedness.

This philosophy is closely related to minimalism.[5] We feel the prototype we're creating should provide users with only the necessary information for affording navigational behavior.

5. Minimalism is a movement where the infamous architect Mies van der Rohe popularized the concept of "less is more" – achieving the maximum effect with the minimum of means (**?**).

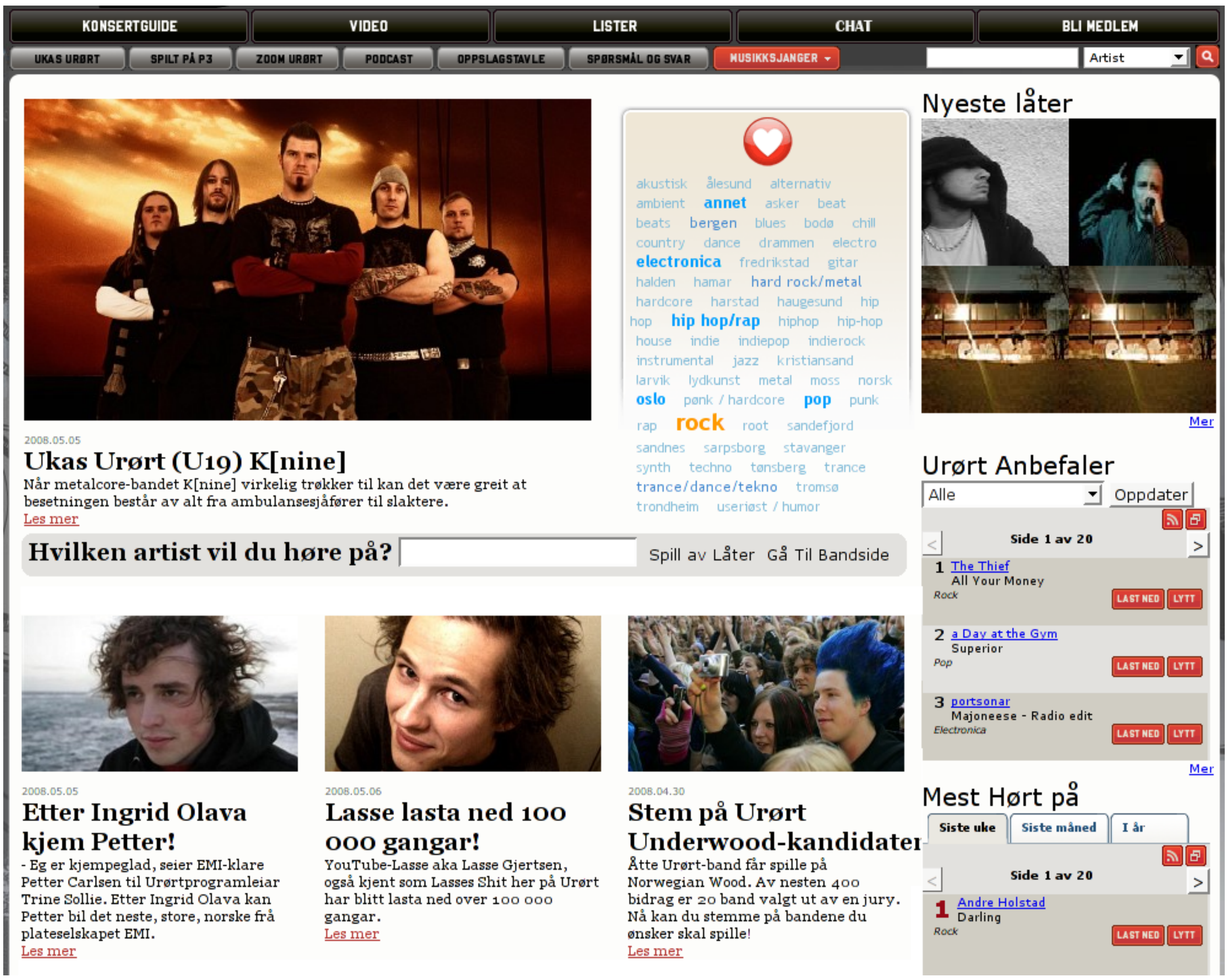
? wrote a book called *The Paradox of Choice: Why More Is Less* where he explains that we in our modern society have an overabundance of choice. All these choices can produce psychological distress. As **?** explains this problem can be resolved by limiting the amount of choices we are presented with.

Our design could potentially introduce even more choices. We're introducing more information, and more navigational choices, into the Urørt web site. A possible solution could be to remove some of the choices presented to the user if we felt the choices we're providing for navigation were of higher importance than those provided by default. We decided against this since we wanted to test our navigational design in opposition to the navigational designs already provided. By eliminating some of the default navigation we would not be sure if potentially more satisfied users was the result of our removal of choices, the new choices we provided, or a combination.

4.2.2 Activity stream for Urørt

According to **?**, p. 193 "There is enormous and growing interest in the consumption of up-to-the-moment streams of newly published content of various forms: news articles, posts on blogs or bulletin boards, and multimedia data such as images, songs, or movie clips".

In the Urørt web page there is currently not an easy way to discover what is new for the areas that you personally care about. The main page as seen in Figure 4.2 displays the latest editorial articles. In some sense these activities represent new content. At the right margin there are also lists of both editorial song selections and the most popular by listenings


KONSERTGUIDE
VIDEO
LISTER
CHAT
BLI MEDLEM
UKAS URØRT
SPILT PÅ P3
ZOOM URØRT
PODCAST
OPPSLAGSTAVLE
SPØRSMÅL OG SVAR
MUSIKKSJANGER
Artist
akustisk ålesund alternativ ambient annet asker beat beats bergen blues bodø chill country dance drammen electro electronica fredrikstad gitar halden hamar hard rock/metal hardcore harstad haugesund hip hop hip hop/rap hiphop hip-hop house indie indiepop indierock instrumental jazz kristiansand larvik lydkunst metal moss norsk oslo pønk / hardcore pop punk rap rock root sandefjord sandnes sarpsborg stavanger synth techno tønsberg trance trance/dance/tekno tromsø trondheim useriøst / humor
Nyeste låter
Mer
Urørt Anbefaler
Alle
Oppdater
Side 1 av 20
Mest Hørt på
Siste uke
Siste måned
I år
Side 1 av 20
LAST NED
LYTT
Spill av Låter
Gå Til Bandside
Les mer

2008.05.05
Ukas Urørt (U19) K[nine]
Når metalcore-bandet K[nine] virkelig trøkker til kan det være greit at besetningen består av alt fra ambulansesjåfører til slaktere.

Hvilken artist vil du høre på?

1 The Thief — All Your Money — Rock
2 a Day at the Gym — Superior — Pop
3 portsonar — Majoneese - Radio edit — Electronica

1 Andre Holstad — Darling — Rock

2008.05.05
Etter Ingrid Olava kjem Petter!
- Eg er kjempeglad, seier EMI-klare Petter Carlsen til Urørtprogramleiar Trine Sollie. Etter Ingrid Olava kan Petter bil det neste, store, norske frå plateselskapet EMI.

2008.05.06
Lasse lasta ned 100 000 gangar!
YouTube-Lasse aka Lasse Gjertsen, også kjent som Lasses Shit her på Urørt har blitt lasta ned over 100 000 gangar.

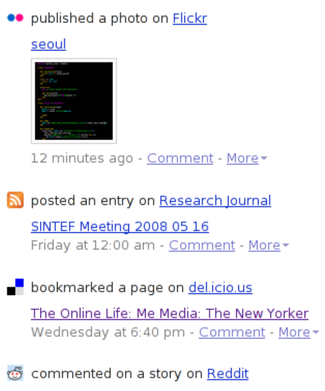
2008.04.30
Stem på Urørt Underwood-kandidater
Åtte Urørt-band får spille på Norwegian Wood. Av nesten 400 bidrag er 20 band valgt ut av en jury. Nå kan du stemme på bandene du ønsker skal spille!

Figure 4.2: The main page of Urørt with mostly editorial content. Retrieved May 8, 2008, from http://www11.nrk.no/urort/default.aspx.

and downloads. Whether you're logged in to the web page by a registered user handle or browsing the page anonymously, you're presented with the same information. Navigating onto your personal profile page yields the same results: no good pointers to fresh content which you are likely to especially care about.

When we conducted our study of the Facebook social network site we became intrigued by how easy it was to keep updated on the latest developments for our friends by using its news feed, as described in § 3.2.2 (p. 39).

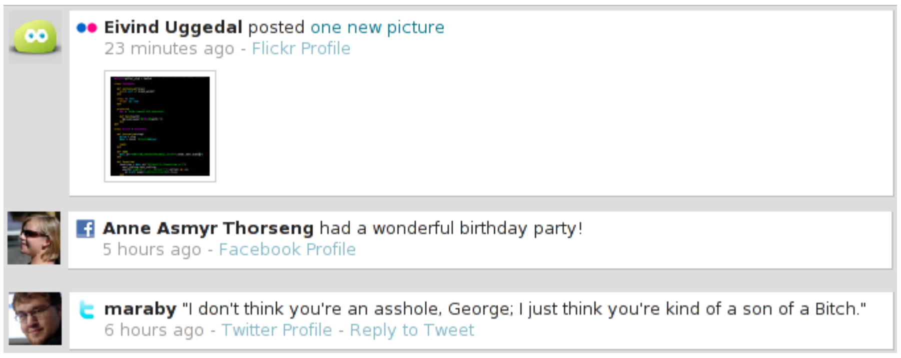
Such streams of activity seem to have become popular also outside Facebook. Socialthing! and FriendFeed are two services that aggregates such streams from several sources and provides an unified stream. Their distinguishing factor is that Socialthing! fetches activities for your existing friends on services as Facebook and Flickr. FriendFeed on the other hand only collect activities from the people you decide to follow


published a photo on Flickr
seoul
12 minutes ago - Comment - More

posted an entry on Research Journal
SINTEF Meeting 2008 05 16
Friday at 12:00 am - Comment - More

bookmarked a page on del.icio.us
The Online Life: Me Media: The New Yorker
Wednesday at 6:40 pm - Comment - More

commented on a story on Reddit


Eivind Uggedal posted one new picture
23 minutes ago - Flickr Profile

Anne Asmyr Thorseng had a wonderful birthday party!
5 hours ago - Facebook Profile

maraby "I don't think you're an asshole, George; I just think you're kind of a son of a Bitch."
6 hours ago - Twitter Profile - Reply to Tweet

Figure 4.4: Activity stream for the author and his friends on Socialthing!, retrieved May 18, 2008, from https://socialthing.com

on FriendFeed itself. FriendFeed can in addition display your personal stream as an aggregation of all supported services you're using. Both activity stream aggregators are shown in Figure 4.3 and Figure 4.4.

When the news feed was introduced on Facebook in September 2006 users immediately responded negatively since they felt their privacy was compromised. Even though all data available through the news feed had previously been available to the same users, the aggregated and efficient display of this information led to concerns (?, pp. 13–14). Interestingly these negative first reactions seemed to vanish as "Users quickly adjusted to the new architecture; they began taking actions solely so that they could be broadcast across to Friends' News Feeds" (?, p. 19). ?, p. 1031 found that the news feed was indeed used by 241 respondents to a usage survey conducted by the author. He feels these results indicate a greater acceptance of the news feed today compared to when it was introduced. An ethnographic study conducted by ?, p. 3126 of several social network sites including Facebook found the news feed to be one of the primary means of interacting with the web site.

Based on both our own observations and other's of the perceived usefulness of activity streams, we decided on using such a navigational design in our prototype implementation for Urørt.

Relevant activities

The approach seen at Facebook for letting users know what's fresh and relevant through an activity stream (or news feed as Facebook calls it) seemed a good fit for Urørt. But Urørt does not have the concept of friends as seen in social network sites. Urørt exists for people to freely share their music and let others find music they like – and not making new friends or keeping in touch with old friends. How do we then find recent activities which potentially could be interesting for our users when we don't have any friends to consult?

We tried to answer this question by using a feature on Urørt that allows any user to *favor* an artist. If a user conducts such an action he or she becomes a *fan* of that artist. We don't know why people signify artists as their favorites on Urørt, but the main reason may be that they simply like the music the artist is publishing. Another reason could be that people know the artist personally and therefore add them as a favorite – not dependent on whether they like their music or not.

But regardless of the motives for adding an artist as a favorite it requires a conscious effort from the user. We're therefore led to believe that the list of artists a user have favored are more important than other artists on Urørt. Our design is therefore based on the activities of a person's favorite artists. Initially we toyed with making a friend like concept on Urørt by saying that people which share one or more favorite artists with you are in some way similar to yourself. We had to throw away this idea because of technical problems with getting such a solution to scale.

Activity types

There were many activities taking place associated with an artist which we could potentially use in our activity stream. We decided to select those which contained information about when the activity occurred and was technically easy[6] to retrieve. Those that matched our criteria was:

6. Easy as in only requiring access to content freely available on Urørt.

- *Songs* being published by an artist.
- *Song reviews* being written by users.
- *Concerts* being performed by an artist.
- *Blog posts* being written by an artist.

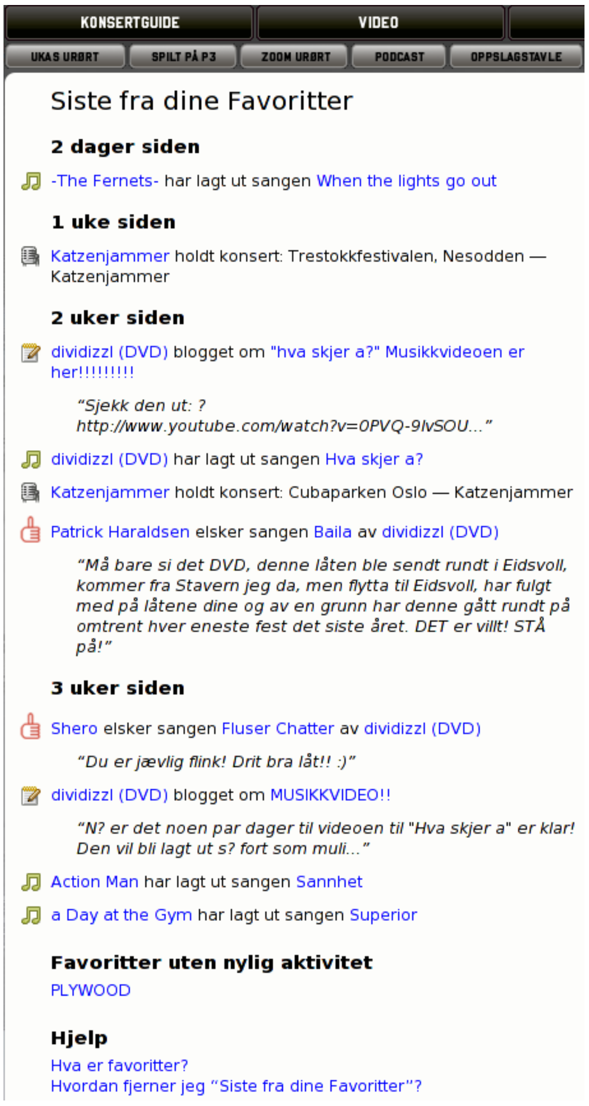
As you can see three of these activities are conducted by artists themselves while reviews are conducted by other users.

Activity filtering

There was quite bit of activity information available based on our four categories for a typical artist. When a user had several artists as favorites the amount of activity information got large. We therefore had to filter out some activities and only show what we thought would be most relevant.

Firstly we were only concerned with historical information in our activity list. Future concert events were therefore filtered out. Secondly we were only showing the most recent activities believing that fresh content is more important than aged content. As activities were sorted chronologically in our stream, we simply cut off all activities after a preset number of the most recent activities were displayed. The version of our prototype which was tested by real world users showed only the ten most recent activities.

In the case of Facebook, they are not showing the complete picture of friend activity in their news feed. It's believed that up to 80% of



Siste fra dine Favoritter

2 dager siden

-The Fernets- har lagt ut sangen When the lights go out

1 uke siden

Katzenjammer holdt konsert: Trestokkfestivalen, Nesodden — Katzenjammer

2 uker siden

dividizzl (DVD) blogget om "hva skjer a?" Musikkvideoen er her!!!!!!!!!

“Sjekk den ut: ? http://www.youtube.com/watch?v=0PVQ-9lvSOU...”

dividizzl (DVD) har lagt ut sangen Hva skjer a?

Katzenjammer holdt konsert: Cubaparken Oslo — Katzenjammer

Patrick Haraldsen elsker sangen Baila av dividizzl (DVD)

“Må bare si det DVD, denne låten ble sendt rundt i Eidsvoll, kommer fra Stavern jeg da, men flytta til Eidsvoll, har fulgt med på låtene dine og av en grunn har denne gått rundt på omtrent hver eneste fest det siste året. DET er villt! STÅ på!”

3 uker siden

Shero elsker sangen Fluser Chatter av dividizzl (DVD)

“Du er jævlig flink! Drit bra låt!! :)”

dividizzl (DVD) blogget om MUSIKKVIDEO!!

“N? er det noen par dager til videoen til "Hva skjer a" er klar! Den vil bli lagt ut s? fort som muli...”

Action Man har lagt ut sangen Sannhet

a Day at the Gym har lagt ut sangen Superior

Favoritter uten nylig aktivitet

PLYWOOD

Hjelp

Hva er favoritter?
Hvordan fjerner jeg “Siste fra dine Favoritter”?

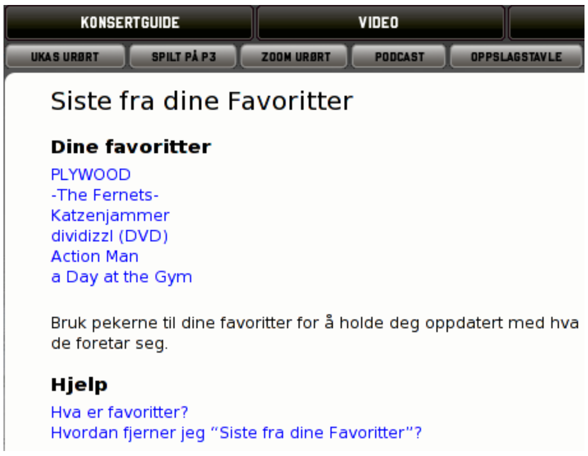
Figure 4.5: Activity stream for experiment users on Urørt, retrieved Jun 2, 2008, from http://www11.nrk.no/urort/default.aspx

all recent activities are filtered out (**?**). Facebook's approach for only showing a selection of recent activity is probably driven by technical scalability problems, and not a desire to only present a selection of the recent activity of friends. In our activity stream we gave users a complete picture of all types of activities within the cut-off point.

We named the implementation “Latest from your Favorites” – emphasizing the focus on providing the latest activities for a user's favorites. The complete design of our activity stream for Urørt can be seen in Figure 4.5.

4.2.3 *Favorite list*

As we'll describe in § 5.2.1 (p. 67) we decided to used a control group in our experiment setting. The navigational design we presented them with did not include an activity stream. In stead they were simply pre-



Siste fra dine Favoritter

Dine favoritter

PLYWOOD
-The Fernets-
Katzenjammer
dividizzl (DVD)
Action Man
a Day at the Gym

Bruk pekerne til dine favoritter for å holde deg oppdatert med hva de foretar seg.

Hjelp

Hva er favoritter?
Hvordan fjerner jeg "Siste fra dine Favoritter"?

Figure 4.6: Favorite list for control users on Urørt, retrieved Jun 2, 2008, from http://www11.nrk.no/urort/default.aspx

sented with a list of their favorites so that they manually had to keep up with what recent activity their favorite artists were conducting. This implementation was also named "Latest from your Favorites" so that it would appear similar to the full featured version. The design of the control group implementation can be seen in Figure 4.6.

To give both our control and experiment group the same navigational possibilities – excluding the activity stream – we decided to show a list of the favorites without recent activity for our experiment group. If all their favorites had activities present in the stream, no such list was displayed. Figure 4.5 (p. 51) shows the listing of one such artist without recent activity.

4.3 PROCESS

This sections details the development process we used for building our activity stream implementation for Urørt.

4.3.1 *Prototyping*

A *prototype* is an early version of an application that are used for finding out more about the problem at hand and its possible solutions (**?**, p. 409). The software developed for our research on social navigation fitted these characteristics. It was not supposed to be used after its behavior was evaluated. For that it was to inefficient and relied on specific web browser environments and extensions. This did not mean that the prototype couldn't have impact on how Urørt evolved in the future. If our evaluation favored our design decisions the developers of Urørt could take advantage of such potential improvements in their web site design.

?, pp. 409–410 explains that a prototype can be used for (i) gathering more sound requirements from users during a requirements phase, (ii) evaluating the feasibility of a proposed design during a design phase, and (iii) testing the final system by verifying it against the prototype. We followed his second example of prototype usage by making a prototype for what we believed to be a sound social navigation design. This was then evaluated. If time had permitted (sadly one has limited time and resources available during master thesis research) the results of such an evaluation could have been input in a new design process and a new prototype system.

As **?**, p. 114 explains, prototyping can mean different things based on context. Often it's used to explain systems where one writes the least amount of code to get a solution and throw it all away when the design question is answered. This was not our intent. We tried to make the system fully operational and make the code we authored comprehensible and valid. More precisely we were creating a *high-fidelity* prototype (**?**, p. 78) with a robust architecture.

The reason for creating a robust application was twofold. Firstly we were developing the prototype as part of our master thesis and it was meant to show how proficient we were in both programming and designing applications. We imagined delivering a barely working and poorly constructed software package would not be beneficial for our examination results. Secondly the application could potentially see some stress under user testing depending how large the test population were. Having a crashing or non-functioning application during such testing could have proved to be costly with regards to our experiment results.

We did therefore create a prototype to demonstrate our ability (or inability) to develop software. This was not our sole reason, and probably not the most important. As **?**, p. 292 describes, prototyping is not successful if one have not learned something during the process about what one set out to investigate at the outset. We wanted to investigate navigational designs and the prototype was with such a focus nearly a means to an end.

4.3.2 Testing

One can verify that the application one are building functions as it's supposed to do as code is added and changed by using automated tests. We have personally experienced benefits with using automated testing on earlier projects. *Test-driven development* is a development technique where one writes an automated test for a non-existent feature or improvement before one actually implement the feature itself – development is literally driven by tests. So the focus is not primarily on the test, but to better design software through a test-first approach. **?** conducted studies on both developers in companies and university students for finding out whether a test-driven approach improved the quality of software design. Groups which wrote a test before the accompanying code were measured against groups which wrote their test after implementing the solution. They found that code size decreased when a test-first approach was used – both classes and methods were smaller and simpler. In addition test-first developers had better coverage of the implementation code in their tests (?, p. 81). This could indicate that it's easier to opt out from testing after you have a piece of working code. The discipline that test-driven development dictates makes it impossible not to write tests.

Behavior-driven development is a response to the test-driven development approach. It was introduced by **?** for shifting the focus from writing tests to writing specifications of behavior (?). By writing specifications one is able to more clearly describe the intent of an application than when one are writing traditional tests.

We were convinced of the benefits of behavior-driven development based on earlier development projects and were therefore developing our prototype application with such a development process. Having specifications that could be automatically run to check how our application conformed to our expectations was very valuable. This was especially so when working with Urørt as an external data source since we did not have control over the stability of its structure.

4.4 ARCHITECTURE

Our implementation basically needed to do two things:

1 Collect existing data from various places on the Urørt web site.
2 Display this data in existing web pages on the Urørt web site in a way that we hoped would enhance navigation.

4.4.1 *Extending an established web site*

As we've described earlier in this chapter we were creating a prototype application. Inspired by solutions in the Hoodwink.d community we set out to implement our system on top of already established web pages.

? makes a case for using the means of such a browser extension – Greasemonkey[7] – and custom scripts for realizing projects that without such technology would never have come to existence. As he describe there is a sweet spot where such an approach really shines. He therefore created a framework for determining if a given project embodies the factors that would make this kind of an implementation a particularly suitable solution. What follows is a recitation of his selection factors and how our project adhered to these.

7. See § B.1.1 (p. 122) for details on Greasemonkey like extensions.

Do you have access to the source code of the web application?

When the source code of the target web site is available it would probably be easier to just change that. The case is made for using Greasemonkey when the source code is not available. Since we did not have access to the internals of Urørt we saw this factor as favorable for Greasemonkey in our project.

If the application is under your control and source code is available, is updating the application a risky endeavor?

This factor does not apply to our implementation since we obviously did not create Urørt ourselves. Had been the authors we could have used Greasemonkey for adding new features without putting the existing web site in danger, as it would be externalized from the original implementation.

How critical is the feature to be added?

If the web site is not functional or complete without the new feature it is advisable to defer from using Greasemonkey. The reason is the difficulty of ensuring that all users have the extension installed and the newest version of the custom script. Since we will be able to ensure that our test users have the right extension and scripts installed this is of no concern to us. In addition Urørt is functioning fine without our feature enhancement which makes Greasemonkey a sound technical solution for our means regarding this factor.

Is Firefox available for the potential users, and does it work with the target application?

The Firefox web browser should be available to install on most platforms and we can report that it works fine on the target web site. We cant expect all potential users to install Firefox on their computers. This limits the reach we have with a Greasemonkey based implementation.

To what degree are the target users computer literate?

Installing a web browser, an extension for it, and finally a custom script can be a bit complicated for certain users. We have to expect the test users to be a selection of our general population and some could therefore have trouble with achieving such a setup. On this aspect a solution that alters the original application would work better since users are not required to change their computing environment. We hope to partly solve this problem with providing clear instructions for how users can configure their environments to support custom Greasemonkey scripts.

What size is the user population that needs the new feature?

This factor is based around the fact that server applications can be more easily updated since they seldom require intervention by the user. It's much harder to ensure that client applications like Greasemonkey are kept up-to-date by its users. It's argued that such an approach works better for smaller populations and one should therefore keep users of custom Greasemonkey scripts to a minimum.

Since we're not foreseeing the use of our implementation outside user testing the application does not have to be kept up to date. In addition we're anticipating a fairly limited user base due to our test setting. Greasemonkey should therefore not put any hindrance in place for our implementation in relation to this factor.

How often is the page structure in the web application changing?

Since implementations based around Greasemonkey often is dependent on the underlying web site and its structure it's important that this remains stable while the implementation is in use. While we in our project had a fairly small time window of actual use, we were concerned with changes breaking our implementation.

We saw two possible solutions for this problematic aspect of Greasemonkey implementations. Firstly, communication between the developers of Urørt and ourself about upcoming changes would allow us to anticipate them and handle them gracefully when they arrived. Secondly one could introduce a meta structure in the web site by using agreed-upon class names of certain HTML elements in the style of what *microformats*[8] are trying to achieve.

8. Microformats are a set of simple and open data formats for better structuring of web content. All microformats adheres to the principles of solving a specific problem, starting as simply as possible, being designed for humans first – machines second, reusing elements from already established standards being modular and embeddable, and encouraging decentralized development, content, and services (**?**, p. 7). To us the essence of microformats seems to be the usage of semantic class names in HTML elements. By semantic class names we mean naming classes for what the elements they belong to represent.

Does the new feature require many changes to the existing web application?

Using Greasemonkey to enhance and alter existing web pages is not as straight forward as altering the source of the web page. Therefore massive changes to an existing web site is best handled directly in the source code leaving Greasemonkey to be the better alternative when only smaller alterations and addition are needed.

The changes we proposed to introduce in the Urørt web site was not earth shattering in scope and size. We therefore believed and hoped our implementation could be implemented without too much additional effort with Greasemonkey.

Does the target web application already have JavaScript code that mutates the page?

This factor tries to capture the fact that the behavior implemented with custom Greasemonkey scripts are run before any additional behavior implemented in the web site itself. It's therefore hard to manipulate a web site that mutates over time.

In our project we were quite fortunate as we were only adding behavior to the Urørt web site, not changing any existing behavior. We therefore did not have to concern ourselves with the dynamically already present on Urørt.

Does the feature require communication with a server in a different network domain than the web application?

Standard web pages can asynchronously retrieve information using JavaScript and the `XMLHttpRequest` object. To enforce a certain level of security browsers does not allow such requests to retrieve information from other domains than what the request was sent from. This limitation is eliminated in Greasemonkey scripts as one can request information from other domains through a similar asynchronous request object existing in the extension.

This feature was vital for our project. As we'll see in the next section we were dependent on requesting information from a resource that we ourselves had control over from the custom Greasemonkey script. This resource was however not hosted in the domain that Urørt is operating within. Without cross-domain requests in Greasemonkey our implementation would be infeasible.

Based on how our project positioned itself with regard to the important factors for the feasibility of using Greasemonkey according to **?**, we did not see any major objections for doing so. We believe the benefits of a prototype implementation based on Greasemonkey outweighed the disadvantages with such a technical solution.

Another solution would be to create a web proxy that altered pages specific to Urørt. **?**, p. 1109 took this approach when developing a collaborative bookmarking service. As the authors explain, a proxy-based solution ensures universal access across browsers and platforms requiring no installation of additional browser extensions or plugins. The drawbacks to such and approach for our prototype were quite clear. A user having enabled our hypothetical web proxy would have to take a round trip to our server for every web page he or she visited, regardless of whether it was Urørt related. This would introduce substantial loads on our servers and users would probably notice reduced page load times when browsing the Web.

4.4.2 Client-server model

A client-server model is an architecture where one is separating a system into two logical parts: one or more clients and one or more servers (**?**, p. 3). The client is a "computer system or process that requests a service of another computer system or process" (**?**, p. 11) and the server is a "provider of resources" (**?**, p. 49). More specifically, a client presents the user interface, uses a predefined language for querying the server, communicates these queries through a given communication method, performs computation on the the results of queries sent by the server, and displays these through the user interface (**?**, pp. 78–79). A server is characterized by providing a service to the client, responding to queries constructed by the client, and hiding away its underlying technical system for the client (?, p. 79).

The client and server thus have disparate responsibilities – the client is a consumer and the server is a producer (?, p. 3). This delegation is the essential part of client-server computing and enables one to focus on one aspect of a problem as one does when adhering to the concept of *separation of concerns* (?, p. 61). A client-server model also enables one to scale both horizontally and vertically[9] – an impossible feat with monolithic systems (?, pp. 7–8).

9. To scale horizontally entails adding more servers to a client-server architecture. When one increases the resources of a single server one is scaling vertically.

We therefore decided to use a client-server architecture so that we could offload some of the more computationally expensive operations off the client and onto a dedicated server. Another benefit of such an architecture is that it allows us to cache data globally – shared by all clients. This means that data collection is handled on the server-side, while data display is handled on the client-side.

Following the characterization of client-server systems provided by ? our client is presenting a user interface by altering the structure of the Urørt web page. The client sends queries to the server in the form of HTTP[10] formatted requests. Our server takes these queries in HTTP form and goes to the Urørt web site to collect the data it needs. Based on the data the server finds at the external Urørt web site, the server does some computations on the data before a response is sent with the HTTP communication method to the client. The client take this data (which is represented in a predefined format) and uses it to alter the user interface by introducing new information.

10. Short for Hyper Text Transfer Protocol – the protocol used for transferring data on the Web.

?, p. 887–888 implemented a system for re-finding places one have already visited on the Web. What's interesting about their system is that their architecture is strikingly similar to our own. They use a client-server model with a Greasemonkey enabled browser as their client. The client facilitates the users as they're navigating by supplying additional information alongside existing web pages. One of the ingredients in helping users in their browsing is suppling information from third party content providers. By separating this computationally heavy part of the application into the server-side they were able to offload the clients in a similar manner we used when fetching data from the Urørt web site for our prototype implementation of an activity stream.

4.4.3 Caching

In the planning stages of our implementation and its early development we decided to store all information retrieved from the Urørt web site in a relational database. We did some study into selecting the best ORM[11] for connecting our application to the underlying database. We had this nice model where the underlying relational database engine was abstracted away and interaction to this database was done with a clear

11. ORM, an acronym for object-relational mapping, is "the technique of converting records in a relational database into object instances in an object-oriented programming environment" (?, p. 889). This is possible since "relational databases can be represented reasonably in object-based code if you simply think of database tables as classes, table rows as objects, and table fields as object attributes" (?, p. 2).

and explicative DSL,[12] eliminating the need for constructing complex SQL.[13]

Then we had a sudden realization when coding on the part of our application that retrieved information from the external Urørt web site, pushed it into our relational database, and made our data available in a structured manner through our ORM layer. It was not critical if we lost some or the entirety of this data. We could just retrieve it again from our external source. And since the data had to be kept up to date by retrieving the data from our source within certain intervals, we saw no need to make it available in a relational database. We were not trying to persist data, but rather keeping a *cache*[14] of what we had retrieved.

As described we came to the realization that no persistent data about activities was needed in our implementation. We therefore sat out to find the best cache solution for our needs as detailed in § B.1.2 (p. 130).

We only cached one type of data in our implementation: a chronologically pre-ordered list of all activities for a given artist. Let us explain how this works when user X requests his list of recent favorite activities. Our system first determines that user X have artist A and artist B as favorites. It then consults the cache and asks if it can get a list of activities for artist A. The cache have a list of activities for artist A and promptly returns it to its caller. Our system then tries to do the same for artist B, but this time there is no list of activities for that artist in the cache. This is a *cache miss*, and we are therefore forced to retrieve information about artist B's activities from the Urørt web site. From this information we calculate a new list which are stored in the cache with a certain *time to live*. The next time a request for artist B's list of activities are handled the cache should now be able to return that list if the request were made within the time to live interval.

In our system we set a time to live of 12 hours. This meant that activity lists stored in the cache were retrievable within 12 hours.

12. DSL or domain-specific languages are programming languages shaped for a specific domain. By doing so one can offer more expressiveness in a limited application domain. This results in better usability and increased productivity within the limited domain, compared to using a general-purpose programming language (**?**, p. 317).

13. SQL (initially named SEQUEL) stands for structured query language and is now the *de facto* language for interacting with relational databases. It was invented by **?**, p. 250.

14. According to **?** a cache is "a temporary storage area where frequently accessed data can be stored for rapid access".

4.4.4 Persistence

As detailed in § 4.2.3 (p. 51) we designed two versions of our prototype: one delivering an activity stream and another only listing favorites. This introduced the need to deliver two versions of the client software, one for our main experiment group and one for our control group. To handle this we had to introduce persistence in our application to know what kind of client software the last user received to determine what software the next user should be provided with. This way every second user received a full-blown client version and the other users received a control version.

4.4.5 *Background processes*

With our cache architecture in place it became apparent that it was too expensive time wise for users to retrieve the activities of all their favorites when multiple cache misses occurred. We therefore decided to "warm up" the cache by running several background processes that populated our cache with fresh data from the Urørt web site several times a day.

Since we had implemented persistence for the various experiment participants of our system and their version of client software, we simply used their unique user names on the Urørt web site as input to our cache warm up processes.

After testing the system casually we found a need for shortening the response times in the event we received a request from a user with our full-blown client software where no relevant activity lists was present in cache. Initially we retrieved information from the Urørt web site for all artists that the requester had favored. Under this scheme a user with several favorites could expect response times to be several minutes in length. We therefore decided to deliver only the activities of the user's first favorite if non of the activity lists of his or her favorites were present in cache. When doing so we registered the non returned artists and put them in to a similar background process as we used for our cache warm up.

This meant that the first time a user made a request to our application he or she would be presented with the activities of only his first favorite if no other users previously requesting the system had similar favorites. After the background processing of this user's favorites was complete he or she would get a full picture of the activities of all his or her artists. We think this is an acceptable price to pay for reduced latency.

4.4.6 *Asynchronous requests*

In the traditional style of AJAX applications we requested information on the client-side of our prototype asynchronous.[15] This means that requests handled by the `XMLHttpRequest` object are independent of other requests the browser is making – they are non-blocking. This enables developers to create web pages where additional data is requested after the page is loaded by a normal HTTP request. This is often coupled with the ability to detect user behavior, and contents are requested only when needed. **?**, pp.281–282 argues that the increased complexity (and thereby size) of todays web pages have created problems in perceived responsiveness for users due to network latency. They think asynchronous request of small content items solves this problem – bringing greater interactivity to the user. Another solution that can be used is to move some processing into the client with JavaScript (**?**, p. 9)

15. While it's possible to create synchronous requests with the `XMLHttpRequest` object it's discouraged. With such requests the entire web browser will be locked while it's waiting for a response for its request (**?**).

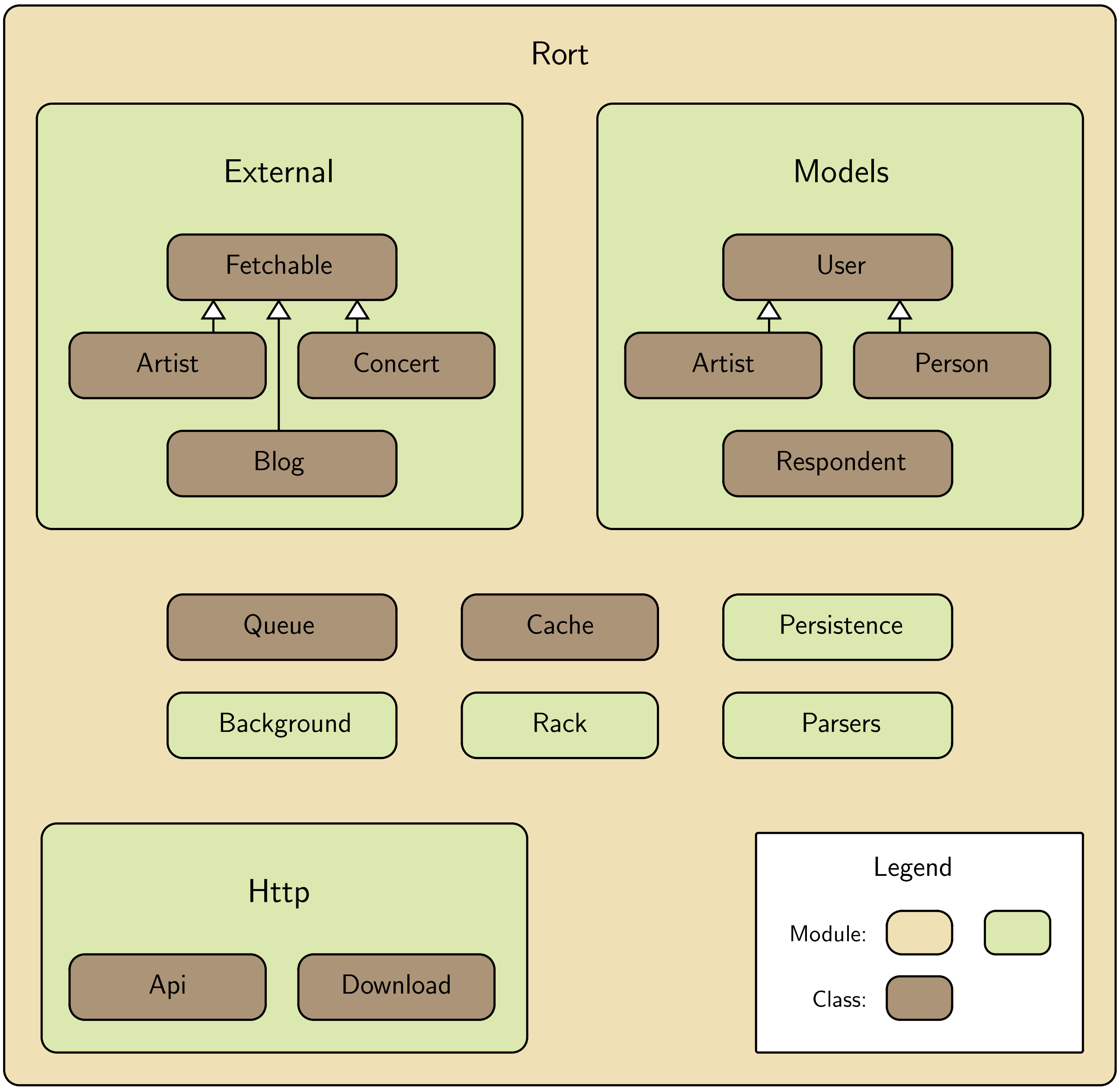


Figure 4.7: Diagram of classes and modules in the prototype server software.

was coined by ? sometime in 1967. The motivations leading to object-oriented programming were mainly to find "a better module scheme for complex systems involving hiding of details" (?, p514). The essence of object-orientation lies in the concepts of abstraction, classes, encapsulation, inheritance, objects, message passing, methods, and polymorphism (?, pp. 124–126).

4.4.7 *Code structuring*

Both the client-side and server-side programming languages we used are as we'll see in § B.1.1 (p. 123) and § B.1.2 (p. 125) *object-oriented*.[16] Since our client software was pretty dense in code size (see § 4.6 (p. 63) for details) and only had to handle a small set of operations we did not use object oriented features as classes and objects to structure our client code. Instead we used functions as a partitioning scheme when implementing on the client side. Our client side software was contained in two files – one file for each of our two versions (control and full-featured).

However, on the server side we had to write more code and our code had to handle quite a large set of operations. We therefore implemented our server side software with the use of modules, classes, objects, and methods. A figure of how the various classes and modules are related to each other can be found in Figure 4.7 (p. 61). The abstractions those constructs gave us, made it easier to precisely handle the complexity and disparity of our solution. As ?, p. 864 puts it:

the purpose of abstracting is not to be vague, but to create a new semantic level in which one can be absolutely precise.

To make the handling of our source code easier we used a hierarchy of directories and files that resembled our module and class constructs. One file and possible directories were used for each module and class as can be seen in Figure 4.8.

```
|-rort
|-|-external
|-|-|-blog.rb
|-|-|-concert.rb
|-|-|-fetchable.rb
|-|-|-artist.rb
|-|-queue.rb
|-|-cache.rb
|-|-background.rb
|-|-core_ext.rb
|-|-persistence.rb
|-|-external.rb
|-|-parsers.rb
|-|-models.rb
|-|-http
|-|-|-download.rb
|-|-|-api.rb
|-|-models
|-|-|-user.rb
|-|-|-person.rb
|-|-|-respondent.rb
|-|-|-artist.rb
|-|-rack.rb
|-|-http.rb
|-rort.rb
```

Figure 4.8: Hierarchy of the server prototype software. The code for producing this hierarchy listing can be found in § D.4 (p. 142).

4.4.8 *Data structure*

We'll shortly summarize the data structure used in our database. We stored informations about respondents to our survey in a `respondents` table. We used a `id` column as an identifier for each stored respondent; an `email` column for identifying respondents in relation to their survey answers; a `group` column to distinguish between experiment and control respondents; a `slug` column to store the unique identifier a respondent had at the Urørt web site so that we could warm up our cache (see § 4.4.5 (p. 60) for details); a `requests` column to count the times the respondent requested data from our server; and a `created_at` column to keep track of when a respondent first registered his email when downloading our client software. In Table 4.1 we give further details about the data we stored in a database.

4.5 PERFORMANCE

Initially we did not architect our application with performance in mind. We wanted to create a working prototype first and then try to increase

	Type	Constraints
`id`	Integer	Auto incremented primary key
`email`	Text	Unique, not empty
`group`	Text	Not empty
`slug`	Text	
`requests`	Integer	Initial default of zero
`created_at`	Time stamp	

Table 4.1: Prototype persistent data structure, by field name

it's performance if need be.

It became apparent that generating an activity stream for a user with several favorites was very time consuming. We identified the major bottleneck to be the actual retrieval of information from the Urørt web page. We could not do much code and algorithm wise to remedy this problem. The only solution was to increase the bandwidth allowed on our internet connection. As can be seen in Table 4.2 our retrieval times decreased as we moved to a production server with better internet connectivity.

	Speed (MB/s)	Time (s)
Development	0.4	4.0
Production	10.2	2.8

Table 4.2: Retrieval time and speed of a typical artist profile page (423kB in size), by system. Note that the speedup is not relative to the transfer speed as there is overhead in establishing a connection to the servers at Urørt.

The next bottleneck we became aware of was also related to the cost of retrieving web pages from Urørt. We did some careful counting of the number of retrieval attempts to Urørt and were able to minimize these to a considerably lower number.

The last bottleneck we encountered was the actual parsing of the retrieved web pages from Urørt. We were able to implement some minor optimizations, but did not see any major improvements.

But the major performance gains were seen when we introduced a fast cache solution and a system for periodically warming this cache up, as described in § 4.4.3 (p. 58) and § 4.4.5 (p. 60). With these two pieces in place we felt we could present users with bearable response times.

4.6 SOURCE CODE

We'll wrap up the discussion of our implementation with showing some statistics related to our source code:

		Number of lines of		
	Files	code	comments	blank lines
Client implementation	2	373	42	87
Server implementation	21	850	9	189
Server specification	16	714	0	179

Table 4.3: Prototype source code statistics, by type

By dividing our number of lines of specification code with implementation code we get a ratio of 84:100 between specification/implementation on the server side.

EMPIRICAL STUDY OF A SOCIAL NAVIGATION PROTOTYPE

5

This chapter details an empirical study of the prototype implementation we developed in Chapter 4 (p. 45). First we'll frame what we think this study should answer through research problems and hypotheses. Then we'll go on to describe the method we used for conducting this study before the results of the study are presented. Based on the results we'll discuss our findings before we close this chapter with some words about the limitations of this particular empirical study.

In this chapter we'll use the following symbols and abbreviations:

N	number of occurrences
Mdn	median
Rng	range
σ	standard deviation
$\overline{x}$	mean for x
U	Mann-Whitney U-test statistic
T	smallest of the two sums of ranks for the Wilcoxon signed-rank test
Z	standard score
p	probability value
α	level of significance

Table 5.1: Statistical Symbols and Abbreviations

5.1 RESEARCH PROBLEMS AND HYPOTHESES

Our main research question deals with how a specific social navigation technique, activity streams, could potentially influence usage of a web site:

RP0: *Can social navigation through activity streams influence usage of an established web site?*

From this main research question we also proposed more specific problem statements that more clearly states different ways of influencing usage:

RP1: *Do users perceive social navigation through activity streams as helpful in order to keep up-to-date on favorites' activities on Urørt?*

This question deals with the way users keep up-to-date on what their favorites are doing on Urørt. We want to investigate if activity streams can help users in this task.

We were also concerned with how activity streams influenced the frequency of keeping up-to-date on activities:

RP2: *Does social navigation through activity streams lead users to more often keep up-to-date on favorites' activities on Urørt?*

The next research question deals with with how activity streams could potentially influence the importance of favorites on Urørt:

RP3: *Does social navigation through activity streams lead users to make more artists on Urørt their favorites?*

We also had a more technical research question relating to how one can conduct experiments on established web sites with Greasemonkey:

RP4: *Can prototyping with Greasemonkey be considered a viable technical option when testing user behavior in an established web site?*

This question seek to investigate whether Greasemonkey prototyping should be considered as one of potentially many technical alternatives in future research experiments, where user behavior is tested. As this is a new way to test user behavior in relation to social navigation, we're mainly concerned with gaining experience with using such a technical solution.

We created hypotheses for the majority of our research questions. After having presented our results in § 5.3 (p. 74) we'll test these hypotheses in § 5.4 (p. 86).

Our $H1$ hypothesis deals with how easy respondents can keep-up-to date with favorites' activities with and without an activity stream. B is the degree respondents can easily keep up-to-date on favorites' activities without an activity stream. A is the degree respondents can easily keep-up-to-date on favorites' activities with an activity stream.

- $H1_0$: $B \geq A$
- $H1_A$: $B < A$

Relating to the hypotheses about the degree respondents can keep up-to-date on activities we have a *H*2 hypothesis concerning the frequency of keeping up-to-date. *B* is the frequency respondents keep up-to-date on favorites' activities without an activity stream. *A* is the frequency respondents keep up-to-date on favorites' activities with an activity stream.

- $H2_0$: $B \geq A$
- $H2_A$: $B < A$

A *h*3 hypothesis about how activity streams influences favorite usage is stated next. *B* is the amount of favorites for respondents without an activity stream. *A* is the amount of favorites for respondents having used an activity stream.

- $H3_0$: $B \geq A$
- $H3_A$: $B < A$

5.2 METHOD

This section outlines the methodology we used for testing our research hypotheses. We wanted to test our hypotheses in a real usage situation, while having control of effects that could influence the results. We therefore decided to conduct a real world experiment with a pretest and posttest, in addition to using a control group. We'll dive into the design of the experiment, how we collected the data, and how data was analyzed.

5.2.1 *Experiment design*

?, p. 78 describes an experiment as a process where:

- The experimenter assigns subjects to different conditions.
- The experimenter manipulates one or more variables. These variables are *independent variables*.
- The experimenter measures the effect of the manipulation of the independent variables on other variables. These other variables are *dependent variables*.

We are conducting a real world experiment, meaning that our subjects are studied in their natural habitat – not in a laboratory. The advantages of a real world experiments are firstly that it's easier to generalize results to a wider real world population since one does not have an artificial setting as found in laboratories. Secondly, real world experiments are not as prone to gaming by its participants. Lastly, it may be easier to find willing subjects in the real world.

Real world experiments have some shortcomings compared to laboratory experiments. Seemingly most important is the lack of control of various variables which could interfere with the independent variables. For more information about the merits and disadvantages of real world experiments see **?**, pp. 80–87.

In addition to a real world experiment we're using a two-group experiment design with a test before and after the independent variables are manipulated through a *treatment*. The two groups are:

- An *experiment group*: *E*. This group are given a treatment by manipulating an independent variable.
- A *control group*: *C*. This group are not given a treatment but are instead given a *placebo* which does not manipulate the independent variable.

Using two groups means that we can measure the difference between those that underwent treatment and those that were given a placebo. This gives us a way to measure any difference induced by the actual treatment since we can detect possible *placebo effects* or *observer effects*. We will then be able to see if respondents are answering positively simply because they are given something new or since they know they are observed.

By using a before and after design we are also able to use pre-post differences as a basis for measuring the effect of treatment or no treatment.

In our case the treatment is analogous with the prototype implementation with an activity stream for Urørt as described in § 4.2.2 (p. 47). The placebo on the other hand is the prototype implementation with a favorite list as seen in § 4.2.3 (p. 51). The favorite list is an integrated part of the treatment implementation. This means that the only difference between the two prototype versions are the activity stream – the independent variable we as experimenters are manipulating (by introducing the feature).

Bearing in mind the details of our experiment design, Figure 5.1 gives an overview of how the experiment process was carried out. In addition to the pretest and posttest we conducted a follow-up survey of our respondents to gauge whether they managed to install our prototype implementation. The timeline of the various parts of the experiment were as follows:

1. Pretest: day 1
2. Follow-up: day 2
3. Posttest: day 11

5.2.2 *Subjects*

The subjects were the 789 latest users of Urørt which had signed in with their user name and password on the Urørt page as of when the experiment was started.

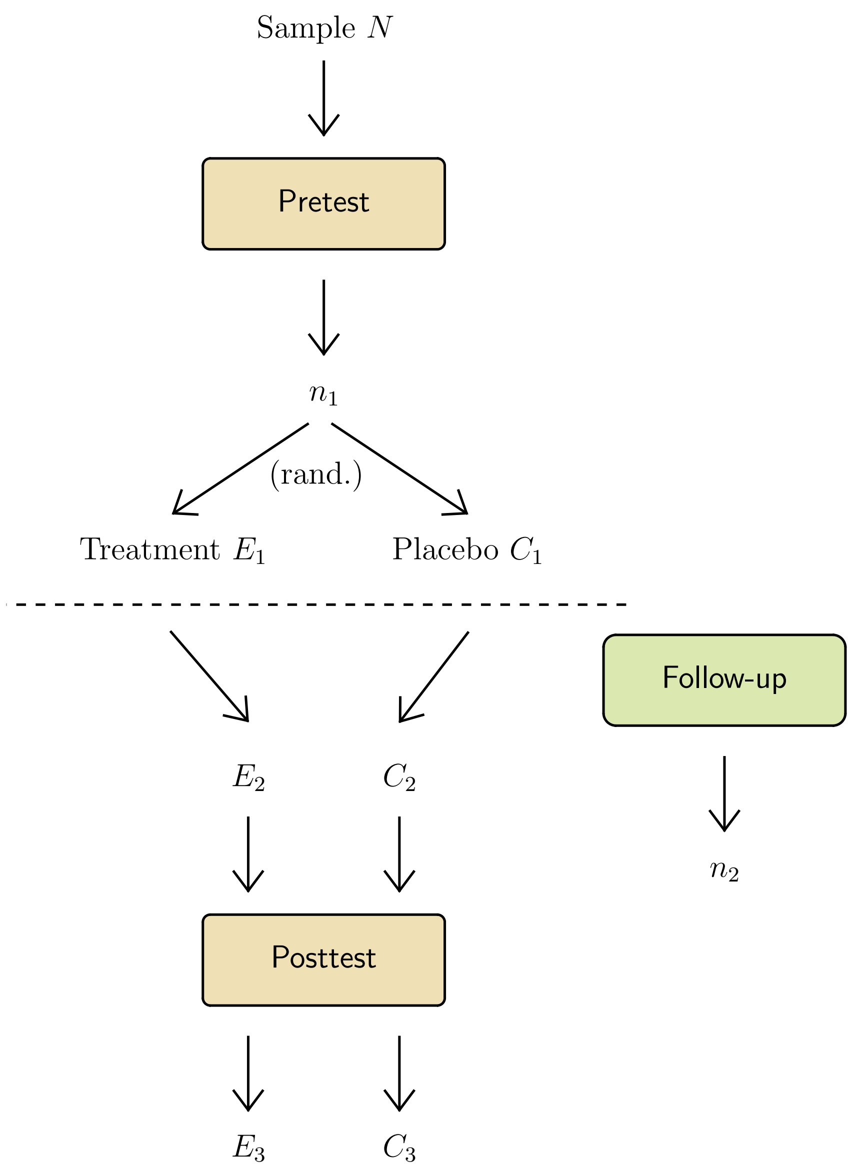


Figure 5.1: Overview of the various parts of the experiment. The sample N are given a pretest. n_1 completes the pretest. E_1 are given an treatment prototype while C_1 is given a placebo prototype by randomization. After one of the two types of prototypes are provided, respondents are followed-up to check if they had problems installing the prototype software. n_2 answers the follow-up questions. E_2 successfully installed the treatment prototype and C_2 successfully installed the placebo prototype. Both E_2 and C_2 are given a posttest. E_3 of the treatment sample E_2 and accordingly C_3 of the placebo sample C_2 completes the posttest.

We wanted about 200 participants to answer our pretest and guessed that 100 would be bothered to install our prototype. Through randomization that would amount to 50 users in the experiment group and 50 users in the control group.

Since we were only concerned with users of the Firefox web browser we had to contact a fairly high number of potential respondents to yield a sufficient number of respondents using this particular browser. We had read that a little more than 13% of world wide internet users use Firefox (**?**).

5.2.3 *Data collection*

We collected all our data through questionnaires. An online survey system was used for creating a pretest, posttest, and follow-up questionnaire. The various questionnaires can be found in their original language and wording in Appendix C (p. 137). What follows are translations to English of the most important questions and the response options.

Pretest & posttest questions

These questions were asked both in the pretest and posttest. First we asked questions to investigate the usage of favorites on Urørt by our respondents:

- How many favorites do you have on Urørt? – we expected a numerical value of the amount of favorites the respondent had.
- How often do you update yourself on what your favorites on Urørt are doing? – the respondents could select between the following frequency categories: daily, several times a week, weekly, monthly, and seldom/never.

Then we asked the respondents to qualify several statements which investigated how easy it was for them to keep up-to-date on favorites. These statements was to be rated on a 5-point Likert scale (**?**):

- Fully disagree
- Somewhat disagree
- Neither agree nor disagree
- Somewhat agree
- Fully agree

With the statements:

- It's easy to keep up-to-date on what my favorites are doing on Urørt.
- It's easy to keep up-to-date on whether my favorites publishes new songs on Urørt.

- It's easy to keep up-to-date on whether my favorites publishes new blog posts on Urørt.
- It's easy to keep up-to-date on whether my favorites are performing at concerts.
- It's easy to keep up-to-date on the reactions other users at Urørt have towards my favorite artists' songs.

Pretest-only questions

These questions were asked only in the pretest. Here we asked questions to get an impression of our pretest respondents:

- Age? – a numerical value was expected.
- Gender? – either male or female.
- Firefox user? – selection between the following frequency of use categories: always, regularly, sometimes, and seldom/never.
- How often do you use Urørt? – selection between the following frequency categories: daily, several times a week, weekly, monthly, and seldom/never.
- Do you sign-in (with user name and password) when using Urørt? – selection between the following frequency categories: always, regularly, sometimes, and seldom/never.

Posttest-only questions

These questions were asked only in the posttest. First we asked specifically about usage of the prototype implementation.

- How frequently have you used "Latest from your Favorites" when you are signed-in on Urørt? – selection between the following frequency of use categories: have not used, only a few times, almost every time, and every time.

Next we wanted to investigate the perceived usefulness and ease of use for our prototype implementation. This well tested approach for conveying technological acceptance was introduced by **?**. Like **?**, p. 340 we used a 7-point scale as possible answers:

- Extremely unlikely
- Unlikely
- Slight unlikely
- Neutral
- Slight likely
- Likely
- Extremely Likely

We shorted the statements of perceived usefulness down to four alternatives which we felt made sense for our implementation:

- “Latest from your Favorites” would enable me to keep up-to-date on my favorites in an efficient manner.
- “Latest from your Favorites” would enable me to keep up-to-date on more favorites.
- “Latest from your Favorites” would make it easier to keep up-to-date on favorites.
- “Latest from your Favorites” would be useful for keeping up-to-date on favorites.

The statement for perceived ease of use was also shorted down to four alternatives:

- It would be easy to learn to use “Latest from your Favorites”.
- It would be easy to make “Latest from your Favorites” do what I want.
- It would be easy to become skillful at using “Latest from your Favorites”.
- “Latest from your Favorites” would be easy to use.

Lastly we wanted to investigate if respondents would like “Latest from your Favorites” to be a standard feature on Urørt. We used a 5-point Likert scale (**?**) to gauge respondents reactions to our question:

- Fully disagree
- Somewhat disagree
- Neither agree nor disagree
- Somewhat agree
- Fully agree

With the specific question:

- Do you think “Latest from your Favorites” should be a standard feature of Urørt?

Follow-up questions

We also asked the participants to answer a short questionnaire regarding how the installation process went. This was done to investigate how easy or hard installation of Greasemonkey based prototypes were:

- Did you manage to install “Latest from your Favorites”? – the respondent had to choose between these categories: yes – it was an easy and quick process, yes – but I experienced small problems, yes – but I experienced large problems, and no – I gave up.

5.2.4 *Data analysis*

We used two statistical tests for analyzing the data we collected. Since our sample sizes were fairly small we could not make any claims about the forms of the distribution for the Urørt population from where our sample was drawn. This makes it suitable to use *non-parametric* statistical tests as they don't make any assumptions about the distribution of the variables to be tested (?, p. 34).

Most of our data are of an ordinal measurement – meaning that one can infer the rating of variables, but not the distance between ratings. To represent the central tendency of our data we'll therefore use the median (?, p. 27). Ordinal data is another characteristic where non-parametric tests are best suited (?, p. 35).

Mann-Whitney U-test

The Mann-Whiteney U-test is a non-parametric test for comparing two independent conditions and can be seen as the non-parametric alternative to the independent t-test (?, p. 522). More formally the Mann-Whiteney U-test can "be used to test whether two independent groups have been drawn from the same population" (?, p. 128). According to ?, chapter 11a the assumptions of the Mann-Whitney U-test are:

1 The two samples under test should be randomly and independently drawn.
2 The dependent variable should be intrinsically continuous.
3 The measures of the two samples should be at least of an ordinal scale.

We used the Mann-Whitney U-test when comparing differences between our experiment group and control group.

Wilcoxon signed-rank test

The Wilcoxon signed-rank test is a non-parametric test for comparing two related conditions and is a non-parametric alternative to the dependent t-test (?, p. 534). The assumptions of the Wilcoxon signed-rank test are similar to those of the Mann-Whitney U-test (?, chapter 12a):

1 The paired values of X_A and X_B should be randomly and independently drawn.
2 The dependent variable should be intrinsically continuous.
3 The measures of the paired values X_A and X_B should be at least of an ordinal scale.

The Wilcoxon signed-rank test was used for comparing differences within groups between the pretest and posttest.

		Gender (%)							
	N	female	male	$\overline{Age}$	σ	U	Z	p (2-tailed)	
E	13	14.3	85.7	26.23	10.902	68.5	-0.174	0.876	E – C
C	11	—	100.0	24.55	7.917	580.0	-0.468	0.645	E – Rest
Rest	97	8.2	91.8	26.81	9.324	475.0	-0.595	0.559	C – Rest

Table 5.2: Respondents gender and age. Comparison of age between experiment group, control group, and those not given treatment or placebo.

Level of significance

For the two statistical test we've described we decided on a level of significance α of $p \leq 0.05$. This is the level for α we worked with based on our sample size and the importance of the results which will be obtained (**?**, pp. 8–9).

Outliers

We went through the collected data looking for outliers – "deviant scores that [. . .] are uncommon score or they range far from the central tendency of the data set" (**?**, p. 84). We were only concerned with such numerical outliers for ratio values (age, number of favorites) where it would be meaningful to talk about means. For ordinal values we were not concerned with such outliers as we operated with medians for conveying the center of a distribution. If outliers were found, the value was simply deleted from the sample.

Another form of outliers can be respondents which respond very monotonous – indicating that they just tick off questions without any thought. In such cases the respondents motivation is to complete the questionnaire as fast as possible. We did not eliminate such responses from the sample as it's hard to know the exact motivations of the respondents. In addition we don't have a particular large set of values for each respondent to base such analysis on. It would therefore be hard to distinguish such outliers.

5.3 RESULTS

Figure 5.2 gives an overview of how many respondents we had to our pretest, posttest, and follow-up questions. As seen in the figure the non-achievement rate was quite high from the initial sample N to the respondents of our pretest n_1. The non-achievement rates was more or less constant from pretest respondents n_1 to treatment E_1 and placebo C_1 obtention, to successful treatment E_2 and placebo C_2 installation, and finally to posttest respondents E_3 and C_3. The non-achievement rate for going through all steps from a pretest ($n = 123$) to the posttest ($n = 14 + 11 = 25$) is quite high at 79.7%.

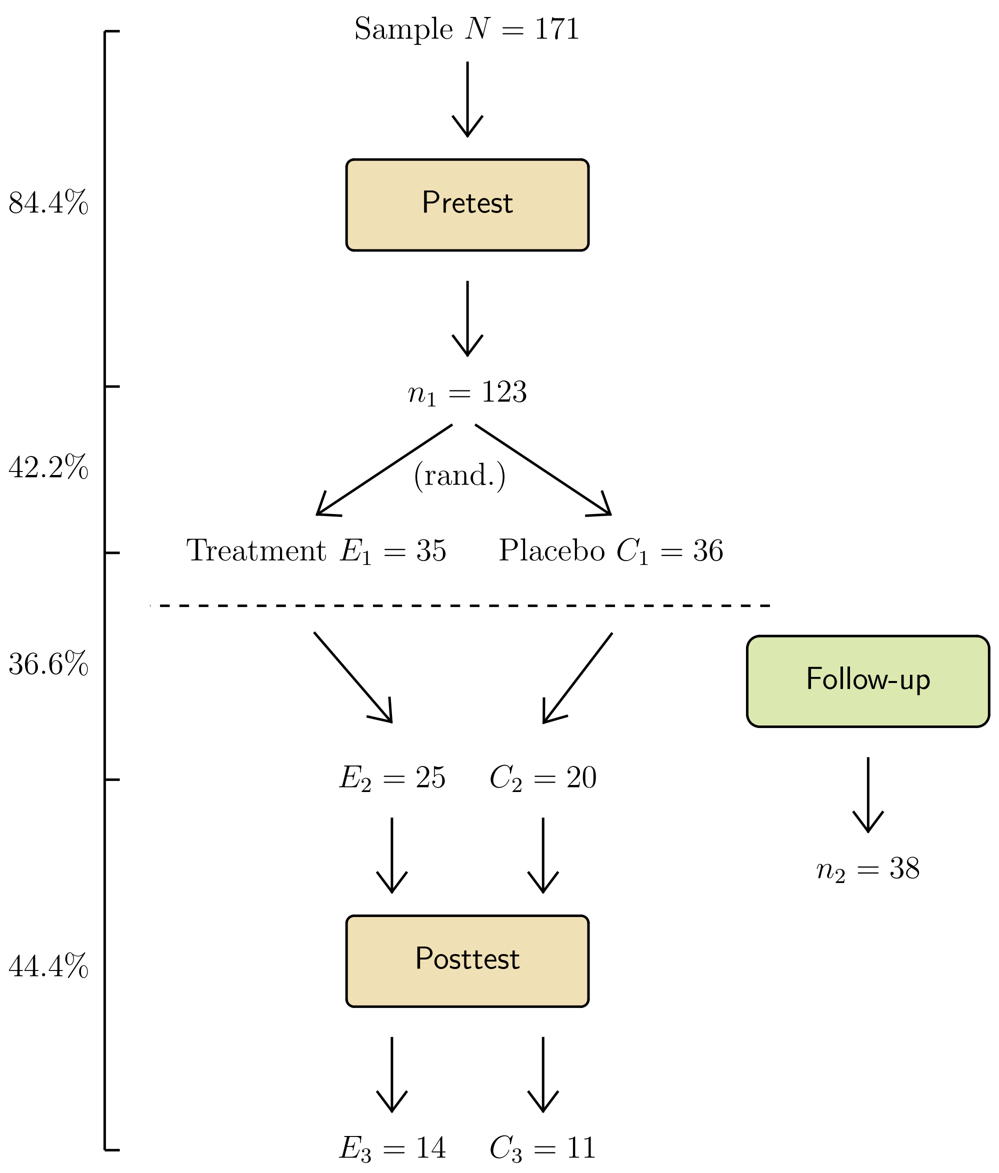


Figure 5.2: Non-achievement rates for the various parts of the experiment. The sample N of 171 Urørt users were given a pretest and a sample n_1 of 123 completes the pretest. By randomization a sample E_1 of 35 were given a treatment while a sample C_1 of 36 were given a placebo. The sample n_1 were given a follow-up questionnaire for checking how the installation of the prototypes went. n_2 completed the follow-up questionnaire. A sample E_2 of 25 and a sample C_2 of 20 managed to install the treatment and placebo prototype respectively. Of those, a sample E_3 of 14 from the treatment sample E_2 and a sample C_3 of 11 from the placebo sample C_2 completed a posttest.

	N	*Mdn*	*Rng*	*U*	*Z*	*p* (2-tailed)		
E	14	5	3	70.5	-0.407	0.707	*E* – *C*	
C	11	5	2	668.0	-0.181	0.887	*E* – Rest	Firefox usage
Rest	98	5	3	508.5	-0.352	0.748	*C* – Rest	
E	14	3	4	47.5	-1.757	0.085	*E* – *C*	
C	11	4	2	653.5	-0.294	0.773	*E* – Rest	Urørt usage
Rest	98	3	4	353.5	-1.918	0.055	*C* – Rest	
E	14	4	3	63.5	-0.774	0.462	*E* – *C*	
C	11	3	3	572.5	-1.035	0.306	*E* – Rest	Authenticated usage
Rest	98	3	3	527.5	-0.125	0.920	*C* – Rest	

Table 5.3: Respondents Firefox and Urørt usage. Comparison between experiment group, control group, and those not given treatment or placebo.

Table 5.2 (p. 74) shows the gender distribution and mean age of both our experiment group, control group, and the respondents which never got a treatment or placebo. The age differences between groups are unnoticeable and far from statistically significant. The data shows that the majority of respondents were male.

In addition to gather characteristics about age and gender we also investigated how frequent respondents used Firefox as their browser, how often they used Urørt, and how often they used Urørt in an authenticated state. Based on the questions:

- Do you use Firefox?
- Do you sign-in (with user name and password) when using Urørt?

we graded respondents answers as follows:

- Always: 5
- Regularly: 4
- Sometimes: 3
- Seldom/never: 2

Based on the question: "How often do you use Urørt?" we graded respondents answers as follows:

- Daily: 5
- Several times a week: 4
- Weekly: 3
- Monthly: 2
- Seldom/never: 1

The data in Table 5.3 shows little difference in Firefox usage between groups – the differences are far from statistically significant. The same

	N	Mdn	$Mdn_{\sum E} - Mdn_{\sum C}$	Rng	U	Z	p (1-tailed)
E	12	5	$>$ 1	4	45	-1.353	0.089
C	11	4		3			

Table 5.4: Up-to-date on favorites' activities. Comparison between experiment and control group for the posttest.

trend can be observed for whether the respondents are authenticated when they use Urørt.

In actual Urørt usage, we observe that control group respondents had higher usage frequencies, both compared to experiment respondents and respondents which never were given a treatment or placebo. The difference is close to statistically significant, compared between the control group and those without treatment or placebo (respondents not having tried to install our prototype).

5.3.1 RP1: *Do users perceive social navigation through activity streams as helpful in order to keep up-to-date on favorites' activities on Urørt?*

Our H_0 stated that we would not see any positive change in how easy respondents felt it was to keep up-to-date on favorite's activities after introducing an activity steam. Our H_A said that we would indeed observe a change in how respondents rated this task.

Activities in general

Based on the statement "It's easy to keep up-to-date on what my favorites are doing on Urørt" we graded respondents answers as follows:

- Fully disagree: 1
- Somewhat disagree: 2
- Neither agree nor disagree: 3
- Somewhat agree: 4
- Fully agree: 5

First we compared how easy respondents felt it was to keep up-to-date on favorites' activities after they were given a treatment (an activity stream) or a placebo. See Table 5.4 for the results.

The results show a tendency towards higher acceptance scores for experiment respondents having used an activity stream, than control respondents with a placebo. This difference is however not statistically significant.

		N	Mdn	Post − Pre	$Mdn_{\sum E} - Mdn_{\sum C}$	Rng	T	Z	p (1-tailed)
E	Pre	14	3	2	1	4	10	-1.513	0.086
	Post	12	5			4			
C	Pre	11	3	1		3	16	-0.787	0.258
	Post	11	4			3			

Table 5.5: Up-to-date on favorites' activities. Comparison between pretest and posttest within the experiment and control group.

Then we tested if there was a change in acceptance within the respondent groups from before they were given a treatment or placebo and after. The results can be seen in Table 5.5.

The data shows a tendency towards higher acceptance rates from the pretest to the posttest for both experiment and control respondents. The acceptance of the statement about how easy it was to keep up-to-date on favorites' activities have risen more for the respondents having used an activity stream than those with a placebo. Neither of these increases are statistically significant.

Specific activities

We'll now look at respondents answers to more specific statements about keeping up-to-date on activities. Based on the statements:

- It's easy to keep up-to-date on whether my favorites publishes new songs on Urørt.
- It's easy to keep up-to-date on whether my favorites publishes new blog posts on Urørt.
- It's easy to keep up-to-date on whether my favorites are performing at concerts.
- It's easy to keep up-to-date on the reactions other users at Urørt have towards my favorite artists' songs.

we graded respondents answers as follows:

- Fully disagree: 1
- Somewhat disagree: 2
- Neither agree nor disagree: 3
- Somewhat agree: 4
- Fully agree: 5

We compared how easy respondents felt it was to keep up-to-date on favorites' specific activities after they are given a treatment or a placebo. See Table 5.6 for the results.

The reported degree of how easy it is to keep up-to-date on new songs, blogs posts, and reviews is higher for those which used an activity stream than those without. Respondents using and activity stream

	N	Mdn	$Mdn_{\sum E}$ $-Mdn_{\sum C}$	Rng	U	Z	p (1-tailed)	
E	12	5	1,0	4	59	-0.479	0.340	Song
C	11	4		2				
E	12	4.5	1.5	4	59	-0.460	0.289	Blog
C	11	3		2				
E	12	3.5	-0.5	4	66	0.000	0.517	Concert
C	11	4		3				
E	12	3.5	0.5	4	60	-0.385	0.372	Review
C	11	3		3				

Table 5.6: Up-to-date on favorites' specific activities. Comparison between experiment and control group for the posttest.

		N	Mdn	Post – Pre	$Mdn_{\sum E}$ $-Mdn_{\sum C}$	Rng	T	Z	p (1-tailed)	
E	Pre	14	4	1	0	4	8.0	-1.780	0.049	Song
	Post	12	5			4				
C	Pre	11	3	1		3	4.0	-1.709	0.055	
	Post	11	4			2				
E	Pre	14	3	1.5	1.5	3	9.5	-1.872	0.039	Blog
	Post	12	4.5			4				
C	Pre	11	3	0		3	13.0	-1.150	0.166	
	Post	11	3			2				
E	Pre	14	3	0.5	-0.5	3	7.5	-1.127	0.188	Concert
	Post	12	3.5			4				
C	Pre	11	3	1		3	5.5	-1.063	0.203	
	Post	11	4			3				
E	Pre	13	3	0.5	1.5	4	4.5	-1.298	0.125	Review
	Post	12	3.5			4				
C	Pre	11	4	-1		3	9.5	-0.780	0.281	
	Post	11	3			3				

Table 5.7: Up-to-date on favorites' specific activities. Comparison between pretest and posttest within the experiment and control group.

reported a lower values of how easy it was to keep up-to-date on concerts compared to those without such treatment. Neither the tendencies towards increases and decreases in degrees of keeping up-to-date, from the control group to the experiment group, were statically significant.

As for general activities, we also tested whether respondents perception of how easy it was to keep up-to-date on specific activities from favorites had changed from before a treatment or placebo was given, to after. See Table 5.7 (p. 79) for the results of the in group comparisons.

The within group data shows that experiment respondents and control respondents find it easier to keep up-to-date on recent songs and concert performances after the treatment or placebo was given. The increases are of the same order for both those which used an activity stream and those without. Neither of the two groups increases for concerts is statistically significant. The experiment groups' increase is significant for songs while the control groups' increase for songs is insignificant. Note that the difference between *p* values for the two groups related to songs are marginal.

Respondents having used an activity stream saw an increase in median response from "neither agree or disagree" to "fully agree" regarding how easy it was to keep up-to-date on blog posts of favorites. This difference is statistically significant. The control group saw no such increase after they were given a placebo.

On the topic of how easy respondents could keep up-to-date on reviews of favorites' songs the experiment group saw no change after treatment while the control group saw a decrease after having used a placebo. This decrease is not statistically significant.

Perceived usefulness

Like **?** we asked respondents about the perceived usefulness of the prototype they were given (an activity stream or a placebo) in relation to keeping up-to-date on favorites. Based on the statements:

- "Latest from your Favorites" would enable me to keep up-to-date on my favorites in an efficient manner.
- "Latest from your Favorites" would enable me to keep up-to-date on more favorites.
- "Latest from your Favorites" would make it easier to keep up-to-date on favorites.
- "Latest from your Favorites" would be useful for keeping up-to-date on favorites.

were graded respondents answers as follows:

- Extremely unlikely: 1
- Unlikely: 2
- Slight unlikely: 3
- Neutral: 4
- Slight likely: 5
- Likely: 6
- Extremely Likely: 7

As these statements were only given in the posttest we compared how efficient, quantifiable, easy, and useful respondents felt it was to keep up-to-date on favorites' specific activities after they had used an

	N	Mdn	$Mdn_{\sum E}$ $-Mdn_{\sum C}$	Rng	U	Z	p (1-tailed)	
E	12	6.5	$>$ 0.5	3	58.0	-0.547	0.355	Effective
C	11	6		2				
E	12	6	$>$ 0	3	62.0	-0.270	0.387	More
C	11	6		2				
E	12	6.5	$>$ 0.5	3	55.5	-0.707	0.286	Easier
C	11	6		2				
E	12	6	$>$ 0	3	60.5	-0.370	0.400	Useful
C	11	6		2				

Table 5.8: Perceived usefulness of "Latest from your Favorites". Comparison between experiment and control group for the posttest.

activity stream or a placebo. In other words we compared the difference between the experiment and control group. See Table 5.8 for the results.

The data shows no differences in ranking regarding "Latest from your Favorites" perceived ability for enabling respondents to keep up-to-date on more favorites and general usefulness for keeping up-to-date on favorites, between the experiment and control group. The experiment group shows a small increase in the ranking of "Latest from your Favorites" effectiveness and easefulness for keeping up-to-date on favorites compared to the control group. These differences is not statistically significant.

Perceived ease of use

Borrowing from the works of **?** again, we asked respondents about the perceived ease of use of the prototype they were given (an activity stream or a placebo). Based on the statements:

- It would be easy to learn to use "Latest from your Favorites".
- It would be easy to make "Latest from your Favorites" do what I want.
- It would be easy to become skillful at using "Latest from your Favorites".
- "Latest from your Favorites" would be easy to use.

were graded respondents answers as follows:

- Extremely unlikely: 1
- Unlikely: 2
- Slight unlikely: 3
- Neutral: 4
- Slight likely: 5
- Likely: 6
- Extremely Likely: 7

	N	Mdn	$Mdn_{\sum E}$ $-Mdn_{\sum C}$	Rng	U	Z	p (1-tailed)	
E	12	6	0	3	57.5	-0.561	0.320	Easy to learn
C	11	6		3				
E	12	5.5	-0.5	3	50.5	-1.025	0.177	Flexible
C	11	6		4				
E	12	6	0	3	58.5	-0.486	0.343	Become skillful
C	11	6		3				
E	12	6	0	4	59.5	-0.429	0.372	Easy to use
C	11	6		3				

Table 5.9: Perceived ease of use for "Latest from your Favorites". Comparison between experiment and control group for the posttest.

Just as the perceived usefulness statements, these statements were only given in the posttest. We compared the responses to these statements between the experiment and control group to see if the introduction of an activity stream or placebo made a difference in how easy the respondents perceived "Latest from your Favorites" would be to use. See Table 5.9 for the results.

The ease of use data shows little differences between the groups. The respondents perception of how flexible "Latest from your Favorites" would be in use are actually higher for the control group – does not using an activity stream but a placebo. This difference is not statistically significant.

Activity stream as standard feature

We wanted to measure how well perceived the prototype was, by asking if people would want the functionality we provided to be a standard feature on the Urørt web site. Based on the question:

- Do you think "Latest from your Favorites" should be a standard feature of Urørt?

we graded respondents answers as follows:

- Fully disagree: 1
- Somewhat disagree: 2
- Neither agree nor disagree: 3
- Somewhat agree: 4
- Fully agree: 5

This question was naturally only asked in the posttest after usage.

	N	Mdn	$Mdn_{\sum E}$ $-Mdn_{\sum C}$	Rng	U	Z	p (1-tailed)
E	11	5	0	1	55.0	-0.607	0.500
C	11	5		1			

Table 5.10: The prototype as a standard feature. Comparison between experiment and control group for the posttest.

	N	Mdn	$Mdn_{\sum E}$ $-Mdn_{\sum C}$	Rng	U	Z	p (1-tailed)
E	13	2	-1	2	24.5	-2.955	0.002
C	11	3		2			

Table 5.11: Frequency of keeping up-to-date on favorites' activities. Comparison between experiment and control group for the posttest.

Table 5.10 compares responses between the experiment and control group.

The standard feature data shows equally positive results for both groups. There is no significant difference in the results.

5.3.2 RP2: *Does social navigation through activity streams lead users to more often keep up-to-date on favorites' activities on Urørt?*

Our H_0 stated that we would not see any increase in how frequent experiment respondents kept up-to-date on favorites' activities after giving them an activity stream. The alternative H_A contradicted this and said an activity stream would increase the frequency of how often experiment respondents were keeping up-to-date.

Keeping up-to-date frequency

Based on the question "How often do you update yourself on what your favorites on Urørt are doing?" we graded respondents answers as follows:

- Daily: 5
- Several times a week: 4
- Weekly: 3
- Monthly: 2
- Seldom/never: 1

First we compared how frequent respondents having used an activity stream kept up-to-date with those which hadn't used an activity stream. The results are displayed in Table 5.11.

		N	Mdn	Post − Pre	$Mdn_{\sum E} - Mdn_{\sum C}$	Rng	T	Z	p (1-tailed)
E	Pre	13	1	1	1	2	10	-1.941	0.046
	Post	13	2			2			
C	Pre	11	3	0		2	8	-1.134	0.227
	Post	11	3			2			

Table 5.12: Frequency of keeping up-to-date on favorites' activities. Comparison between pretest and posttest within the experiment and control group.

	N	Mdn	$Mdn_{\sum E} - Mdn_{\sum C}$	Rng	U	Z	p (1-tailed)
E	12	3	1	2	40.0	-1.771	0.056
C	11	2		2			

Table 5.13: Frequency of using "Latest from your Favorites". Comparison between experiment and control group for the posttest.

As can be seen from the data, respondents which had used an activity stream reported lower frequencies of use than those which had used a placebo. This difference is highly significant.

As for our previous two research questions and hypotheses we've also checked if there were changes in the frequency of keeping up-to-date within the experiment and control groups from before they were given a treatment or placebo, to after. The results can be seen in Table 5.12.

The results show a different picture than the between group posttest results. Here we see that the frequency of keeping up-to-date have increased over time for those respondents which used an activity stream. Those respondents with a placebo show a stagnation in the frequency of keeping up-to-date. The increase for experiment respondents is statistically significant.

Prototype usage frequency

We did also collect information in the posttest of how frequent respondents had used the prototype. Based on the question:

- How frequently have you used "Latest from your Favorites" when you are signed-in on Urørt?

we rated respondents answers as follows:

- Have not used: 1
- Only a few times: 2
- almost every time: 3
- every time: 4

	N	Mdn	$Mdn_{\sum E} - Mdn_{\sum C}$	Rng	U	Z	p (1-tailed)
E	13	8	0	49	68.5	-0.174	0.438
C	11	8		38			

Table 5.14: Number of favorites. Comparison between experiment and control group for the posttest.

		N	Mdn	Post − Pre	$Mdn_{\sum E} - Mdn_{\sum C}$	Rng	T	Z	p (1-tailed)
E	Pre	13	8	0	-2	50	39.0	0	0.515
	Post	13	8			49			
C	Pre	11	6	2		35	22.0	-0.059	0.5
	Post	11	8			38			

Table 5.15: Number of favorites. Comparison between pretest and posttest within the experiment and control group.

Table 5.13 lists an comparison between the experiment and control group for this question and ratings.

The data shows that the experiment respondents used the prototype more frequently when they were logged in to Urørt than the control respondents. This difference is however not statistically significant.

5.3.3 RP3: *Does social navigation through activity streams lead users to make more artists on Urørt their favorites?*

Our H_0 hypothesized that the amount of favorites would be greater for the experiment group (due to their activity stream usage) in comparison to the control group. First we looked at the differences between the experiment and control group, as can be seen in Table 5.14.

The data shows no notable nor significant difference between the amount of favorites for the two groups. The range of favorites are greater for the experiment group, but their medians are equal.

We also looked at differences in the amount of favorites within both the control and experiment group from the pretest to the posttest. The data is summarized in Table 5.15.

The within group data shows that there have been minimal development in the amount of favorites for the experiment group after the activity stream was used. The control group have seen an increase in the median amount of favorites. This change is far from statistically significant.

N	%	
26	68.4	Yes, it was an easy and quick process
3	7.9	Yes, but I experienced small problems
0	0.0	Yes, but I experienced large problems
9	23.7	No – I gave up

Table 5.16: Successful and non-successful installation of the prototype.

	Successful (%)	Failed (%)
Follow-up questions	76.3	23.7
Actual non-accomplishment rates	63.4	36.6

Table 5.17: Drop off for installation of the prototype.

5.3.4 *RP4: Can prototyping with Greasemonkey be considered a viable technical option when testing user behavior in an established web site?*

We collected responses about how the prototype installation process went in a separate survey, in between the pretest and posttest. The results can be seen in Table 5.16.

The data shows that 26 + 3 = 29 of 38 respondents (76.3 %) to this particular survey managed to install the prototype.

Another data source for how respondents fared when trying to install our prototype are the non-accomplish rates found in Figure 5.2 (p. 75). This data shows that of the 71 ($E_1 + C_1$) respondents which bothered trying installing the prototype, only 45 ($E_2 + C_2$) managed to do so. This equates to 63.4% of the participants. Table 5.17 shows a comparison of the actual non-accomplishment rates and the follow-up survey.

5.4 DISCUSSION

This section will discuss our various research questions in relation to the results we've presented. We'll start with looking at several aspects of activity streams as a social navigation mechanism before we discuss prototyping with Greasemonkey on an established web site.

As stated earlier, we worked with a level of significance α of $p \leq 0.05$.[1] **?**, p. 1277 argues that "surely, God loves the .06 nearly as much as the .05". We are therefore going to discuss results with values of p approaching α in a flexible fashion in this section.

1. See § 5.2.4 (p. 74) for details.

Based on our profiling of the respondents[2] we found the two groups which were using our implementation to be representative of the general sample in all aspects, except in how frequent they used Urørt. The control group used Urørt more frequent than than both the general sample and the experiment group with a probability of 0.055 compared to the general sample. We argue that this makes the control group more

2. See Table 5.3 (p. 76) for details.

experienced Urørt users than the experiment group. This is an important aspect which we'll try to keep in mind in the discussion about activity streams which follows.

5.4.1 RP1: *Do users perceive social navigation through activity streams as helpful in order to keep up-to-date on favorites' activities on Urørt?*

We hypothesized that usage of activity streams would improve the level of which participants could keep up-to-date on what their favorites on Urørt were doing.

Activities in general

Comparing how easy respondents felt it was to keep up-to-date on activities within the experiment and control group,[3] we observed a higher increase in agreement for the experiment group. The increase of agreement for the control group could be explained as a placebo effect. Since the probability of making a type I error are $p = 0.086$, we simply take this result as an indication and not as hard proof for validating our alternative hypothesis.

3. See Table 5.5 (p. 78) for details.

In addition the comparison within groups, we also compared the differences between the two groups. We observed similar positive results for activity streams with p approaching α with 0.089. We take the higher increase of agreement for the experiment group as an indication of the appropriateness of an activity stream for keeping up to date on activities. Due to lack of significant evidence we can not reject $H1_0$ regarding keeping up-to-date on activities in general.

Specific activities

When asking participants to qualify more specific statements of how easy they felt they could keep up-to-date on different types of activities, we observed differences between[4] the groups and within the groups[5]. The between group data showed no significant nor borderline significant differences between groups. This contradicts the borderline significant results we found with the same comparison on activities in general.

4. See Table 5.6 (p. 79) for details.

5. See Table 5.7 (p. 79) for details.

When comparing within groups, the only notable and significant difference appeared for activities relating to publishing new blog posts. Based on this evidence we reject $H1_0$ in favor of $H1_A$ for specifically keeping up-to-date on blog posts.

We did find significant evidence of a an increase in ease of keeping up-to-date on songs for respondents which used an activity stream while those without had insignificant increases. The difference of significance were quite small, so H_0 stands accepted for songs.

We found no significant differences while using an activity stream for both concerts and reviews. H_0 therefore stands accepted for concerts and reviews.

Why do users of activity streams so strongly feel that the stream help them to better keep up-to-date on recent blog posts and not the other types of activities? One possible explanation could be that the activity stream shows an excerpt of blog posts after the blog author and title.[6] This means that the user can get a glimpse into the content of the blog post without actually navigating to the post itself. But reviews from other users are also displayed with an excerpt of its content. We therefore find this explanation to be highly suggestive. It might be that respondents from the experiment group answered with such a strong tendency towards the usefulness of activity streams with regards to blog posts by chance.

6. The activity stream with excerpts can be seen in Figure 4.5 (p. 51).

Our data shows no indicators as why we experienced contradicting results when asking how easy users could keep up-to date on activities in general compared to specific activities. A possible explanation could be that respondents only formed a general impression of the usefulness of the activity stream due to the short period respondents had access to the feature. They were possibly unable to use all parts of the activity stream and have therefore no specific meanings about these.

We note this as a potentially useful lesson when conducting questions about the ease of conducting a task. In our case, asking more generally gave larger differences than asking more specific questions.

Perceived usefulness

We tried to characterize in what way activity streams could be better than no such feature for keeping up-to-date on favorites by asking respondents to gauge statements of perceived usefulness.[7] The results of asking in this manner yielded no noticeable nor significant differences between experiment and control respondents. Similarly to asking for specific activities, asking for specific qualities of keeping up-to-date seems to result in only minor differences between groups. The $H1_0$ can not be rejected regarding perceived usefulness of an activity stream as a means to keeping up-to-date on favorites.

7. See Table 5.8 (p. 81) for details.

Perceived ease of use

Our investigation into the perceived ease of use[8] showed only minor differences between experiment and control groups. On the issue of how flexible the prototype was, respondents from the control group actually reported higher acceptance to our statements. In light of this evidence $H1_0$ can not be rejected regarding perceived ease of use for an activity stream as a means to keeping up-to-date on favorites.

8. See Table 5.9 (p. 82) for details.

We think the reason for this lies in the nature of asking respondents about ease of use without mentioning for what task the prototype should be easy to use. Our placebo prototype had fewer features (lacking activity streams) and was more simple than the experiment prototype with an activity stream. Surely the simplest application would be easier to use as there are less information and less navigational possibilities available. The statements of perceived ease of use investigates the overall ease of using the application, not how easy it to use for keeping up-to-date on activities. They measure technological acceptance and not actual acceptance of the application to perform a particular task.

Activity stream as standard feature

Relating to perceived ease of use and technological acceptance is our question about whether respondents wanted the prototype to be a standard feature on Urørt. The results[9] indicated a dead race between the prototype with an activity feed and the prototype without. While this data does not show a higher liking of activity feeds, it's interesting to note how high the acceptance as a standard feature are from our respondents. Even the placebo – composed of only a list of a user's favorites – seems to be so useful for respondents that they want it included together with Urørt's standard features.

9. Table 5.10 (p. 83) lists the results.

Activity streams for keeping up-to-date on favorites

Our data have showed inconclusive results for whether activity streams help users in keeping up-to-date on activities. There seem to be an indication of the usefulness of activity streams in this regard. Activity streams clearly makes the task of keeping up with blog posts on Urørt easier.

We've seen how one asks respondents questions can make a difference in the results one are able to obtain. In our case, more specific questions yielded answers further from where we believed them to be based on our hypotheses. A more general question resulted in data more in line with our expectations.

5.4.2 *RP2: Does social navigation through activity streams lead users to more often keep up-to-date on favorites' activities on Urørt?*

We hypothesized that usage of an activity stream would result in higher frequencies of keeping up-to-date on favorites' activities on Urørt.

Keeping up-to-date frequency

As we indicated in the beginning of our discussion the control respondents seemed more experienced with using Urørt. This is evident in how frequent they keep up to date compared to experiment respondents for the posttest.[10] The control group's more frequent action of keeping up-to-date is highly significant. When we compared the shift in the frequency of keeping up-to-date from the pretest to the posttest[11] we noticed, that the control group's frequency is practically unchanged. Interestingly the experiment respondents frequency of keeping up-to-date have increased significantly over the same period. This is a good example of how our pretest and posttest experiment design have enabled us to look at change over time within groups, without jumping to inconclusive inferences by looking only at the state after the treatment or placebo was introduced.

10. See Table 5.11 (p. 83) for details.

11. See Table 5.12 (p. 84) for details.

In light of the significant within group increases for keeping up-to-date for experiment respondents compared to control respondents, we reject our $H2_0$ in favor of the $H2_A$.

Prototype usage frequency

Our data concerning actual usage frequencies of the prototype[12] supports the increase of keeping up-to-date for experiment respondents. The increase in usage approaches α with $p = 0.056$.

12. See Table 5.13 (p. 84) for details.

This data is however different than our findings of how often the same groups kept up-to-date on favorites' activities. The prototype usage data showed that the experiment group had a higher usage rate of "Latest from your Favorites" than the control respondents. The results approaches α with $p = 0.056$.

Does this mean that these two data sources contradict each other? Not necessarily. The first frequency of use statistics shows how often the two groups kept up-to-date on activities while the second shows how often the two groups used our prototype. We believe the lower control group usage of the prototype is related to the lower usefulness of the placebo implementation.

Why do the control group then report higher frequencies for keeping up-to-date on activities? There could be several reasons, but we believe this could indicate that the control group are keeping up-to-date on favorites' activities with other means than the placebo prototype implementation alone. This could then indicate that the prototype without an activity stream is less useful for keeping up-to-date on activities.

Keeping up-to-date more often with activity streams

To summarize, we believe that an activity stream makes respondents more often keep up-to-date on activities than without. Those having used such a tool reported larger changes over time in how frequent they conducted such tasks than those which did not. When one takes into account how often the prototype implementation was used, it seems like the prototype without an activity stream is less suitable for keeping up-to-date than that which implements such a feature. Respondents without an activity stream would then need to keep up-to-date on activities with other means than solely relying on the prototype.

5.4.3 RP3: *Does social navigation through activity streams lead users to make more artists on Urørt their favorites?*

We hypothesized that usage of activity streams on Urørt would make respondents add more artists to their list of favorites.

As our data[13] showed, we found no evidence whatsoever for our claims of increases in number of favorites after having used an activity stream. We believed that the increased focus on favorites and their activities through an activity stream would lead users to make more artists their favorites. Based on the data we've provided, the H_0 for the number of favorites related to an activity stream can not be rejected.

13. See Table 5.13 (p. 84) for details.

One reason for this mismatch between our hypotheses and the evidence could be that there is no explanation of the benefits of adding favorites on Urørt. When a user is browsing the profile page of an artist there is no statement along the lines of: "become a favorite of this artist and get automatically updated on their latest activities". Such a reminder of one benefit of adding artists as favorites could possibly lead to more favoring.

Another reason for this discrepancy could be the short time our prototype was in use by experiment respondents. Just over a week could be too small a time window for measuring changes in how many favorites respondents were adding.

5.4.4 RP4: *Can prototyping with Greasemonkey be considered a viable technical option when testing user behavior in an established web site?*

When we conducted the experiment of our prototype with real world users we got valuable feedback on how well such a technical solution works. Since Greasemonkey enabled client with a dedicated server back-end is (to our knowledge) a new way of experimenting with social navigation, we did not make any hypotheses about the failure or success of the solution. We consider this early work on such prototypes where a few lessons were learnt.

Limitations in browser selection

As described in § B.1.1 (p. 122) there exists Greasemonkey-like implementations for all major browsers. Greasemonkey for Firefox is the only implementation that supports sending requests to other domains than the domain a user-script is running under. This means that if one have to use a server back-end to handle the heavy lifting (as we did), one are limited to using Firefox as a browser platform.[14]

14. Unless one have the opportunity to run the server software under the domain of the web site one are prototyping. We had no such luxury.

Targeting only Firefox as a platform means that one are unable to reach all potential users. The amount of Firefox users lies somewhere between one tenth and one quarter of all web citizens.[15] Installing a new web browser to take part in an experiment is a lot to expect from end-users. One therefore have to take into account the limited outreach Firefox have when deciding on a Greasemonkey based prototype.

15. The amount of users which use Firefox can vary between different populations. According to **?** 13.8% of users world wide used Firefox in February 2008. The market share for Firefox in Europe lied on 19.7% in February 2008 according to **?**. **?** reports the usage to be 28.8% for Europe in March 2008 while 20.3% for Norway (our candidate country) for the same period.

We specifically asked for users of Firefox when we contacted 789 potential respondents. 171 responded to our request, while 123 completed our pretest survey.[16] Those who completed the pretest survey reported a median usage frequency of Firefox as "always".[17] This means that from 789 people, 171 (22%) bothered to take our survey and was likely a Firefox user. This means that either almost 100% of all potential respondents with a Firefox web browser decided to answer our survey (highly unlikely) or Firefox usage for our sample of Urørt users were above the norm for Norway.

16. See Figure 5.2 (p. 75) for details.

17. See Table 5.3 (p. 76) for details.

Difficulties with installing Greasemonkey and user-scripts

Installing our prototype software consisted of two steps:

1 Installing the Greasemonkey Firefox extension.
2 Installing our prototype user-script for Greasemonkey.

The first step in the installation process consisted of (i) navigating to the Greasemonkey installation page, (ii) clicking an installation button on the page, (iii) clicking on another installation button in an installation dialog, and (iv) restarting the Firefox browser. The second step in the installation process consisted of (i) filling out an email address,[18] (ii) pressing an install hyperlink on the user-script installation page, and (iii) clicking on an installation button in a user-script installation dialog. It's evident that this multi-step process can be a bit complicated for the average user. We tried to make the separate steps as seamless as possible by providing screen dumps with descriptions for all parts of the process.

18. We needed the respondents' email addresses for associating answers to the pretest with the group each respondent was randomly distributed to if they decided to install our prototype.

Our non-accomplish rates showed a drop off rate of 36.6% while our follow-up survey showed a drop off rate of 23.7%.[19] This discrepancy probably shows that frustrated respondents which did not manage to install the software were less inclined to take a follow-up survey.

19. See Table 5.17 (p. 86) for details.

These are quite high rates of unsuccessful installations and one should keep this in mind when one designs an experiment where respondents have to install Greasemonkey and user-scripts themselves. To solve this problem one could design an experiment where participants were invited to a lab (where all software were pre-installed). A lab study have its shortcomings too, as we've described in § 5.2.1 (p. 67).

During our development of a prototype application with Greasemonkey for enhancing an established web page we also got a feel for its pros and cons from a development perspective.

Unobtrusive for the established implementation

Prototyping with Greasemonkey is unobtrusive for both the creators of a web site and its users. One are only manipulating pages on an already existing web site when having explicitly installed Greasemonkey and an accompanying user-script. This means that one can conduct experiments by altering a web site without contacting its authors, nor having to inform the web site's existing users. One are in other words only altering content on the client side – in the browser. Normal users are presented with the web site as served by its the web server.

We had contact with the creators of Urørt during our prototype implementation phase. But the prototype could just as well be implemented without having contact with the creators.[20]

20. Easy access to experiment participants was the main benefit we had with keeping in contact with the creators of Urørt.

The creators of Urørt benefited from a Greasemonkey approach since they could keep going on with their work without worrying about potential breakage caused by our prototype. Since we did not access their implementation we could not make any harm on the product that was delivered to non-experiment users.

Requires little knowledge of the established implementation

If we had been permitted access to the Urørt implementation we imagine there would be a large upfront investment in learning how the Urørt architecture worked before we could start conducting any implementation work. With a Greasemonkey approach, there was some learning which had to be completed before we could dive in to implementation work, but we believe this to be less than what would be required to change the Urørt implementation.

Could require more work than altering the established implementation

In our work with creating a navigational prototype we had to investigate how Urørt worked through its outward facing interface. This was a somewhat convoluted experience. We believe it would be much easier to change the Urørt implementation directly had we been well acquainted with its inner workings. If one knows the underlying implementation of an established web site it's probably easier to change that directly.

Fragile when the established implementation is changed

A Greasemonkey based prototype have to hook in to the structure of the established web site one are prototyping. The prototype can cease to function properly if this underlying structure changes. The amount of harm such a change imposes depends on what parts of a client-server Greasemonkey prototype it hits:

- *Server side:* If changes to the established web page breaks functionality on the server side, the prototype would be unusable while the changes are compensated for in the server side implementation. The experiment would probably not be jeopardized if one are able to sort out such problems in a timely manner.

- *Client side:* If changes to the underlying web site hinder the client side Greasemonkey user-script to hook in to the web site and change its contents, one could be in a world of pain. Regardless of how easily the breakage can be fixed, one would need to publish a new user-script version and get experiment users to install this. Needless to say, this could have major ramifications on how well the experiment goes.

When we were developing our prototype, before the experiment was conducted, the developers of Urørt imposed a change which made our client side user-script malfunction. The fix was easy and no harm were done since the user-script had not been deployed to experiment participants.

We did not experience any changes which imposed bugs in our server side prototype component. Nevertheless, we were aware of the possibility of such changes and tried to keep updated on how our prototype functioned at all times. Since we used automated testing in our development (see § 4.3.2 (p. 53) for details) we could verify our server side implementation at all times.

5.5 GENERALIZABILITY AND VALIDITY

There is a few factors which have to be taken into consideration when reading the conclusions of our empirical study which we've recently discussed.

5.5.1 *Scale of experiment*

As presented in § 5.3 (p. 74) we saw fairly high non-accomplishment rates in our study. This meant that we only had 25 respondents to our posttest. This number were four times lower than what we had expected.[21]

21. See § 5.2.2 (p. 68) for details about our participation expectations.

5.5.2 *Selection of subjects*

We were only concerned with respondents which were users of the Firefox web browser due to the technical solution we had selected for our prototype. There is a strong possibility for Firefox users not being representative for the Urørt populace. Installing a custom web browser[22] requires some technical knowledge. One could therefore argue that Firefox users are more technical on average compared to people which use their operating system's default browser.

22. Unless one are using an operating system like GNU/Linux where often Firefox is the default browser.

5.5.3 *Technical seeding*

In addition to recruiting only Firefox users we may have recruited the more technical respondents due to our complicated installation process. We have reason to believe that the more technical respondents had a better chance to understand and successfully complete our prototype installation process. This means that the respondents which took our posttest have a strong possibility of being more technical apt than those respondents which only completed the pretest.

5.5.4 *Motive for participation*

A fairly technical installation process and two surveys can take an considerable amount of time to complete. We have reason to believe that only the most active Urørt users bothered to partake in our experiment. Users who often visit Urørt are naturally more experienced in how Urørt works and are therefore possibly not representative for the Urørt populace.

5.5.5 *Implications*

The factors we've listed are all limitations of our study which could interfere with the validity of our findings. In light of these limitations we can not generalize our results to the general Urørt populace. The work we've provided is early research on activity streams and prototyping with Greasemonkey, and should be considered as preliminary indications of the usefulness of such solutions.

PART III

SUMMARY

CONCLUSION

6

The results of the research which we have provided in this thesis can be categorized into three venues:

1 We have given a structured overview of the field of social navigation as seen both in academic literature and in some noticeable social web sites.
2 We have provided details of how one can implement unobtrusive prototypes in established spaces and the feasibility of such a technical approach.
3 We have contributed knowledge of how activity streams functions as a social navigation technique on the Urørt web site.

Based on these three venues we'll provide the most important lessons to take away from our research before we discuss possible future work in these fields.

6.1 LESSONS LEARNT

We believe there are both some theoretical and practical lessons to take away from our research.

6.1.1 Social navigation

Social navigation can mean different things as viewed in academic literature. We proposed a new definition of social navigation based on our belief in the importance of peers in a social navigation system.[1] This means that the information given from other people which guide navigation have to come from peers within the system where navigation are conducted to be considered social navigation.

1. The definition can be found in § 3.3.3 (p. 42).

Information given by the creators or editors of a web site when used for navigational purposes are therefore not social navigation. The creators can however implement structures in their web pages where users of the system can impose information which can be used for social navigation. One example of this divide can be found in recommender systems. Content based recommendations is not social navigation since the information used in the navigational process are given by the editors of the web page. Recommendations given by collaborative filtering is on the other hand social navigation since the navigational information is given by peers in the system.

6.1.2 *Unobtrusive prototyping*

Creating unobtrusive prototypes with Greasemonkey have its advantages and disadvantages when used in real world experiments.

Greasemonkey is best fitted for situations where one don't have access to the established web site one are prototyping on. If one have access to the inner workings of a web site, it would probably be more efficient and easier to implement the prototype within the established implementation. Another benefit of modifying the web site implementation itself is the elimination of the Greasemonkey and user-script installation process, in addition to wider browser support.

We found the major disadvantage of using Greasemonkey in a real world experimental setting to be this complicated installation process and limited browser support. We contribute this as the major factors for the high non-accomplish rates we witnessed. Having conducted experiments in a laboratory setting where users used pre-configured web browsers would have mediated this problem.

6.1.3 *Activity streams*

Based on our experiment with activity streams on Urørt we have provided inconclusive findings of the success of such a social navigation technique. We take our results as indications of the usefulness of activity streams. Further research is needed to abandon the idea or recommend its usage. Since we did not find any noticeable negative results towards activity streams we can recommend implementations or prototypes of this feature in web sites with similar dynamics to Urørt.

6.2 FUTURE WORK

Our new definition of social navigation in light of the essentialness of peers were based on cursory observations and more detailed analysis of two social web sites. We regard our findings of how social navigation is used in social web sites as early work in this area which needs to be expanded on. More widespread collection and analysis of social navigation in modern web sites is needed to see if our observations holds true.

We've described social navigation as a disparate field. We hope our work to some extent can remedy this problem. One venue for further work to make social navigation a better understood term would be to create a taxonomy of social navigation types. A set of design patterns for when, where, how, and why these various types of social navigation should be used could accompanying such a taxonomy.

In light of the problems we experienced with adopting sufficient number of experiment participants we see the need for a laboratory study of activity streams. Had time permitted in our master thesis work its quite plausible that we had conducted an in-lab experiment where factors as technological ability of the respondents would not interfere with the generalizability of our study.

There should also be studies conducted of the use of activity streams over a longer period than 11 days as for our study. Its possible that activity streams become more useful (or perhaps annoying and intrusive) after prolonged use. We would also like to see what effects longer usage periods could have on the importance of favorites on the Urørt web site.

APPENDICES

CONTENT INVENTORY

A

A.1 FLICKR

The following content inventory detailed in Table A.2 represents the state of the Flickr web site as of the 20th of September 2007. The information we've collected could very well have changed since then as web sites like these are known to have a rapid development cycle where changes often are imposed on the user base at quite a frequent rate.

As detailed in § 3.1.1 (p. 32) we introduced variables for abstracting similar page. A complete listing of such variables for Flickr and their meaning can be found in Table A.1.

When starting the content inventory of Flickr it became apparent that the architects had chosen to use a RESTful approach for their URL scheme.[1] The sites' structure was clearly illustrated by the hierarchy of directories represented in the URL. As noted in§ 3.1 (p. 31) we took note of the URLS as an aid during our inventory phase, but decided not to display them in this thesis.

1. A system that adheres to the principles of REST (Representational State Transfer) (?, p. 76) are sometimes called RESTful. Such principles can be applied to URLS making them represent resources (?, p. 110).

Variable	Description
`$user`	Unique nick-name for a user
`$photo-id`	Unique numerical identifier for a photo
`$photo-title`	Textual title of a photo
`$set-id`	Unique numerical identifier for a set (of photos)
`$set-title`	Textual title of a set (of photos)
`$tag`	Unique name for a tag
`$group`	Unique textual name for a group
`$camera-make`	Manufacturer of digital cameras
`$camera-model`	Model number of a particular digital camera
`$date`	A given date (year, optional month, and optional day)
`$topic-id`	Unique numerical identifier for a discussion topic
`$topic-title`	Textual title of a discussion topic
`$member-count`	A variable number of members of a group
`$license-type`	One of several different Creative Commons licenses

Table A.1: Variables used in Flickr inventory

Table A.2: Content Inventory of Flickr

Id	Page Title	Link Name	Link Location
0	Welcome Page		
1	Photos from `$user`	You	Global navigation
1.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
1.1.1	Photos from `$user`	`$user`	Content (comments list)
1.1.2	Photoset: `$set-title`	`$set-title` (Set)	Right sidebar
1.1.3	`$group`'s pool	`$group` (Pool)	Right sidebar
1.1.4	`$user`'s photos tagged with `$tag`	`$tag`	Right sidebar (tag list)
1.1.4.1	All photos tagged with `$tag`	public photos tagged with `$tag`	Left sidebar
1.1.5	`$user`'s geotagged photos on a Map	View `$user` map	Right sidebar (details list)
1.1.6	Everyone's geotagged photos on a Map	see more photos here	Right sidebar (details list)
1.1.7	Camera finder: `$camera-model`	`$camera-model`	Right sidebar (detail list)
1.1.8	Archive of `$user`'s photos taken on `$date`	`$camera-model`	Right sidebar (detail list)
1.2	Photoset: `$set-title`	`$set-title`	Left sidebar
1.3	`$user`'s photosets	Sets	Local navigation
1.3.1	Photoset: `$set-title`	`$set-title`	Content area
1.1.1.1	Photo detail in set: `$photo-title`	Photo thumbnail	Content area
1.4	`$user`'s tags	Tags	Local navigation
1.4.1	`$user`'s photos tagged with `$tag`	`$tag`	Content (tag cloud)
1.4.1.1	All photos tagged with `$tag`	public photos tagged with `$tag`	Left sidebar
1.4.1.2	Photo detail: `$photo-title`	Photo thumbnail	Content area
1.5	`$user`'s geotagged photos on a Map	Map	Local navigation
1.5.1	Photo detail on map: `$photo-title`	Photo count icon	Map
1.5.1.1	`$user`'s photos tagged with `$tag`	`$tag`	In-line dialog
1.5.1.2	Photo detail: `$photo-title`	View photo page	In-line dialog
1.6	Archive of `$user`'s photos on Flickr	Archives	Local navigation

Table A.2: (continued)

Id	Page Title	Link Name	Link Location
1.6.1	Archive of `$user`'s photos taken on `$date`	`$date`	Content area (Taken on)
1.6.1.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
1.6.2	Archive of `$user`'s photos posted on `$date`	`$date`	Content area (Posted on)
1.6.2.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
1.7	`$user`'s favorite photos on Flickr	Favorites	Local navigation
1.7.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
1.8	`$user`'s most popular photos, interestingness	Popular	Local navigation
1.8.1	Photo detail: `$photo-title`	Photo thumbnail or photo title	Content area
1.8.2	`$user`'s most popular photos, views	Views	Sub local navigation
1.8.2.1	Photo detail: `$photo-title`	Photo thumbnail or photo title	Content area
1.8.3	`$user`'s most popular photos, favorites	Favorites	Sub local navigation
1.8.3.1	Photo detail: `$photo-title`	Photo thumbnail or photo title	Content area
1.8.4	`$user`'s most popular photos, comments	Comments	Sub local navigation
1.8.4.1	Photo detail: `$photo-title`	Photo thumbnail or photo title	Content area
1.9	Profile: `$user`	Profile	Local navigation
1.9.1	Photos from `$user`	`$user`	Content (Groups)
1.9.2	Group: `$group`	`$group`	Content (Groups)
2	Organize your photos	Organize	Global navigation
3	Photos from your contacts	Contacts	Global navigation
3.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
3.2	Photos from `$user`	`$user`	Content area
4	Groups	Groups	Global navigation
4.1	Group: `$group`	`$group`	Content
4.1.1	Discussion: `$group`	Discussion	Local navigation
4.1.1.1	Topic: `$topic-title` in `$group`	`$topic-title`	Content (topic list)

Table A.2: (continued)

Id	Page Title	Link Name	Link Location
4.1.1.2	Photos from `$user`	`$user`	Content (topic list)
4.1.2	`$group`'s pool	Pool	Local navigation
4.1.2.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
4.1.2.2	Photos from `$user`	`$user`	Content area
4.1.3	Geotagged photos from `$group`	Pool	Local navigation
4.1.3.1	Photo detail on map: `$photo-title`	Photo count icon	Map
4.1.3.1.1	`$user`'s photos tagged with `$tag`	`$tag`	In-line dialog
4.1.3.1.2	Photo detail: `$photo-title`	View photo page	In-line dialog
4.1.3	Members of `$group`	`$member-count` Members	Local navigation
4.1.3.1	Photos from `$user`	`$user`	Content area
5	Explore	Explore	Global navigation
5.1	Photo detail: `$photo-title`	Photo thumbnail or `$photo-title`	Content (highlighted photo)
5.2	Photos from `$user`	`$user`	Content (highlighted photo)
5.3	Interesting photos from the last 7 days	last 7 days	Content area
5.3.1	Photo detail: `$photo-title`	Photo thumbnail or `$photo-title`	Content area
5.3.2	Photos from `$user`	`$user`	Content area
5.4	Interesting photos from `$date`	`$date`	Content area
5.4.1	Photo detail: `$photo-title`	Photo thumbnail or `$photo-title`	Content area
5.4.2	Photos from `$user`	`$user`	Content area
5.5	Everyone's geotagged photos on a Map	a map of the world	Content area
5.5.1	Photo detail on map: `$photo-title`	Photo count icon	Map
5.5.1.1	`$user`'s photos tagged with `$tag`	`$tag`	In-line dialog
5.5.1.2	Photo detail: `$photo-title`	View photo page	In-line dialog
5.6	Popular Tags	popular tags	Content area
5.6.1	Photos tagged with `$tag`	`$tag`	Content (tag cloud)

Table A.2: (continued)

Id	Page Title	Link Name	Link Location
5.6.1.1	Photos tagged with `$tag`	Most interesting	Left column
5.6.1.1.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
5.6.1.1.2	Photos from `$user`	`$user`	Content area
5.6.1.2	Clusters for `$tag`	`$tag` clusters	Left column
5.6.1.2.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
5.6.1.2.2	Photos from `$user`	`$user`	Content area
5.6.1.2.3	Clusters for `$tag`	`$tag`	Content (cluster list)
5.6.1.2.4	Photos in cluster: `$tag` `$tag` `$tag`	See more of this cluster…	Content area
5.6.1.2.4.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
5.6.1.2.4.2	Photos from `$user`	`$user`	Content area
5.6.1.3	Photo detail: `$photo-title`	Photo thumbnail	Content area
5.6.1.4	Photos from `$user`	`$user`	Content area
5.7	Camera Finder	Camera finder	Content area
5.7.1	Camera finder: `$camera-make`	`$camera-make`	Content area
5.7.1.1	Camera finder: `$camera-make`: `$camera-model`	`$camera-model`	Content area
5.7.1.1.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
5.6.1.1.2	Photos from `$user`	`$user`	Content area
5.8	Photos from everyone	most recent uploads	Content area
5.8.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
5.8.2	Photos from `$user`	`$user`	Content area
5.8.3	Popular Tags	Popular tags	Right sidebar
5.8.4	Creative Commons	Creative Commons	Right sidebar
5.8.4.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
5.8.4.2	Photos from `$user`	`$user`	Content area
5.8.4.3	Photos with Creative Commons `$license-type`	See more	Content (`$license-type`

Table A.2: (continued)

Id	Page Title	Link Name	Link Location
5.8.4.3.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
5.8.4.3.2	Photos from `$user`	`$user`	Content area
5.9	Photos tagged with `$tag`	`$tag`	Content (tag cloud)
5.10	Interesting photos from `$date`	`$date`	Content (A year ago)
5.10.1	Photo detail: `$photo-title`	Photo thumbnail	Content area
5.10.2	Photos from `$user`	`$user` and see more photos	Content area
5.10.3	Profile: `$user`	profile	Content area
5.10.4	Clusters for `$tag`	`$tag`	Content area
5.11	Photo detail: `$photo-title`	Photo thumbnail or `$photo-title`	Content (A year ago)
5.12	Photos from `$user`	`$user`	Content (A year ago)
5.13	Profile: `$user`	profile	Content (A year ago)
5.14	Photoset: `$set-title`	`$set-title`	Content (Sets)
5.14.1	Photo detail in set: `$photo-title`	Photo thumbnail	Content area
5.15	Photos from `$user`	`$user`	Content (Sets)
5.16	Groups	loads of groups	Content (Groups)
5.17	Group: `$group`	`$group`	Content (Groups)
5.18	`$group`'s pool	Pool	Local navigation
5.19	Members of `$group`	`$member-count` Members	Local navigation
6	Photo detail: `$photo-title`	Photo thumbnail	Content area
7	Photos from `$user`	`$user`	Content area

A.2 FACEBOOK

The inventory of the Facebook web site represents the state it was in the 14th of May 2008. We're also using variables for our Facebook inventory and their representations can be found in Table A.3.

Note that we've described the users of Facebook as people. At Flickr one can have an online, almost anonymous, nickname. On Facebook on the other hand one have to provide a real name[2] and the account have to represent an existing individual.[3]

2. From the Facebook help pages: "Facebook disallows certain names and words in names that tend to be associated with fake accounts (e.g. Paris Hilton)" (**?**).

3. From the Facebook help pages: "using the name of a group or organization is not permitted, as Facebook accounts are for individual use only" (**?**).

Variable	Description
`$person`	Full name of a person
`$network`	The name of a network
`$group`	The name of a group
`$member-count`	A variable number of members of a group or network
`$photo-count`	A variable number of photos
`$video`	Title of a video
`$video-count`	A variable number of videos
`$posted-count`	A variable number of posted items
`$comment-count`	A variable number of comments on a posted item
`$discussion-topic`	Topic of a discussion topic in a discussion board
`$discussion-count`	A variable number of discussions on a discussion board
`$event`	The name of an event
`$guest-count`	A variable number of events
`$guest-count`	A variable number of guest for an event
`$wall-post-count`	A variable number of posts on a wall
`$classified`	The name of a classified in the marketplace
`$gender`	The sex of a person: male, female, or unspecified
`$relationship`	Status of a person's relationship: Single, in a relationship, engaged, married, it's complicated, or in an open relationship.
`$gender-interest`	What gender a person is interested in: men or women.
`$looking-for`	What a person is looking for from other people: friendship, dating, a relationship, or networking.
`$birth-date`	The date of a person's birth
`$birth-year`	The year of a person's birth
`$home-town`	The town a person calls home
`$home-country`	The country a person calls home
`$political-view`	A person's political view: very liberal, liberal moderate, conservative, very conservative, apathetic, libertarian, or other.
`$religious-view`	A person's religious view.
`$album`	Title of a photo album.
`$album-count`	A variable number of photo albums.
`$page`	Title of a fan page.
`$fan-count`	A variable number of fans of a fan page.

Table A.3: Variables used in Facebook inventory

Table A.4: Content Inventory of Flickr

Id	Page Title	Link Name	Link Location
0	Home	Login form	Login page
1	$person	Profile	Global navigation
1.1	My networks	$network	Person info box
1.1.1	Browse My Networks	$member-count	Network info box
1.1.1.1	$person	Profile picture	Member list
1.1.1.2	$person	$person	Member list
1.1.1.3	$network	$network	Member list
1.1.1.4	$network	Send message	Member list
1.1.1.5	$person's Friends	View Friends	Member list
1.1.1.5.1	$person	Profile picture	Member list
1.1.1.5.2	$person	$person	Member list
1.1.2	Networks on Facebook	Browse Networks	Network info box
1.1.2.1	$network	$network	Network List
1.1.3	Popular Today (Posted Items)	See What's Popular	Network info box
1.1.3.1	Popular Today (Groups)	Groups	Global content area navigation
1.1.3.1.1	$group	Group picture	Group list
1.1.3.1.1.1	Photos from $group	$photo-count	Photo box
1.1.3.1.1.1.1	Photos from $group (View single)	Photo thumbnail	Photos list
1.1.3.1.1.1.1.1	Photos from $group (View single)	Previous	Photo navigation
1.1.3.1.1.1.1.2	Photos from $group (View single)	Next	Photo navigation
1.1.3.1.1.1.1.3	$person	$person	Tagged people in photo
1.1.3.1.1.1.1.4	$person	Added by $person	Photo meta-data
1.1.3.1.1.1.1.5	$person	Profile picture	Comment
1.1.3.1.1.1.1.6	$person	$person	Comment
1.1.3.1.1.2	Photos from $group	See All	Photos box

Table A.4: (continued)

Id	Page Title	Link Name	Link Location
1.1.3.1.1.3	Videos from $group	$video-count	Videos box
1.1.3.1.1.3.1	Videos from $group $video	Video thumbnail	Video list
1.1.3.1.1.3.1.1	Videos from $group $video	Previous	Video navigation
1.1.3.1.1.3.1.2	Videos from $group $video	Next	Video navigation
1.1.3.1.1.3.1.3	$person	$person	Tagged people in video
1.1.3.1.1.3.1.4	$person	Added by $person	Video meta-data
1.1.3.1.1.3.1.5	$person	Profile picture	Comment
1.1.3.1.1.3.1.6	$person	$person	Comment
1.1.3.1.1.3.2	Videos from $group $video	$video	Video list
1.1.3.1.1.3.2	$person	by $person	Video list
1.1.3.1.1.4	Videos from $group	See All	Video box
1.1.3.1.1.5	Posted Items on $group	$posted-count	Posted items box
1.1.3.1.1.5.1	Posted Items on $group	$comment-count comments	Posted item
1.1.3.1.1.5.1.1	$person	Profile picture	Comment
1.1.3.1.1.5.1.2	$person	$person	Comment
1.1.3.1.1.6	Posted Items on $group	See All	Posted items list
1.1.3.1.1.7	$group Discussions	$discussion-count discussion topics	Discussions list
1.1.3.1.1.7.1	$discussion-topic	$discussion-topic	Discussions list
1.1.3.1.1.7.1.1	$person	Profile picture	Comment
1.1.3.1.1.7.1.2	$person	$person	Comment
1.1.3.1.1.8	Group Members	$member-count	Member box
1.1.3.1.1.8.1	$person	Profile picture	Member list
1.1.3.1.1.8.2	$person	$person	Member list
1.1.3.1.1.9	$group Wall	$wall-post-count	Wall posts list
1.1.3.1.1.9.1	$person	Profile picture	Wall post

Table A.4: (continued)

Id	Page Title	Link Name	Link Location
1.1.3.1.1.9.2	`$person`	`$person`	Wall post
1.1.3.1.1.10	`$person`	`$person`	Officer list (right sidebar)
1.1.3.1.1.11	`$person`	`$person`	Admin list (right sidebar)
1.1.3.1.2	`$group`	`$group`	Group list
1.1.3.2	Popular Today (Events)	Events	Global content area navigation
1.1.3.2.1	`$event`	Event picture	Event list
1.1.3.2.1.1	Photos from `$event`	`$photo-count`	Photo box
1.1.3.2.1.1.1	Photos from `$event` (View single)	Photo thumbnail	Photos list
1.1.3.2.1.2	Photos from `$event`	See All	Photos box
1.1.3.2.1.3	Videos from `$event`	`$video-count`	Videos box
1.1.3.2.1.3.1	Videos from `$event` `$video`	Video thumbnail	Video list
1.1.3.2.1.3.2	Videos from `$event` `$video`	`$video`	Video list
1.1.3.2.1.3.2	`$person`	by `$person`	Video list
1.1.3.2.1.4	Videos from `$event`	See All	Video box
1.1.3.2.1.5	Posted Items on `$event`	`$posted-count`	Posted items box
1.1.3.2.1.5.1	Posted Items on `$event`	`$comment-count` comments	Posted item
1.1.3.2.1.6	Posted Items on `$event`	See All	Posted items list
1.1.3.2.1.8	Event Guests	`$guest-count`	Guests box
1.1.3.2.1.8.1	`$person`	Profile picture	Member list
1.1.3.2.1.8.2	`$person`	`$person`	Member list
1.1.3.2.1.9	`$event` Wall	`$wall-post-count`	Wall posts list
1.1.3.2.1.9.1	`$person`	Profile picture	Wall post
1.1.3.2.1.9.2	`$person`	`$person`	Wall post
1.1.3.2.1.10	`$person`	`$person`	Other invited list (right sidebar)
1.1.3.2.1.10	`$person`	Profile picture	Other invited list (right sidebar)

Table A.4: (continued)

Id	Page Title	Link Name	Link Location
1.1.3.2.1.11	`$person`	`$person`	Admin list (right sidebar)
1.1.3.2.2	`$event`	`$event`	Event list
1.1.3.3	Popular Today (Notes)	Notes	Global content area navigation
1.1.3.3.1	`$person`'s Notes	`$note`	Note list
1.1.3.3.1.1	`$person`	Profile picture	Note comment
1.1.3.3.1.2	`$person`	`$person`	Note comment
1.1.3.3.2	`$person`	Profile picture	Note list
1.1.3.3.3	`$person`	`$person`	Note list
1.1.4	Discussions	View Discussion Board	Network info box
1.1.3.1.1.8	Group Members	`$member-count`	Member box
1.1.3.1.1.8.1	`$person`	Profile picture	Member list
1.1.3.1.1.8.2	`$person`	`$person`	Member list
1.1.4	Browse My Networks	`$member-count`	People in `$network` box
1.1.5	`$person`	Profile picture	Member list
1.1.6	`$person`	`$person`	People in `$network` box
1.1.7	`$event`	`$event-count`	Upcoming events box
1.1.8	`$event`	`$event`	Upcoming events box
1.1.8	Popular Today (Posted Items)	See All	Network info box
1.1.9	Popular Today (Groups)	See All	Popular in `$network` box
1.1.10	`$group`	`$group`	Popular in `$network` box
1.1.11	Discussions	`$discussion-count` discussion topics	Discussion box
1.1.12	`$discussion-topic`	`$discussion-topic`	Discussion box
1.1.13	`$network`'s Wall	`$wall-post-count`	Wall posts box
1.1.14	`$person`	Profile picture	Wall post
1.1.15	`$person`	`$person`	Wall post

Table A.4: (continued)

Id	Page Title	Link Name	Link Location
1.1.16	Marketplace	See All	Marketplace box
1.1.16.1	Marketplace - `$classified`	`$classified`	Classified listing
1.1.16.1.1	`$person`	`$person`	Classified info
1.1.16.2	`$person`	`$person`	Classified info
1.1.17	Marketplace - `$classified`	`$classified`	Marketplace box
1.2	Browse My networks (by gender)	`$gender`	Person info box
1.2.1	`$person`	Profile picture	Person listing
1.2.2	`$person`	`$person`	Person listing
1.2.3	My networks	`$network`	Person listing
1.3	Browse My networks (by gender interest)	`$gender-interest`	Person info box
1.3.1	`$person`	Profile picture	Person listing
1.3.2	`$person`	`$person`	Person listing
1.3.3	My networks	`$network`	Person listing
1.4	Browse My networks (by relationship status)	`$relationship`	Person info box
1.4.1	`$person`	Profile picture	Person listing
1.4.2	`$person`	`$person`	Person listing
1.4.3	My networks	`$network`	Person listing
1.5	Browse My networks (by looking for)	`$looking-for`	Person info box
1.5.1	`$person`	Profile picture	Person listing
1.5.2	`$person`	`$person`	Person listing
1.5.3	My networks	`$network`	Person listing
1.6	Profile Search Results (by birthday)	`$birth-date`	Person info box
1.6.1	`$person`	Profile picture	Person listing
1.6.2	`$person`	`$person`	Person listing
1.6.3	My networks	`$network`	Person listing

Table A.4: (continued)

Id	Page Title	Link Name	Link Location
1.7	Browse My networks (by birth year)	`$birth-year`	Person info box
1.7.1	`$person`	Profile picture	Person listing
1.7.2	`$person`	`$person`	Person listing
1.7.3	My networks	`$network`	Person listing
1.8	Profile Search Results (by home town)	`$home-town`	Person info box
1.8.1	`$person`	Profile picture	Person listing
1.8.2	`$person`	`$person`	Person listing
1.8.3	My networks	`$network`	Person listing
1.9	Profile Search Results (by home country)	`$home-country`	Person info box
1.9.1	`$person`	Profile picture	Person listing
1.9.2	`$person`	`$person`	Person listing
1.9.3	My networks	`$network`	Person listing
1.10	Browse My networks (by political view)	`$political-view`	Person info box
1.10.1	`$person`	Profile picture	Person listing
1.10.2	`$person`	`$person`	Person listing
1.10.3	My networks	`$network`	Person listing
1.11	Profile Search Results (by religious view)	`$religious-view`	Person info box
1.11.1	`$person`	Profile picture	Person listing
1.11.2	`$person`	`$person`	Person listing
1.11.3	My networks	`$network`	Person listing
1.12	Photos of `$person`	View photos of `$person`	Left sidebar
1.13	`$person`'s Friends	View `$person`'s Friends	Left sidebar sidebar
1.14	`$person`	Profile picture	Friends sidebar box
1.15	`$person`	`$person`	Friends sidebar box
1.16	`$person`	Profile picture	Mutual friends sidebar box

Table A.4: (continued)

Id	Page Title	Link Name	Link Location
1.17	$person	$person	Mutual friends sidebar box
1.18	$person's Friends	View $person's Friends	Friends in other networks sidebar box
1.19	$person's Photos – $album	Album picture	Photos sidebar box
1.19.1	$person's Photos – $album	Photo thumbnail	Photos list
1.20	$person's Photos – $album	$album	Photos sidebar box
1.21	$page	Page picture	Pages sidebar box
1.21.1	Fans of $page	$fan-count	Supporters box
1.21.1.1	$person	Profile picture	Fan list
1.21.1.2	$person	$person	Fan list
1.21.2	Fans of $page	See All	Supporters box
1.21.3	$person	Profile picture	Supporters box
1.21.4	$person	$person	Supporters box
1.21.5	$page's photos	$album-count	Photos box
1.21.6	$page's photos	See All	Photos box
1.21.6.1	$page's Photos – $album	Album picture	Album list
1.21.6.2	$page's Photos – $album	$album	Album list
1.21.6.3	$page's Photos – $album	View Album	Album list
1.21.7	$page's Notes	$note	Notes box
1.21.8	$page's Wall	$wall-post-count	Wall posts box
1.21.9	$page's Wall	See All	Wall posts box
1.21.3	$person	Profile picture	Wall posts box
1.21.4	$person	$person	Wall posts box
1.22	$page	$page	Pages sidebar box
2	$Friends	All Friends	Global navigation
2.1	$person	Profile picture	Friends list

Table A.4: (continued)

Id	Page Title	Link Name	Link Location
2.2	$person	$person	Friends list
3	Photos	Photos	Left sidebar
3.1	$person's Photos – $album	Album picture	Album list
3.1	$person's Photos – $album	$album	Album list
3.3	$person	$person	Album list
4	Groups	Groups	Left sidebar
4.1	$group	Group picture	Group list
4.2	$group	$group	Group list
4.3	Group Members	$member-count	Group list
4.4	$person	$person	Group list
5	Events	Events	Left sidebar
5.1	$event	Event picture	Event list
5.2	$event	$event	Event list
6	$person	$person	News feed
7	$group	$group	News feed
8	$person	Profile picture	News feed
9	$person's Wall-to-Wall with $person	Wall-to-Wall	News feed
10	$event	$event	News feed
11	$person's Photos – $album	photo	News feed
12	$person's Photos – $album	Photo thumbnail	Photos list
13	$person	$person	Status updates, right sidebar
14	$person	$person	Birthdays right, sidebar
15	$person	$person	People you may know, right sidebar

SELECTION OF THIRD PARTY SOFTWARE

B

This appendix will first describe the ingredients in our prototype software stack. Afterwards we'll account for the tools we've used while developing this software stack.

Firstly one important aspect with the software used in our thesis work – from operating system to third party libraries – is that it should only consist of *open source* [1] software. In our experience it's invaluable to have sources available for all involved software. If one encounter abnormal behavior or bugs it's much easier to locate them when one have sources available and one can trivially (depending on the complexity of the problem) create a patch that sorts them out. One of the motivating factors of open source contributors is the opportunity for other users to find and fix failures and provide improvements on their code (**?**, p. 87).

1. *The Open Source Definition* (**?**) dictates the terms software needs to follow to be accepted as open source. Some call open source for *Free Software* and the term *Free/Libre/Open Source Software* have been used to reconcile these different wordings. We're pragmatics like others (**?**, p. 8) and are not going into the political details of these terms and are going to describe such software as open source throughout this thesis.

It's also our experience that one can find third party software that fits one's problem domain more easily if one chooses to use open source software because of the vast availability of such software. **?** found that the availability of open source code and projects had grown exponentially from January 1995 to December 2006.

Lastly it's of importance to keep the act of conducting science open so that future researchers easily can discuss, falsify, and improve on previous research. Software is often an essential part of computer science research and **?**, p. 430 therefore argues that open source is a property to strive for when conducting such research.

B.1 PROTOTYPE SOFTWARE STACK

Based on the architectural decisions made in Chapter 4 (p. 45) we'll now go on to make more fine-grained choices of what specific third party software components to utilize in our prototype application. We've decided to harness some of the seemingly best freely available software components in this software stack. The issue of such code reuse was introduced by **?**, pp. 138–142 when he voiced a need for the software industry to become industrialized. His proposed technique for enabling mass-production of software was to offer components – families of program routines that can be used for any given job. These components should be created in a way so that they can fit together as building blocks.

The developer should be able to treat these components as *black boxes*.[2]

2. In the field of electronics black boxes are used to describe electronic circuits with a fixed set of terminals where one is deliberately ignoring the internals of the circuit. Only the external properties of the circuit given by the electronic properties of its terminals are emphasized (**?**, p. 171). Paralleling with computer science we can think of a component, module, object, or routine (electronic circuit) as being a black box when we're only concerned with its input and output characteristics through its interface (terminals).

After object-oriented programming became the most popular programming paradigm[3] it's become commonplace to offer software components in the form of classes and modules which easily can be integrated into a new software project. As **?**, p. 37 puts it "code reuse is what object-orientation [is] all about in the first place". Such libraries or frameworks ought to provide more efficient development, higher code quality, and easier maintenance (**?**, p. 12). We're therefore leveraging several such freely available software components in our prototype system. We'll first survey system components utilized on the client-side and then those selected for the server-side implementation.

3. According to TIOBE's list of of how popular different programming languages are (based on the number of engineers using them, courses given for them, and third party vendors endorsing them) 65.463 percent of language use were object oriented (**?**). The languages counted towards object orientation were: Java, PHP, C++, Perl, Python, C#, Ruby, Delphi, JavaScript, D, FoxPro, Ada, and ColdFusion.

B.1.1 Client-side

Platform

The platform for the clients is in essence a web browser. We are making changes to a web page (more correctly the DOM of a web page) after all. The web browser have to be explicitly chosen to be one that readily supports scripting existing web pages – a term often called *user scripting*. The Firefox[4] web browser was the first browser providing a plug-in for handling such scripting of web pages and seems to have the most mature implementation in our view. Since Firefox also is the most adopted[5] cross-platform open source web browser the platform choice was quite easy.

4. Firefox is available at http://www.mozilla.com/en-US/firefox.

5. Firefox was the second most used web browser in February 2008 only surpassed by Microsoft's Internet Explorer (**?**).

Firefox provide user scripting through the means of the Greasemonkey[6] browser extension. Essentially all it provides is the ability for a user to install a script which can manipulate the behavior and properties of an existing web page using the *DOM*.[7] When a user have such a script for a specific web page installed its instructions will be executed on the next visit to the given site, enabling all kinds of modifications to the DOM. Some have predicted that Greasemonkey could enable users to finally "take back the Web" – making the decision of how a web page behaves and what information it presents the choice of the user of a web page, not the creator (**?**, pp. 3–4.) Although most often used for customizing the appearance of a given web site (**?**, p. 39), Greasemonkey can also be used for creating new navigational designs on the Urørt web site.

6. Greasemonkey is available at http://www.greasespot.net.

7. The DOM is a three of objects representing the hierarchical structure of nested tags (with text and attributes) in HTML documents (**?**, pp. 307–310).

Although we've settled on the Firefox and Greasemonkey platform there is a certain possibility that our implementation could work in other browsers providing user scripting. The Opera browser provides user scripting without any plugins,[8] the Safari browser can handle user script with the GreaseKit[9] plug-in. Our prototype user script would not work in the Opera browser since it does not allow a `XMLHttpRequest` for another domain than what the user script is currently running in. As described in § 4.4.1 (p. 57) this is an essential feature for our implementation. Recent versions of GreaseKit removed the possibility for such

8. For more information about Opera's user-scripting capabilities see http://www.opera.com/support/tutorials/userjs/examples.

9. GreaseKit for Safari is located at http://8-p.info/greasekit.

kinds of requests to take place. We're therefore left with Greasemonkey for Firefox as our only deployment platform.

Programming language

The ability to programmatically alter behavior inside web browsers was first introduced by Netscape in their 2.0 version of the web browser with the same name. JavaScript was first intended to be a lightweight scripting language for gluing together HTML and applets written in the Java programming language (?).[10] Java applets never took off and JavaScript soon became the *de facto* standard for enabling behavior on the Web and was standardized as ECMAScript in 1997 (?).

10. Sun Microsystems, the creators of Java, had negotiated with Netscape about including it in their second major web browser release. The development of JavaScript, then called Mocha, was already underway and people inside Netscape wondered why one needed two languages. "The answer was that two languages were required to serve the two mostly-disjoint audiences in the programming ziggurat who most deserved dedicated programming languages: the component authors, who wrote in C++ or (we hoped) Java; and the 'scripters', amateur or pro, who would write code directly embedded in HTML" (?).

Because of this we had no say in what programming language to use on the client-side. That is not to say that JavaScript is a poor programming language. Contradictory to its name, JavaScript bears few similarities to the Java language.[11] Despite its origins as a scripting language JavaScript is now considered a full-featured modern programming language (?, p. 2; ?, p. 3) including object-orientation.

11. The name was more of a marketing decision when Netscape teamed up with Sun (?, p. 2).

Convenience library

We decided to use a JavaScript library to make interactions with the DOM simpler. In addition there recent JavaScript convenience libraries provide a unified interface to the browser – abstracting away inconsistencies between browser vendors. Lately a myriad of such frameworks have appeared, but the most interesting ones seems to be Prototype, Yahoo! UI Library (YUI for short), MooTools, MochiKit, and jQuery.[12] There are other frameworks available that provide everything but the kitchen sink but we needed a lightweight or modular solution.

12. Available, in respective order, at http://www.prototypejs.org, http://developer.yahoo.com/yui, http://mootools.net, http://mochikit.com, and http://jquery.com.

As can be seen in Figure B.1 (p. 125) we summarized the size of the most current version for each library of this writing. These are not exact metrics – we selected not to include certain widgets and logging facilities for the modularized libraries – but should provide clear guidance. To keep a level playing feel in this comparison we did not use minified (removal of comments and unnecessary spaces) or packaged (compressed) versions of the libraries. All comments and documentation was stripped with a small script presented in Source Code Listing D.2 (p. 142) since the in-line documentation and commenting varied amongst the libraries.

We played around a bit with the different libraries to get a feel for how they worked. What follows is a comparison of simple DOM manipulation for the different libraries. We followed the official documentation for the various libraries and tried to solve or problem as succinct and clearly as possible. We tried to add a `class` attribute of "highlight" to all `em` elements with an descendant `p` element:

When we compare these rather trivial problem solutions it becomes apparent that choosing a JavaScript library can have major impact on

```
getElementsBySelector("p em").each(function(em) {
  em.addClassName("highlight");
});
```

Source Code Listing B.1: DOM manipulation in JavaScript with the Prototype library

```
var em = YAHOO.util.Selector.query("p em");
YAHOO.util.Dom.setClass(em, "highlight);
```

Source Code Listing B.2: DOM manipulation in JavaScript with the Yahoo! UI library

```
$$("p em").each(function(em){
  em.addClass("highlight");
});
```

Source Code Listing B.3: DOM manipulation in JavaScript with the MooTools library

```
var p = getElementsByTagAndClassName("p");
for (i = 0; i < p.length; i++) {
  em = getElementsByTagAndClassName("em","*", p[i]);
  for (j = 0; j < em.length; j++) {
    addElementClass(em, "highlight");
  }
}
```

Source Code Listing B.4: DOM manipulation in JavaScript with the MochiKit library

```
$("p em").addClass("highlight");
```

Source Code Listing B.5: DOM manipulation in JavaScript with the jQuery library

how easily implemented and understood your code will be. Four of the five libraries have support for selector syntax based on that found in CSS.[13] This is what makes the MochiKit example the most complex one, requiring the developer to do two queries into the DOM and construct two loop structures for iterating over the results. Prototype and MooTools also requires the developer to loop over a single result set, but the iteration is abstracted into an `each` function making the logic a bit more clearer. Yahoo! UI Library's DOM functions works on both single elements and collections of elements – eliminating the need for an explicit loop structure. Notice though that the library from Yahoo! relies heavily on namespacing – which is a good thing for interoperability with other libraries – but can be a bit verbose at times.

13. CSS is short for Cascading Style Sheets – a stylesheet language most commonly used for describing the presentation of HTML documents.

The solution written with jQuery provides even more clarity. Every query into the DOM returns a special jQuery object which means that one can call methods like `addClass` directly on this object regardless if

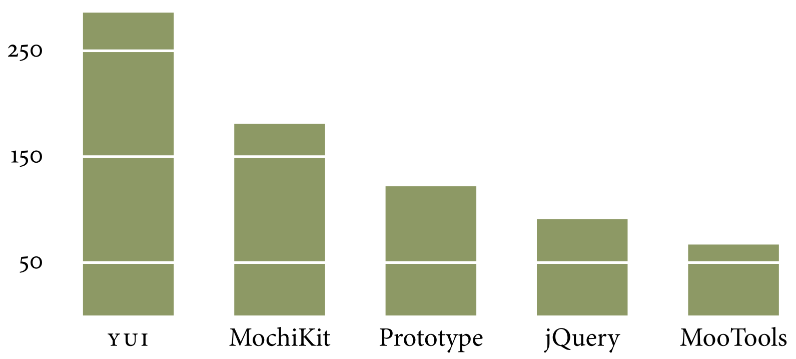


Figure B.1: Comparison of JavaScript library file size, in kB.

the jQuery object holds a single or multiple elements. Also unique to jQuery is the fact that every method call returns a new jQuery object. This means that one can *chain* methods together, expressing succinctly and clearly what you intend to accomplish with your code. We can extend our initial problem and add some punctuation inside our `em` element:

```
$("p em").addClass("highlight").append("!");
```

Source Code Listing B.6: Chaining multiple methods together in jQuery

We decided to select jQuery as the JavaScript library for our implementation. Firstly jQuery have a very minimal file size compared to the largest library ▮▬ we tested. It was only beaten in this regard by MooTools, and the difference ▮▮ was minor. Secondly its unique syntax makes for succinct and clear code which we value highly. It seems others have take jQuery and its virtues to hart as many large corporations like Google, Intel, Dell, and BBC have used it in their public facing offerings.[14]

14. For a complete list see http://docs.jquery.com/Sites_Using_jQuery.

B.1.2 Server side

Platform

Based on the following survey of server side software we needed an UNIX-based operating sytem as the plattform of our server. The particular operating system was pre-selected for us as SINTEF already had a server we could use. This system was running Debian[15] GNU/Linux – a perfect fit for the rest of our server side software stack.

15. Debian GNU/Linux is freely available at http://debian.org.

Programming language

When doing prototype work it's important that the programming language one uses is efficient to work with. This means that programmer efficiency is more important than computational efficiency (a language's native performance). **?** argues that the true measure of a language's productivity is how little code you need for solving a given problem. Since we didn't have time to invest in learning a new language we had to do with those we knew from before. Of those Ruby,[16] Python,[17] and Common Lisp[18] were the ones with language features that fitted our development process. All these languages supports multiple programming-paradigms though Common Lisp is most functional of nature while Python and Ruby are more inclined towards object-orientation.

16. The Ruby language is available at http://ruby-lang.org.

17. The Python language can be found at http://python.org.

18. Common Lisp, the prevalent Lisp dialect today, is a standard (**?**) and has many implementations. A gateway to this language and its many implementations can be found at http://common-lisp.net.

They are all *latent typed*[19] and have quite expressive syntax. This makes for concise source code. **?** argues that the worst thing that can happens to a code base is size which often is the result of code bloat. In addition, both Ruby, Python, and Common Lisp are *interpreted* languages.[20] This means that the programmer don't have to go through a compilation process before he can see the results of his labor. When prototyping rapidly it's quite convenient to make small changes and see the results instantanously.

19. Latent typing "is a style of typing that does not require (or perhaps even offer) explicit type declarations"(**?**).

20. This is only partly true since Common Lisp implementations incrementally compile code and extensions or new implementations for Python and Ruby implements just-in-time compilers. In both cases the developer does not need to explicitly invoke a compile process before using a program, therefore resembling interpreted languages.

Disussion of the virtues of different flavors and implementations of programming languages have been the subject of endless debate. In the end we think it comes down to personal preference and making a pragmatic choice for the tool best suited for the job at hand. If we had to select a programming language based on our list of candidates based on the languages syntax and posibilities in itself we would probably have gone with Common Lisp. **?**, p. 27 have called it "the programmable programming language"based on the fact that program code in Lisp is data and can be manipulated with the same constructs one are using on data. This makes it immensly powerfull and is the reason why it's survived for over 50 years (**?**, p. 217) and been able to adopt new paradigms in programming as they've appered. Even though we walue programming efficiency over computational performance, it should be noted that Common Lisp have been described as the only performant dynamic language (**?**) compared to statically compiled languages.[21]

21. In *The Computer Language Benchmarks Game* (see http://shootout.alioth.debian.org) several programming languages are pitched against each other in several tests to determine their computational performance. As of this writing (April 2, 2008) Common Lisp is 1.8 times slower than the fastest language: C++. Python and Ruby are respectively 18 and 56 times slower than the leader.

As it turns out, the most important criteria for choosing our implementation language was its library support.

? shares this viewpoint::

The programming environment (development/target platforms, intended audience, and most importantly, available libraries) is the primary factor in determining one's choice of programming language.

In the next section we discuss our options of such libraries or frameworks. Based on our findings there we landed on Ruby as the language of our server-side implementation.

Data extraction library

The core library we need is one that handles data extraction from existing web pages, so called HTML *scraping*. While it's possible to handle such problems with regular expressions, this becomes tedious after a while. We therefore prefer a special purpose library.

The major deciding factor when we selected the implementation language was the availability of such a library and its usefulness. We've already revealed Ruby as our implementation language and are therefore killing the suspense. Our data extraction library of choice is called Hpricot[22] and makes HTML parsing a blissful endeavor in our opinion.

22. Hpricot can be obtained from http://code.whytheluckystiff.net/hpricot. A curious note: Hpricot is written by the same person who created Hoodwink.d – our inspiration for a transparent prototype implementation.

The Python alternative for web page scraping is Beautiful Soup. We were not able to find any libraries specially made for HTML scraping implemented in Common Lisp. There exists several *XML*[23] libraries that could handle our tasks, but none as well integrated as the Ruby and Python options.

23. Extensible Markup Language. General purpose markup language specification that enables implementors to create custom markup languages. HTML is not a subset (specified in) XML (?). *XHTML* on the other hand, a reformulated version of HTML, is a subset of XML (?).

To get a feel for the difference between Hpricot and Beautiful Soup we tried them out on some trivial examples. Under you'll see the listings for one of these examples. We are trying to find an `em` element with a class of `citation`, which have a `p` element as its parent, in a HTML document contained in the `html` object:

```
html('p').content.findNextSiblings('em', 'citation')
```

Source Code Listing B.7: HTML parsing in Python with Beautiful Soup

```
  html/'p > em.citation'
```

Source Code Listing B.8: HTML parsing in Ruby with Hpricot

We feel that Hpricot's syntax is much clearer than that of Beautiful Soup. This could be a personal preference since we've used CSS for a long time and Hpricot's selector syntax is based on CSS and Xpath, just as jQuery. Hpricot was in fact initially based on jQuery's selector syntax (?). This means that we can use the same syntax for selectors on the server and client-sidea cognitive advantage.

When we started developing our prototype application we came over what in some way can be seen as a bug of Hpricot. We feel the fault lies with the server-side plattform Urørt uses. Simply put, Microsoft's ASP.NET web application framework uses a hidden HTML input element to maintain the state of HTML forms between stateless HTTP requests. This hidden input element can be quite large[24] in size since it contains a serialized version of the state of the current page's HTML forms. Such large single HTML elements have been described by Microsoft itself as a problem (?). Hpricot sets aside a buffer of 16kB for storing each HTML element. When Hpricot encounters an element with the size we're seeing on Urørt it simply chokes.

24. On the Urørt web site (http://www11.nrk.no/urort/Artist/dividizzIDVD) the hidden input element used for maintaining state weighted in at 122kB!

Since Hpricot is open source software someone had thankfully experienced the same problem and provided a patch to dynamically increase the buffer if an enormous HTML element was encountered. All we had to do for properly using Hpricot on Urørt was to use a version patched with this change instead of using the standard vanilla version.

Data fetching library

Since we've selected Ruby as our development language of choice we used `open-uri`, part of the standard Ruby library, for fetching documents over HTTP. `open-uri` is trvial to use and integrates nicely with Hpricot:

```
require 'hpricot'
require 'open-uri'

html = Hpricot(open('http://redflavor.com'))
(html/'address.vcard > .fn').inner_html
# => "Eivind Uggedal"
```

Source Code Listing B.9: Fetching a HTML document with `open-uri` and parsing it with Hpricot to find the first and last name of a hCard Microformat

JSON library

Since we'll mainly be serving requests for our JavaScript based client implementation we found it sound to transfer this data as JSON.[25] Luckily for us there exists a libary for encoding Ruby objects into JSON format simply called json.[26] The next code listings show how simple objects can be encoded and the resulting JSON format.

25. JSON, short for JavaScript Object Notation, is specified in RFC 4627 (?). Shortly put it's a lightweigh data interchange format based on the object literals of JavaScript.

26. The Ruby JSON can be found at http://json.rubyforge.org.

```
msg = {:interjection => 'hello',
       :noun         => 'world',
       :suffix       => '!'}

msg.to_json
```

Source Code Listing B.10: Encoding a Ruby hash to JSON format

```
{"interjection": "hello",
 "noun":         "world",
 "suffix":       "!"}
```

Source Code Listing B.11: The result of the JSON encoding of a Ruby hash

HTTP *framework*

A web framework or rather HTTP framework is needed to make the generated activity data in JSON format available for our client. In addition we need to provide some traditional HTML pages and take action on input we receive from some of these.

There is numerous frameworks for easing the creation of HTTP applications available for Ruby. The most popular[27] is Ruby on Rails.[28] Other actively developed frameworks include Merb, Camping, Sinatra, and Ramaze.[29] These frameworks have a varying degree of complexity, but none of them is as simple as Rack,[30] a web server interface that sits between a web framework and a web server or can be used as a very light weight web framework on its own. Since we our neeeds were a bit specialized we decided to use Rack for handeling HTTP requests. Because of its simplicity it's very flexible and allowed us to create exactly the HTTP interface we wanted. By using Rack we can also easily swap between different web servers that support the protocoll mandated by Rack's interface layer.

27. A search for Ruby on Rails books on Amazon revealed 35 titles as of 20 May, 2008.

29. Available at http://m com, http://code.whyth net/camping, http://sin com, and http://ramaze. tively.

HTTP *server*

Since our web framework builds on Rack we have several HTTP servers to select from. The most prominent alternatives of Ruby servers are[31]:

- *Webrick* is the original Ruby web server included in all recent versions of the Ruby environment. Because of its simplicity it has been widely used when developingweb applications. Unfortunately it's painstakingly slow compared to the other alternatives.
- *FastCGI* was the de facto way of running Ruby web applications for production systems before Mongrel, the following alternative, was released. It's basically an improvment over the well known CGI model. Contrasted to its ancestor the processes of FastCGI are persistent and are reused when handeling requests (**?**).
- *Mongrel* was introduced as a faster alternative to Webrick. One of the distinguishing factors of the library is that it handles each request in its own thread – enabeling it to handle many concurrent requests. The project is now very mature and can be concidered as the most stable Ruby web server.
- *Evented Mongrel* is a fork of Mongrel which uses an event loop using the Reactor pattern (**?**, p. 529) instead of using threading to handle multiple concurrent requests. The requests are therefore handeled sequentially.
- *Thin* is a fork of Evented Mongrel, promising even better performance than the original Mongrel. It's an sequential server which includes various convenience functions for easily starting, stopping, and managing a wide array of Thin servers.
- *Ebb* is the newest kid on the block and are proving to be faster than its predecessors. It's entrirely written in C and uses C libraries in contrast to

Mongrel and the Mongrel forks which are only partly written in C with Ruby libraries. It supports both sequential and threaded processing of requests. This project is fairly new and it's not recommended to utelize it in production yet.

As we've seen the Ruby web server offerings are many and it can be quite hard to find out the best option. We first and foremost wanted a stable solution with decent performance. This eliminated Ebb in spite of its promising performance characteristics. Webrick and FastCGI is fairly outdated and leaves something to be desired in the performance department. Thin seems to have taken over the space Evented Mongrel used to occupy in the Ruby server world because of its ease of use and improved performance.

This leaves us with two choices: Thin and Mongrel. The most distinguishing feature between them is that Thin processes requests sequentially and Mongrel processes them in threads, enabeling concurrent processing. Wether sequential or concurrent processing is best depends on your application. In general a concurrent model wins when you have varying response times. If for instance one request takes several seconds to process all other requests to you application have to wait until the first request have finished processing when one are using a sequential model. For a concurrent model the first request can be processed in its own thread while new requests can be accepted and processed in their own threads.

Why then not use a concurrent model all the time? The drawback with a cuncurrent model is that creating threads is costly and introduces some overhead – resulting in longer response times. If your application only have fairly short response times you can take advantage of a sequential model where the time to process a response will be lower compared to a scenario where one have to create a new thread for each request.

? recommends using Mongrel, the concurrent Ruby web server, for general purpose applications. He goes on to say that the treshold for wether one should use a sequential server is response times that are no longer than two too three seconds. Seeing as our application could very well fall outside this treshold for some requests[32] we followed the recommendations from **?** an opted to use Mongrel as our Ruby web server. By doing so we're on the safe side as we can deliver simultaneous requests even if one of the requests are taking overly long to process. In addition we believe the overhead of using a concurrent model with threading would not introduce to much overhead in response time for average timed requests compared to a sequential model.

32. When the data the user is requesting can not be found in the cache we have to retrieve the data from Urørt and perform calculations on it. This can be quite time consuming and could very well exceed the 2–3 second treshold **?** recommends to use as guidance in server selection.

Cache library

As detailed in § 4.4.3 (p. 58) we had no need to persist our activity data since it was changing as time went by. We did however want to cache this data for a given time so that users would get berable request times when using our prototype implementation.

We did not test various cache solutions, but instead went with a proven cache solution called Memcached.[33] Memcached was developed at LiveJournal[34] to handle caching of data for their large user base. Memcached is used by many large web sites and Facebook is currently the largest user of the caching solution (?). Memcached is very performant since it stores all its cached items in memory. To interact with the Memcached server we used the memcache-client[35] Ruby library.

34. LiveJournal is a place where people can keep a blog or journal with some social network features baked in. As of May 20, 2008 LiveJournal had over 15 million registered users (?). LiveJournal is available at http://livejournal.com.

Database

Since we were going to only use a single database table (see § 4.4.7 (p. 62) for details) and did not need any special features, we could basically have used all freely avaiable relational database systems. We selected Sqlite[36] as it have a much smaller footprint than it's competitors. This is mainly due to it's limited featureset. Another usefull characteristic of Sqlite is it's storage format. Sqlite simply writes to a normal disk file that can be backed up and handeled with normal file system tools.

36. Available at http://sqlite.org.

For accessing the database we had a choice between several libraries that makes interaction with the database easier and abstracted based on database software. We selected the library we found had the most flexible and lightweight ORM capability: Sequel.[37]

B.1.3 Overview of client & server components

With all the pieces in place of our third party software puzzle we present a high level view of the architecture, from the client to the server, in Figure B.2 (p. 132). Table B.1 lists the versions we used of the various third party software.

B.2 DEVELOPMENT TOOLS

As with the implementation platforms, languages, and third party libraries our first criterion for selecting development tools is freedom.

B.2.1 Version control

We've found it indispensable to use *version control* when writing code and even used it when authoring this thesis. We'll not spend time to discuss the merits of version control since we feel its benefits are major and using one induces almost zero overhead in your working process. Sometimes we feel that the use of version control can guide you when conducting complex tasks.

There are however several different forms of version control system one can use. One of the most used version control implementations

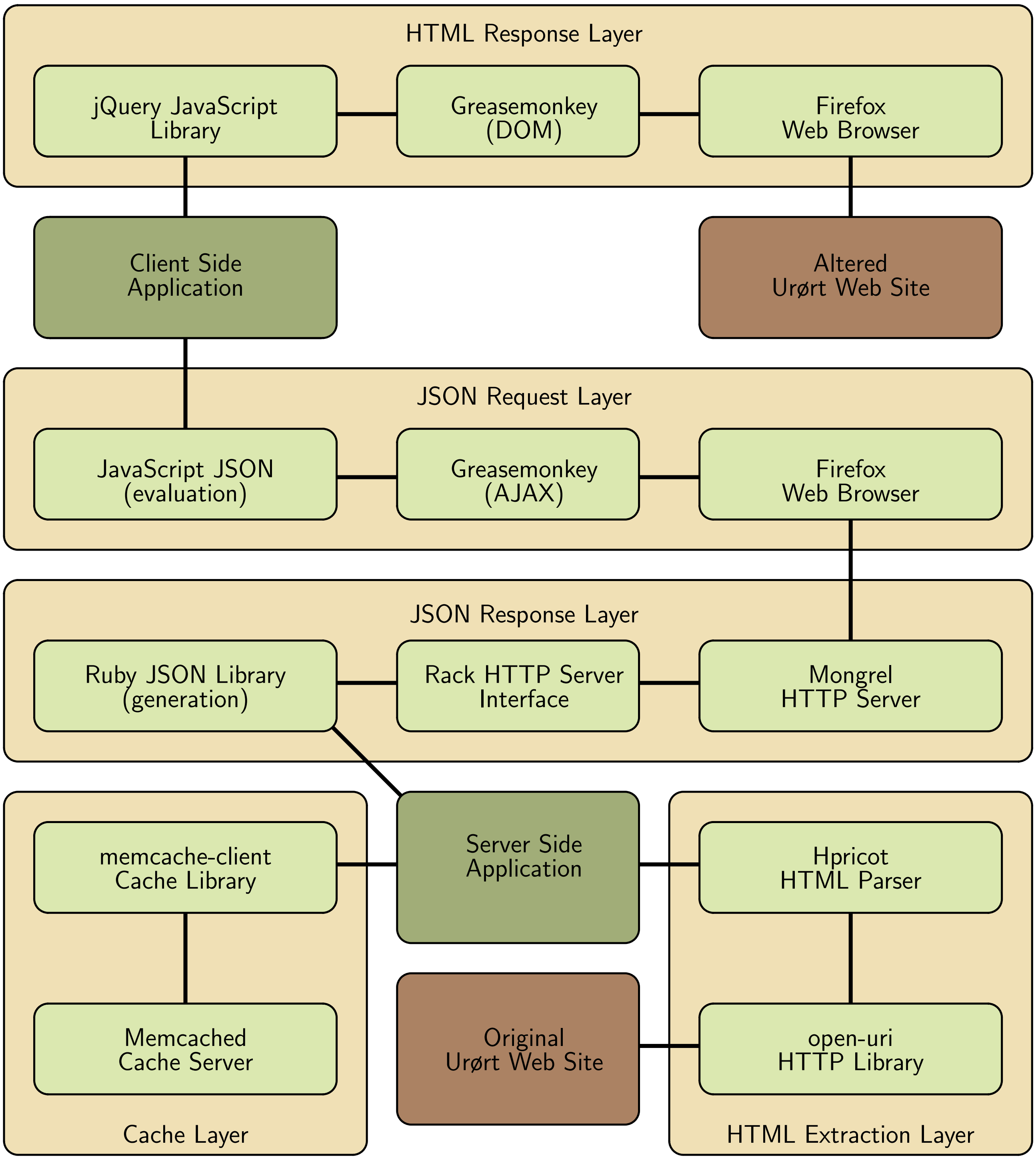


Figure B.2: High level view of the overall prototype architecture.

	Name	Version
Web browser	Firefox	User dependant (tested on 2.0.0.14 and 3.0 RC1)
Client-side language	JavaScript	Browser dependant (tested on 1.7 and 1.8)
User script extension	Greasemonkey	User dependant (tested on 0.7.20080121.0)
JavaScript library	jQuery	1.2.4
Server-side plattform	Debian GNU/Linux	4.0
Server-side language	Ruby	1.8.5
HTML scraping library	Hpricot	0.6 (with buffer overflow patch)
HTTP fetching library	open-uri	1.8.5 (included with Ruby)
JSON library	json	1.1.2
HTTP framework	Rack	0.3.0
HTTP server	Mongrel	1.1.4
Cache server	Memcached	1.1.12
Cache access library	memcache-client	1.5.0
Database server	Sqlite	3.3.8
Database access library	Sequel	1.5.1

Table B.1: Versions of third party software used in the prototype stack, by type

the last years in open source circles was Subversion[38] – a *centralized version control system* meaning that one central server holds the version controlled code repository and its history.[39] Recently *decentralized version control systems* have become more popular amongst developers. A decentralized model means that every developer can have their own repository consisting of all history.[40] Code is then shared either in a push or pull fashion between such individual repositories. This enables a much better model for collaboration. We favor this last model of version control and so have projects like Linux, X, Mozilla, and OpenSolaris.[41]

Based on criteria of performance and current adoption there are in our view only two interesting decentralized version control systems: Git[42] and Mercurial.[43] Both are unique in that they don't track meta-data, they just track content and meta-data are thereby inferred from the content. At a very high level view Mercurial have a better user interface and Git supports some advanced features the former don't have. We opted to used Mercurial for this development project since we've substantial experience in using it and did not need any of Git's advanced features.

42 Available at http://git.or.cz

43 Available at http://www.selenic.com/mercurial

B.2.2 *Editor*

A developer's main tool for authoring software is his editor. Sometimes the language of implementation warrants a specialized editor with aids for handling cumbersome tasks specific to that language. Such an editor is often called an IDE[44] and are used most often for languages like Java and C#. **?** found that developers mostly use an IDE for navigating large collections of source code, refactoring code, debugging code, and

44 A good example of an IDE (integrated development environment for short) is Eclipse (available at http://eclipse.org). It was first used for Java development but since extended with plugins for handling other programming languages and families.

interacting with revision control systems in addition to normal editor usage. Development environments found in Lisp[45] and Smalltalk[46] are surpassing IDE types in integration and interactiveness even though they preceded them.

45. ?, p. 69 describes the nature a
benefits of the Lisp environment
“The ‘residential’ design of progra
ing systems, whereby all facilities
for the user are integrated into o
system with which the user comm
nicates during the entire interact
session, offers great possibilities
user convenience”.

The programming languages we previously settled on, JavaScript and Ruby, are very expressive and dynamic in their nature in addition to being interpreted instead of compiled. Our experience is that IDE usage for such languages stands more in the way than aid you as a programmer during your problem solving process. **?** conducted a rather unscientific survey of 1000 Ruby programmers. Despite of the surveys shortcomings it showed that the majority of Ruby programmers used non-IDE editors for their development.

The interactive experience provided by Lisp and Smalltalk implementations are sadly missing[47] from JavaScript and Ruby implementations. This means that we're left with finding a good editor which enables us to focus on writing code as efficiently and safely as possible. Editor selection is highly a matter of preference and finding one that matches your work process. Powerful editors have a reputation of being quite hard to learn. But if you get over the steep learning curve the benefits the editor gives you are worth it.

47. Ruby has an interactive interpreter similar to those found in Lisp and Smalltalk environments called `irb`. It's not integrated into an overall programming environment and therefore is mostly used for testing out small ideas.

? have experienced how much effort programmers can invest in something seemingly trivial as an editor:

If the thought of switching editors doesn't fill you with quite a bit of dread, what you're using now is almost certainly under powered, and you definitely haven't customized it enough.

B.2.3 Testing suites

As described in § 4.3.2 (p. 53) we're firm believers of using automated testing when developing applications.

Greasemonkey user scripts inherit a strict security model where the DOM one is interacting with are a special copy of the browser's DOM. In the case of testing this is unfortunate since it's very complicated to get a testing library to properly run within this secure model.

We initially tried to adapt the Screw Unit[48] JavaScript behaviour-driven development library to the intricacies of Greasemonkey user scripts, but had to give up. We therefore had to develop without automated tests on the client-side. Fortunately the most complicated logic of our application are on the server side and our client-side development process did not get to hard despite the lack of a proper testing suite.

48. Screw Unit is available at http://github.com/nkallen/screw-unit.

On the server-side we had better success with integrating a testing suite into our application. There are several options available when selecting amongst Ruby testing suites. We're ignoring traditional test-driven libraries as we're proponents of a behaviour-driven style (§ 4.3.2 (p. 53)). We looked at[49]:

49. These behaviour libraries can be found at respectively http://rspec.info, http://rubyspec.org, http://chneukirchen.org/repos/testspec, and http://chneukirchen.org/repos/bacon.

- *RSpec* is the original behaviour-driven suite for Ruby. It's therefore the most mature project and have the best integration with other tools. It does seem to suffer from too much complexity in its code base. The following suties adresses this complexity problem.
- *MSpec* have more features than RSpec in spite of having clearer and more understandable source code (**?**).
- *test/spec* is an interface for writing specifications of behavior on top of the original Ruby unit testing library and are therefore compatible with tests written with it.
- *Bacon* is the smallest of the suites when counting source code but still implements the majority of the fatures of its big brothers. It's still a young project and have therefore not seen much usage.

Based on the tool support and code maturety we decided to use RSpec for our development. This allowed us to use autotest, a part of ZenTest[50] which continously runs your test suite as you make changes to your source code files. This way you can keep your focus on the editor and only glance over the status of your test suite as you move along.

50. Available at http://www.zenspider.com/ZSS/Products/ZenTest/.

B.2.4 *Debugger & profiler*

Since we were unable to utilize automated tests on the client-side we had to resort to a debugger checking for correctness in our code while developing. There are currently two JavaScript debuggers for the Firefox browser: Venkman[51] and Firebug.[52] The latter have a considerably less intrusive interface than the former and seems to be much more actively developed as of this writing. Firebug have the most advanced features with a better user interface. Our choice of a client side debugger was simple.

On the server-side we never saw the need for a debugger since we developed all our code in a behaviour-driven way enabeling us to both catch bugs and form solutions through our specification suite. But as described in § 4.5 (p. 62) we stumbled upon some major performance problems. For locating the bottlenecks in our application – the places that consumed the most time – we used a *profiler*[53] called ruby-prof.[54] This is the fastest profiler for Ruby and it can generate various forms of reports which enables a developer to understand the time related aspects of his code.

51. Located at http://www.mozilla.org/projects/venkman/.

52. Firebug has its home at http://www.getfirebug.com.

53. A profiler is a tool used by developers for identifying the execution time of various parts of a program or how often these parts of the program are utilized (**?**, p. 120). It's used when trying to improve the performance of a segment of a given program and should be used in an iterative way (**?**, p. 125).

54. Available at http://rubyforge.org/projects/ruby-prof.

QUESTIONNAIRES

C

The questionnaires was given to Norwegian users and are therefore represented here in their original language and tone. After every question we list the possible responses. Response options are separated by commas (,). When response options are enclosed within angle brackets ([]) only one answer was allowed for that particular question. For response options enclosed in curly braces ({}) multiple answers was allowed for that question. An asterisk (*) at the beginning of a question denotes that an answer was required. [0–N] means that the response have to be a positive number. [*] indicates a free text response.

C.1 PRETEST SURVEY

These set of questions were asked before users were given an option to install and use our prototype application.

C.1.1 User profile

1 Er du mann eller kvinne? [Mann, Kvinne]
2 Hvor gammel er du? [0–N]
3 * Bruker du Firefox nettleser? [Alltid, Som regel, Av og til, Sjelden/aldri]
4 * Hvor ofte besøker du nettstedet Urørt? [Daglig, Flere ganger i uken, Ukentlig, Månedlig, Sjelden/aldri]
5 * Når du besøker Urørt, pleier du å logge deg på (med brukernavn og passord)? [Alltid, Som regel, Av og til, Sjelden/aldri]

C.1.2 Favorites on Urørt

1 * Er du kjent med begrepet "Favoritter" på Urørt? [Ja, Nei]
2 Hvor mange Favoritter har du på Urørt? [0–N]
3 Hva gjør at du velger å legge artister på Urørt til dine Favoritter? {Artistens musikk, Artistens popularitet, Venner med artisten, Kjennskap til artisten}
4 Hvor ofte opdatererer du deg på hva dine Favoritter på Urørt foretar seg? [Daglig, Flere ganger i uken, Ukentlig, Månedlig, Sjelden/aldri]

C.1.3 Being up-to-date on favorites

1 Jeg synes det er enkelt å holde meg oppdatert på hva mine Favoritter foretar seg på Urørt. [Helt uenig, Litt uenig, Verken enig eller uenig, Litt enig, Helt enig]
2 Jeg synes det er enkelt å holde meg oppdatert på hvorvidt mine Favoritter legger ut nye sanger på Urørt [Helt uenig, Litt uenig, Verken enig eller uenig, Litt enig, Helt enig]
3 Jeg synes det er enkelt å holde meg oppdatert på hvorvidt mine Favoritter legger ut nye blogg innlegg på Urørt. [Helt uenig, Litt uenig, Verken enig eller uenig, Litt enig, Helt enig]
4 Jeg synes det er enkelt å holde meg oppdatert på hvorvidt mine Favoritter holder konserter. [Helt uenig, Litt uenig, Verken enig eller uenig, Litt enig, Helt enig]
5 Jeg synes det er enkelt å holde meg oppdatert på hvilke reaksjoner andre Urørt brukere har på mine favoritters sanger. [Helt uenig, Litt uenig, Verken enig eller uenig, Litt enig, Helt enig]
6 Har du noen ønsker for hvordan Urørt kunne gjort det enklere å holde seg oppdatert på Favoritter? [*]

C.2 FOLLOW-UP SURVEY

The following pre-installation questions were asked 24 hours after users were given their initial survey and the option to install our prototype application.

1 * Klarte du å installere den nye funksjonen "Siste fra dine favoritter"? [Ja – det gikk fort og greit, Ja – men jeg opplevde små problemer underveis, Ja – men jeg opplevde store problemer underveis, Nei – jeg gav opp]
2 Dersom du opplevde problemer, vennligst fortell kort om hva som var problemet. [*]

C.3 POSTTEST SURVEY

The posttest survey was given to users 11 days after they took part in the initial survey.

C.3.1 Favorites on Urørt

1 Hvor mange Favoritter har du på Urørt? [0–N]
2 Hva gjør at du velger å legge artister på Urørt til dine Favoritter? {Artistens musikk, Artistens popularitet, Venner med artisten, Kjennskap til artisten}
3 Hvor ofte opdatererer du deg på hva dine Favoritter på Urørt foretar seg? [Daglig, Flere ganger i uken, Ukentlig, Månedlig, Sjelden/aldri]

C.3.2 Being up-to-date on favorites

1 Hvor mye har du brukt den nye funksjonen "Siste fra dine Favoritter" når du har vært pålogget Urørt? [Har ikke brukt, Kun noen få ganger, Nesten hver gang, Hver gang]

2 Jeg synes det er enkelt å holde meg oppdatert på hva mine Favoritter foretar seg på Urørt. [Helt uenig, Litt uenig, Verken enig eller uenig, Litt enig, Helt enig]

3 Jeg synes det er enkelt å holde meg oppdatert på hvorvidt mine Favoritter legger ut nye sanger på Urørt [Helt uenig, Litt uenig, Verken enig eller uenig, Litt enig, Helt enig]

4 Jeg synes det er enkelt å holde meg oppdatert på hvorvidt mine Favoritter legger ut nye blogg innlegg på Urørt. [Helt uenig, Litt uenig, Verken enig eller uenig, Litt enig, Helt enig]

5 Jeg synes det er enkelt å holde meg oppdatert på hvorvidt mine Favoritter holder konserter. [Helt uenig, Litt uenig, Verken enig eller uenig, Litt enig, Helt enig]

6 Jeg synes det er enkelt å holde meg oppdatert på hvilke reaksjoner andre Urørt brukere har på mine favoritters sanger. [Helt uenig, Litt uenig, Verken enig eller uenig, Litt enig, Helt enig]

7 Hvordan påvirker funksjonen "Siste fra mine Favoritter" din bruk av Urørt? [*]

C.3.3 Perception of new functionality

1 Funksjonen "Siste fra dine Favoritter" vil gjøre at jeg kan holde meg oppdatert på mine favoritter på en effektiv måte. [Svært usannsynlig, Usannsynlig, Litt usannsynlig, Nøytral, Litt sannsynlig, Sannsynlig, Svært sannsynlig]

2 Funksjonen "Siste fra dine Favoritter" vil gjøre det mulig å holde seg oppdatert på flere favoritter. [Svært usannsynlig, Usannsynlig, Litt usannsynlig, Nøytral, Litt sannsynlig, Sannsynlig, Svært sannsynlig]

3 Funksjonen "Siste fra dine Favoritter" vil gjøre det enklere å holde seg oppdatert på favoritter. [Svært usannsynlig, Usannsynlig, Litt usannsynlig, Nøytral, Litt sannsynlig, Sannsynlig, Svært sannsynlig]

4 Funksjonen "Siste fra dine Favoritter" vil ære nyttig for å holde seg oppdatert på favoritter. [Svært usannsynlig, Usannsynlig, Litt usannsynlig, Nøytral, Litt sannsynlig, Sannsynlig, Svært sannsynlig]

5 Funksjonen "Siste fra dine Favoritter" vil være enkel å lære seg å bruke. [Svært usannsynlig, Usannsynlig, Litt usannsynlig, Nøytral, Litt sannsynlig, Sannsynlig, Svært sannsynlig]

6 Funksjonen "Siste fra dine Favoritter" vil la meg bruke den slik jeg vil. [Svært usannsynlig, Usannsynlig, Litt usannsynlig, Nøytral, Litt sannsynlig, Sannsynlig, Svært sannsynlig]

7 Funksjonen "Siste fra dine Favoritter" vil være enkel å bli dyktig til å bruke. [Svært usannsynlig, Usannsynlig, Litt usannsynlig, Nøytral, Litt sannsynlig, Sannsynlig, Svært sannsynlig]

8 Funksjonen "Siste fra dine Favoritter" vil være enkel i bruk. [Svært usannsynlig, Usannsynlig, Litt usannsynlig, Nøytral, Litt sannsynlig, Sannsynlig, Svært sannsynlig]
9 Synes du funksjonen "Siste fra dine Favoritter" burde være en standard funksjon på Urørt? [Helt uenig, Litt uenig, Verken enig eller uenig, Litt enig, Helt enig]

SOURCE CODE

D

D.1 UNOBTRUSIVE SOCIAL NAVIGATION PROTOTYPE FOR URØRT

Due to size considerations we're not going to list the source code of our prototype implementation for Urørt called "Latest from your Favorites" in this appendix. The source code and its entire history is available at http://bitbucket.org/uggedal/rort/.

D.2 REDDIT COLLABORATIVE FILTERING ALGORITHM

Here is the source code of the algorithm that decides the score of a submission. We made syntactical changes to make the code's intent clearer.

```python
from datetime import datetime, timedelta
from math import log

def seconds_since_cutoff(date):
    cutoff = datetime(2005, 12, 8, 7, 46, 43)
    td = date - cutoff

    seconds = td.days * 24 * 60 * 60
    seconds += td.seconds
    seconds += float(td.microseconds) / 1000000
    return seconds

def score(up_votes, down_votes, submitted_date):
    score = up_votes - down_votes
    order = log(max(abs(score), 1), 10)
    sign = 1 if score > 0 else -1 if score < 0 else 0

    age = seconds_since_cutoff(submitted_date)

    return round(order + sign * age / 45000, 7)
```

Source Code Listing D.1: The collaborative filtering algorithm used on Reddit

D.3 JAVASCRIPT COMMENT STRIPPER

This small script was written to help compare different JavaScript libraries:

```
#!/usr/bin/env ruby

js_files = Dir['*.js']

ignore_pattern = /^[\s\t]?(\/\*|\*|\/\/)/

js_files.each do |file|
  File.open("#{file}.out", 'w') do |out|
    out.puts File.readlines(file).reject do |line|
      line =~ pattern
    end
  end
end
```

Source Code Listing D.2: Strips comments from JavaScript libraries

D.4 SHELL FILE AND DIRECTORY HIERARCHY

This one-liner was used for conveying the directory and file hierarchy of our server side software:

```
find . | sed -e 's/[^\/]*\//|--/g' -e 's/-- |/|/g'
```

Source Code Listing D.3: File and directory hierarchy with standard UNIX tools