# Lightweight Initiation and Seamless Transitions in the CoopFox Collaborative Browser Extension

by

Ralf Michael Strobel

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the Computing in the Humanities Program
of the Faculty Information Systems
and Applied Computer Sciences
at the University of Bamberg

under the Supervision of Prof. Dr. Tom Gross

# Abstract

Activities of internet users continue to shift more and more towards social and collaborative interaction. Still, groupware tools for every-day collaborative browsing are barely available, let alone mainstream. In search for such a solution, my thesis explores three key concepts from CSCW research, which have not been addressed systematically by other cobrowsing prototypes so far: lightweight session initiation, seamless transitions between solitary and group work, as well as contextual collaboration. I have developed my own cobrowsing solution, CoopFox, specifically to adhere to all three concepts. A user study conducted with the finished prototype found that all participants were quickly able to comprehend each concept and use it productively in web research, outperforming separate browsers and screen sharing in terms of user satisfaction.

# Acknowledgements

I would like to thank all of my user study participants, as well as everyone who provided feedback on the SpreadVector and CoopFox prototypes during their development. I would further like to thank the team at the Cooperative Media Lab, Mirko Fetter, Sascha Herr and Professor Tom Gross, for their support in developing the concept for this thesis and the contained user study.

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

Over the past decade, activities of internet users have continued to shift more and more towards social and collaborative interaction. However, most of these new activities and the services they have created (e.g. microblogging, social recommendation, Q&A sites) still rely solely on asynchronous communication. Synchronous communication services such as Skype have also increased in popularity, but remain separate and generic tools.

The process of exploring and researching information on the web is still considered a single-user activity, and web browsers are built as single-user software, much to the regret of the many CSCW researchers, who have pointed out the need for distributed real-time collaboration on the web for nearly two decades [Greenberg & Roseman 1996, p. 271] [Chong & Sakauchi 2000, p. 803] [Morris 2008, p. 1657] [Niederhausen et al. 2010, p. 396].

There have been a great number of prototypes, which successfully implemented popular groupware features in web browsers (see chapter 3) and all of them received very positive feedback in user tests under laboratory conditions. Sadly though, few of these projects ever made it out of their experimental stage and none eventually gained enough popularity among the average internet users to become mainstream.

## 1.1 Motivation

The purpose of my thesis is to explore ways in which the established principles for synchronous collaborative browsing may be adapted in a form that is more appealing to current every-day internet users, who are often engaged in spontaneous and highly dynamic interactions, rather than planned long-term collaboration.

To this end, chapter 2 first establishes a requirement profile of the average web users and their needs for real-time collaboration, by reviewing literature on both web usage behaviour as well as synchronous collaborative work. Chapter 3 contrasts these findings with a review of different past solutions for real-time collaborative web browsing. Based on the previous two chapters, Chapter 4 then derives the interaction design for my own prototype solution, CoopFox. Chapter 5 describes the details of its implementation. Chapter 6 presents the findings from my user study, in which I compare participant satisfaction with CoopFox against other common forms of casual online collaboration. Finally, chapter 7 summarizes my conclusions on the potential for every-day use of collaborative web browsing and gives an outlook on the future development of CoopFox.

In the long run, I hope that my approach will provide perspectives to help satisfy the growing demand for more real-time collaboration on the web. A recent online survey of 167 adult internet users found that 49% of them already engage in collaborative web-related activities at least once per week [Morris 2013]. A similar prior survey placed this number at 38% [Capra et al. 2011]. Commonly reported scenarios include:

- Researching different aspects of a work-related problem, sharing results via email.
- Online-shopping or travel planning as a group in front of a single computer.
- Sharing links to media via text chat and discussing them by phone or voice chat.

This demonstrates how often people now utilize generic ways of communication in order to support collaboration, suggesting that they might benefit even more from dedicated support tools. As younger web users show much greater affinity towards collaboration [Morris 2013, p. 1188], this need is likely to increase even further in the future.

## 1.2 Outline

The structure of my thesis is based around three key principles: *lightweight initiation*, *seamless transitions*, and *contextual collaboration* (defined below). They have been presented, independently, as essential for user-friendly collaboration support of highly dynamic and informal activities, such as web browsing.

Each principle focusses on events at a different scale of the collaborative process: *initiation* of the collaboration as a whole, *transition* between different work modes, and contextual *integration* of individual collaborative actions into the work environment. My assumption is that usability must be optimized at all three scales, in order to create a solution that is broadly appealing to the targeted casual web user.

The separation into concerns of *initiation*, *transition* and *integration* therefore continues as a pattern throughout my thesis. Chapters 2, 4, 5, and 6 all contain a dedicated subchapter for each concern, presenting the respective problems, solutions or results.

### 1.2.1 Lightweight Initiation

This term originated from a criticism of shared workspace systems in office environments by Carl Gutwin and Saul Greenberg. They pointed out that these systems "are designed to support planned, formal collaboration sessions, but it appears that much of the shared work (…) is informal, unplanned, and opportunistic" [Gutwin et al. 2008, p. 1412].

According to the authors, in order to support such informal group work, users must be able to initiate collaboration at any time, with little cognitive effort, and carry over their current state of work into the collaborative process. Cumbersome manual steps such as setting up a dedicated workspace or copying existing work content must be avoided. (Their concrete proposals are discussed in section 2.2.2.)

### 1.2.2 Seamless Transitions

The criterion of *seamlessness* in collaborative systems was first coined by Mark Stefik and then developed further by Hiroshi Ishii [Ishii & Miyake 1991, p. 37]. It refers to the cognitive ease of transitioning between different work "modes" during collaboration.

For instance, participants may wish to complete large parts of the shared task alone, but intermittedly come together in close interaction to share results or exchange feedback. For highly dynamic environments, these transitions between solitary and group work are crucial for efficient division of labour. Though not relevant to my thesis, seamlessness may also concern other transitions such as synchronous to asynchronous or co-located to distributed work [Borghoff & Schlichter 2000, pp. 127-128].

### 1.2.3 Contextual Collaboration

This industry-related term was coined by META Group analyst Matt Cain, who used it for different techniques of embedding collaboration functions directly into the user interface of single-user applications [Hupfer et al. 2004, p. 21]. The intention is two-fold: Firstly, users can keep using the software they already know [Xia et al. 2004, p. 163]. Secondly, the embedded functions allow users to collaborate directly in context of a piece of related content, without the cognitive cost of switching between applications. Popular examples include embedded chats in text editors or software IDEs, as well as augmentation of person names with one-click communication [Hupfer et al. 2004, p. 21].

# 2 Background

This chapter analyses the demand of internet users for collaborative web browsing, and identifies key requirements for a co-browsing support tool. To this end, I review selected literature on current web usage patterns as well as synchronous collaborative work. In the same process, I also introduce and disambiguate all necessary terminology for the upcoming chapters, as well as relevant scientific theories.

## 2.1 Terminology

This first subchapter focusses on all general definitions required for the entire scope of my thesis. Definitions which have a concrete association with one of my outline principles (*initiation*, *transition* and *integration*) are then introduced separately, as soon as they are required, in the respective subchapters 2.2 to 2.4.

The subject of collaborative browsing lies at an intersection between the fields of computer-supported collaborative work (CSCW) and information retrieval (IR). This is why it is especially important to carefully choose and disambiguate any terminology in this context, as even seemingly simple terms like "search" have been used with a variety of different meanings, depending on the orientation of their literary context.

### 2.1.1 Collaboration

According to Peter Denning and Peter Yaholkovsky, *collaboration* occurs as the highest form of interaction, when people are "working together synergistically". This means that they are giving active support to each other, in order to achieve a common goal with greater effectiveness or efficiency, compared to a simple division of labour, where each person contributes to the goal individually [Denning & Yaholkovsky 2008].

This pure division strategy is what the authors define as *cooperation*, a reduced level of interactivity, which still involves common goals and established "rules of interaction". Finally, *coordination* is defined as the lowest level of working together, where people merely "regulate" or "harmonize" their actions and resource requirements, so that each of them is able to achieve their often independent goals.

Coordination is therefore also a necessary subset of cooperation and collaboration. Participants may have to invest significant effort into negotiating and maintaining the conditions of their interaction. This includes aspects such as goals, roles, responsibilities, rights, schedules, procedures, terminology and resource allocation [Schmidt 2011, p. 56]. The resulting "articulation work" is a mandatory overhead to all cooperative and collaborative work. Supporting coordination and reducing its effort to a minimum is therefore an important requirement for all CSCW applications.

When tasks are simple and short-lived, involving few people, coordination support is typically covered by simple communication support. In such conditions, people are capable of coordinating themselves through informal conversation [Schmidt 2011, p. 97] [Ellis et al. 1991, p. 52]. Complex tasks require more sophisticated support systems, geared towards common problems such as scheduling, resource allocation, as well as storage and retrieval of artefacts and documentation. The combination of all these aspects is commonly referred to as a *shared workspace* [Soliman et al. 2005, pp. 368-369].

### 2.1.2 Groupware

Application which actively support cooperative or collaborative work are commonly referred to as *groupware* [Johansen 1988]. It should be mentioned, however, that the use of the term *group* varies across literature [Forsyth 2009, pp. 2-5]. One common definition merely states that a group is "any number of people who interact with each other, are (…) aware of each other, and perceive themselves to be in a group" [Pennington 2002, p. 3].

The term *team* (or *workgroup*), on the other hand, has a more precise definition in the context of CSCW, namely that its members "must be engaged in a common task and interact within a shared environment to perform it" [Borghoff & Schlichter 2000, p. 92]. Combined with the definition from the past session, it can be said that cooperation and collaboration are therefore always performed by a *team*. However, large parts of CSCW literature use *team* and *group* synonymously. I do the same throughout this thesis.

Johansen classified groupware systems using a 2x2 time-space-matrix, depending on whether they were designed to support collaboration between people acting at the same time (synchronous, real-time) or at different times (asynchronous), and whether people were in the same place (co-located) or in different places (distributed). My thesis focusses almost exclusively on *real-time distributed groupware* (RTDG) solutions.

However, strict classification of groupware via the space-time-matrix is becoming increasingly difficult [Cockburn & Greenberg 1993, pp. 34-35]. Especially since the appearance of mobile computing devices, people now often transition between co-located and distributed work modes during a single collaboration. The same is true regarding synchronous and asynchronous interaction, for example in the case of instant messengers, which allow users to delay their responses to incoming communication if their attention is bound otherwise [Nardi et al. 2000, p. 83]. This has been called "semi-synchronous" [Gross 1998, p. 43]. However, for the purpose of my thesis, I refer to any interaction as "synchronous" or "real-time", as long as it could potentially be executed synchronously.

### 2.1.3 Web Browsing, Web Search and Web Research

Today *browsing* is used colloquially as an umbrella term for all activities performed using a web browser. The expression originally dates back to the early days of the WWW and before, when browsing was literally the only way of exploring information on a computer network. More scientifically established and precise is the term *web navigation*, which commonly refers to the passive exploration of hypertext by directly entering URLs or following links, but not by interactive means such as query-based lookup [Levene 2011, pp. 38-39]. However, many scientific publications have also used *web browsing* and the even more colloquial *web surfing* synonymously to *web navigation*.

A similar problem exists for the term *search*, which was originally tied very closely to the use of a global web search engine. But as the complexity the web has grown, many large web sites now also provide local search, filter and recommendation functions, which have often even replaced the traditional navigation structures on these sites. "Navigational searches" also make up an increasing amount of queries on global search engines [Rose & Levinson 2004, p. 15], blurring the line between navigation and search. In addition, it has been pointed out that the process of searching often goes beyond the simple evaluation of search engine results. It can be a complex iterative process of global and local searches, navigation, query and goal refinement, and also social feedback through forums or Q&A sites [Kuhlthau 1991, pp. 362- 363] [Morris 2013, p. 1188].

These combined ambiguities have led some authors to abandon the word *search* for more specific expressions such as "online information seeking" [Levene 2011, p. 24] [Golovchinsky et al. 2009] [Kuhlthau 1991]. However, many other authors still use *search* to describe both simple lookup tasks and complex exploratory research. These may also include leisure activities with undefined goals, in which people are simply looking for inspiration or entertainment, by aimlessly querying or navigating web sites.

Throughout my thesis, I therefore use the term *web research* to describe any complex process, which requires a person to visit or make use of multiple webpages, in order to satisfy a well-defined, pre-existing demand for information. This may include fact-based questions or the need for a solution to a specific problem, but also consideration of multiple options, for instance during online shopping or travel planning.

I use the term *web browsing* to describe all activities which do not fit into the research category. This includes simple lookups like the current weather forecast, consumption of news and media content, or just exploration without a specific goal.

Generally excluded from the scope of my thesis are *transactional activities*, such as writing emails via web interfaces, commenting, online banking or online games. The reason is that these activities either do not offer themselves for collaboration, or would require very task-specific support, such as distributed collaborative text editing.

According to a recent survey of web user activities, these three categories (research, browsing, transactions) account for more than 98% of web browser usage, wherein transactions make up 46.7%, browsing 38.2% and research 13.4% [Kellar et al. 2007].

### 2.1.4 Collaborative Browsing and Collaborative Research

In accordance with the previously established definitions, I only use the term *collaborative web browsing* (cobrowsing) for activities and solutions, when they revolve around simple tasks, for which users primarily need to be able to visit web pages simultaneously and communicate in their context. For activities and solution which involve the gathering, evaluation and aggregation of results, in order to fulfil a common pre-existing information goal, I use the term *collaborative web research*.

This second aspect has been scientifically investigated under various names, including "collaborative web search", "collaborative information seeking" and "social search". Earliest related work even predates the web age, going back to the 1960, when researchers first analysed social information sharing among scientists [Twidale et al. 1997, p. 762].

Twidale et al. observed the collaborative behaviour of users of a physical library and classified activities by how close people were interacting [Twidale et al. 1997, p. 766]. They identified "joint search" as the closest collaborative state, in which group members were using the same resources, discussing content and planning next steps together. During "coordinated search", group members were using separate resources, but intermittedly comparing and discussing their results. In the "free query" state, group members would work separately, but peripherally monitor the actions of others and sometimes ask them for feedback or help. Finally, "chance contact" led to brief collaboration when two individuals were coincidentally using the same resource.

Brynn Evans and Ed Chi took a closer look at search tasks performed under tight collaboration and identified several distinct phases [Evans & Chi 2008, pp. 487-491]: After an initial phase of goal and method negotiation, they observed repetitively alternating phases of "foraging", "sensemaking" and "organizing".

During a *foraging* phase, participants look for new content, typically using a process of iterative query refinement at varying degrees of interaction. Other authors have suggested that keyword awareness and page visitation awareness support might be crucial to increase the efficiency of collaboration during this phase [Morris 2008, p. 1660].

A *sensemaking* phase is used to discuss and evaluate findings from the last foraging phase and integrate new discoveries with previous knowledge. The intensity of interaction is typically higher during this process, as it requires group decision making. The involved decision making tasks can be categorized as "choosing" and "negotiating", according to McGrath's 1984 group task circumplex [Baltes et al. 2002, p. 162]. Known support mechanisms such as voting may therefore apply to aid this process.

As soon as a sufficient amount of valuable new content has been identified, an *organizing* phase follows, during which participants format and compile their findings into a more persistent state, create meta-level artefacts such as process documentation, and make plans for next steps. Once more, group decision tasks are involved, in this case of the category "generating" (ideas, plans), for which appropriate support mechanisms such as brainstorming have been applied successfully in the past.

Foraging, sensemaking and organizing can repeat many times before the team is satisfied with the results. Also, division of labour strategies vary strongly between teams. Some prefer to perform each phase simultaneously, while other teams develop dedicated foragers, analysts and organizers [Golovchinsky et al. 2009, pp. 49-50].

The final phase identified by Evans and Chi is *distribution*, which follows as soon as the main research is complete. During this process, the participants ensure that all members receive appropriate copies of all findings. They may also wish to archive their results as well as process documentation, to be reused in similar future researches or in case another refining research becomes necessary.

## 2.2 Initiation

This subchapter focusses on all events which lead up to a person being able to browse or research collaboratively. Since standard web browsers do not provide support for this, the first step is always a decision towards an additional software product, followed by its installation and configuration. These aspects are investigated in subsection 2.2.1.

Next, several users need to be able to find each other and begin a collaborative session. The term *session* in this context always refers to "a period of synchronous interaction" [Ellis et al. 1991, p. 46] from the user's perspective, independent of how it may be modelled or implemented in various systems. This is the subject of subsection 2.2.2.

### 2.2.1 Technology Acceptance

One of the most popular theoretical models for the adoption of new software tools by computer users is the "Technology Acceptance Model" (TAM) [Davis 1989]. It suggests that people, in their decision whether to adopt a new tool, weigh up the amount of future effort it will save them ("Perceived Usefulness") against the amount of effort required to learn and operate it ("Perceived Ease of Use"). This is of course a simplification and has since been amended with various social and situational factors [King & He 2006] [Yi et al. 2006]. However, the basic assumption is still considered valid and is therefore used as the base of further considerations.

On the side of *usefulness*, any support for collaborative browsing and research can be considered an improvement over no support (only generic communication tools), which is still the standard method the majority of people are using. As to be shown in chapter 3, this assumption generally holds for all studies of past prototypes, in that participants always perceived some level of utility in the applications they were testing. However, the extent of this perceived advantage may in fact not be very great.

This becomes evident in comparison to some of the most successful CSCW applications today, such as workflow management and group decision support systems. These tools focus on coordination support for complex tasks in medium to large groups, using asynchronous communication. Their usefulness is obvious, because any attempt at handling the same tasks using informal collaboration would likely result in chaos.

In contrast, most real-time collaboration tools do not play to these strengths of CSCW, because they focus on small groups and synchronous communication. In these settings, coordination support is hardly necessary, because participants can easily coordinate their actions through informal communication [Schmidt 2011, p. 97]. Collaborative web browsing and research present quite the worst case examples, with a typical group size of 2-4 people [Morris 2013, p. 1184], as well as usually uncomplex and informal tasks.

This constellation plays into the hands of another general problem for real-time collaboration support tools: Users always have an easy alternative to using the system, in the form of informal collaboration. Whenever they encounter even the slightest difficulty of use, it may already outweigh their perceived usefulness of the system, and they will default to generic forms of interaction, such as making a phone call or simply walking up to other people to talk to them in person [Grudin & Palen 1995, p. 276]. This is best exemplified by a quote from a 2013 survey, in which a participant stated that a collaborative browsing tool would only be interesting if it "was faster and required fewer mouse clicks than copying a link into an e-mail message." [Morris 2013, p. 1188].

Regarding *ease of use*, the context of collaborative web browsing also presents a number of modifying factors which increase the difficulty for support tool acceptance. First of all, this is simply the web-context itself. It has been shown that users expect much higher simplicity and ease of use for any application related to the internet. This goes so far that perceived usefulness and effective use can in fact be negatively correlated in TAM studies of online tools, simply because more useful tools are also likely to be more complex and therefore difficult to master [Dasgupta et al. 2002, p. 95].

A second problem is the "perceived critical mass" bias [Van Slyke et al. 2007], which is specific to all tools for communication support through non-standard protocols. It means the subjective impression of users, how many other people (in their environment) also use the application and how likely it is that more will use it over time. This strongly influences perceived usefulness, but also perceived ease of use, through the expectation of quickly receiving peer support in case of future difficulties.

Lastly, an important aspect to consider is that studies may have confirmed user appreciation of collaborative browsing support, but this does not necessarily mean that users are unhappy with their current ways of collaboration. In fact, 78% of participants in a recent study indicated satisfaction with their use of generic communication tools [Morris 2013, p. 1185]. This knowledge is crucial with respect to "production bias" [Carroll & Rosson 1987, pp. 80-82], i.e. the unwillingness of users to switch to a new system, as long as they still know a working method to produce the desired result.

Put together, the aspects listed above clearly show that achieving widespread user acceptance for a collaborative web browsing and research tool is inherently a very challenging task, especially regarding the required ease of use. Given the relatively low perceived usefulness, it seems unlikely that potential users will be willing to invest much effort into setup procedures and learning phases for a new tool.

This demands that both installation as well as the first steps of operation have to be extremely quick and intuitive, to leave the best possible first impression regarding ease of use. Brief interactive tutorials, which encourage exploration of program features, can help minimize frustration and at the same time help to demonstrate the usefulness of a new tool [Carroll & Rosson 1987, pp. 83-84]. As soon as this hurdle is taken, the mentioned production bias will gradually begin to work in favour of the application, as users are less likely to abandon a software they have already invested time into.

In order to overcome the perceived critical mass barrier, it should be considered to offer backward compatibility to pre-existing technologies, so that the entire setup process can be presented more like a valuable upgrade rather than a need to start over from zero [Grudin 1994, p. 100]. Ideally, users should retain any messenger contact lists or settings, and be able to interact with contacts who are still using a different system.

### 2.2.2 Session Start

Every collaboration tool has to provide a mechanism by which its users can contact each other and enter a collaborative state. This has become known as the "session paradigm" [Arantes et al. 2010, p. 124]. Each session is typically defined by its participants ("who"), a specific time frame ("when"), a physical or virtual location and its resources ("where"), as well as often a specific task or goal ("what", "why") [Arantes et al. 2010, p. 125].

Two common explicit mechanisms of session creation are *initiator-based* systems, where one person begins the session and then invites other known contacts to join, and *joiner-based* systems, where existing sessions are accessible through a repository and anyone with sufficient rights can join [Edwards 1994, p. 324]. Such explicit mechanisms are usually appropriate when the desired activity can be given a natural name, and when the required participants and methods are well-known [Edwards 1994, p. 324].

On the other hand, there are also implicit methods of session creation, such as *location-based* systems, which automatically initiate contact between users who visit the same physical or virtual space, or *artefact-based* systems, which initiate contact based on concurrent access of a specific resource like a document [Edwards 1994, p. 325]. The advantage of these systems is that they allow for informal and even serendipitous collaboration, without any manual administrative actions [Edwards 1994, p. 325].

A metaphor which has become very prominent for both explicit and implicit sessions is the "room" (e.g. chat room, virtual meeting room, virtual office, virtual workspace) [Greenberg & Roseman 2003, pp. 207-214]. Rooms ensure privacy and access control, allowing communication only between people who are currently inside. They also provide heightened awareness about who else is present and what they are doing, through event and status notifications. Lastly, rooms can act as containers for tools and artefacts which are part of the current activity inside the room. In some cases, people and artefacts can even be spatially arranged, to express their relations and the current state of work. If these configurations are stored persistently, rooms can also allow for very intuitive transitions between synchronous and asynchronous collaboration.

A downside of the room metaphor is that it does not capture spontaneous and informal collaboration very well. Firstly, users are usually unable to see what people in other rooms are doing, so that serendipitous collaboration is unlikely [Gutwin et al. 2008, p. 1414]. Secondly, rooms are inflexible, in that they must always pre-exist before collaboration can occur, either by explicit creation or by implicit natural existence.

A common scenario which is not covered by this metaphor involves two people who have already begun to work on a task separately, and then suddenly realize a chance for collaboration [Gutwin et al. 2008, p. 1412]. While most room-based collaboration systems allow users to join another person or group in another room and even move artefacts between rooms [Greenberg & Roseman 2003, pp. 225-226], it is generally not possible to fully merge two existing rooms and their content when necessary. Here, the strictly separated nature of rooms gets in the way of the actual abstract concept that collaborating people have in mind, namely that of common activities.

An entirely different approach, which is in many ways the exact opposite of the room metaphor, is the instant messenger (IM). While rooms express the "where" and "what" of a session, instant messengers focus exclusively on the "who". A session is initiated between two people, and more participants can usually be added to it later. Additionally, most messengers also provide peripheral awareness information about known contacts, such as their availability for interaction or even their current occupations.

Several authors have pointed to text-based instant messaging as one of the best examples of lightweight session initiation for informal tasks [Gutwin et al. 2008, p. 1421] [Nardi et al. 2000, pp. 83-84] [Carroll et al. 2003, p. 606]. For the initiator, contact is established instantly, without any formal setup or invitation procedure. For the recipient, first messages arrive only as an unobtrusive notification, which can be ignored until later. The downside of most instant messaging applications is that they are limited to simple communication functions and do not provide support for structured collaboration. That is why Carl Gutwin and Saul Greenberg, in their call for *lightweight initiation*, suggest the need for a hybrid approach [Gutwin et al. 2008, p. 1422]:

Collaborative tools should provide awareness information about the current activities of others. When people see an opportunity for collaboration, they should be able to initiate contact as easily and quickly as with an instant messenger. From this first contact phase, they should be able to move with little effort to a more advanced collaborative state, in which they can share their existing artefacts from their previous independent work.

People already regularly follow this pattern, using their generic communication tools. It has often been observed that collaboration originates in an informal fashion, such as through instant messenger or personal contact, then moves to email to develop ideas in a more organised and persistent fashion, and is then eventually implemented in a structured document or workflow management system [Muller et al. 2004, p. 375]. However, since these tools are usually not capable of sharing data natively, each transition leads to overhead effort, as participants have to manually copy and reformat their content. As a consequence, people tend to procrastinate the transition to a more appropriate medium until long after their collaboration has become inefficient, resulting in an even worse effective performance [Cockburn & Greenberg 1993, p. 34].

In conclusion, the ideal system for informal collaborative tasks such as web research should allow users to gradually increase the complexity of their collaboration, from peripheral awareness, to lightweight IM-like initiation, to a structured room-like state.

## 2.3 Transition

This subchapter focusses on all events which occur within a collaborative session and change the mode of collaboration between participants. Most commonly, these are transitions between phases of loose interaction (strong division of labour) and phases of tight interaction like group decision making, as already mentioned in section 2.1.4.

In order to analyse these transitions, it is first necessary to establish a general model of the processes within a collaborative session. This is the subject of section 2.3.1, in which I review different process and task analysis models from the field of HCI and specifically CSCW, in order to define the terminology for the remaining thesis.

Section 2.3.2 presents the mentioned distinction between loose and tight coupling as well as important concepts of seamlessly transitioning between the two work modes. Based on these concepts, section 2.3.3 takes a closer look into awareness support functions which can contribute to seamless transitions, particularly during web research.

### 2.3.1 Tasks

In the field of CSCW, the objective of task analysis and process modelling has to be approached from two different perspectives. One is the general perspective of HCI, which focusses on the interaction between people and the collaboration support tools they use. The second perspective is that of social science, which focusses on the interaction between different people and the ways in which they communicate and collaborate.

On the HCI side, *hierarchical task analysis* (HTA) is a common approach. It suggests that tasks can be recursively decomposed into a tree of subtasks, if necessary down to individual motoric actions [Annett 2003, p. 18]. Each task is specified by a *goal* with specific conditions and feedback parameters. Tasks of the same level are related by a *plan*, which defines their serial or parallel execution. Since HTA is primarily intended for process optimization of well-defined tasks, it is typically combined with models which predict how humans process a given task tree, such as GOMS [Annett 2003, p. 32].

On the side of social science, *activity theory* (AT) [Kuutti 1991] has become one of the most widely used models in the field of CSCW. An *activity* is defined as the smallest scope of work which carries "meaning", in that the motivation of participants can be understood without any further context information (typically synonymous to the term *session*, as defined in subchapter 2.2). In contrast, activities are comprised of *actions*, which may have *goals*, but only receive meaning through their context.

An activity is specified from the perspective of a *subject* who interacts with *objects* in the context of a *community* of other subjects. *Tools* mediate the relationships between subject and objects, *rules* mediate the relationships between subject and community, and *division of labour* mediates the relationships between object and community.

While actions are thought of as hierarchical, similar to tasks in HTA, there is no notion of programmatic execution. Rather, the entire interdependent system of subject, objects and community is jointly moved towards a desired state by the execution of actions, as determined by rules and division of labour within the dynamic context. AT is therefore not a predictive model like HTA, but primarily a descriptive model of work relationships.

A popular unification of HTA and AT for the field of CSCW has been developed under the name *groupware task analysis* (GTA) [van der Veer et al. 1996]. Throughout my thesis, I adopt its terminology and definitions.

GTA retains the hierarchical *task* as the basic concept of work, defined by a *goal*. However, tasks are no longer decomposed down to motoric actions, but only to the level of "unit tasks", which are the smallest scope within the abstract concept that users have of their work, independent of the system they are working with. Unit tasks may then be decomposed further into *user actions* and *system events* to describe the interface level.

To specify the social environment of tasks, GTA introduces the common concepts of *actors* and *roles*, as well as *organization*, which is the set of relations between the two. Complex tasks can be split up between actors according to predefined or situated roles. GTA further distinguishes between *rules*, which are defined explicitly by the actors of a task (but may not necessarily be followed in practice), as well as *protocols*, which describe the de-facto way in which people choose to interact repeatedly and predictably.

From activity theory, GTA retains the concept of objects, which can be either material or virtual entities, and which can be gradually transformed during task execution. However, the object concept of GTA is more unified and closer to object-oriented software development, in that there is no distinction between subjects and objects, but both actors and proactive automated systems are simply defined as "active objects".

By using an object type hierarchy, GTA can define arbitrary influence relationships between different object types as well as between tasks and object types, allowing models to capture very dynamic collaborative processes. Compared to the simple stack model of HTA, this makes task processing less predictable, but arguably more realistic, especially for less formal tasks with loosely defined goals and rules.

As multiple authors have pointed out, user goals during informal tasks "are not organized in any strict hierarchy and are radically transformed as events unfold" [Whiteside & Wixon 1987, p. 359], and groups may even change entire task definitions in response to external or internal events [Mandviwalla & Olfman 1994, pp. 252-253]. Therefore, many researchers have recommended that developers of collaboration support systems should identify common low-level tasks as well as their relations and support them in a flexible way, but not attempt to model strict workflows for high-level tasks [Schmidt 2011, p. 60] [Bernstein 2000, p. 280] [Grudin 1994, pp. 98-99].

Web research is a prime example of a task which does not follow a regular hierarchical structure, but unfolds dynamically during its execution. This is mostly due to the fact that researchers often do not even start out with a well-defined goal and success condition, but rather with an abstract motivation, according to the *anomalous state of knowledge* hypothesis [Kuhlthau 1991, p. 362]. It describes a researching person as someone who realizes a need for information and will recognize proper information once found, but who is at first unable to determine a straightforward process to acquire it. Therefore, the research task has no predetermined structure and continues indefinitely until it either succeeds or the person gives up. Utilized methods may be chosen and gradually refined dynamically, depending on acquired knowledge and experience. However, this does not mean that there are no reoccurring subtasks and subtask relationships at all, some of which have already been described in section 2.4.1.

A system to support collaborative web research must therefore present a work environment which is inherently stateless and support users in the execution of common subtasks (e.g. foraging, sensemaking) at any time, in any order, and even in parallel. Roles do not have to be considered explicitly, as small groups can coordinate informally. Still, the formation of ad-hoc roles such as leaders and followers should be possible.

### 2.3.2 Coupling

The term coupling can be seen from two different perspectives in the context of CSCW, one being the user's requirements and the other being the capabilities of a support tool.

From the user perspective, coupling means "the amount of work that one person can do before they require (…) consultation with another person" [Baker et al. 2001, p. 132]. During tight coupling, participants require constant exchange of information and simultaneous access to task objects. Inability to communicate promptly leads to a breakdown in work progress [Olson & Teasley 1996, p. 422]. At the other extreme, during loose coupling, work proceeds primarily in parallel and requires little exchange in information or awareness of the actions of others. Inability to use resources independently leads to conflicts and stalls in work progress [Olson & Teasley 1996, p. 422].

From the system perspective, coupling describes the extent to which resources and interface actions are synchronized between users [Dewan & Choudhary 1995, p. 2]. With a tightly coupled groupware system, users might for instance always see the exact same screen content, even when working on physically distinct computers (WYSIWIS). A more relaxed coupling might only synchronize some windows or allow independent scrolling of synchronized window content. Very loosely coupled systems perform no automatic synchronization at all, but simply notify users in the event of updates.

Typically, tightly coupled work is best supported by tightly coupled systems and loosely coupled work by loosely coupled systems. The challenge lies in the fact that many collaborative tasks do in fact require both modes in constant alteration, as people often choose to divide labour most of the time, but come together intermittedly to coordinate their progress or share their results [Scott et al. 2003, p. 165] [Baker et al. 2001, p. 132] [Gaver 1991, p. 295]. This is what Mark Stefik and Hiroshi Ishii addressed in their calls for *seamless transitions* between solitary and group work [Ishii & Miyake 1991, p. 37].

The TeamWorkSpace system by Ishii & Miyake presented a very straight forward solution, by providing users with two screens, one private and one shared between all team members. This allowed workers to act solitarily most of the time, but very elegantly switch to tight collaboration by moving content to the shared screen. Other authors have made similar suggestions to divide workspaces into private and shared areas and allow content to be moved between them [Scott et al. 2003, p. 165] [Sarin & Greif 1985, p. 33].

Past implementations have usually approached the problem either from the side of loose coupling, by introducing limited collaboration into existing single-user software [Hupfer et al. 2004, p. 21], or from the side of tight coupling, by introducing features such as individual note-taking into groupware [Mandviwalla & Olfman 1994, p. 246]. Generally, well-working solutions have often been very task-specific.

In the case of collaborative web research, one important specificity to notice is its already very high frequency of subtask transitions for each person. During foraging phases, avid web researchers have been observed to spend only an average of 12 seconds on each web page [Hawkey & Inkpen 2005, p. 1445]. The emergence of tabbed browsing has additionally introduced a very high degree of parallelism and task branching behaviour into the research process [Huang et al. 2012, p. 203].

A collaboration support system for web research must integrate its coupling policy with these existing work patterns in a natural way, allowing unobstructed use of multiple tabs during loosely coupled foraging phases, but also tightly coupled discussions of specific page content during sensemaking and organizing phases.

### 2.3.3 Awareness

Awareness is an umbrella term in the field of CSCW, which encompasses all forms of perception and comprehension [Endsley 1988, p. 97] of entities within the current work environment of a person. This then provides context for this person's own activity [Dourish & Bellotti 1992, p. 107]. Numerous subcategories of awareness have been defined in CSCW literature and also social science literature [Gross et al. 2005].

Though partly incoherent in detail, these definitions roughly fall into two categories: Firstly, awareness of *people*, their physical or virtual presence (informal awareness), their interests, intentions, emotional or attentional state (social awareness), as well as their roles and occupation regarding the collaborative task (group-structural awareness). Secondly, awareness of the *workspace*, including the current task and solution strategies, as well as the presence, meaning and change of artefacts [Gutwin & Greenberg 2002].

Other fundamental categorizations include the distinction between synchronous (What is happening?) and asynchronous awareness (What has happened?), as well as tightly coupled (What is happening at my focus of attention?) and loosely coupled awareness (What is happening elsewhere?) [Borghoff & Schlichter 2000, p. 177].

Collaboration support systems can improve awareness in both passive and active ways [Gutwin & Greenberg 2002, pp. 423-424]: Passive awareness support is achieved by providing users with good mechanisms for intentional communication (speech, gestures) as well as consequential communication (making actions easily observable for others). Active awareness support is provided by *feedthrough* systems, which directly notify users when there are relevant changes within the workspace or in the status of other people.

For simple synchronous collaboration tasks, such as cobrowsing, systems only need to provide very little active support, since many awareness aspects are covered by informal communication [Ellis et al. 1991, p. 52]. Regarding awareness of people, only *informal awareness* must be supported actively (i.e. "the pervasive experience of who is around, what these persons are doing") [Gross et al. 2005, p. 327]. Regarding workspace awareness, some form of *action awareness* must be provided (i.e. knowledge of what actions are being performed by group members) [Carroll et al. 2003, pp. 611-612].

However, the implementation of these active support strategies always depends on the task at hand. For the case of cobrowsing, several specific subcategories of awareness have been defined in literature: Most importantly, users must be able to see which web pages other group members are looking at (*location awareness*), in order to allow efficient division of labour while foraging, as well as coordinated co-viewing of pages in sensemaking phases [Gianoutsos & Grundy 1996, p. 16]. The ability to determine if a page has already been seen by other group members (*visitation awareness*) is important to reduce redundant effort in foraging phases [Morris 2008, p. 1660]. This form of awareness can additionally be supported in passive form, by allowing page annotations as a form of communication [Gianoutsos & Grundy 1996, p. 16]. This allows users to share specific assessments of page content. Finally, in organizing phases, users must be able to associate each result with the person who found it, as well as its page of origin (*attribution*) [Wexelblat & Maes 1999, p. 272] [Paul & Morris 2009, pp. 1775-1776].

As a last side note, it should be mentioned that the term "awareness" is also often used in software engineering. A "location aware" system has access and responds to the location of its user. However, for the scope of my thesis, both meanings are strongly intertwined, in that any location aware subsystem is always related to user location awareness.

## 2.4 Integration

This subchapter focusses on all events which occur at the unit task level (as defined in section 2.3.1) of a collaborative web browsing or research session. Many of these low level tasks require user interaction with both the standard web browsing functions as well as the collaboration support functions presented in the past subchapters. Therefore, such tasks potentially benefit from a tight integration between the respective user interfaces, allowing users to complete them in less time and with less cognitive effort.

With respect to the technology acceptance considerations made in section 2.2.1, this effort reduction crucially determines the perceived usefulness of cobrowsing tools. Because collaborative browsing and research can also be performed acceptably using existing software in the eyes of many users, the most obvious advantages a groupware solution has to offer are therefore simplicity and speed.

The first upcoming section 2.4.1 determines which common unit tasks are most likely to receive a performance boost from integration. The two remaining sections present possible technical solutions. Section 2.4.2 investigates possible extensions of the browser application user interface, while section 2.4.3 does the same for extensions of any web page content loaded within the browser tabs.

### 2.4.1 Task Performance

At the unit task level, GTA is effectively identical to single-user task analysis, so that common predictive performance models can be applied. The base of my following considerations is the common keystroke-level model (KLM) [Card et al. 1980].

From the known scenarios for collaborative web research (chapter 1), it should be fair to assume that unsupported users are often using at least three separate applications simultaneously: the web browser, a text communication tool (email, chat) and an office application to gather and organize results. The copy-paste function of the underlying operating system is most likely used to transfer content between these applications.

A copy-paste sequence between two maximized applications on a Windows computer can be formulated in the KLM as $MH_{[mouse]}\ P_{[source]}\ K_{[mouse]}\ H_{[keyboard]}\ K_{[ctrl]}K_{[c]}\ MP_{[taskbutton]}\ K_{[mouse]}\ P_{[dest]}\ K_{[ctrl]}K_{[v]}$ which would add up to 8 seconds of execution time. This is certainly a very rough estimate and will vary greatly between users and specific content to be transferred, but it gives an idea of the complexity and cognitive overhead required. From here on, I am only using the copy-paste cycle as an abstracted base unit.

Using copy-paste cycles as a metric, the most expensive unit tasks include *referencing specific web page content in results / message*s, which requires copying both the content and usually also the current URL for source attribution. The reverse task is *finding referenced web page content*, which includes copying the URL to the browser (it may not always be a clickable link) and then the referenced content to the search field.

A second tier of tasks which always require one copy-paste includes *referencing an entire page in results / messages*, *navigating to a referenced page*, and *copying content between results and messages*. Lastly, a third tier of tasks do not require copy-pastes but still require switching to a different application, such as *reading and writing messages* or *reviewing results* while the browser is currently in foreground.

However, the overhead effort for tier 3 tasks can be mitigated by aligning windows side by side, whereas tiers 1 and 2 can only be truly improved by proper integration.

### 2.4.2 Contextual Collaboration

As described briefly in section 1.2.3, contextual collaboration is the approach to extend the graphical user interface of single-user application with additional groupware functionality [Hupfer et al. 2004, p. 21]. Retrofitting software like this used to be technically challenging, but has become much more feasible over the past years, as applications are now increasingly modular and allow plug-in code to be loaded dynamically at start-up or even at run-time [Cheng et al. 2005, pp. 148-149].

Contextual collaboration is different from adding classical *collaboration awareness* [Dewan & Choudhary 1995, pp. 177-179] to software, in that the core functionality of the application is not modified to be collaborative, but only extended with new collaborative features, which integrate contextually with the application content. A common example is an integrated chat, which can be used to annotate and discuss selected text passages in an office application, while the actual text editing is still controlled by a single user.

This approach seems fitting for collaborative browsing, since actual concurrent control over a web browser instance is not required and may even be detrimental, according to the requirements established in the past subchapters. Leaving the core browser functions untouched also decreases the chance of incompatibility issues in case of future browser software updates, which are very frequent today.

Any tasks categorized as tier 3 by the past section can be easily optimized by this integration approach, since both messaging as well as result collection could be housed in a single unified interface within the browser, so that the need to switch between applications is eliminated. However, as mentioned, this is no categorical advantage over aligning windows side by side, unless it leads to other synergistic effects. Reduced flexibility compared to free window positioning might even become a disadvantage.

Tasks from tier 2 can also be optimized by this approach, if messenger and result collection are made aware of the web page which is currently loaded in the browser, so that manual copying and pasting of URLs can be combined into single interface commands. The same potential exists for moving content between results and messenger.

In order to improve the performance of tier 1 tasks, messenger and result collection have to be integrated not just with the browser UI, but also dynamically with the content of each loaded web page. This requires a specialized approach, which is not part of contextual collaboration in general and is presented in the next section.

### 2.4.3 Web Augmentation

Comparable to augmented reality in the physical world, web augmentation is the concept of overlaying or modifying 3rd party web pages in the browser, in order to customize them for the local user or dynamically combine them with other services [Díaz 2012, p. 79]. This is achieved by injecting additional JavaScript code or style sheets into the pages as they are loaded, using local browser extensions, frames or proxy servers.

The same principle provides a solution to improve tier 1 task performance, by allowing bidirectional low-level access to the web page content. Common JavaScript APIs enable page code to capture input events from mouse and keyboard, which facilitates selection of content passages by the user. Conversely, content structures can also be navigated, searched and modified programmatically, facilitating content indication and markup.

The challenge is to develop a concept which combines both extensions to the browser UI as well as the page content into a unified contextual collaboration solution.

## 2.5 Summary

This chapter has defined the exact purpose and scope of my work, as well as the specific requirements for the upcoming design and implementation of CoopFox.

Subchapter 2.1 defined this scope as *synchronous distributed collaborative web research*. The term *research* was distinguished from general browsing as an activity which aims to satisfy a concrete pre-existing information need in the sense of the anomalous state of information hypothesis. The process of obtaining the required information is a complex iterative progress, which has no straightforward solution and involves access to and use of many separate web pages. Still, several common phases of collaborative research can be identified: *foraging*, *sensemaking*, *organizing* and *distribution*. Each of these has its own set of common subtasks and must be supported in different ways.

Subchapter 2.2 focussed on *initiation* and developed requirements for all program aspects beginning at *installation* and leading up to the start of a collaborative *session*. A key problem in this area is to achieve user acceptance, given that *perceived usefulness* of synchronous online collaboration tools is generally low, and so is the threshold for acceptable *perceived ease of use*. Combined with the known critical mass barrier and production bias, a successful solution has to be a backward-compatible extension of existing technologies, with quick setup and very easy use. This lightweight approach is consistent with the requirements for *session initiation* of informal collaborative tasks, such as web research. As most interaction begins spontaneously or even serendipitously between people, the solution must abstain from strict room metaphors, but offer a first contact method focussed on people, similar to *instant messaging*. It must further allow users to quickly *combine their independent work* states. However, such an informal session should still be able to evolve into a structured state when necessary.

Subchapter 2.3 focussed on *transition* between solitary work (*loose coupling*) and group work (*tight coupling*). In general, foraging phases offer themselves for division of labour strategies and therefore profit from a loosely coupled collaborative interface, allowing independent navigation in *multiple tabs*, and providing active support for decoupled awareness (*location awareness*, *visitation awareness* and page *annotation*). Sensemaking and organizing phases profit from a tightly coupled interface, allowing fine-grained discussion of page content, as well as *result management* with attribution. The main challenge lies in the fact that web research tasks have *no predictable workflow*, meaning that loosely and tightly coupled work occur in rapid alteration. This calls for an interface concept which is inherently stateless and allows users to engage in any supported subtask (tightly or loosely coupled) at any time and even in parallel.

The final subchapter 2.4 focussed on *integration* between browser, messenger and result collection functions, as provided by separate applications in unsupported settings. Several low-level tasks were identified, which have potential for faster execution in an integrated user interface, *eliminating manual copy and paste* transfers. The greatest benefit is achieved for tasks which involve *fine-grained references* to web page content. In order to tab into these synergistic effects, it is necessary to *embed* both the messaging interface as well as the result collection within the browser UI and enable the exchange of web location information between these systems. The content of each loaded web page must also be included in this integration concept and must be augmented with interactive functionality, to provide the desired ability for fine-grained referencing.

# 3 Related Work

This chapter presents a review of past cobrowsing solutions and research prototypes, with a particular focus on the requirements established in chapter 2. Each subchapter is dedicated to a different system, following the chronological order of their publication. This order was chosen because the features and concepts of each system were strongly influenced by the changes in available technology over the years, and also other related work which had been published before. To classify common interaction concepts throughout this chapter, I often refer to the compilation of design patterns for CSCW, developed by Till Schümmer and Stephan Lukosch [Schümmer & Lukosch 2007].

## 3.1 ComMentor (1995)

One of the first academic groupware experiments for web browsing was ComMentor, which used a modified version of the NCSA Mosaic web browser [Röscheisen et al. 1995] and a custom server to provide shared annotation [Schümmer & Lukosch 2007, p. 291]. Similar ideas had already been discussed in the Mosaic community before [NCSA 1993].

ComMentor allowed users to attach text annotations to arbitrary content elements on web pages. Annotations could also be annotated themselves, creating a simple reply mechanism. In addition, users were able to group annotations into ordered sets and share them with other users or user groups, effectively creating guided web tours.

Tightly coupled collaboration was not supported well, as annotations would only be retrieved from the server whenever a web page was loaded. On the other hand, carrying over independent work into collaborations was easy, because no distinction was made between private and shared annotations. Users were able to begin an annotation set as simple private notes, and then give read or write access to other users or user groups later.

## 3.2 GroupWeb (1996)

Presented in April 1996, GroupWeb was one of the earliest academic projects on synchronous collaborative browsing [Greenberg & Roseman 1996]. Visited web pages were tightly synchronized between group members. Independent window sizing and scrolling was possible, unless users actively chose to lock their viewport to that of another person. Remote scroll bars [Schümmer & Lukosch 2007, p. 342] provided awareness of the current viewport position of other participants while relaxed scrolling was active.

GroupWeb provided shared annotations, but not as fine-grained and contextual as ComMentor. The UI contained a shared text editor [Schümmer & Lukosch 2007, p. 219], which stored content separately for each visited URL, so that users were able to make collaborative notes and comments to each page, and also exchange text messages.

Tightly coupled interaction was further supported through real-time telepointers [Schümmer & Lukosch 2007, p. 359], by which participants were able to gesture and guide the attention of others. However, these did not function as remote cursors [Schümmer & Lukosch 2007, p. 353], but merely as a method for passive pointing.

GroupWeb was built using the GroupKit framework as a stand-alone web browser, which further required a specialized server with registered user accounts. Sessions had to be created explicitly and could then be joined by other users.

## 3.3 W4 (1996)

The World Wide Web for Workgroups (W4) browser was presented in August 1996 and was the first system to provide flexible transitions between loosely and tightly coupled work [Gianoutsos & Grundy 1996]. From the technical perspective, it was built similarly to GroupWeb and also included the same major features for tightly coupling, such as telepointers and remote scroll bars. The annotation system of W4 was closer to ComMentor, with notes and even chat windows or collaborative whiteboards attachable to individual page elements. These widgets updated in real-time for other group members.

Users were free to visit different pages at the same time, unless they explicitly chose to follow another user, in which case both page transitions and scrolling were coupled. Spontaneous transition into tight coupling was further possible by location. Widgets like the telepointer activated automatically whenever two users visited the same page.

In order for participants to be able to coordinate during loose coupling, the system provided support for decoupled awareness. This included a page history view for each participant, providing location and visitation awareness, as well as the ability to jump to any page another user had seen before. A shared bookmark list allowed participants to jointly collect important pages. A global text chat enabled cross-page communication and a global collaborative text editor allowed for results to be collected and organized.

The system further provided visual and audible feedthrough notifications for actions of users on different pages. For instance, if a note was added to a page, a notification message would also appear in the global chat window (if enabled by the author).

## 3.4 CoWeb (1996)

A third cobrowsing system from 1996, CoWeb, was the first to feature a form of lightweight session initiation [Jacobs et al. 1996]. While all of the previous systems relied on a specialized web browser and server with pre-existing user accounts, CoWeb was a Java applet based web application, which could be started from a standard browser by visiting a special start page. Any person was able to join the collaboration instantly without authentication. Explicit sessions did not exist, and all users who were simultaneously connected to a server were interacting with each other.

The applet allowed dedicated collaboration on single pages, which had to be manually requested using the “coophttp://“ URL schema. These pages would be loaded via the java server, where they were modified by exchanging common HTML elements with smaller java applets, which provided a range of groupware functions depending on the element they replaced. This included collaborative drawing of overlays for images, or collaborative text editing for input fields.

CoWeb offered different coupling modes which users could switch between manually. In WYSIWIS mode, page navigation and scrolling were synchronized between participants. A more relaxed mode allowed independent scrolling while collaborative elements were still synchronized in real time. A time-relaxed mode allowed independent editing until a commit button was pressed manually.

Finally, a private mode allowed fully independent page navigation between users. Similar to W4, a global chat between all connected users allowed for communication between loosely coupled participants. A shared document history provided location and visitation awareness, enabling users to join each other on pages or revisit them.

## 3.5 CoNavigator (1997)

Part of the larger MetaWeb toolkit project [Trevor et al. 1997], CoNavigator also used a lightweight login mechanism, running as a java applet in a standard browser. It focussed on very loosely coupled and even serendipitous collaboration, as users only became aware of each other automatically, when they visited the same web page at the same time. However, an additional global search function allowed users to find other people, see their current web location, and even obtain contact details for messenger applications.

Once two or more users met on a page, they automatically became part of an implicit session, providing a group chat. In addition, they could also explicitly launch "sessions" with each other, which was a MetaWeb term for a variety of small collaborative tools in separate windows, such as collaborative text editors or drawing boards. All of these features were not interconnected, however, and meant for quick disposable interaction.

## 3.6 CSCW3 (1998)

The CSCW3 prototype focussed on loosely coupled work similarly to CoNavigator, also using implicit page-based sessions ("rooms") with a local chat, combined with global search functions [Gross 1998]. However, more asynchronous and decoupled awareness information was provided in the context of pages, such as which other pages users had visited directly before and after viewing the current page, as well as how long they viewed each page. This data could also be visualized and shared, in the form of navigation history maps, showing the paths taken through web pages by individual users.

Limited tight collaboration was supported through the ability to couple multiple browsers together and visit pages synchronously. Real-time coordination with other people was further possible by setting up a persistent dedicated chat, sharing bookmarks or page annotations. The combination of these features was intentionally left open to the user, to allow for many flexible collaboration strategies during informal use.

## 3.7 Proxy Approaches (1999 & 2000)

Around the year 2000, several technologically similar solutions were published, which employed a reverse proxy server to inject java applets into arbitrary loaded web pages. This required the user to manually change their browser configuration once, but worked as a persistent solution, without modifying the standard browser itself. As a downside, activating and deactivating the functionality spontaneously was impractical. The provided interactive functions were quite standard at this point, such as switching between loosely and tightly coupled navigation, text chat, as well as page annotation. However, each solution presented very different and innovative awareness features.

Footprints [Wexelblat & Maes 1999] displayed a radar-like map, showing navigation routes from and to the current page, as well as how often they had been used. Both pages and routes could be annotated, allowing users to comment on the relations between pages. Cooperative Proxy [Cabri et al. 1999] chose a more contextual approach, modifying the visual style of links in delivered pages, to indicate whether other group members had already visited them. E-coBrowse [Chong & Sakauchi 2000] introduced a different kind of telepointer, in the form of a miniature overview of the current web page, indicating the mouse position of each co-located user as a colour-coded marker.

## 3.8 UsaProxy (2007)

The proxy concept was revisited in 2007 through UsaProxy [Atterer et al. 2007]. Due to improved web technology, it no longer required java applets but only JavaScript with AJAX support. The additional scripts were injected into each loaded page by the proxy. This also made it feasible to integrate the delivery directly with the server of a web site, providing instant local collaboration in arbitrary browsers without setup.

The system also first addressed the problem of personalized, login-protected and dynamic web pages. Whenever a document was requested by one browser, the proxy ensured that an identical copy was also delivered to all other session members. Further, any user input events and dynamic page alterations were replicated between page copies. Due to this tightly synchronized approach, strict WYSIWIS (switchable between a master-master and master-slave mode) was the only collaboration form available.

UsaProxy provided a telepointer, which acted as a true remote cursor, due to the input event slaving. As a side effect, users were often observed using text selection as another pointing method, since their mouse selections were also replicated across browsers. Further collaborative features included the common chat box and shared whiteboard.

Session initiation was lightweight, as soon as the proxy setup was complete. Users accessed a special start page, presenting a list of all other users who were currently browsing through the proxy. Coupling of two browser windows then followed a simple invite-accept handshake. In addition, it was possible to generate and send http links, which directly opened a coupled browser with a specific user. Ending a collaborative session was possible at any time, simply by closing the embedded chat box.

## 3.9 SearchTogether (2007)

The first work to specifically address collaborative web research, in the sense of my definition (2.1.4), was SearchTogether [Morris & Horvitz 2007]. However, the system focussed very strongly on the use of search engines as the central task and provided few of the established collaborative browsing features, besides an integrated text chat.

Collaboration was always loosely coupled, including tabbed browsing, with no direct awareness of the web pages currently viewed by other group members (which turned out to be one of the most common complaints among test users). On the other hand, users were made aware of search queries already performed by others. In addition, search result listings were extended with avatar pictures of users who had already seen the respective pages. Hovering the mouse atop an avatar revealed the exact visitation time.

A voting system (thumbs up/down) allowed users to rate results after viewing them, indicated to following group members through icons in their result listings. Annotation of result pages was provided and visualized in a similar way. A separate summary tab provided a sorted list of all results which had been annotated or rated as relevant by any session member. Alternatively, users could also directly recommend pages to specific members, which would then show in a recommendation queue in their browsers.

Division of labour was supported actively through a “Split Search” button, which synchronously executed a search query in a new tab of all browsers and automatically assigned each result to a different group member for evaluation. However, in user studies, this feature was rarely used. Participants rather chose to coordinate through the various annotation functions and also praised these capabilities in their explicit feedback.

SearchTogether was implemented as an extension to Internet Explorer 7 and used the Windows Live Messenger protocol for communication, thus allowing most Windows users to keep their standard web browser and existing messenger contact list. Sessions had to be created explicitly before collaboration could begin, by first specifying a topic name as well as known contacts from the buddy list who would participate. A once created session was persistent on the server and allowed asynchronous collaboration.

In 2009, the same research team created CoSense, an additional tool designed to support sensemaking and organizing phases [Paul & Morris 2009], which worked on session data previously created during a foraging phase with SearchTogether. The tool generated a variety of different visualisations from this session data.

A timeline view showed the combined history of the search process, including both chat messages as well as search queries and page visitations of all members, in order to retrace the collaborative process as a whole. A chat-centric view associated each chat message with the web page the sender was looking at while writing. Both views allowed users to re-visit any of the referenced pages, simply by clicking the respective entry.

A separate workspace view allowed users to organize and edit the foraged results from SearchTogether, by editing comments and ratings or assigning tags. It also featured two collaborative text editing areas for notes and to-dos. Re-importing these modified results into SearchTogether was not intended, suggesting that the authors did not expect an iterative alteration between foraging and sensemaking/organizing phases for their tools.

## 3.10 PlayByPlay (2009)

Seamless coupling mode transitions during informal cobrowsing were addressed explicitly by a prototype called PlayByPlay [Wiltse & Nichols 2009]. The author's innovative approach was to integrate browser content synchronisation with the instant messenger concept. Instead of immediately replicating all user actions in remote browsers, they were printed out as human readable descriptions in the text chat box next to the browser content (e.g. "went to www.google.com", "entered 'cafe' into the search box", "clicked the search button"). This allowed other group members to either follow these actions of others synchronously, or focus on their own work and reconstruct the events later by reading through the chat history. Individual actions could also be replicated in the local browser at any time, by clicking an icon next to the respective chat message, for instance to jump to a page visited by another group member. Alternatively, users could explicitly enable a tightly coupled mode, in which all actions performed by another person were automatically replicated in their local browser. To respect the privacy of users, there was also an option to pause the local recording of actions at any time.

For synchronous location awareness, PlayByPlay displayed miniature images of the web pages currently seen by remote session members. Users were further able to take small screenshots of web page content and post them to the chat using a special "clip" function, in order to show others exactly what they were currently seeing on their screen.

Usability tests with PlayByPlay generally yielded positive results, but also revealed a number of problems. Action descriptions in the chat were too plenty and too fine-grained to be followed by others. Their meaning was also sometimes hard to comprehend, depending on the quality of page element captions. Participants further often expressed a lack of feedback on whether other group members were currently in sync with them.

Based on the instant messenger analogy, PlayByPlay session initiation worked simply by opening chats with contacts from a buddy list [Schümmer & Lukosch 2007, p. 109]. More than two session participants were not supported. The used message protocol was non-standard, so that account creation on a specialized server was required. The client application was a Firefox extension, although a limited HTML-only version also existed.

## 3.11 CoFox (2012)

Another approach to seamless transitions was CoFox [Rodriguez Perez et al. 2012]. Instead of capturing user actions, the system captured and broadcasted the entire window of each person as a video stream. Only one remote user was supported and the respective video stream was presented side by side with the local tabbed browser.

Telepointing by moving the mouse over the remote video was not implemented, although the authors had hinted at this possibility before [Rodriguez Perez et al. 2011]. This earlier proposal also included the ability to rewind both the remote as well as the locally recorded video, to review the process. An annotation function was proposed, which would allow users to add comments to specific positions in the video history.

The evaluated version from 2012 only contained an embedded chat, as well as a separate shared link list to collect results. Still, test results showed significant improvements in both research performance as well as user satisfaction, compared to the use of either separate browsers or a single shared browser. CoFox can therefore be seen as an example for the great impact that synchronous awareness alone can have on collaborative research, as well as the general demand for synchronous cobrowsing tools.

## 3.12 Commercial Solutions

Over the recent years, several commercially developed solutions for synchronous collaborative web browsing have emerged. However, most of these focus on very tightly coupled work, effectively providing instant screen-sharing of a single browser tab, very similar to the implementation of UsaProxy. Typically, these products are JavaScript based and integrated directly into a single web site, for the purpose of instant customer support (e.g. LiveLOOK, Firefly, unblu, synchronite, CoBrowser.net).

One of the few exceptions is Samesurf, which is usable with any web site, due to its implementation as a stand-alone modified Google Chrome browser. Like many academic prototypes, it provides an integrated chat and real-time telepointers. But like the other commercial products, it only supports tight coupling, with one single tab shared using a master-slave policy. Participants can take turns at being in control, but slave browsers are merely able to point passively or browse independently in separate unshared tabs.

Another similar service is called CoVu, which originally started out as a very ambitious project in the form of a social network, offering asynchronous messaging and asynchronous link sharing, as well as tightly coupled navigation on demand, as evident by the demonstration videos still available on YouTube. However, the currently available version only provides simple tight URL synchronization between browsers.

Session initiation with either Samesurf or CoVu does not require registered user accounts. Once the applications are launched, a session with a unique ID is automatically created, which other users can enter in their respective instance to join.

## 3.13 Summary

| | Lightweight Initiation | | | | Seamless Transitions | | | | | | | | | Context. Collab. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Std. Browser | Std. Server | Spontaneous Init. | Carry Over Work | Loose Coupling | Tight Coupling | Transition L/T | Informal Aw. | Action Aw. | Location Aw. | Visitation Aw. | Multiple Tabs | Result Mgmt. | Embedded UI | Annotation | Referencing |
| ComMentor | − | − | O | + | ++ | −− | −− | − | − | − | O | − | + | − | ++ | ++ |
| GroupWeb | − | − | − | − | − | ++ | − | + | + | ++ | O | − | − | + | + | O |
| W4 | − | − | − | − | + | ++ | ++ | + | + | + | + | − | + | + | ++ | + |
| CoWeb | ++ | −− | ++ | O | + | + | + | + | O | + | − | − | − | O | O | O |
| CoNavigator | ++ | − | + | O | ++ | O | ++ | ++ | O | + | O | − | − | O | − | O |
| CSCW3 | − | − | O | + | ++ | + | ++ | ++ | O | + | ++ | − | + | + | + | + |
| Coop. Proxy | O | −− | − | − | + | + | + | + | O | + | ++ | − | − | ++ | − | O |
| Footprints | O | − | − | − | ++ | −− | −− | O | − | − | ++ | − | − | + | + | O |
| E-coBrowse | O | − | − | − | O | ++ | + | + | O | O | O | − | − | ++ | ++ | O |
| UsaProxy | ++ | − | ++ | O | −− | ++ | − | + | ++ | O | − | − | − | + | O | ++ |
| SearchTogether | + | + | − | − | ++ | − | − | ++ | O | − | ++ | + | ++ | ++ | + | + |
| PlayByPlay | − | − | + | O | + | + | + | + | ++ | O | O | O | − | ++ | O | ++ |
| CoFox | − | − | − | − | ++ | O | ++ | + | ++ | ++ | O | + | + | ++ | − | + |
| Samesurf | O | O | + | O | −− | ++ | O | ++ | + | ++ | − | O | − | ++ | − | O |
| CoVu | ++ | O | ++ | O | − | + | − | + | − | + | − | − | − | + | − | − |

Table 1. Comparison between past cobrowsing solutions, regarding requirements established in chapter 2. The ++ grade is awarded for fulfilling a requirement in particularly sophisticated or very easy to use ways (e.g. location-based tight/loose switching; page element annotations). The O grade states that the requirement is not fulfilled, but users can achieve the effect using workarounds (e.g. carrying over work by revisiting the private history in collaboration). The −− grade signifies that the approach is entirely antithetic to the requirement (e.g. enforcing strict WYSIWIS). Green highlighting signifies that a solution overall excelled regarding one of the three key principles (several ++) without having significant drawbacks (−).

Almost all past solutions have excelled regarding at least one of the key principles from subchapter 1.2, but none have systematically addressed all three of them together (Table 1).

Especially lightweight initiation and seamless transitions are never provided very well at the same time. One group of systems (CoWeb, CoNavigator, UsaProxy, CoVu) has focussed on quick but simple collaboration, usually visiting and operating pages in a WYSIWIS fashion. A second group of systems (W4, CSCW3, CoFox) permitted flexible and complex long-term collaborative work, but setup and session initiation were usually slow (assuming the system is not used as the standard browser in every-day web activity).

No system has yet provided an outstanding solution for users to begin research independently and carry over results into collaboration. SearchTogether stands out as the only system to use a standard server and by providing active result management support.

# 4 CoopFox Concept

This chapter presents the interaction design for CoopFox. Because the development of my solution began prior to this master thesis, in a student project under the name SpreadVector, the first subchapter 4.1 briefly describes the design of this forerunner prototype as well as initial user feedback gathered during this phase. The remaining subchapters then develop the design of CoopFox, following the established trisection into concerns of initiation, transition and integration, where each subchapter presents my solutions to the requirements established in the respective part of chapter 2.

Where inspiration was taken from a related work (chapter 3), I highlight similarities and differences of my design. For design aspects which were already part of SpreadVector, I point out how they have been improved or extended in CoopFox. Like in the previous chapter, the catalogue of design patterns for CSCW by Schümmer and Lukosch is once more used as an important general source of reference [Schümmer & Lukosch 2007]. It further goes without saying that CoopFox was also designed respecting common guidelines for HCI systems [Wasserman 1973] [Nielsen 1994, p. 30].

## 4.1 Early Prototype (SpreadVector)

The purpose of my initial student project was to explore interaction concepts for synchronous cobrowsing in a modern multi-tabbed environment, with primary focus on casual loosely coupled work. From a technical perspective, I also wished to explore possible implementations for cobrowsing systems based entirely on open-source technologies for both clients and server. The client-side was implemented as a Firefox extension, since this approach provided the most powerful API, allowing complex collaborative features. I chose XMPP for communication due to its flexibility and extensibility, which allowed me to use an unmodified standard open-source server.

The interaction design for SpreadVector was inspired by various other prototypes. Many of the basic concepts for loose coupling support are similar to W4 (3.3), but adapted for use with multiple parallel browser tabs. Instead of tightly synchronising actions between browsers, I chose the PlayByPlay approach (3.10) of posting them to the chat in human-readable form. However, the amount and detail of such messages was much lower to increase practicality for casual use. I further chose to associate each chat message with the web page the author was looking at and allow users to jump back to these sources, similar to CoSense (3.9). Lastly, I decided to monitor and react to DHTML events, as suggested by UsaProxy (3.8), but only to inform other users of remote page changes and not to synchronize them, since my focus was to remain on loosely coupled work.

The Firefox UI was extended by two areas, a contact list panel to the right, and a chat panel underneath the web content (Figure 1). Double-clicking a contact opened the chat with this person or add the person to the already open chat, creating a group conversation. A second function of the contact list was to provide location awareness, by displaying links to the pages other contacts were seeing. Visitation awareness was supported through an overlay in the top-right corner of each web page, indicating if other contacts had visited it before (yellow background), were looking at it right then (green), or were looking at the same URL but with personalized page content (orange). Each page transition was also posted to the chat history, allowing basic page annotation through subsequent messages.

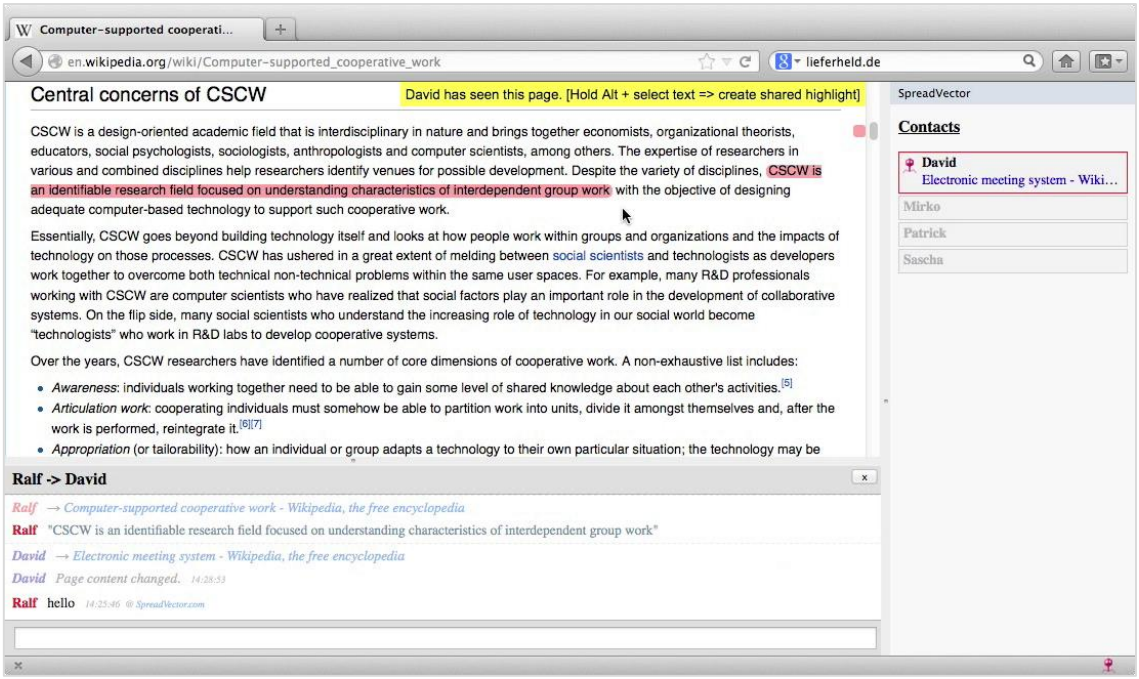



Figure 1. Firefox browser window with active SpreadVector extension.

Features for tightly coupled cobrowsing such as synchronized navigation, telepointers, or collaborative form filling were not considered. Still, it seemed plausible that users would want to reference and discuss individual text excerpts and not just entire pages. For this reason, a different metaphor for pointing was chosen, namely that of the *highlighter pen* [Shen & Sun 2002]. By holding down the Alt key and selecting arbitrary text on a web page with the mouse, users could highlight the passage in their colour, visible to all chat participants. The selected text was also quoted to the chat, to notify participants who were currently not looking at the same page. Clicking the quote jumped the active browser tab directly to the referenced passage.

A brief formative feedback questionnaire, handed out during a public presentation of the finished prototype, returned very encouraging results. Of the 17 participants who handed in valid responses (13 male, 4 female, mean age 25.2, SD 4.37), 15 positively mentioned the highlighting function or the direct jump capability. The integrated chat was mentioned 6 times, location awareness 4 times and visitation awareness twice.

Negative comments were very diverse and mostly comprised of feature requests or already known technical problems, some of which I explain in the rest of this chapter. The few reoccurring complaints included too much confusing information in the chat (4 mentions) as well as too much vertical space occupied by the chat frame (2 mentions). Several participants also verbally voiced privacy concerns over sharing visited pages.

Additional personal discussions with test users eventually resulted in a more abstract understanding of what they liked about SpreadVector and what they expected from a solution for every-day casual cobrowsing. After further literary research, this culminated in the three key principles of my master thesis: lightweight initiation, seamless transitions, and contextual collaboration. SpreadVector already adhered to each concept reasonably well, but still left room for improvements, especially regarding the ability to carry over independent work into collaborations, as well as active result management. These various improvement are detailed throughout the upcoming subchapters.

## 4.2 Initiation

Consistent with subchapter 2.2, this subchapter presents all design aspects of CoopFox which influence the events leading up to a collaborative session. This includes any prerequisites for the surrounding system (4.2.1), the installation of the extension in Firefox (4.2.2), its customization (4.2.3), and finally the actual initiation of a session (4.2.4).

### 4.2.1 System Prerequisites

This first section elaborates the base technologies for CoopFox's web browsing and communication functions, as far as these choices influence the installation procedure (4.2.2) or the general user experience. The related requirements section 2.2.1 has suggested compatibility with or extension of existing technologies as a way to maximize user acceptance, avoiding the need to install and learn an entirely new system.

This agrees with the *browser extension* approach already chosen for SpreadVector. A proxy implementation similar to UsaProxy or E-coBrowse might have been superior regarding setup speed and platform independence. However, since a proxy can only modify the loaded web content but not the browser UI, this approach was deemed incompatible with various other requirements regarding fluent contextual collaboration (2.4.2) as well as parallel multi-tabbed browsing (2.3.2). A third option would have been a stand-alone modified browser application, such as Samesurf. But given the low subjective usefulness of cobrowsing tools (2.2.1), users will likely be relunctant to leave behind their known browser environment (customizations, bookmarks, etc.) for the infrequent need to collaborate [Ishii & Miyake 1991, p. 37] [Xia et al. 2004, p. 163].

The choice of *Firefox* as a base system was made since it offered the most powerful API and therefore provided an ideal development environment for a prototype like this. However, more browsers will eventually have to be supported. Otherwise this would create the same problems as the stand-alone approach. In the feedback form for SpreadVector, several participants explicitly requested support for Google Chrome.

The technical reasoning for *XMPP* as a communication protocol was already detailed in 4.1. From the user perspective, XMPP has the advantage of an already very well built out infrastructure with many 3rd party server providers, but also free downloadable server implementations for expert users. Further, since major companies like Google and Facebook use XMPP for their chat implementations, many users are able to continue using their existing accounts and contact lists. Still, in the long run, more alternatives should be considered here as well. Skype was requested several times in user feedback.

### 4.2.2 Installation

In accordance with the section 2.2.1 requirements, the installation and XMPP login steps were kept very quick and easy. Since CoopFox is only a prototype, it must be installed via manual download instead of the Mozilla add-on store. However, the entire scenario still consists of only two steps and takes place at runtime, without a browser restart:

- The user enters the download URL to the `coopfox.xpi` file in the address bar.
- Firefox display a security warning dialog, which the user confirms.

After the installation, a new button appears at the far right of the main browser toolbar (Figure 2). Clicking it toggles the CoopFox functionality and user interface on and off for each individual browser window, indicated by a changing button icon.

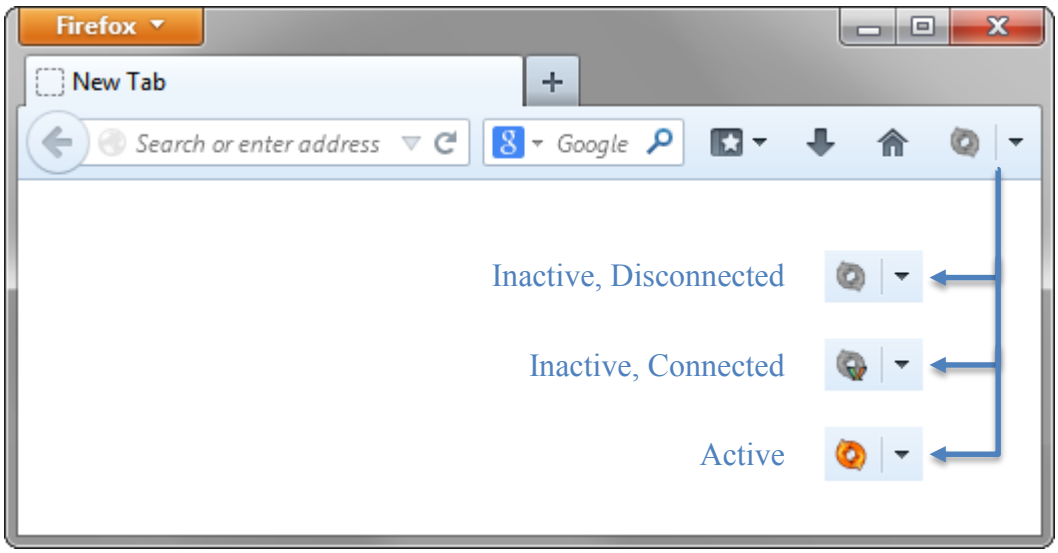



Figure 2. Firefox window immediately after the installation of CoopFox, showing the new toolbar icon.

Inspired by the instant messenger paradigm (2.2.2), users have the option to *hide CoopFox during independent work* but still remain available for spontaneous sessions. For this purpose, the drop down menu attached to the toolbar button contains a checkbox to maintain an XMPP connection even when CoopFox is not active in any windows. A second checkbox automatically enables the extension for each new browser window, and a last menu entry allows users to reset their current XMPP login credentials.

Entering the *login credentials* becomes necessary on the first attempt to connect, at which point CoopFox presents a custom modal login dialog (Figure 3). This dialog accepts multiple types of credentials. Expert users can manually specify the exact network location of an XMPP server, a user ID and password, as well as an encryption policy. Average users can switch to alternative forms for either Google or Facebook accounts, which only require the respective user name and password. In either case, the dialog offers to save the entered credentials in the internal encrypted password store of Firefox.

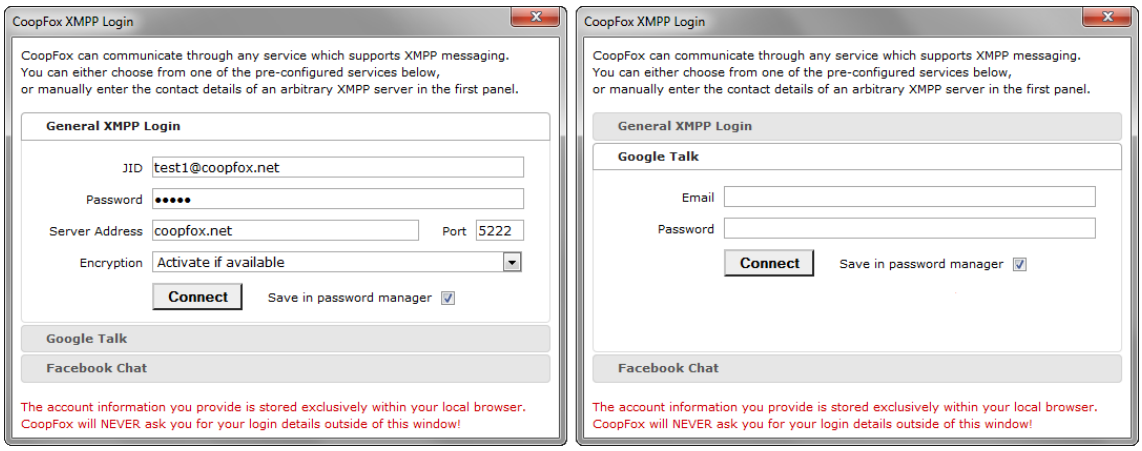



Figure 3. Login dialog, accepting generic XMPP server credentials (left) or a specific service login (right).

The entire process to establish full functionality of CoopFox can easily be completed in under 30 seconds, without disrupting the state of existing browser tabs, so that users can install the extension and begin collaboration at any time during independent work.

### 4.2.3 UI Customization

Once CoopFox is activated for a browser window, it displays a new *sidebar* to the right of the tabbed browser (Figure 4), which is itself vertically divided into a *contact list panel* at the top and a *session panel* at the bottom. Different from SpreadVector, the chat is now part of the session panel, so that the tabbed browser space is no longer reduced vertically. The session panel can contain multiple *tabbed areas*, so that several separate session functions can be accommodated without using up too much space.

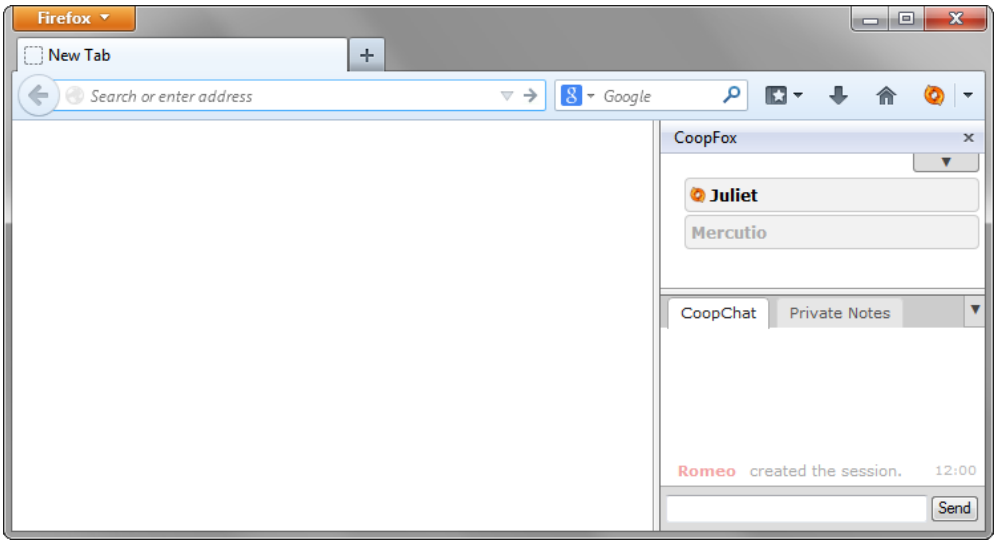



Figure 4. Firefox window with activated CoopFox sidebar, showing the contacts panel and session panel.

The sidebar *width* as well as the *height* ratio between contacts and session panel can be adjusted by clicking and dragging the separators. This is stored persistently. The session tabs can be reordered in the same way, by dragging them to a new position.

*Contact list management* similar to other IM clients takes place via the contact list menu (Figure 5). Users can change their own *display name*, and current *availability status* [Schümmer & Lukosch 2007, p. 170]. Users who have privacy concerns about sharing their visited pages with their collaboration partners can also disable this function here. Lastly, *new contacts* can be added to the contacts by their XMPP JID. Removing and renaming them is possible via the context menu by right-clicking a contact list item.

The session panel can be customized via its own menu (Figure 5). Each tab can be hidden separately. In the extension settings dialog, expert users can further choose whether participant colours are synchronized, or whether the local user is always red.

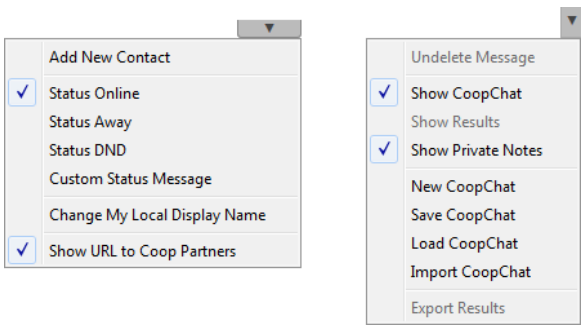



Figure 5. Menus located within the contact list panel (left) as well as the session panel (right).

### 4.2.4 Session Persistence

Users should have the ability to maintain multiple parallel sessions and quickly re-open a session after closing it, analogue to instant messaging (2.2.2). Specific to cobrowsing, they also still need to be able to use their browser privately in parallel to collaborating. To resolve these issues, CoopFox builds on the already existing mental model of separate windows, associating each collaborative session with a distinct (tabbed) browser window. Windows for which CoopFox is deactivated can be used for normal private browsing.

The state of each session is persisted in the context of its window. If a window is closed and later restored from the Firefox history, the session is also resumed. In addition, it is naturally also possible to save sessions to files and reload them later (Figure 5).

Another fundamental principle of CoopFox sessions is that their entire state is always represented by the chat history. Every persistent action always has a corresponding chat entry, creating a comprehensive timeline [Schümmer & Lukosch 2007, p. 377], similar to PlayByPlay or CoSense. Because the chat thereby holds such a central role and is effectively synonymous to the session itself, it was given the special name "CoopChat". All prompts and menus use this term, since "session" might be too abstract for some users.

### 4.2.5 Session Merging

As the key requirement for *lightweight initiation*, users must be able to begin working independently and carry over their state of work into a collaboration at any time (2.2.2). This is exactly the purpose of CoopFox's novel *session merging* concept: When CoopFox is activated for a window, it already contains a fully functional CoopChat, which can be used to collect and comment on web content solitarily. If the opportunity for collaboration arises, several users can merge their solitary CoopChats into one joint session, evaluate each other's progress and then continue their web research together. The strict timeline form of CoopFox sessions (4.2.4) comes into play at this point, allowing for two or more sessions to be combined, simply by interleaving the chat histories chronologically.

An additional 1-on-1 *private chat* function was included in CoopFox to compensate for a downside of the session merging approach, namely that establishing first contact is no longer as quick and reversible as with an instant messenger (2.2.2). Because merging is by definition an irreversible process, it requires prior negotiation in the form of an invitation procedure (described further below). The separate private chat allows people to contact each other in the usual lightweight form, determine their readiness for collaboration, and then merge their CoopChats. Private chats appear as named tabs in the session panel (Figure 6), although they are not part of a specific session. Rather, their content is persisted per contact, so that any former personal correspondence is preserved.

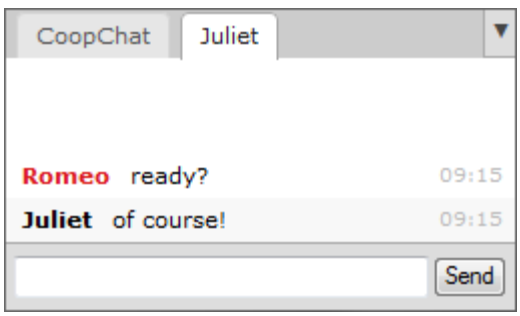



Figure 6. CoopFox session panel with an active private chat between Romeo (local) and Juliet (remote).

The tab of new incoming private chats appears in the background and blinks red until it has been activated, but does not disrupt the user in any way. Moving the mouse cursor over the tab reveals a close button, by which the chat can be quickly discarded. Lastly, private chats can also be used to communicate with contacts who are not using CoopFox, whereas joint CoopChats can only be created between CoopFox clients. This should reduce the critical mass barrier for CoopFox, as users may adopt the extension a general replacement for their usual stand-alone XMPP messengers.

*Initiating* either a joint CoopChat or a private chat occurs through the contact list, by double-clicking any contact who is also online. CoopFox prompts the user for the type of chat to engage (Figure 7). Alternatively, to bypass this prompt, the desired chat type can also be started directly from the context menu of each contact entry.

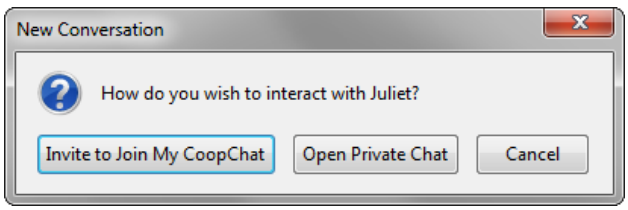



Figure 7. Chat type selection prompt, displayed when users double-click an entry from their contact list.

Merging CoopChats is initiated via *invitation* [Schümmer & Lukosch 2007, p. 255]. The invited contact is asked via a modal dialog whether to open the incoming CoopChat in a new browser window, or to merge it with the CoopChat in their current active window, (Figure 8). If several browser windows exist, an alternative prompt with a selection list is presented, allowing the user to select where to open or merge the session.

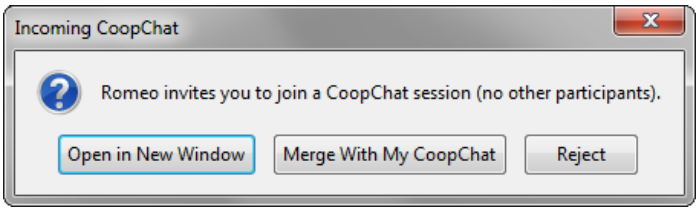



Figure 8. Option prompt for an incoming CoopChat invitation. Button captions may vary contextually.

The entire scenario for session initiation in CoopFox therefore has the following form (assuming both users are logged in and have already begun research independently):

- User1 double-clicks the contact list entry of User2.
- CoopFox1 displays the “New Conversation” prompt dialog (Figure 7).
- User 1 selects “Open Private Chat”.
- CoopFox1 adds a new private chat tab for User2 to the session panel.
- User1 sends a text chat message, asking what User2 is currently doing.
- CoopFox2 adds a new blinking private chat tab for User1 to the session panel.
- User2 replies and as the conversation unfolds, the two users discover that they are working on the same research topic and decide to merge their CoopChats.
- User1 double-clicks the contact list entry of User2 again, this time selecting the CoopChat invitation option in the prompt dialog.
- CoopFox2 displays the “Incoming CoopChat” prompt dialog (Figure 8).
- User2 selects to merge the incoming CoopChat with the existing local one.

Any session member can invite *additional participants* from his or her contact list at any time, even if some of the other participants do not know them by their contact lists. This guarantees that collaboration can always occur spontaneously and is not encumbered by the need to exchange credentials first. A common practical scenario for this is that one member of a workgroup wishes to include another external domain expert for feedback, whom he or she has worked with before, but who is unknown to the other team members.

Lastly, lightweight *re-initiation of a previous collaboration* is also important, because teams often interrupt and later continue their work sessions [Bardram et al. 2006, p. 213]. For this reason, session participation in CoopFox is persistent. If several users reload an old shared session at the same time, they automatically re-join and are back in contact. This even works for people who do not normally know each other by their contact lists. If any user has made updates to the CoopChat in the absence of others, these changes are automatically merged back into the shared history, so that everyone is up to date.

While this means that it is effectively not possible to remove a user from a session, it is of course possible to leave and *abandon participation*, simply by deactivating CoopFox for the respective window or closing the window altogether. Users who have left receive no further updates on the session, unless another participant explicitly re-invites them.

## 4.3 Transition

This subchapter presents the interaction design for all aspects of CoopFox which relate to seamless transitions between solitary and group work. The first four sections are dedicated to the different forms of awareness support, introduced in 2.3.3. Section 4.3.5 describes the way CoopFox manages tight and loose coupling (2.3.2). The last two sections describe group decision support for sensemaking and organizing phases (2.1.4).

### 4.3.1 Informal Awareness

A prerequisite for any collaborative work is *informal awareness* (i.e. "the pervasive experience of who is around, what these persons are doing, and what they are going to do") [Gross et al. 2005, p. 327]. CoopFox supports this form of awareness through the contact list panel. Like in any other instant messenger, it shows which contacts are currently online and indicates their availability (e.g. away, do not disturb). Contacts who are also signed in using CoopFox are marked with the extension logo (Figure 9, left).

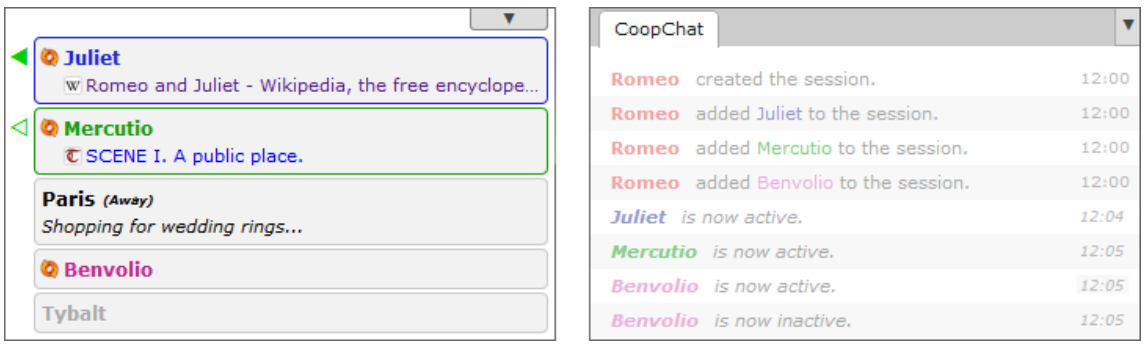



Figure 9. Contact list and corresponding user status messages in the chat, on the left side from top to bottom: (1, 2) CoopFox users who are part of the current session; (3) A contact who is signed in using a different client and has set the "away" state along with a status message; (4) A CoopFox user who used to be part of the session, has closed the session, but is still online. (5) A contact who is offline.

The contact list further indicates which contacts are participating in the current session, by highlighting them with a coloured name and outline. These assigned colours are globally consistent and also used for *attribution* in the CoopChat and in user annotations on web pages. Contact lists are therefore not redundant between different windows, but tightly integrated with the session context that each window represents. As usual for instant messengers, any status changes are also logged to the chat history (Figure 9, right).

### 4.3.2 Action Awareness

A second general prerequisite for collaborative work is *action awareness* (i.e. knowledge of what actions are being performed within the group) [Carroll et al. 2003, pp. 611-612]. This represents an essential part of workspace awareness for synchronous systems.

In SpreadVector, this type of awareness was only provided through the chat, by logging events such as page changes to the message history, similarly to PlayByPlay. This led to two common complaints among users: Firstly, they often felt overwhelmed by the constant flow of incoming information in the chat. Secondly, users often missed new relevant entries in the chat history while they were predominantly focussed on browsing and merely paying peripheral attention to the chat.

CoopFox addresses the information overload problem by abandoning the automated action logging (see next section) and by introducing separate tabs for the session panel, to split off more information from the CoopChat. Inactive tabs indicate new content through red highlighting, so that users cannot miss important updates (Figure 10). If Firefox itself is currently not in foreground, new incoming content now also triggers a platform-specific notification to draw the user's attention (e.g. blinking the Firefox Taskbar button on Windows or jumping the Firefox icon in the MacOS Dock).

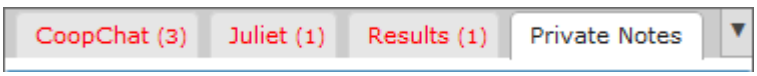



Figure 10. Red highlighting and counters in the session panel tab selectors indicate new unseen content.

CoopFox addresses the peripheral awareness problem by introducing animations to the chat, giving users a longer time window to notice changes [Ellis et al. 1991, p. 50]. For instance, new messages first appear in red and then slowly fade to their actual colour over several seconds (Figure 11, a), while deleted messages slowly fade to white before disappearing (b). If new changes occur outside of the current scroll viewport, these effects are delayed until the corresponding entry comes into view the next time. Markers (c) at the scrollbar notify the user of new unseen changes above or below. Lastly, CoopFox provides a remote keyboard activity indicator (d) [Schümmer & Lukosch 2007, p. 363] as common in most instant messengers, so that users can expect new content in advance.

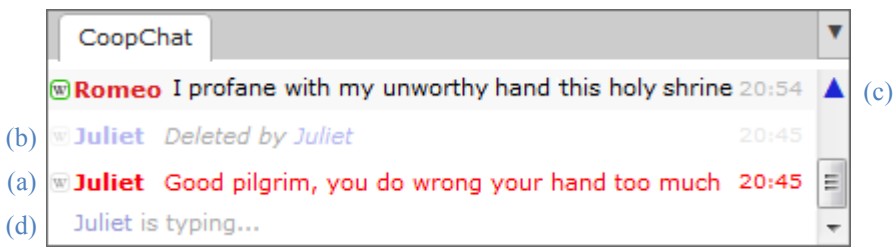



Figure 11. Visual fade effects as well as scrollbar markers in the chat support peripheral action awareness.

### 4.3.3 Location Awareness

Awareness of the web pages viewed by other session members was one of the central requirement established in 2.3.3, highly important for both foraging and sensemaking. CoopFox provides this form of awareness by contextually augmenting the underlying UI paradigms of *tabbed browsing* and *instant messaging*. In particular, location information was introduced in three places: contact list entries, chat messages, and browser tabs.

The *contact list* provides basic synchronous location awareness. List entries of session members contain a link to the web page the respective person is looking at in his or her currently active tab (Figure 12). The link shows both the human readable page title as well as the web site icon (favicon). Clicking the link allows users to jump directly to each other's location, which was already one of the most popular features in SpreadVector. The additional favicon was added in CoopFox for faster visual recognition.


Figure 12. Current web locations of other session members are contextually added to the contact list.

For quick page annotation, each *chat message* now carries a reference to the page its author was looking at while writing, shown in the form of a small prefixed page icon link (Figure 13). This replaces SpreadVector's approach of logging each page change to the chat explicitly, which was deemed too distracting. It also gives users deliberate control over the location information they expose, whereas SpreadVector users often voiced discomfort over the fact that their entire browsing history was made visible persistently. Hovering the mouse on top of the icon reveals the full page title as a tooltip. Clicking either the icon or the title opens the referenced page in the local browser, allowing all session members to quickly revisit any location which has been part of the discussion. Messages which were written on the same page as the one currently active in the local browser receive an additional highlight in the form of a green outline for the page icon, thus providing a visual cue for consecutive discussions about a single web location.

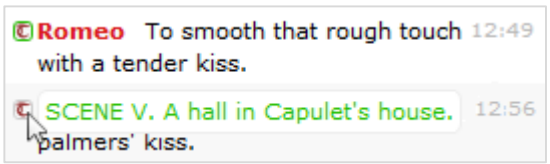



Figure 13. Location icons attached to chat messages. Top: A message written on the currently active page. Bottom: A chat message written on a different page. Hovering the mouse atop icons reveals the page title.

Augmented *browser tab selectors* allow users to monitor the actions of other session members across several pages at once. When someone posts content to the CoopChat that relates to one of the locally opened pages, the according tab selector colour is changed to the author's colour. Tabs with unseen changes appear in bold italic font until clicked.

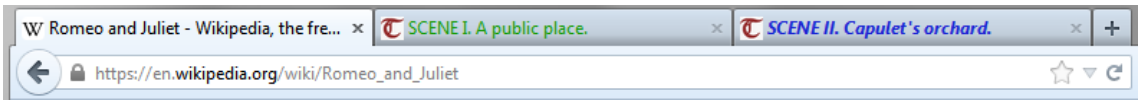



Figure 14. Tab colouring indicates the session member who was last active on a page.

### 4.3.4 Visitation Awareness

The opposite direction of location awareness (Where are the other group members?) is visitation awareness (Who is/was already here?). This knowledge is primarily required to allow division of labour and avoid redundant effort during foraging phases (2.3.3). CoopFox once more uses contextual augmentation to support this kind of awareness, introducing new information in two places: contact list entries and hyperlinks.

The *contact list* provides visitation information for each session member, regarding the page in the locally active tab. Contacts viewing the same page are indicated by a blinking green arrow left of the list entry (Figure 15). Contacts who have seen the local page before are marked with an unfilled unblinking arrow. Hovering the mouse atop an arrow displays a tooltip, indicating how long the contact has been viewing the page or when the last visit occurred, analogue to these hints in CSCW3 and SearchTogether. In SpreadVector, all this information was presented as an overlay in the page corner, but this solution was deemed to obtrusive and distracting by several users. The indication of deviating page content between local and remote browsers was also discontinued, since few users seemed to understand its purpose when asked about it, and even fewer found it useful, given there was no fine-grained indication of what exactly was different.


Figure 15. The visitation status of session members for the current local page is shown in the contact list. Top: A user who is looking at the same page. Bottom: A user who has seen the current page before.

*Hyperlinks* within web pages indicate which session members have already seen the target page, using colour-coded checkmarks (Figure 16). This solution is similar to the one in Cooperative Proxy (3.7). If page-related content has been posted to the CoopChat by a session member, this is indicated via an additional pen icon.

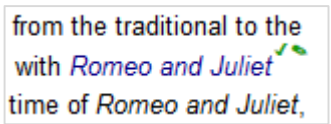



Figure 16. Text link within a web page, which has already been visited and annotated by the green user.

The same checkmarks are also shown in the CoopChat, in the tooltip which appears when hovering the mouse atop the location icon of a message (Figure 17). The page icons themselves also provide visitation information for the local user, in that icons to unvisited pages are shown in greyscale, so that users can more easily recognize new content.

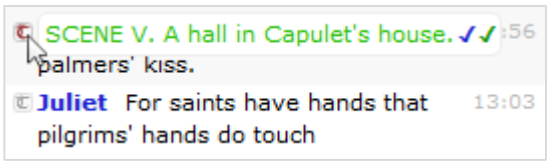



Figure 17. Visitation information in the message location icons. Top: A message from a page visited by the local user, as well as the blue and green users. Bottom: A message from a page not visited by the local user.

### 4.3.5 Location-Based Coupling

To allow easy transitions between tightly and loosely coupled work at any time, CoopFox uses a location-based approach, similar to other systems (W4, CSCW3, CoNavigator): Features for tight coupled interaction activate automatically, as soon as two session members visit the same web page at the same time. Users can leave the tightly coupled state whenever they wish and return to solitary browsing, simply be leaving the page.

However, CoopFox extends this principle to *tabbed browsing*, providing even faster transitions, by allowing users to maintain the co-located tight coupled state in some tabs while using other parallel tabs for solitary browsing at the same time. In combination with the location awareness support described in 4.3.3, this concept enables convenient and fast-paced alterations between foraging and sensemaking (2.3.1), where users can spend the majority of time in solitary browsing but also quickly convene on one page in a new tab for real-time feedback and discussion, before going their separate ways once more.

Any tight coupling in CoopFox is always *relaxed*, providing real-time synchronized actions between remote page instances, but leaving both scrolling and page transitions in the deliberate control of the user. This decision was made since introducing alternative more tightly coupled WYSIWIS modes would necessarily require a manual mode switch function (similar to CoWeb). However, this would contradict the general seamless approach, in which all transitions occur implicitly, based solely on web locations. Features such as collaborative form filling were dismissed for the same reason.

The primary tight-coupled feature in CoopFox is real-time *pointing* to page content, also provided by many of the related systems (GroupWeb, W4, UsaProxy, Samesurf). However, in the tradition of SpreadVector and its success with highlighting, CoopFox still does not use telepointers or remote scroll bars. Instead, the highlighting approach was further simplified to a quick remote selection [Schümmer & Lukosch 2007, p. 348]: As soon as a user selects any text on a web page with the mouse, this selection is immediately replicated in the browsers of other session members looking at the same page, using the colour of the selecting person (Figure 18). Due to the relaxed scrolling environment, a blinking indicator next to the page scroll bar notifies other users of highlights outside of their view and lets them jump directly to the selection by clicking it.

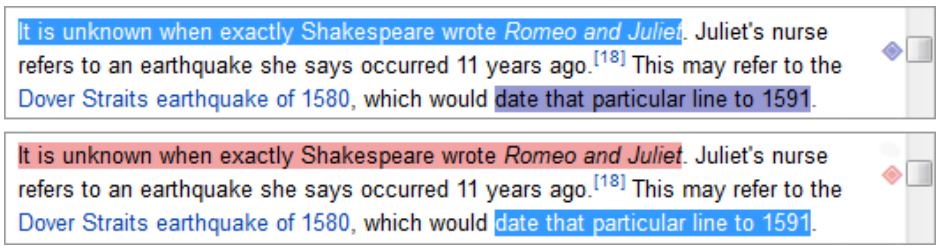



Figure 18. Transient highlighting of web page content by two users, as seen on their respective screens.

The persistent highlighting function from SpreadVector for fine-grained referencing and annotation also still exists (described in 4.4.3). The selection highlights, however, are merely transient and vanish when the text is unselected again. Neither do they appear in the CoopChat. Although the highlighting implementation has been vastly improved (see 5.4.2), sometimes a user may still select text which either contains unsuitable HTML or simply does not exist in the remote user's page. In this case, an unobtrusive notification popup is displayed for a few moments next to the mouse pointer, explaining the problem.

### 4.3.6 Result Review

To provide better support for result reviews during sensemaking phases (2.1.4), two new features were added to the CoopChat: threaded discussions and message pointing.

Attaching comments to existing messages allows users to create subthreads within the conversation. They can thus lead organized discussions about subtopics or give dedicated feedback on the results foraged by someone else [Schümmer & Lukosch 2007, p. 281] [Röscheisen et al. 1995, p. 329], as it was already possible in ComMentor, for instance. In the expectation that this function will be used very often during complex researches, it has been made very quick and easy to access: Double-clicking any message in the chat history displays an embedded text input field indented underneath (Figure 19, a), which works exactly like the normal chat input and can be submitted with the enter key, inserting the new comment message in its location (Figure 19, b). Nested comments are not allowed, to keep the function as simple as possible. A double-click on a comment simply appends another comment to the parent message. Comments further do not have location information attached, since they already receive context through their parent message. This allows users to comment at any time, without exposing their current web page.

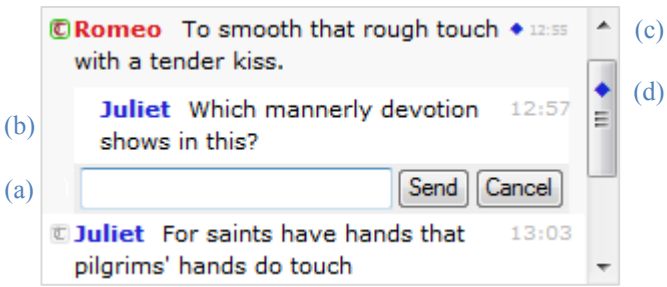



Figure 19. Threaded discussions and message pointing in the CoopChat. (a) Active comment input field. (b) Comment made by Juliet on original message by Romeo. (c, d) Message selected by remote user Juliet.

Message pointing (message remote selection [Schümmer & Lukosch 2007, p. 348]) allows users to direct the attention of other session members to a specific message during discussions. Clicking on any message in the chat history displays a small coloured dot to the right of the message and in the scroll bars of all session participants (Figure 19, c, d). The scrollbar marker guarantees that all participants notice the selection, independent of their current scroll viewport. Clicking it scrolls directly to the selected message, thus facilitating the feature for long and complex chat discussions, where it is most useful. Selections are transient and vanishes as soon as the user clicks on something else.

### 4.3.7 Result Management

During organizing phases (2.1.4), session members must perform group decision making and collaborative editing tasks to manage their results. In distributed settings, this requires specific active support, as exemplified by SearchTogether/CoSense (3.9). CoopFox currently provides a small number of basic collaborative operations for this purpose: Result collection, prioritization, workspace clean-up (deletion of non-results), as well as distribution (export). A general conceptual challenge is presented by the nonlinear workflow of web research (2.3.1), which demands that all functions must be accessible at any point, in between or even parallel to foraging and sensemaking phases.

CoopFox supports *result collection* in a separate session panel tab, which is available at all times, but remains hidden until the first result is added. Any message from the CoopChat can be declared a result at any time and added to the list. The results tab only lists these messages, effectively acting simply as a special pruned view of the same data (Figure 20). Since both the author and location information is preserved, *attribution* (2.3.3) is automatically ensured. Adding a message to the results occurs either through the right-click context menu or by double-clicking it with the middle mouse button.

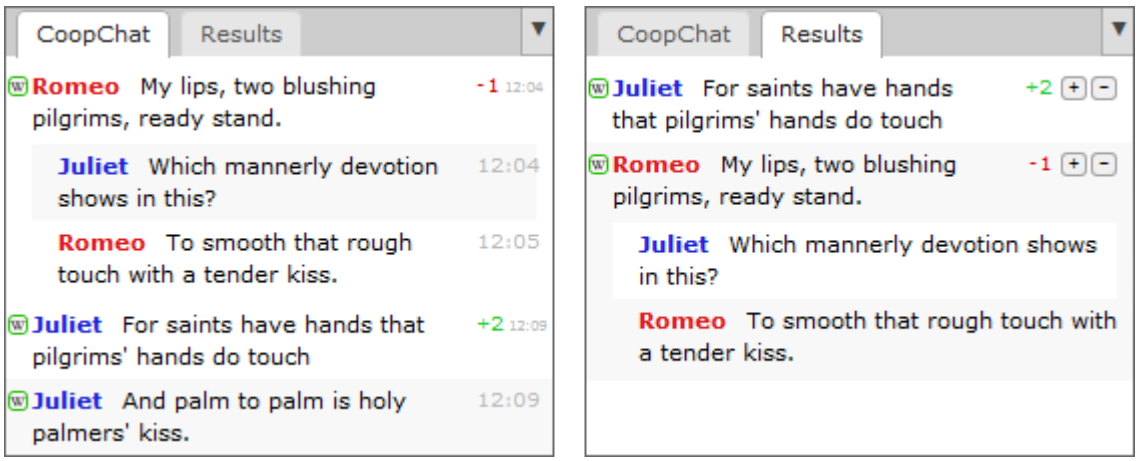



Figure 20. Session results tab (right), which presents an alternative view of the original CoopChat (left).

The *sorting* of result tab entries is determined by a weight value, representing the importance of each result. This value is initialized to +1 for each new result and is shown to the right of the message in both the result view as well as the original CoopChat. The value for each result can be adjusted by any session member using the +/– buttons in the result list (or the context menu) and is not limited in either direction. No access rights policy is applied either, since it is assumed that collaborating small groups are able to negotiate a social protocol for coordinated use of the system. It should be pointed out that this approach is not intended as a voting system [Schümmer & Lukosch 2007, p. 208], but rather as lightweight collaborative flagging [Schümmer & Lukosch 2007, p. 287]. This is also the reason why no form of attribution or feedthrough notification is provided for modifications, as it is assumed that users operate the weighting system in very tight collaboration after they have already reached consensus through verbal discussion.

Workspace *clean-up* (i.e. deletion of unnecessary or rejected content) is another common part of organizing phases. CoopFox supports this operation in the same tightly synchronized way. Any user can delete any CoopChat message or comment at any time via the right-click context menu. To correct accidental deletions, there is also a shared undelete function, which can be activated via the session panel menu and always restores the last message deleted by any person in a stack-like fashion. While results can also be deleted via the context menu, there is no undo function for this, but merely a confirm prompt, since lost results can easily be added again from the original CoopChat entry.

Lastly, CoopFox also supports the *distribution* of results, although so far only in a very minimalist way, by allowing users to export the result list as a CSV file, containing the message or quote text, location URL as well as page title. Future versions of CoopFox will have to explore more sophisticated solutions to organize and export results.

## 4.4 Integration

This subchapter describes the interaction design for the contextual collaboration functions of CoopFox, which integrate synchronous communication directly with web content. The described solutions relate directly to the copy-paste overhead problem identified for various unit tasks in subchapter 2.4, if these tasks were to be performed using separate applications for browsing, messaging and result collection functions.

### 4.4.1 Application Switching

CoopFox eliminates the need to switch between browsing, messaging and result document applications, by integrating all of these functions into a unified user interface. This affects all unit tasks identified as "tier 3" in section 2.4.1, such as parallel browsing and message reading, which are encumbered by application switching overhead but do not involve any copy paste cycles for content transfer.

In CoopFox, the browser always stays in view, accompanied by either the chat or the results tab. Viewing all three at the same time is still not possible, which is a deliberate compromise to reduce required space and information load. The update notifications in all session panel tab selectors ensure that users do not miss any content (see 4.3.2).

### 4.4.2 Location Referencing

CoopFox provides several quick ways of referencing entire web pages to other people. Section 2.4.1 classified such tasks as "tier 2", since they typically involve an overhead of one copy-paste cycle between applications. As a common scenario, one person is looking at a particular web page and wishes for another person to visit it as well, to discuss its contents, as part of a guided tour, or to provide support. Without specific tool support, this requires an additional application like an instant messenger and might proceed as follows:

- User1 copies the URL from the browser address bar.
- User1 switches to a text messenger application session with User2.
- User1 pastes the URL into the input field and clicks the "Send" button.
- User2 switches to the text messenger application session with User1.
- User2 clicks on the received link in the chat history.

The past subchapters have already introduced two ways to achieve the same goal in simpler ways using CoopFox. The first one uses the location link in the contact list:

- User1 verbally tells User2 to open his or her current page.
- User2 clicks on the link in the contact list within the entry of User1.

A second way to reference a web page in CoopFox is to use the location information displayed with each chat message. This does not even require any verbal communication:

- User1 posts the text message "look at this page" into the CoopChat.
- User2 clicks on the page location icon in front of the received message.

However, since referencing an entire web page is such a common task in cobrowsing, a third way to achieve this has been added to CoopFox, which requires only a single mouse click for both the sender and receiver. If a user clicks the "Send" button of the CoopChat without entering text first, a link to the currently open web page is posted to the chat (Figure 21). It shows the human readable page title of the page. To allow users to discover this feature intuitively, a preview of the link is already displayed in the input field whenever it is empty and not currently in focus.

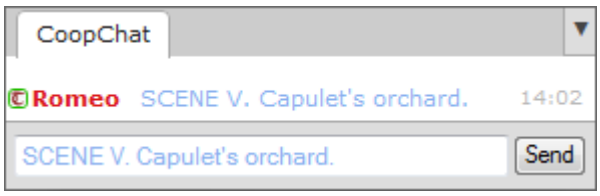



Figure 21. A page link is posted into the chat by sending empty text (a preview is shown in the input field).

The entire scenario of directing another person to one's current web page using this method is shortened to two steps, involving only one mouse click on each side:

- User1 clicks the "Send" button of the CoopChat without having entered text.
- User2 clicks on the posted link.

While this method is not substantially faster than using the link in the contact list, it provides the benefit of persistency, storing the page reference in the chat for later revisits. It also allows User1 to continue browsing, in case User2 is not instantly ready to follow the link. Combined with the comment function (4.3.6), this further offers a quick way to start a discussion subthread about an entire page, using the link as the parent message.

### 4.4.3 Content Referencing

The greatest task performance advantage of CoopFox is achieved in referencing specific web page content to other people. Such tasks are identified as "tier 1" in section 2.4.1 and can be estimated with two copy-paste cycles of overhead when performed using separate applications. Common example scenarios involve one person trying to direct another to a specific part of a web page. The scenario begins like the one from the previous section, but then further requires either a verbal description by User1 of the position User2 must scroll to, or a text excerpt which must be copied and sent over for User2 to search.

To simplify this process, CoopFox still provides the ability to mark text excerpts in web pages and post links to their exact position into the chat. Called simply "highlight" in SpreadVector (4.1), this feature was now given the name "Direct-Quote" and made available from the context menu when right-clicking on selected text in web pages. Holding down the Alt-key during selection is also still available for shorthand access. The quoted text is persistently highlighted in the web page using the sender's colour (Figure 22, top), and posted to the CoopChat in the form of a link (Figure 22, bottom), which opens the target page and also scrolls directly to the highlighted passage. Using this feature, the content referencing scenario shortens to two steps:

- User1 selects a text passage he or she wants to show to User2, while holding Alt.
- User2 clicks on the received link in the CoopChat.

Because the quotes are stored persistently in the chat history, together with their author and origin page, Direct-Quoting is also the primary method intended for gathering results during foraging phases. They can be added to the result list like any other message. Direct-Quote highlights can be distinguished from transient pointing (see 4.3.5) by their round edges and a pen as a scrollbar marker instead of the diamond symbol. Permanent highlights further show a close button when hovered above with the mouse, which removes the highlight and is equivalent to deleting the quote message in the chat.

The quote function further doubles as a quick method for detailed annotation, showing any chat comments made in relation to a quote upon hovering the mouse (Figure 22, top). In reverse, double-clicking the highlight opens the comment input in the chat.

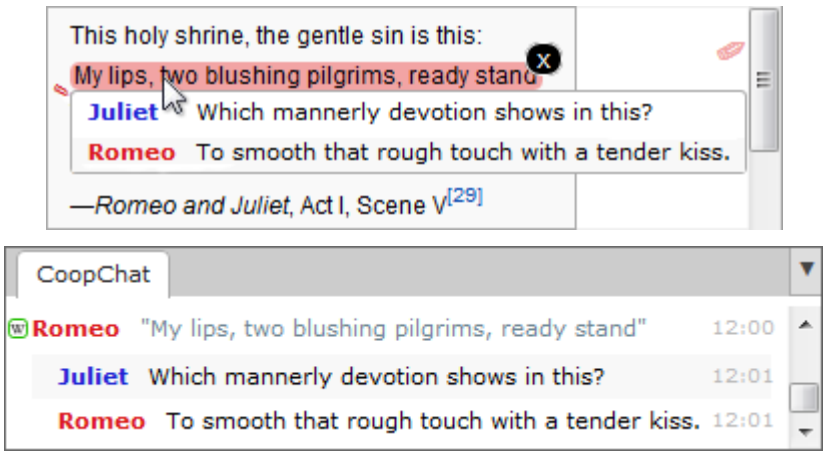



Figure 22. Direct-Quotes are persistently highlighted in the web page (top) and posted to the chat (bottom). Hovering the mouse on top of the highlight reveals any comments made in the chat.

## 4.5 Summary

In this chapter, I have described the interaction design for CoopFox, in relation to similar projects from chapter 3 and the requirements from in chapter 2. The central design concern for CoopFox was to create a holistically flexible experience for every-day use, based on the three key principles of lightweight initiation (quick and spontaneous sessions), seamless transitions (free choice between tight and loose coupling), and contextual collaboration (fine-grained interaction with web page content). Many details of my design were inspired by earlier systems which excelled regarding some of these principles. CoopFox combines these successfully aspects and, in addition, also introduces novel interaction concepts for each of the three key principles:

*Lightweight initiation* is achieved through the novel approach of *session merging*. CoopFox does not distinguish between single- and multi-user sessions, so that users can first begin researching on their own, using all functions of the extension, and combine several single-user sessions and their results into a multi-user session when necessary. The concept is intended to combine the easy person-to-person session initiation of instant messengers with the convenient content sharing of room-like workspaces.

*Seamless transitions* between solitary and group work are achieved by augmenting the existing paradigms of *tabbed browsing* and *instant messaging* with location awareness. Location-aware tabs allow users to collaborate tightly in one tab, simply by visiting the same web page as their peers, while maintaining other private pages open in separate tabs. The location-aware messenger allows users to jump quickly to any page one of their contacts is seeing, or to a page associated with any received message. CoopFox further extends the messenger paradigm with new functions for threaded discussions, pointing, and result collection. All of these functions can be used at any time, imposing no strict workflows, so that foraging, sensemaking and organizing can occur freely as seen fit.

*Contextual collaboration* is provided through the ability to reference entire pages or selected text excerpts via the chat with very few clicks, thus creating a quicker alternative to manual copying and pasting of content between applications. In real-time synchronized form, the same remote text selection approach also serves as CoopFox's lightweight alternative to the telepointer. During tightly coupled work, session members viewing the same web page can use it as a fine-grained synchronous pointing tool.

# 5 CoopFox Implementation

This chapter describes the implementation of CoopFox. The contents of upcoming subchapters are therefore largely correlated with the structure of chapter 4, where the extension was already presented from the conceptual perspective. The focus lies on specific engineering challenges that had to be overcome in order to deliver the desired functionality, since the entirety of the system cannot be presented here.

Large parts of the low-level code were directly reused from the forerunner project SpreadVector (see 4.1), so that this chapter once more contains many back references and comparisons to my former student project. However, most of the user interface was entirely re-developed from the ground up. In total, the code base of CoopFox is now comprised of approximately 12600 pure lines of code (excluding blanks and in-code documentation), compared to merely 3700 for SpreadVector.

The pros and cons for the choice of Firefox and XMPP as base technologies are not discussed any further in this chapter but simply treated as given. The decision for this approach was already made at a very early stage during the development of SpreadVector, for the reasons detailed in sections 4.1 and 4.2.1. To my knowledge, only a single related project has used the same combination of Firefox and XMPP communication before [Lowet & Goergen 2009], but barely any details of the implementation were published, so that there are very few references to other cobrowsing systems in this chapter.

UML-style diagrams are used throughout this chapter to illustrate code structures. However, since the program logic of Firefox extensions is implemented in JavaScript, some adaptions to the diagram patterns are unavoidable. Some of these unique aspects are presented in the architecture overview in subchapter 5.1, before the following subchapters then continue in the established aspect trisection from chapters 2 and 4.

## 5.1 System Architecture

CoopFox is developed as an extension for Firefox browsers based on Gecko Engine 2+ (Firefox 4+) [Shepherd 2010b], which introduced JavaScript 1.8.5 as the current language standard [Shepherd 2010c]. More specifically, at least Firefox 22 is required, since CoopFox already makes use of various draft language constructs, such as the WeakMap and the for-in loop [Schuster 2012]. This does not present a problem in terms of practical availability, however, since Mozilla has implemented a very strict automatic update policy for their browser, so that the vast majority of Firefox users across all operating systems can always be expected to have the latest version installed.

In general, CoopFox does not include any operating system specific code, but relies entirely on the abstraction layer provided by the Gecko Engine. During development, the extension was tested on Windows, MacOS and Ubuntu Linux to ensure it operates properly on all of these platforms. No functional problems were discovered in this process. Visual styles of windows and menus tend to vary strongly with the underlying graphics libraries, but this did not result in any major usability problems.

The overall architectural approach of CoopFox can be described as strongly component-based (see 5.1.3) and event-driven (see 5.1.4). This results from the fact that all program activity is initiated either by a user interface action or an incoming network message, both of which can be modelled as distinct self-contained events.

### 5.1.1 External APIs and Libraries

Firefox extensions have very extensive and generic access to all low-level functions of the web browser via the XPConnect (Cross Platform Connect) layer [Kimbro 2005]. It allows the extension code, written in JavaScript, to create instances of native (C++) Gecko classes and interact with them through JS proxy objects. Throughout this chapter, I refer to such native classes by their Netscape interface name, beginning with `nsI`.

CoopFox was further developed using the Mozilla Add-on SDK 1.4 [Bamberg 2013], which imposes a modern object-oriented code structure and provides a library to emulate class hierarchies in JavaScript. Since the Gecko Engine provides a dynamic module loader based on CommonJS [Dangoor 2009], each class can further be separated out into its own file and namespace. Altogether, this gives the code a very Java-like structure, which can be presented within the semantics of standard UML class diagrams (e.g. Figure 25).

The SDK further includes a variety of libraries (formerly known as *Jetpack* project) which either wrap the low-level Gecko API into more abstract classes or combine them to provide common advanced functions. This provides extension developers with easy access to various essential browser functions like windows, tabs, clipboard, hotkeys, password store and so on. I refer to these as *SDK classes* in contrast to my own.

Testing and deployment of CoopFox was performed using the SDK build scripts, written in Python. These were not modified, but simply wrapped in Windows batch script files, in order to pass arguments about code location and default settings for CoopFox, such as the XMPP credentials to use for testing. Although possible within the SDK framework, automated unit tests were not performed, because relevant test cases usually involved network communication or local UI events, which are difficult to simulate.

The entire user interface of CoopFox (side bar content and login dialog) was written in HTML and CSS, as well as normal web page JavaScript without SDK library support. The jQuery JavaScript framework was used in this context, in order to assist with common DHTML tasks, such as DOM traversal and visual effects [jQuery 2006]. In addition, jQuery UI was used to create common interface elements such as tabs and accordions [jQueryUI 2008]. Both libraries were manually included in the extension, using a custom build of jQuery 2.0, which excluded unnecessary library parts such as AJAX support or compatibility with browsers other than the latest Firefox version.

Finally, the XMPP protocol implementation used in CoopFox is loosely based on the existing *xmpp4moz* library, available as an open source Firefox extension [Mirra 2009]. However, this library is based on an old Gecko API and has not been updated since 2009. Throughout the development of SpreadVector and then CoopFox, it was entirely rewritten from the ground up, as described further in section 5.2.3.

### 5.1.2 Internal Libraries

Some generic functionality, which has not been integrated in into the SDK so far, was implemented within the CoopFox namespace itself, but separated out into files under the subfolders `utils` and `browser`. The first folder contains mostly single utility functions. Very often these are convenience wrapper calls to Gecko API service class methods, or small code fragments copied from various online tutorials. The second folder includes JavaScript class wrappers to various special parts of the browser UI (tabs, context menus, dialogs, toolbar buttons), which are currently not available through SDK classes or where the SDK implementation was not sophisticated enough for CoopFox.

### 5.1.3 Components

During the transition from SpreadVector to CoopFox, the system has been completely refactored and modularized. The old system consisted of only three distinct subsystems (XMPP protocol stack, chat / sidebar UI, as well as browser content augmentation). The new system is strictly divided into a backend system (`CFBackend`), written in SDK-facilitated classes, and a frontend system (`CFFrontend`), written in functional JS based on jQuery (Figure 23). Both systems provide only a relatively small core implementation, while the actual functionality is then added by distinct modules.

The `CFBackend` core consists of the `main.js` file (5.2.1) which initializes the `XMPP Stack` (5.2.3), creates the toolbar button and watches over the association between windows and sessions. As soon as a session is opened for a window, the central session class `CoopFox` takes over (5.2.2). It initializes the `CFFrontend` and modules.

The `CFFrontend` core consists of a very basic contact list ("roster") and an empty session panel, which provides basic support for tabs and notifications. The modules then use the `XMPP Stack` to communicate with other clients and load further JS and CSS files into the `CFFrontend` to create tabs or roster extensions (see upcoming sections).

Modules are rarely aware of each other. They rather augment each other's behaviour, by adding markup to existing elements or additional data to outgoing XMPP messages.

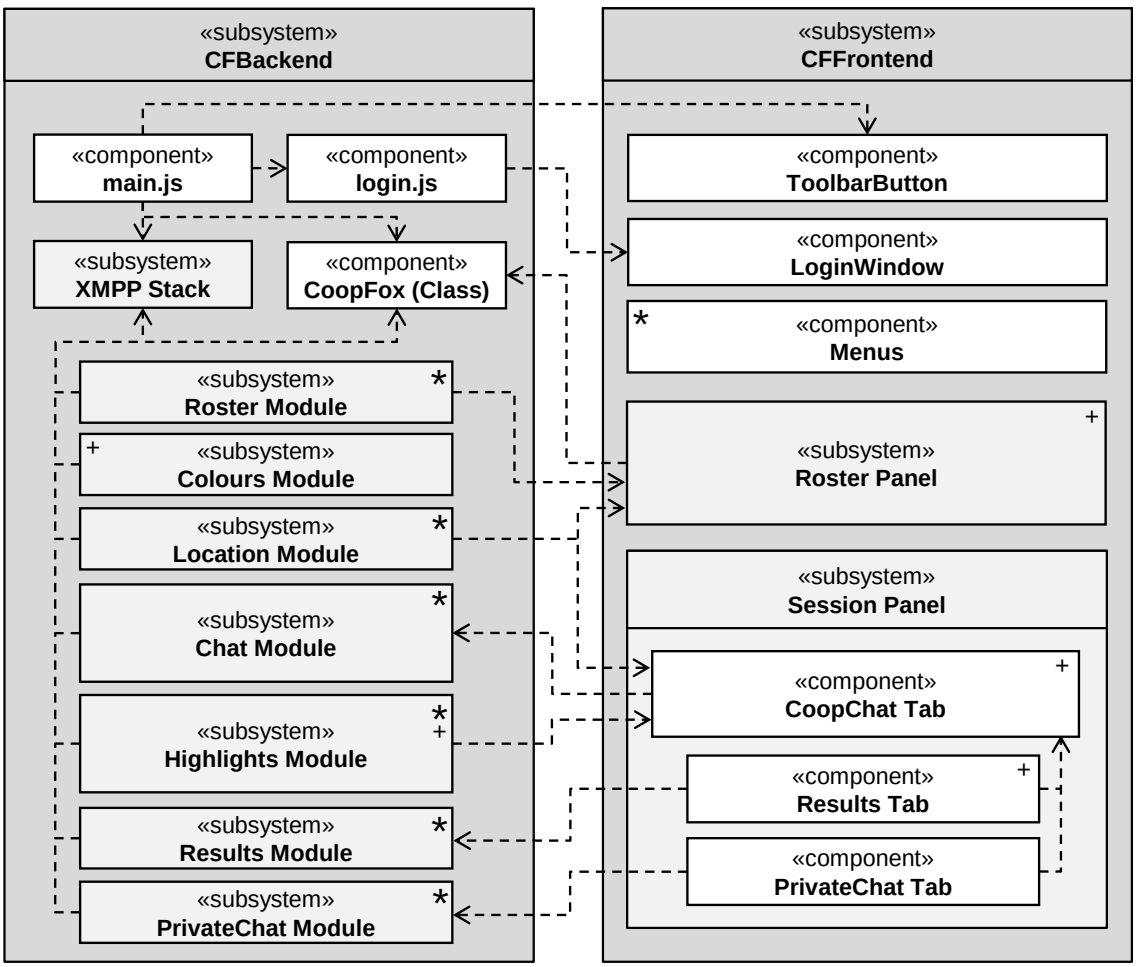



Figure 23. Component diagram for CoopFox. Arrows point in the direction of dependencies / use. Dependencies on the `Menus` component are indicated by a * symbol in the right corner for better visibility. A + symbol is used in the same way for dependencies on the `Colours` module.

### 5.1.4 Events

The publish-subscribe pattern is pervasive throughout the entire code base of CoopFox, because all program activity is always initiated by distinct and self-contained events (user actions, network messages, web page state changes). Since JavaScript does not require strict class signatures and allows to pass and store method references as variables, it provides a very powerful environment for loosely coupled event-driven architectures.

CoopFox embraces this power in the form of a generic `EventHub` class (Figure 24), which is able to both publish events and subscribe to those of other `EventHub` instances. Almost all other classes within the extension are derived from this base class.

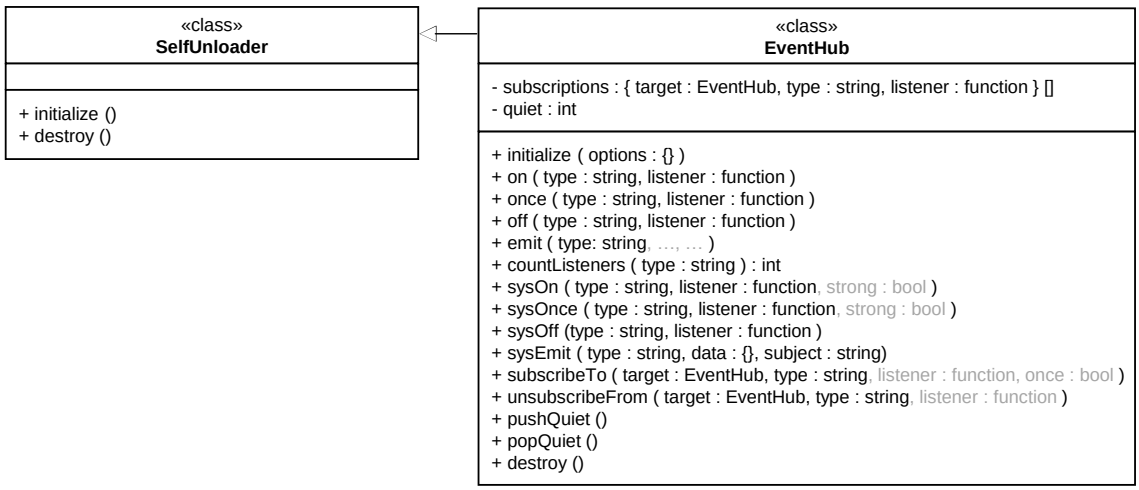



Figure 24. Class Diagram for EventHub and SelfUnloader, which form the base for all CoopFox classes.

Events are identified by name strings, so that any external class may subscribe to the events of a hub at any time, simply by passing an event name and listener method to `on()`. However, the growing complexity of subscription networks required a more controlled solution, so that subscriptions between `EventHub` instances are typically handled by calling `subscribeTo(eventHub, eventName)` on the subscribing instance. By default, the instance then routes future events to its method `_onEventName()`.

The advantage of this solution is that the subscribing instance remembers the subscription and automatically unsubscribes when `destroy()` is called, thus ensuring all instance associations are safely removed. `EventHub`'s parent class `SelfUnloader` ensures that `destroy()` is called at Firefox shutdown or extension uninstall events.

Events can be published on a specific instance using `emit(eventName, arg1…)`, and are passed synchronously to all subscribers. Exceptions are events passed between `CFBackend` and `CFFrontend` (Figure 23), which are processed asynchronously and cannot contain any non-serializable arguments, due to the general sandboxing restrictions between extension code and web document code within the Gecko environment.

From the perspective of the backend code, the roster and session panel documents are wrapped in a `Symbiont` object (SDK class), which exposes a `port` property with the usual `on()` and `emit()` methods. A corresponding `port` is also visible within the global namespace of the sandboxed document, so that arbitrary events can be exchanged.

A local synchronous event system has also been created within both panel documents, using a jQuery object `globalEvents` with its methods `on()` and `trigger()`.

## 5.2 Initiation

Like in the previous chapters, this subchapter covers the basic infrastructure of CoopFox, which determines its installation and session behaviour. Key challenges in this area included the persistent association of sessions to browser windows and the ability to have multiple session and private windows in parallel (5.2.1 - 5.2.2). Next, the XMPP protocol had to be implemented in JS and extended to allow peer-to-peer multi-user chats as well as session merging (5.2.3). Finally, the sidebar panels had to be created using only DHTML, while fitting seamlessly with the surrounding browser UI (5.2.4 - 5.2.7).

### 5.2.1 Main.js File

The `main.js` file manages all global session-independent functionality, such as the toolbar icons, XMPP connection and of course the ability to create and persist sessions. It is executed after the extension has been installed and whenever Firefox is started. To provide quick installations without a browser restart (4.2.2), CoopFox uses the Gecko 2.0 bootstrapping API [Shepherd 2010a], which no longer loads any other files automatically, so that `main.js` is solely responsible for entire initialization process.

Because sessions are tightly associated with browser windows (4.2.4), toolbar buttons and sidebars must be managed individually for all windows. The `main.js` file keeps track of browser windows using an instance of the `BrowserWindowsMonitor` utility class, attaches instances of the `ToolbarButton` utility class to each window and creates instances of the `CoopFox` session class (see next section) when necessary.

It also maintains the central `XMPPThreadHubClient` instance and assigns XMPP threads to sessions (see 5.2.3). New incoming threads (session invitations) are handled with the appropriate dialog, asking the user to open or merge the session. If a connection is required and the user has not provided login credentials, the login dialog is displayed (via `login.js`) and the result stored in the Firefox password manager, if requested.

Lastly, session persistence is also handled in `main.js`, by storing the message history of the XMPP thread when either the CoopFox sidebar or the entire window is closed. Since every persistent action within a session is represented in the chat, sessions can be restored following the “replay by re-execution paradigm” [Manohar & Prakash 1994]. The messages are stored in the context of their window using the `nsISessionStore` service, which allows to retrieve data even after a window has been closed and restored or the entire browser has been restarted. If CoopFox is activated in a window which already contained a session, the user is asked whether the session should be restored, in which case the messages are loaded back into the created XMPP thread instance.

### 5.2.2 CoopFox (Session) Class

Each session is controlled by an instance of the central class `CoopFox`. It receives a browser window instance and an XMPP thread as constructor arguments and initiates all session-dependent functionality. It creates the `CFFrontend`, comprised of the sidebar with roster and panel frames, notifies all available modules to attach their functionality, and then coordinates the initialization sequence between all components with appropriate event messages. During the lifetime of the session, it continues to act as an event bridge, e.g. forwarding roster updates to the `CFFrontend` or link clicks from the `CFFrontend` to the browser tabs. Finally, it also coordinates the shutdown sequence of all components.

### 5.2.3 XMPP Stack

CoopFox contains its own implementation of the XMPP protocol stack, which had to be implemented almost entirely from the ground up, since no existing library was compatible with the current Gecko API (see below) or the desired custom extensions (next page).

Some general JavaScript libraries existed for the purpose of embedded web site chats, but their features were very limited and their code structures vastly different from those of the SDK environment. They were also usually geared towards BOSH as a connection method, rather than native TCP sockets, which would have prevented users from connecting to the majority of XMPP services, including those of Google and Facebook. Merely the *xmpp4moz* library (5.1.1) provided useful code templates, but also had to be rewritten entirely to the newer SDK structures. The basic implementation already took place for the development of SpreadVector, leading to the first four classes in Figure 25. The second four classes provide the new CoopFox functionality, such as session merging.

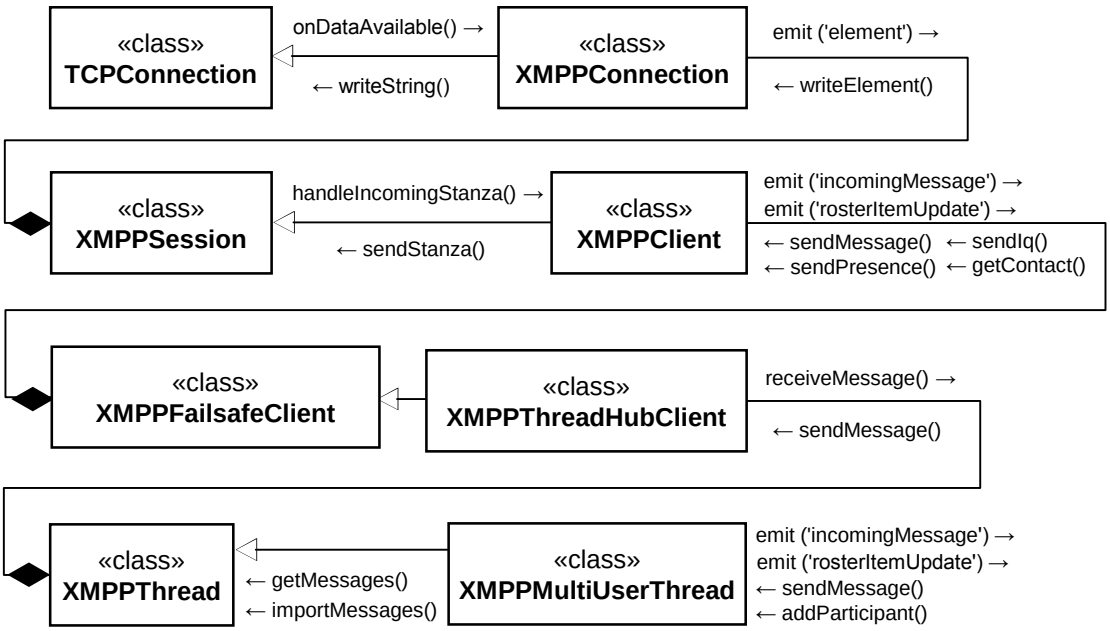



Figure 25. Class diagram for the CoopFox XMPP stack. The top four classes are identical to SpreadVector.

The classes taken over from SpreadVector essentially implement the standard XMPP protocol [Saint-Andre 2011a], starting with an encryption-enabled `TCPConnection` to the server. The `XMPPConnection` class then uses the native Gecko XML parser to capture whole stream-level XML elements including their children and translate them into nested JavaScript objects. `XMPPSession` handles the authentication between client and server, before `XMPPClient` takes over and manages the exchange of message and presence stanzas with the contacts in the user's roster [Saint-Andre 2011b].

The first addition made by CoopFox is the `XMPPFailsafeClient`, a simple transparent wrapper which captures any sort of connection loss or other error in the underlying layers and automatically attempts to re-establish the connection.

`XMPPThreadHubClient` acts as a message multiplexer, which allows users to have parallel conversations, using the standard `<thread>` element in each `<message>` to distinguish them. The hub creates individual `XMPPThread` instances, which have the same interface as an `XMPPClient`, but send and receive only in their own thread.

In order to facilitate persistent sessions, latecomer support and session merging, each `XMPPThread` also stores a chronologically sorted history of its conversation, which can be retrieved using `getMessages()` and loaded using `importMessages()`. Importing does not replace the current state, but merges any new messages into the conversation at the correct chronological position, ignoring messages with identical IDs. The replay paradigm was also implemented at this point, via a `replayMessages()` method, which re-emits the `incomingMessage` event for each history entry.

Finally, `XMPPMultiUserThread` provides CoopFox's innovative lightweight initiation features: peer-to-peer multi-user chat and session merging. There is in fact an official XMPP extension for multi-user chats [Saint-Andre 2012a]. However, this approach was dismissed, since it is based on a strict room metaphor that requires explicit session creation and does not allow users to merge existing sessions into one (see 2.2.2).

My solution uses message multicasting [Hildebrand & Saint-Andre 2004] to emulate a multi-user chat room using only peer-to-peer communication. Session participants address their outgoing messages to all other known participants. The challenging part about this approach is how to add and remove participants without a central registry. Special `<message>` elements are used to communicate participant status changes. They all contain a `<coopfox xmlns="http://coopfox.net/xmpp/namespace">` element, which is a generic container for all protocol extensions introduced by CoopFox. This element then holds an element `<participant jid="…" action="…" />`, with the JID of the affected client and an action (`join`, `leave` or `reject`).

When a user invites a new participant, a `join` message is sent to the new client, where the invitation confirm dialog is shown. At the same time, the message is also sent to all current participants, to notify them of the new participant and the person who sent out the invitation. The invited user can respond with a `reject` message to explicitly deny participation, or of course leave the conversation by closing CoopFox, which automatically triggers a `leave` message to all other participants. If another session member invites a person who is unknown to the local user, the local client automatically attempts a temporary presence decloak handshake [McVittie et al. 2012] with the remote client, so that both peers can see each other's online status for the duration of the session.

Session merging is achieved through another XMPP extension, in the form of a custom synchronisation protocol. Once a client joins a session, it attempts to poll the existing conversation history from its peers by sending them an `<iq>` request stanza with a query namespace of `http://coopfox.net/xmpp/namespace/sync`. A remote client responds by sending its message history in the query result for the new client to import. In return, all existing session participants also poll the new client, leading to a merged conversation if both sides already had messages in their history. To avoid redundant full history exchanges between peers, each sync request contains a checksum, based on the IDs of all known messages. The polled client can compare this checksum against its own history and only send the missing messages or even reply with an empty result.

The sync process further serves two other important purposes. The first is time synchronization between clients. To ensure the correct chronological position of imported messages, all sync replies contain a timestamp. If a client receives a timestamp which lies ahead of its local time, it corrects the timestamps of all its stored and future messages forward accordingly. This way all clients end up agreeing on one effective system time. The chat frontend reverses this offset to display messages in local time again.

The second additional purpose of synchronization concerns re-initiation of former sessions. When a client loads and replays an old message history, it also replays all join messages, thus attempts to synchronize with former participants, and also sends directed presence decloak handshakes to off-roster contacts. If other former participants also currently have the same session opened, their clients interpret the sync request as an implicit sign that the polling client has re-joined the conversation and list the corresponding contact as an active participant again. The synchronization response meanwhile ensures that all clients receive any messages they might have missed.

### 5.2.4 Chat Module

This module provides the core functionality of the CoopChat tab, routes sent and received messages between the `CFFrontend` and the XMPP thread of the session, and provides other generic infrastructure for the chat interface, such as the remote keyboard activity indicator (4.3.2) which must be associated with any input field.

The central engineering challenge herein was to make all functionality easily reusable and extensible by other modules. The `PrivateChat` and `Results` modules reuse many elements in their tabs, like the message list and input forms. Generic constructor functions were therefore written for these elements, which create the necessary DOM elements including some essential custom methods and jQuery event handlers. Generic global port events allow other modules to send and intercept messages. The local events `chatMessageReceived` and `chatMessagePost` allow the other modules to pre-process any raw incoming content and post-process the inserted HTML.

In order to find and change existing messages, each posted history element retains the original XMPP message ID as its HTML element `id` attribute. Other meta-information such as the message sender and message type (text, status update, invitation, etc.) is encoded in HTML classes for easy bulk updates on entire message groups. Classes are further used to mark unseen messages and trigger the fade in animation once they scroll into view (4.3.2). Lastly, messages also have to retain their timestamp as a hidden data property, so that merged content can be inserted at the correct position in the history.

Message deletion is also implemented in this module, once more as a custom extension to the XMPP protocol. To delete messages, a client send out a `<message>` stanza with `<coopfox><chat id="[msgid]" action="delete"/></coopfox>` to the others and itself, causing all clients to hide the indicated item. A message with an `undelete` action causes the item to be shown again. All clients store received delete messages in a stack, so that any user can undo deletions at any time.

A general challenge for the implementation of the sidebar was to make the look and feel of the HTML panels harmonize with the surrounding browser interface elements. In SpreadVector, the added panels still felt very much like additional web pages displayed alongside the browser. Using a number of tweaks, CoopFox is able to emulate normal application user interface elements much better. These measures included:

- Use of jQuery UI to generate common single-user interface elements such as tabs.
- Using native system menus via the Gecko API, rather than HTML overlays.
- Replacing the standard web page context menu with a custom context menu.
- Disabling text selection on UI elements via the CCS3 `user-select` property.
- Enforcing the system standard mouse cursor via the CSS `cursor` property.
- Suppressing any tooltips which revealed the system URL of the panel documents.

### 5.2.5 PrivateChat Module

The purpose of the additional private chat was to allow spontaneous 1-on-1 conversation between users, and also to provide basic communication compatibility with other standard XMPP clients. Its implementation was rather simple, since most of the infrastructure is reused from the `Chat` module. Instead of `XMPPMultiUserThread`, the private chats use the alternative `XMPPContactThread` class, which simply receives all unthreaded messages coming from a specific sender.

Since users may wish to continue an interrupted private conversation, the thread history is also persisted, however, not in context to any session or window but globally. Thus, a private chat tab with a specific person can be opened in any session panel or even in multiple session panels at once and always contains identical content. Incoming chats are always opened in the currently active window first.

In order to guarantee the standard messenger compatibility, private chats were tested between a CoopFox instance and the open source instant messenger Jitsi [Ivov 2011], using both a local OpenFire XMPP server [der Kinderen 2007] as well as the GoogleTalk and Facebook Chat services [Google 2012] [Facebook 2010]. No functional problems were discovered. It merely became apparent that CoopFox will have to support more common XMPP extensions (e.g. avatar pictures) to satisfy as a standard messenger.

### 5.2.6 Roster Module

The `Roster` module provides several advanced management functions for the contact list and the presence information of the local user. This includes changing the user's availability state and custom status message via the roster panel menu.

A `SubscriptionManager` class controls actions such as adding, removing and renaming contacts, by sending and listening for the necessary `<presence>` stanzas through the global `XMPPClient` instance. The corresponding commands are made accessible through the context menu of contact list items and the central panel menu.

### 5.2.7 Colours Module

The colour-assignment of session participants was moved into a separate module, since several other modules depend on this function independently. Initially, this module was also supposed to provide a colour selection user interface, but this has not been implemented so far. At the moment, user colours are selected successively from five predefined sets, each containing a foreground and a lighter background colour. An option in the extension settings allows users to select whether the local colour should always be forced to red, or whether session colours are assigned in the strict join order of participants, thus ensuring that all session members see the same colours.

The `Colours` module maintains a loosely coupled relationship with the other modules, most of which are not even colour-aware. In the sidebar, the corresponding modules merely tag user-related items (roster items, messages, selection markers) with an HTML class based on the user's JID. The `Colours` module then defines a global CSS rule to assign the corresponding colour to the tagged elements.

Some exceptions from this principle have to be made for the `Highlights` and `Location` modules, which assign explicit colours to text markup or browser tabs. For this purpose, the `Colours` module injects a colour lookup method into the `CoopFox` class object, using it similar to a service lookup container.

## 5.3 Transition

With the core infrastructure established, this subchapter describes the modules responsible for CoopFox's awareness and coupling features, as introduced in 4.3. One of the key challenges in this area was the collection, distribution and visualisation of location awareness data across clients (5.3.1). Further extensions to the XMPP protocol were required to implement the threaded discussion and message pointing features (5.3.2), and as well for the separate result collection (5.3.3). Remote text selection is discussed in the next subchapter, in the same context as persistent highlighting.

### 5.3.1 Location Module

This module is responsible for CoopFox's location (4.3.3) and visitation awareness support features (4.3.4), which require the detection, exchange and visualisation of page access events between clients. Because several other modules must also react to such events, their detection is implemented in a general utility class `WindowTabsMonitor` which is instantiated and made publicly accessible by the central `CoopFox` session class. Through it, all modules receive events whenever a new web document has been loaded into a tab (`documentReady`), is unloaded from a tab (`documentUnload`), and also whenever the user switches between tabs (`documentActivate`).

The `Location` module class `WebLocationTracker` subscribes to these events and notifies other clients in the session of page changes. It achieves this by broadcasting a special `<message type="headline">` stanza to other clients, with the content `<coopfox><location/></coopfox>`. The `location` element contains several attributes describing the currently active web page of the user, including the page URL, title, and favicon URL. The XMPP message type `headline` indicates non-chat content, which I use for transient updates not to be archived by `XMPPThread`. However, the same location element is also appended to all regular outgoing message stanzas, so that the receiving client can attribute chat history entries to the page the author was looking at.

To ensure that transmitted URLs are comparable between clients and present no security risks to the local user, each URL is normalized using a series of heuristic rules, which remove user-related information such as session IDs and other surplus tracking parameters. At the moment, only a few example rules for popular web sites have been hardcoded into the normalization filter, including YouTube and Amazon.

Some users may still see it as a violation of their privacy to share their visited web pages with others in real time. This feature can therefore be deactivated, so that the cleartext page title and cleartext URL are not transmitted when the user visits a new page (location attribution for chat messages is unaffected). However, in order to provide basic visitation and location awareness even in this privacy enhanced mode, an md5 hash string of the visited URL is still sent to the other clients. This way the other CoopFox instances can still indicate whether other team members are seeing or have seen a page once their local user visits it, by comparing the local URL hash code to the remote one.

One last engineering challenge to overcome was that many web sites now update their page content entirely via AJAX, without any full document reloads. Facebook and Google Search are such examples. For this reason `WebLocationTracker` attaches an `nsIMutationObserver` to each document, to capture DOM modification events and notify the other clients if these changes have modified the page title or URL.

The frontend implementation of the `Location` module is rather straightforward. Additional jQuery scripts post-process the rendered HTML for contact list items and chat messages, inserting or updating the location links and icons, as described in section 4.3.3. The visitation history for each session member is maintained in the module frontend. The roster and session panel scripts remember the hash codes for URLs each user has already visited in a hash table structure, allowing quick comparisons with locally visited pages. Message entries further receive the hash code of their respective location URL as an HTML class. This allows efficient bulk updates for all messages which belong to a specific web page, for instance to highlight all messages for the currently viewed web page with a green outline, or grey out the icons on pages the local user has yet to visit.

More technically challenging to implement were the visitation indicators for links within web pages (described in 4.3.4). These are inserted by the `LinkTagger` class. The main problem here was performance optimization, since checking every link in every tab whenever any session member visits a new page is computationally expensive. The same goes for checking every link in a newly opened page against every page already visited by any session member. Using hash table lookup for visited URLs reduces the complexity of both computations down to O(m) per page, with m as the number of links. To avoid any perceivable slowdowns in browser responsiveness, these checks are executed asynchronously when Firefox is idle, at most twice per second, and only for the currently active tab. Since link URLs have to undergo the same normalization filter and md5 hashing as the transmitted visitation data to be comparable, an additional cache was introduced for these computations, to avoid redundancy when links are checked again.

Lastly, the colour-coding of browser tab selectors (4.3.3) is performed by another dedicated backend class called `TabTagger`. Since all UI elements of Firefox are themselves specified in an XML document (XUL) with CSS styles, changing the colour and text style of a single selector is a simple process that requires no programmatic overrides to the browser application. This CoopFox feature has since been tested in combination with several other browser extensions which influence tab appearances. No functional problems or unexpected interferences have been observed.

### 5.3.2 Chat Module Extensions

The threaded discussion and message pointing features for the CoopChat (4.3.6) were implemented directly as part of the `Chat` module to which their functions relate.

Hierarchical threading is natively supported by the XMPP protocol, though rarely implemented by any messenger client. Message sub-threads are created by including `<thread parent="[Id]">[subId]</thread>` within a `<message>` stanza. CoopChat comments carry the message ID of their parent message as their subthread ID. This also required a modification to the `XMPPThreadHubClient`, which now also had to evaluate the `parent` attribute when assigning messages to `XMPPThread` instances.

Message pointing is a pure frontend feature, which required no changes to the `CFBackend` components. When a message in the history is clicked, this is broadcasted via another custom message stanza, similar to the mechanism for delete / undelete actions: `<coopfox><chat id="[msgid]" action="select"/></coopfox>`. Deselection is expressed by omitting the `id` attribute. Like the transient location updates, this message is sent with type `headline`, excluding it from the persistent chat history.

### 5.3.3 Results Module

This module provides the results tab (4.3.7) which allows groups to collect important messages from the CoopChat in a separate list and prioritize them collaboratively.

The tab was conceptualized as an alternative view of the already existing content in the CoopChat, similar to a view in a relational database. Since additional comments or other HTML markup might be added to a message after it was added to the results, it was further important that the results view entries update automatically with their originals. However, achieving this in DHTML presented a challenge, because DOM trees do not allow duplication of nodes by reference. To emulate this behaviour, at first, a deep-copy of the original message is inserted into the result list. Then, an `nsIMutationObserver` is attached to the original message, which detects any changes to the underlying DOM subtree and triggers a refresh of the deep-copy for the results entry. At the same time, all user input events such as mouse clicks on the copied element have to be captured and executed on the original, to preserve features like commenting or message pointing.

Synchronization of results between clients is once again achieved via a custom XMPP message: `<coopfox><result id="[msgid]" action="…"/></coopfox>`. The action attribute value can be one of `up`, `down` or `remove`. The `up` action increases the result priority by 1 and also creates a new result if it did not exist before. The `down` action decreases the priority by 1, and the `remove` action deletes the result entry. This simple principle was chosen because the `up` and `down` operations are commutative, so that it does not matter when distributed clients receive them in deviating order. While a non-commutative approach (e.g. drag and drop) could still have been implemented for synchronous work, using operational transformation [Ellis & Gibbs 1989], this would have become significantly more challenging with respect to session replay and merging.

### 5.3.4 Notes Module

The function of this module is to provide a tab with a text box for private local note taking. The text content is persisted in context of each window in the same way as the CoopChat. Further development of this feature was abandoned at an early stage, because users saw no use for it. This is the reason why it was not mentioned in chapter 4 or the component diagram in this chapter, and why the module remained disabled during the final usability study described in chapter 6. However, as of now, it is kept as part of the code base, in case a use for it presents itself in future applications.

## 5.4 Integration

This subchapter describes the implementation of CoopFox's contextual collaboration and web augmentation features, as first suggested in subchapter 2.4 and presented in 4.4. The main challenges in this area revolved around the remote selection and quote features of the `Highlights` module (4.4.2), which required communication of DOM tree selections between clients, with enough tolerance to handle minor page content deviations on personalized web pages. A first organisational challenge in this was also to develop communication paths between `CFFrontend`, `CFBackend`, and the web augmentation overlays within web pages (4.4.1). All of these subsystems have to be able to exchange fine-grained web location information, in order to act as the desired replacement for manual copying and pasting of page content and URLs (2.4.1).

### 5.4.1 Location Module Extensions

The support features for quick location referencing (4.4.2) were implemented directly as part of the `Location` module. Most prominently, this includes the default function of the *Send* button, to post a link to the current web page whenever no message text has been entered explicitly. Since many frontend features require contextual knowledge of the web locations of session members, a very general `contactLocation` event was implemented. It is sent to the global context of both `CFFrontend` panels, as soon as either the local user or any remote user opens a new web page or changes the active tab. The event contains the new location dataset (5.3.1) and the contact dataset of the respective user. Thus, any module frontend is able to include location-contextual features.

In return, users have to be able to open a link from the chat in the browser, optionally within a new tab. The corresponding left, middle or context-menu clicks on `<a>` elements are captured by the `CFFrontend` panels, relayed to the `CFBackend` via a `linkClick` port event, and then executed by calling `WindowTabsMonitor.openUrl()`. The behaviour of this method is slightly more intelligent than the default behaviour of the browser, in that if the requested document is already open in a tab, the specific tab is simply activated, rather than loading the same page again. Further, if the given URL contains a fragment identifier (#), pointing to an HTML element by its ID, the referenced element is scrolled to the centre of the viewport. Both functions combined allow users to jump quickly between specific page positions across several open tabs.

The last piece of infrastructure necessary for the location sharing feature is the ability to post links in the chat. The basic XMPP protocol does not allow HTML messages. While there is an extension for this [Saint-Andre 2012b], a simpler and more generic method is to send plaintext URLs and convert them into links in the receiving client. This approach also supports users who still prefer to copy and paste URLs manually. However, CoopFox can still provide a synergistic advantage, because the location datasets exchanged between clients contain both URLs and page titles. This allows the module frontend to build a map between these two page properties and translate any URL which has already been visited by session members into a link with a human-readable title.

### 5.4.2 Highlights Module

This module provides the ability to remote-select text on web pages, for the purpose of both transient pointing (4.3.5) as well as persistent Direct-Quotes in the chat (4.4.3). Its implementation is directly based on the highlighting function in SpreadVector, but has been improved in several ways to become more flexible and robust.

To my knowledge, no other project has ever used highlighting as a pointing method for collaborative browsing. However, implementation-wise the challenge is very similar to the anchoring of annotations in digital documents, which has been discussed in several publications [Bernheim Brush et al. 2001]. Consistent with the recommendations by Bernheim Brush at al., both SpreadVector and CoopFox have refrained from using absolute path descriptors and character offsets to reference DOM positions. Given that web pages are often partially personalized [Niederhausen et al. 2010, p. 397], this approach would often result in broken references due to deviations in HTML structures between clients. More robust references are provided by descriptors based on the selected text and its surroundings. SpreadVector implemented this approach in a very minimalist way, by transmitting only the user-selected text to other clients when creating a highlight.

The selected text alone can of course occur very ambiguously within a web page (especially for short selections of few characters), so that SpreadVector users often received error messages telling them that their selected text could not be identified uniquely on the remote client side. For this reason, CoopFox uses a more sophisticated approach based on hash signature hints to disambiguate text locations, which has already been implemented in various similar applications [Bernheim Brush et al. 2001, p. 286]. Besides the actual text to select within a web page, each highlight dataset in CoopFox also contains: hash signatures of the preceding and following 50 characters, a hash signature of the text content of the surrounding block level element, hash signatures of the preceding and following block level elements, as well as a hash signature of the absolute XPath [Clark & DeRose 1999] of the referenced content element.

The receiving CoopFox client begins by finding all occurrences of the target text in the target web page using the `nsIFind` service of the Gecko Engine, which delivers an array of `nsIDOMRange` objects. Next, the listed hints are used successively to eliminate mismatching range candidates, until only one such remains. This means that the XPath hint is only used when the selected text including three surrounding text blocks occurs multiple times in a web page. In all other situations, the absolute position of the found passage in the HTML document is not considered, so that the desired robustness against structural deviation between clients is still present. A practical scenario this has been tested with during development is the ranking of search engine results, which often varies between users. CoopFox was always able to highlight excerpts of a specific result independent of its position within the listing, only confronting users with error messages if the selected text could not be found at all in the page seen by another session member.

Communication of highlight datasets between clients occurs once more by means of a custom XMPP `<message>` stanza. If a user selects text in a web page or uses the explicit Direct-Quote function from the context menu, the local client sends out a stanza containing `<highlight type="insert" url="…"><text>…</text></highlight>` (wrapped in `<coopfox>` as usual) to all other session members and itself. Multiple `<text>` elements can be contained in case of a multi-part selection, each of which encloses the text to select and contains the hints as arguments (example stanzas in Appendix C). Persistent highlights (Direct-Quotes) are sent as normal `type="chat"` messages, while transient highlights (pointing) are sent as `type="headline"`. Lastly, remote clients have the ability to return an error to the highlight sender in the form `<highlight type="error" id="[msgid]" url="…" reason="…">`, for instance in case the target text could not be found in the page with the target URL by the remote client. This labels the highlight as failed in the CoopChat for all participants and also display a brief popup notification next to the mouse cursor of the sender.

Once a designated highlight has been received and mapped to a single `nsIDOMRange`, it must be visualized. This occurs by surrounding the range contents with an element of type `<span class="coopfox-highlight">` in the sender's background colour. The challenge in this approach lies in the DOM tree structure, which dictates that only text nodes but not element nodes can be split. To be able to split off the selected content at both ends and wrap it in the `<span>`, the selection must begin and end within the same element node. Again, SpreadVector only delivered an error message if this criterion was not met. CoopFox is now able to salvage these situations, by expanding selections around inline elements and breaking up the selection into parts if it crosses block boundaries.

The coordinator of all this functionality is the module class `DomHighlighter`, which receives events from a `SelectionListener` for the local user input and the `XMPPThread` instance for incoming remote selections. Its main purpose is to keep track of all highlights persistently in the background, so that new highlights can be inserted or existing ones removed, even if their corresponding pages are not currently loaded in any browser tab. As soon as a URL with associated highlights is opened, the highlighter inserts the necessary markup into the DOM, including the highlight wrap itself, but also the close button, the scrollbar radar marker and the annotation overlays (see Figure 22). The class then keeps track of these inserted elements, handles their click events, and removes them again if the corresponding highlight entry is deleted from the CoopChat.

The module extends the CoopChat, in order to print highlight messages in a human readable format. They are rendered as links pointing directly to the highlight wrapper element in the referenced page (which retains the XMPP message ID as its HTML ID). No direct communication takes place between the frontend and backend of the module, but both systems simply listen for the same incoming and outgoing XMPP messages.

## 5.5 Summary

CoopFox presents a number of innovative solutions for collaborative web browsing, not only regarding its interaction design but also its implementation. This chapter has detailed some of the greatest engineering challenges which had to be overcome in the development process. In order to create a browser extension with many complex features, the latest JavaScript technologies and frameworks have been used to create a modern component-based and event-driven architecture which can easily be extended further.

An XMPP client library for this environment had to be created from the ground up, ensuring full native compatibility with a variety of available XMPP servers and services. Based on this infrastructure, an entirely new peer-to-peer multi-user chat protocol has been developed, for the purpose of lightweight session initiation, using a combination of message multicasting, invitation, and peer-to-peer synchronization. The approach rejects the common room metaphor of multi-user chats and allows retroactive session merging instead (i.e. two or more users can combine their existing chat histories into one).

To facilitate seamless transitions between solitary and group work, a location and visitation awareness support system has been devised, which automatically gathers and distributes metadata on visited web pages via a custom XMPP protocol extension. An enhanced privacy mode is available, which only distributes these datasets using obfuscated hash signatures, still allowing basic location and visitation detection.

Lastly, the concept of contextual collaboration has been implemented in an adequate form for the web browser, using web augmentation techniques to facilitate shared text highlighting as both a quick pointing tool and a method for persistent content annotation. By avoiding references based on absolute XPaths or character offsets, the annotation implementation of CoopFox is robust against dynamic and personalised page content. Instead, references use a combination of text search and hash-based context hints, which still allows unambiguous identification of text passages in the vast majority of situations. The highlighting functions have been tightly integrated with chat and location tracking, allowing users to quote by selection, annotate via chat comments or jump directly to quoted passages, thus creating a powerful tool for quick fine-grained content referencing.

# 6 User Study

This chapter presents the results of a user study conducted with the final CoopFox implementation, in order to assess its compliance with the three key principles from subchapter 1.2. Results include both observations of groups performing collaborative web research tasks, as well as direct user feedback through questionnaires.

Separate subchapters are used again to addresses *initiation*, *transition* and *integration*, because these aspects were also addressed in different forms during the study, involving separate tasks and questionnaire designs. The introductory method subchapter 6.1 merely presents the general overview of how the study was conducted. Each following subchapter then contains its own detailed description of the respective tasks and assessment methods, followed by a presentation of the results and their discussion.

Subchapter 6.2 addresses the initiation aspect, based on the criteria of technology acceptance (perceived usefulness and perceived ease of use) with special focus on the session merging approach. Next, subchapter 6.3 addresses seamless transitions between tightly and loosely coupled work, by pitting CoopFox against two baseline systems (screen sharing, separate browsers) in a repeated measures trial. Finally, subchapter 6.4 briefly reviews participant feedback on the specific contextual collaboration features.

## 6.1 Method

Groups of three participants performed a series of short fictional collaborative research tasks under laboratory conditions, and reported on their subjective impressions through both multiple-choice as well as open questionnaires. Task-related feedback was collected immediately after execution, while general usability feedback was collected at the end of each session. Participants were additionally recorded on video during the entire process. In total, each group spent approximately one hour completing both tasks and questionnaires.

### 6.1.1 Participants

Twelve groups (36 individuals) participated in the study (25 male, 11 female; mean age 25.3, SD 4.35), as well as one additional pilot test group. Recruiting took place primarily in person, on the computer science campus at the University of Bamberg, and through several university-related mailing lists and Facebook groups. Potential participants were encouraged to sign up in groups of three or grouped together with other individuals according to their preferences, in order to avoid forced collaboration between strangers. In the end, only three groups contained subjects who did not know each other in person prior to the study. Seven groups were comprised of all-male participants, three groups were all-female and two groups contained one female along with two male subjects.

All participants were either students or PhD students. The vast majority (32 of 36) rated their own computer skills as high or very high, while 4 considered them average. Fields of study mostly included various forms of computer science (28 of 36). However, 12 of these individuals were part of the “Computing in the Humanities” master program, with backgrounds including sociology, psychology, art history, and communication.

All participants indicated they were using the internet to research information at least once per day. Seven subjects also reported collaborating with others during their online researches at least once per day, and 13 more said they were doing so at least weekly.

### 6.1.2 Setting

All study sessions were conducted by the same experimenter over the course of two weeks in November 2013, in the usability lab of the Cooperative Media Lab at the University of Bamberg. Participants were seated at desktop computers with HD1080 screens, facing three different walls at 90° angles in a small room, approximately 1 m apart from each other. Two additional mobile separator walls were placed at 45° angles between the participants, to prevent them from seeing each other's screens peripherally, but still allow verbal communication, thus simulating a distributed setting with audio connection. The same method has also been used by similar studies [Morris & Horvitz 2007, p. 9] [Wiltse & Nichols 2009, p. 1788] [Atterer et al. 2007, p. 85].

All three computers were running Windows 7 in identical configuration, as well as Firefox 22 with the CoopFox extension already installed and configured to connect to a dedicated OpenFire XMPP server [der Kinderen 2007]. Additional software required by specific tasks is described throughout the respective following subchapters.

Another HD1080 screen at the fourth wall of the room was used to play tutorial videos, explaining the use of CoopFox at appropriate times before tasks. All participants were able to see the videos without the need to leave their seats at their workstations.

### 6.1.3 Schedule

Before each study session, all messenger accounts used for tasks were configured to display the first names of the group members (each assigned to a random workstation). Upon arriving, participants were seated at the computers configured with their names.

After signing a separate consent form, every participant received a stapled collection of worksheets (Appendix A, 17 pages total), containing both the task instructions and questionnaires for the entire session in their correct interleaved order, also respecting any repeated measures randomization order for the group. At the bottom of each instruction or questionnaire sheet followed a statement asking participants not to advance to the next page until instructed to. Before doing so, the experimenter always ensured that all group members had finished reading or filling in each page and that there were no questions.

The first page of the worksheet collection introduced participants to a framing narrative for the entire session, explaining that they were preparing to write a guidebook on how to live healthy as a college student, for which they had to perform several web researches together, in order to acquire knowledge on different aspects of this topic. This general assignment was comprised of two main parts, each containing several tasks.

Part one of the study required approximately 40 minutes to complete and contained the tasks and questionnaires addressing seamless coupling mode transitions (6.3.1). Part two required approximately 20 minutes to complete and contained the task and questionnaire addressing lightweight initiation (6.2.1), as well as three pages of general questions addressing the usability and perceived usefulness of CoopFox.

Contextual collaboration (6.4) was not addressed by any specific part of the study, since the concerned specific features of CoopFox were already used during the other web research tasks. The evaluation of this aspect follows primarily from observations, usage statistics logged by CoopFox, as well as qualitative feedback from open questions.

After completing all questionnaires and returning the filled in worksheet collections, participants were encouraged to discuss their group work experience, ask questions about CoopFox or give further feedback. Finally, they each received a formal debriefing letter.

## 6.2 Initiation

This part of the user study assessed whether CoopFox's session merging approach fulfils the criteria for lightweight initiation (2.2.2), i.e. provides a quick and comprehensive way of entering into collaboration while carrying over the state of previous independent work. The measures of this assessment are based on the technology acceptance model (2.2.1): *perceived usefulness* and *perceived ease of use*. These measures are applied not just to session merging, but also to the entire user experience with CoopFox, because generally good usability is crucial to the targeted support of casual and spontaneous collaboration.

### 6.2.1 Task

In the second part of the study, participants were asked to merge three separate CoopChat sessions into one, and then try to comprehend and discuss the content each of them had contributed to the joint chat history. At this point, they had already learned the general operation of CoopFox during part one of the study (see 6.3.1). The specific steps for session merging were demonstrated in a tutorial video (1:10 minutes length).

Upon completion of the video, each participant was given a browser window with an existing CoopFox single-user session. The CoopChat entries of each user belonged to one of three different topics already used during part one of the study. Every participant received one link to an entire web page as well as two specific quoted text passages from different web pages, and was given one minute to familiarize him- or herself with the referenced content without talking to the other group members. Next, the group was instructed to merge their CoopChats, by choosing one group member to invite the others.

Once a joint session was created successfully, each group member was told to pick a favourite result and present it to the others, by visiting the respective page together with them and discussing its content. The purpose of this exercise was to determine how easy it would be for users to identify their own prior contents within the now merged chat history and how easy it would be for others to make sense of new content they had never seen before. To make this task more challenging, the entries from the three single-user sessions were given timestamps which ensured that they would appear interleaved.

### 6.2.2 Observations

All groups were able to join their CoopChats quickly and without major problems. However, some usability issues became apparent in cases where groups did not follow the instruction of picking one group member to invite both others. Situations occurred where two participants attempted to invite each other simultaneously, which led to confusion about who should accept the incoming request. Another similar problem occurred when two group members had already merged their sessions and the third then attempted to invite one of the two. CoopFox does not offer a merge option in this case, because the invitation is directed at only one of two owners of a shared session. Such conflicts had to be resolved by the experimenter. Upon explaining what had happened, it further became apparent that most participants were unable to understand these problems.

An important general observation for the entire study was that none of the participants seemed to have any major problems replicating and abstracting from the operations in the tutorial videos within their own CoopFox instances. There was not a single case in which a person required more help from the experimenter than a reminder of the video content.

### 6.2.3 Questionnaire

Part two of the study was followed up with four pages of questionnaires (14-17), covering both satisfaction with the specific CoopFox session initiation, as well as general usability and learnability aspects. Balanced semantic differential scales were chosen as the dominant item form, because part two of the study did not involve a comparison between CoopFox and other tools. Therefore, statement agreement measures from Likert scales would not have yielded very conclusive results, since there was no baseline to compare them against. SD scales, on the other hand, are at least able to reveal a bias towards one of two given options. Some questions also prompted the subjects to compare CoopFox with their usual method of online research and rate their trial experience as better or worse, thus introducing an implicit baseline condition in the form of an undecided answer.

Page 14 was dedicated to *perceived ease of use* for the session merging task, asking subjects about the intuitiveness of the process, whether they deemed it too formal or informal, and how they were able to cope with the interleaved order of merged CoopChat histories. Page 15 followed up by addressing the general usability and learnability of CoopFox. Three questions on this page were taken directly from the INTUI questionnaire [Ullrich & Diefenbach 2010] (CF_2, CF_3) and two more (CF_5, CF_6) were inspired by the System Usability Scale (SUS) [Brooke 1996].

In order to assess *perceived usefulness*, page 16 included a series of questions asking subjects to rate the usefulness of several initiation-related features and concepts of CoopFox. This included the use of a standard Firefox browser, standard XMPP server compatibility and of course the session merging function. Additionally, page 17 included three questions inspired by SUS, asking participants for a final verdict on whether they would like to use CoopFox in the future for private or collaborative research, and whether they would ask others to use it as well, if they wanted to collaborate with them.

### 6.2.4 Results

The feedback questions about perceived usefulness and perceived ease of use returned very positive answers for both the session merging function in particular as well as for CoopFox in general (see Appendix B for full result histogram set).

The concept of merging existing sessions was received very well, with 24 participants describing it as useful or very useful for their work and 4 even as indispensable (CFF_4). The occasionally observed confusion did not reflect in much negative feedback. 32 of 36 participants found the concept easy or very easy to comprehend (Mrg_1_1) and only 4 indicated feeling confused by it (Mrg_1_5). Further, 27 participants found it easy or very easy to understand the content of other group members after merging (Mrg_2), and 25 said the same about explaining their content to their peers after merging (Mrg_3).

The feedback on general usability and learnability shows that most subjects found the program easy to understand and use. This becomes particularly apparent in the questions inspired by INTUI (Figure 26). Moreover, 18 of the 36 participants indicated that they would have likely understood the extension intuitively without instructions (CF_4).

The final acceptance verdicts were also very positive, with 24 participants indicating they would like to use CoopFox regularly for collaborative research (CFO_1), and 27 stating they would ask others to use CoopFox in order to collaborate (CFO_3).

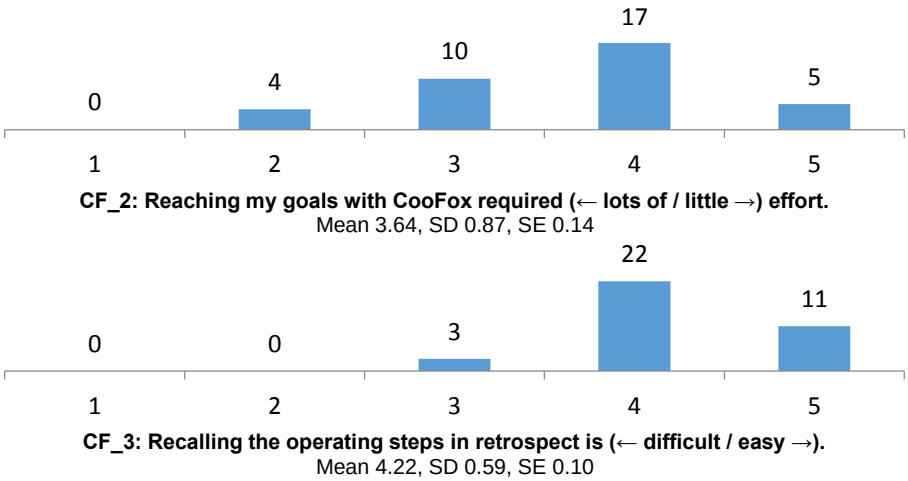



Figure 26. Questionnaire results on general usability / learnability. Captions are paraphrased from German.

Mixed opinions existed on whether the chronological interleaving of the merged CoopChat messages felt logical (Mrg_4). 18 participants found it logical, while 11 found it illogical and 6 were undecided. This also correlates with several qualitative feedback remarks. One participant suggested that it would be nice to switch to an alternative sorting by author, in order to get a better overview of the contents submitted by each person. Very likely, this result also relates to the general impression held by many participants that there was too much information in the CoopChat (CF_7, discussed further in 6.3.4).

Some further unexpected results originated from the question block regarding technology acceptance. Other than anticipated, neither using their existing web browser (CFF_1) nor using their existing instant messenger account (CFF_2) was very important to the participants. The most common classification given to the browser extension approach was “useful” (14 participants), although 7 subjects called it “indispensable”. The XMPP compatibility was most commonly called “dispensable” (13 participants), with only 8 subjects selecting “very useful” as their answer. Even less popular was the private chat feature, which was supposed to provide backward compatible standard IM functionality. 25 subjects rated it “less useful” or “dispensable”.

Another assumption from section 2.2.1 was clearly confirmed, however. Namely that a very quick first start and short learning phase would be crucial to user acceptance. The necessity to receive instructions before being able to use the program was a common complaint which came up in the qualitative feedback (mentioned by 7 participants).

### 6.2.5 Discussion

Results and observations support the hypothesis that session merging is a valid approach to lightweight initiation, providing an easy way to begin collaboration and combine work. The criterion of generally good usability has also been met, though some details such as the conflict-prone invitations and sorting of merged chat histories could still be improved.

A general limitation of this study, which has to be considered, was the high computer proficiency of participants. This has undoubtedly led to overly positive usability results and might also explain the unusual technology acceptance feedback, since avid users are likely more willing to switch to a new web browser and IM account. If the development of CoopFox is continued, it will have to be tested again on a more diverse user base.

## 6.3 Transition

This part of the user study addressed CoopFox's ability to support both tightly and loosely coupled work in a flexible way, providing seamless transitions between both modes (2.3.2). The measure used for this assessment is a direct comparison with two baseline systems: separate browsers as the ideal loose coupling support condition, and screen sharing as the ideal tight coupling support condition. The hypothesis to confirm was that CoopFox supports loose coupling similarly well to the loosely coupled condition and tight coupling similarly well to the tight coupling condition, combining the strengths of both systems.

### 6.3.1 Task

The longer first part of the study followed a repeated measures design, asking participants to complete three short collaborative web researches (each strictly limited to 5 minutes) using three different sets of software tools to support them in their collaborative work:

- Tightly coupled full screen sharing via TightVNC [Kaplinsky 2000], installed as a server on workstation 2 and as clients on workstations 1 and 3. All three participants were controlling the same mouse pointer simultaneously. To collect their results, the group further had access to a text editor (Microsoft Wordpad), which they also had to operate jointly. No restrictions were made on result formats.
- Loosely coupled work in separate browsers with text communication via IRC chat. Identical Nettalk IRC clients [Kruse 2004] were provided on all workstations, configured to connect to a dedicated IRC server. To collect results, each group member further had access to an independent local Wordpad instance, but the group was told that at least one member had to have all results at the end.
- CoopFox, using the inbuilt result list to collect results. In this condition, no other software tools outside the browser were available to the participants.

Subjects were additionally allowed to talk to each other in all three conditions, but were never able to see each other's screens (6.1.2). The first two conditions were chosen as a baseline, under the assumption that the tightly coupled condition would excel at supporting tightly coupled work (sensemaking) but encumber loosely coupled work (division of labour), while the loosely coupled condition would have the opposite effect. In order to encourage not just foraging, but also sensemaking and organizing phases in all three conditions, participants were always told to evaluate their results together and agree on a ranking for their result document, sorting content from most to least relevant.

Each of the three researches was dedicated to a specific sub-topic within the framing student health narrative: reducing muscle tensions, managing stress, and eating healthy. The research topics were always completed by the participants in this exact order, but the order of conditions was randomized between groups, following a balanced Latin square distribution, so that each condition order was assigned to two groups of participants.

In order to standardize conditions, participants did not conduct the researches on real web pages. Instead, they were given a mocked search engine result page for each research as their starting point, which always contained exactly four predetermined result pages of comparable length and quality. These pages were based on real search results for the topic, but mirrored onto a dedicated HTTP server, to ensure availability and consistent content. Irrelevant text and distracting advertisements were removed. If and how participants divided up the research between them was explicitly left open for the group to decide.

Each task was introduced by a worksheet page explaining the research topic and the tools to use. The mocked result page for the topic was provided as the first tab in the browser. This tab was locked, so that it could not be closed and clicked links were always opened in new tabs. As soon as all group members had understood these instructions, the participants were left alone in the room for exactly five minutes to complete the task, while the experimenter watched via video feed from a separate room. The remaining time was announced after two, three and four minutes, as well as ten seconds before expiration.

A special introduction was given for the CoopFox task, in the form of a tutorial video. The total video duration was 3:30 minutes, which were shown in four parts, each demonstrating a different set of functions (location awareness, chat, highlighting, and result collection). After each tutorial part, participants were allowed to try out the functionality they had just seen on their own computers. For this purpose, they were given a browser window in which CoopFox was already active with a joint CoopChat. Before beginning the actual research, they received a new empty joint CoopChat.

After completing each research task, participants were told to advance to the following worksheet page, containing the questionnaire, while the experimenter saved any created result documents (the session and log files in case of CoopFox) and then configured the workstation for the next condition. To minimize the time required for reconfiguration, all necessary application windows for all tasks were already created and minimized to the system tray before the participants arrived (5x Firefox, 2x Wordpad, 1x IRC). For this purpose, another extension (MinimizeToTray) was installed in Firefox [Maier 2009] and a global system tool (RBTray) installed on all computers to be able to do the same with Wordpad [Redko 1998]. A shortcut link for the TightVNC client configuration was added to the desktop, in order to quickly engage screen sharing between the stations. These methods combined allowed the experimenter to change condition setups within 10 to 20 seconds for each workstation, while participants were filling in the questionnaires.

### 6.3.2 Observations

All groups were able to complete the three research tasks without any major problems, always producing several useful results, but the different support conditions had a very noticeable impact on division of labour strategies, communication, and role behaviour. During the use of CoopFox, all browser interactions were captured by a custom logging module, showing that subjects visited an average of 3.64 web pages including revisits (min 1, max 11) and all but three used several parallel tabs for their research. 34 of 36 participants used the Direct-Quote function and all of them participated in creating and sorting the result list. The mean number of quotes posted was 4.67 (max 15).

As intended, all groups engaged in division of labour strategies during both the loose coupling condition in separate browsers as well as the CoopFox condition. Accordingly, many groups expressed discontent during the tightly coupled screens sharing condition, since there were little opportunities for parallelization. Out of the 12 groups, 7 decided to tile several windows across the screen (typically the text editor and the browser), so that different users were at least able to forage and review results in parallel. Two groups even opened three separate browser windows, so that web pages could be read in parallel.

The most obvious difference between the three conditions was the amount and quality of verbal communication which took place. The screen sharing condition induced by far the highest level of verbal interaction, though large parts of it were coordination overhead

to cope with the shared mouse pointer and applications. In addition, the ability to see exactly the same content often led to lengthy and detailed discussions about minor details, such as the formatting of the result document and value of individual results. At the other end of the spectrum was the CoopFox condition, during which there was very little communication (as was also remarked by several participants in qualitative feedback). After establishing agreement on division of labour, most groups entered a prolonged silent phase of divided research, until they were ready to discuss the result ranking.

Leadership and control sharing within the groups also took on very different forms in response to the different conditions. During the loose coupling condition, all groups decided to elect a secretary (usually one member volunteered) to compile all findings of the group in a single document. Feedback from the secretary to the foragers was usually arranged either by reading out result passages aloud or by posting them back to the chat. During the tight coupling condition, 5 of the 12 groups decided to elect a dedicated mouse operator, at least for extended periods of time during the research. The other groups used more or less formalized social protocols to take turns. One of the groups using parallel browser windows developed a very sophisticated system, by which group members had to request and release control over the mouse using strictly defined vocabulary.

The CoopFox condition stood out as far as that there seemed to be no strict roles within any group. In 8 of the 12 groups, one person emerged as the moderator of the result ranking process, responsible for most of the prioritizing clicks, but there was not a single person in any group who did not actively take part in result generation. Very commonly, participants added their own entries from the CoopChat to the results immediately during the foraging phase, as soon as they thought they had identified something important. Ranking always took place in a distinct organising phase, often as soon as the four minute mark had been announced by the experimenter. During this process, the function to increase result priorities was used more than 7 times as often as the decrease priority function, and only two groups chose to delete results again once they had been added.

### 6.3.3 Questionnaire

Since part one of the study followed a repeated measures design, it contained only a single questionnaire, addressing satisfaction with tightly and loosely coupled work as well as overall collaboration. Participants filled in this repeating page independently, each time they had completed one of the three research tasks. The page consisted of 17 statements with attached 5-point Likert scales to indicate agreement, arranged in a large matrix.

Statements were divided into three categories: support of tightly coupled *group* work (G), support of loosely coupled *solitary* work (S), and easy *transition* between modes (T). Items from different categories were interleaved and partially reverse coded on the page. The transition category was specifically aimed at the assumed strength of CoopFox, and was expected to achieve low satisfaction under both other conditions. Since the used statements had not been tested in other studies before, each category further contained a fallback statement, which literally expressed the core assumption to confirm: “Tight collaboration was supported well”, “Independent work was supported well”, and “Transitioning between independent and group work was supported well”. The questionnaire further concluded in two statements assessing *overall* satisfaction (O): “I am satisfied with the collected results” and “I was satisfied with the technical support”.

Since the comparison between CoopFox and the baseline conditions is rather artificial, participants were also asked for a more practical assessment directly, comparing CoopFox against their usual tools and methods for online research. For this purpose, a set of questions was included in the general questionnaire after part two of the study, which prompted the subjects to assign relative adjectives to CoopFox via 5-point SD scales (e.g. faster/slower, more/less complicated, more profound/superficial).

In addition, participants also had the chance to give further qualitative feedback at the end of study part one, on a page with six open questions, which prompted them very generally to describe both positive and negative experiences regarding each of the three support conditions. Only two lines of space were provided for each question, to encourage short and concise answers. An opportunity for longer feedback regarding CoopFox in particular was then given in the final questionnaire after completing both study parts.

### 6.3.4 Results

Pairwise t-test comparison of questionnaire answers between the tree support conditions shows that CoopFox does in fact support both loose and tight coupling similarly well to the respective baselines. The detailed analysis for all statements addressing solitary and group work satisfaction is shown in Table 2. As expected, the loosely coupled baseline achieved significantly better results for all solitary work items, while the tightly coupled condition achieved significantly better results for all group work items. One item was discarded, because there was no significant difference between any conditions.

The results for CoopFox regarding solitary work satisfaction (S) are all significantly better than those of the tight coupling baseline and not significantly different from the loose coupling baseline. Regarding group work satisfaction items (G), all CoopFox results are significantly better than those of the loose coupling baseline and all except one item are not significantly different from the tight coupling baseline. The one exception is presented by a statement addressing awareness of the overall work progress, on which the screen sharing condition still performed significantly better than both others.

The seamlessness of work mode transitions (T) was addressed in a third category of items, which asked participants how easy it was to manage solitary and group work in parallel, and how quickly they could leave and enter the collaborative state. These results are presented in Table 3. The tight coupling condition consistently received the lowest scores in all four valid items from this category. The loose coupling condition performed on par with CoopFox regarding quick transitions between solitary and group work. CoopFox surpassed both of the baseline conditions regarding easy coordination between the two work modes, and also regarding transition support.

Finally, the last two items on each questionnaire assessed the overall satisfaction of participants with their collected results as well as the tool support they had received. These results are shown in Table 4. CoopFox significantly outperformed the two baseline conditions regarding both aspects. The screen sharing conditions achieved a marginally better score than the separate browsers condition regarding result satisfaction, while the opposite is true for tool satisfaction. However both of these differences were not statistically significant. The advantage of CoopFox was particularly expressed regarding the tool satisfaction statement, where it was the only system to receive a positive mean score (4.08) on a 5-point Likert scale between strong dissatisfaction and strong satisfaction. The two other conditions both received average scores below 3.00.

| Ct. | # | Statement (*Reverse Coding) | Loose | CoopFox | Tight |
|---|---|---|---|---|---|
| S | 6 | **Individual work was supported well.** | M 4.19<br>SD 0.95<br>⇢ p 0.000 | M 4.25<br>SD 0.97<br>← p 1.000<br>→ p 0.000 | M 1.58<br>SD 1.00 |
| S | 3 | I often had to adapt my working speed to that of other group members.* | M 2.08<br>SD 1.08<br>⇢ p 0.000 | M 2.00<br>SD 1.10<br>← p 1.000<br>→ p 0.000 | M 3.83<br>SD 1.21 |
| S | 9 | The activity of other group members distracted me from my work.* | M 1.81<br>SD 0.95<br>⇢ p 0.000 | M 2.00<br>SD 1.01<br>← p 0.805<br>→ p 0.001 | M 3.25<br>SD 1.42 |
| S | 12 | I was able to pursue my own work style. | M 4.25<br>SD 0.91<br>⇢ p 0.000 | M 4.14<br>SD 1.10<br>← p 1.000<br>→ p 0.000 | M 1.61<br>SD 0.93 |
| G | 10 | **Tight collaboration was supported well.** | M 2.67<br>SD 1.15<br>⇢ p 0.001 | M 3.81<br>SD 0.86<br>← p 0.000<br>→ p 1.000 | M 3.64<br>SD 1.27 |
| G | 1 | I always knew what the other group members were doing. | M 2.94<br>SD 1.37<br>⇢ p 0.001 | M 3.83<br>SD 1.00<br>← p 0.003<br>→ p 1.000 | M 4.06<br>SD 1.15 |
| G | 4 | It was easy to agree on a shared result list. | M 3.25<br>SD 1.34<br>⇢ p 0.049 | M 3.83<br>SD 0.91<br>← p 0.035<br>→ p 1.000 | M 3.97<br>SD 1.00 |
| G | 7 | I was always aware of the overall progress of our collaborative work. | M 2.50<br>SD 1.16<br>⇢ p 0.000 | M 3.75<br>SD 1.18<br>← p 0.000<br>→ p 0.004 | M 4.50<br>SD 0.70 |
| G | 13 | Coordinating with the other group members was easy. | M 3.39<br>SD 1.18<br>⇢ p 1.000 | M 4.06<br>SD 0.89<br>← p 0.003<br>→ p 0.101 | M 3.61<br>SD 0.96 |

Table 2. Statistical analysis of questionnaire results from study part one, showing items aimed at solitary (S) and group work (G) satisfaction. P-values correspond to the significance of disparity between conditions, according to pairwise t-testing (Bonferroni-corrected). The left column p-value indicates disparity between the two baseline conditions. Coloured in green is the condition with the best result, or the two conditions in case of non-significant disparity. Item 15 was discarded, since no disparity existed between any conditions.

| Ct. | # | Statement | Loose | CoopFox | Tight |
|---|---|---|---|---|---|
| T | 14 | **Transitioning between individual and collaborative work was supported well.** | M 2.69<br>SD 1.01<br>⇢ p 0.000 | M 4.14<br>SD 0.90<br>← p 0.000<br>→ p 0.000 | M 1.69<br>SD 0.92 |
| T | 2 | It was easy to handle both my individual work as well as group coordination. | M 3.58<br>SD 1.08<br>⇢ p 0.000 | M 4.11<br>SD 0.89<br>← p 0.038<br>→ p 0.000 | M 2.36<br>SD 1.16 |
| T | 5 | If I was disrupted in my own work, I was quickly able to resume it afterwards. | M 4.22<br>SD 0.68<br>⇢ p 0.000 | M 4.19<br>SD 0.79<br>← p 1.000<br>→ p 0.000 | M 3.00<br>SD 1.22 |
| T | 11 | If I wanted to focus on a particular text part alone, I was quickly able to catch up with the group again afterwards. | M 3.66<br>SD 0.87<br>⇢ p 0.000 | M 3.97<br>SD 0.86<br>← p 0.443<br>→ p 0.000 | M 2.80<br>SD 1.16 |

Table 3. Statistical analysis of questionnaire results from study part one, showing items aimed at transition between solitary and group work (T) (also see Table 2). Coloured in green is the condition with the best result. Item 8 was discarded, since no disparity existed between any conditions.

| Ct. | # | Statement | Loose | CoopFox | Tight |
|---|---|---|---|---|---|
| O | 16 | I am satisfied with the collected results. | M 3.06<br>SD 1.09<br>⇢ p 0.625 | M 3.94<br>SD 0.75<br>← p 0.000<br>→ p 0.014 | M 3.36<br>SD 1.02 |
| O | 17 | I was satisfied with the technical support. | M 2.78<br>SD 1.17<br>⇢ p 0.163 | M 4.08<br>SD 0.73<br>← p 0.000<br>→ p 0.000 | M 2.31<br>SD 1.12 |

Table 4. Statistical analysis of questionnaire results from study part one, showing the two items aimed at overall satisfaction (O) (also see Table 2). Coloured in green is the condition with the best result.

The questions asking participants to compare CoopFox against their usual web research tools returned mixed results (histograms in Appendix B). Of the 36 participants, a slight majority of 21 described research with CoopFox as faster (CF_1_1), 23 found it to be more organized (CF_1_3), and 22 indicated that gathered results were easier to comprehend in retrospect (CF_1_5). On the other hand, 18 participants stated that the use of CoopFox was more complicated (CF_1_2), opposed to only 12 who found it less complicated than their usual research methods. The question whether using CoopFox for web researches resulted in a more profound or more superficial style of work returned very scattered results (CF_1_4) as well as 18 undecided votes.

Finally, results from the qualitative feedback forms largely paralleled the analytic findings from the study. 12 of the 36 participants literally complimented CoopFox on supporting both individual as well as group work in a very flexible way. 9 subjects particularly mentioned good awareness support regarding what others were doing or had been doing. 5 participants made positive remarks about the shared history and result list.

The most common negative feedback concerned the amount of information in the CoopChat and result list, as well as its organisation. 17 participants indicated they felt overwhelmed by the amount of information, had trouble discerning important from unimportant content, or were unable to follow the work processes and ideas of others.

Another common complaint was the limited functionality of the result list. 5 subjects wished for better ways to structure results and suggested hierarchical grouping as well as full collaborative text editing as solutions. 4 subjects further remarked that the combination of the Direct-Quote feature and result list was good for quick and shallow researches based on references, but insufficient for complex and coherent investigations.

Qualitative feedback on the two baseline conditions closely matched the observations and analytic results. Research in separate browsers was considered to be more efficient and leave greater individual freedom, while coordination and referencing is encumbered. Screen sharing was praised for providing excellent awareness of others and allowing very fine-grained discussions about content, but was also described as inefficient and frustrating to group members who have to sit idly by, because division of labour is hardly possible.

### 6.3.5 Discussion

Test results clearly support the hypothesis that CoopFox supports loose coupling similarly well to the loosely coupled baseline condition, tight coupling similarly well to the tight coupling baseline condition, and further provides a satisfyingly flexible way to transition between the two work modes. In addition, the observation that research using CoopFox requires less verbal communication and explicit roles within the groups further provides evidence for the success of the awareness and result management features. Coordination overhead seems to be much lower in CoopFox than in the other conditions. However, potential for improvement was also evident, specifically regarding the information presentation in the CoopChat and the functional limitations of the result list.

As a potential limiting factor for the validity of the test results, it should be mentioned that the used questionnaires were custom to this study and did not undergo any other validation other than the test run with the pilot group. Therefore, it cannot be excluded that a different set of questions would have revealed more weaknesses of CoopFox.

Secondly, a positive user bias towards CoopFox can be assumed to some degree, since it was difficult to conceal that this was the main focus of the study. Although the name of the extension did not appear explicitly until its introduction, it was the only condition preceded by a ten minute tutorial exercise and by far the most sophisticated tool.

Lastly, a five minute research session on pre-determined result pages under laboratory conditions is certainly a very synthetic test for the very practical problem of collaborative web research. The two other conditions, although inspired by actual popular solutions, were also intentionally designed as extreme as possible, to provide good baselines for the evaluation of CoopFox. It can therefore not be derived how practical CoopFox would be for a real-world application at this point, or how superior it would be to other alternatives. These questions could only be answered by an extended field study.

## 6.4 Integration

This subchapter presents additional findings from the user study, regarding the user acceptance of CoopFox's contextual collaboration features. Since no part of the study was specifically dedicated to these aspects and many details have already been part of the past subchapters, this is kept very brief and without a separate discussion section.

### 6.4.1 Observations

The Direct-Quote feature was very popular throughout the entire study and formed the primary method of result gathering (used 3.7 times as often as sending text messages). The opposite was true for the ability to post a link to the current web page as a whole, which was only used three times in total. Participants rather chose to meet on the same page by instructing others to click on their contact list location link. Also barely used was transient highlighting of page content, which only occurred in very few cases during the second part of the study, when participants were discussing unknown pages.

### 6.4.2 Questionnaire Results

Responses to the "three favourite features" question support the observed popularity of the Direct-Quote function, which was the most common entry with 22 mentions. Qualitative feedback (written and verbal) also shows that users were most impressed by this concept, and considered it a great effort reduction, compared to manual copying and pasting. The only negative remark from one participant was that failed highlights result in a local feedback message but successful highlights do not (transient ones in particular), leading to possible confusion about what other users are seeing on their screens.

## 6.5 Summary

To evaluate the design goals for lightweight initiation and seamless transitions, a user study was conducted, in which twelve groups of three participants each carried out brief collaborative web research tasks. Besides CoopFox, the subjects also used a very loosely coupled baseline condition (separate browsers with text chat communication), as well as a very tightly coupled baseline condition (full screen synchronisation with shared cursor).

Lightweight initiation was evaluated in terms of perceived usefulness and ease of use, by exposing participants to a session merging task and collecting feedback by multiple choice questionnaires. Results show that all users were generally able to understand the concept, and a clear majority (28/36) deemed it valuable for their work. Responses on usability and learnability show that CoopFox is generally easy to learn and operate.

Seamless transitions were evaluated by comparing user satisfaction between the three different research conditions, based on questions regarding solitary work, group work, and transitions between the two modes. Results show that CoopFox generally performs as well as the loose coupling baseline regarding solitary work, as well as the tight coupling baseline regarding group work, and outperforms both in in the transition category.

Limitations of the study included a very computer-skilled sample of participants, the use of non-standard questionnaires, as well as the artificial task setting. These factors combined do not currently permit a judgement about how well CoopFox would perform in a real-world collaborative web research scenario with average-skilled users.

# 7 Conclusions and Future Work

I have presented CoopFox, a Firefox extension to support synchronous collaborative web browsing, optimized for spontaneous and dynamic interaction. My approach is based on three key concepts: lightweight session initiation, seamless transitions between solitary and group work, and contextual integration of communication with web content. I have introduced novel interaction concepts for each of these three aspects. I have further conducted a user study under laboratory conditions to evaluate my solutions.

*Lightweight initiation* is achieved through *session merging*, which allows users to combine their individual work states with one or more of their contacts at any time. It was designed to combine the easy person-to-person session initiation of instant messengers with the convenient content sharing of room-like workspaces. Feedback from the study shows that a majority of users found this approach both easy to comprehend and useful to their every-day collaborative web activities. Potential for future improvement was found, however, regarding conflict resolution in case of competing invitations, as well as better sorting and filtering options to allow better recognition of newly merged content. The idea of a separate 1-on-1 chat for establishing first contact was ignored and judged as unnecessary by the vast majority of users, suggesting that this solution should either be abandoned or rebuilt in a better integrated way, allowing for a direct transition between the first private conversation and the cooperative work session. This would also bring it even closer to the idea of lightweight initiation as by Gutwin and Greenberg (2.2.2).

*Seamless transitions* between solitary and group work are achieved by augmenting the existing paradigms of *tabbed browsing* and *instant messaging* with location awareness. Location-aware tabs allow users to collaborate tightly in one tab, simply by visiting the same web page as their peers, while maintaining other private pages open in separate tabs. The location-aware messenger allows users to jump quickly to any page their contacts are seeing, or to a page associated with any received message. In the user study, this concept proved to be as flexible as working in separate browsers, while also supporting tight collaboration as well as screen sharing. Potential for future improvement was found regarding the amount of information presented to the user. While this aspect had already been improved, compared to CoopFox's forerunner project SpreadVector, some users still felt overwhelmed and distracted by the contextual location hints.

*Contextual collaboration* is provided through the ability to reference entire pages or selected text excerpts via the chat with very few clicks, thus creating a quick alternative to manual copying and pasting of content between applications. In the user study, this was one of the most popular and widely used features, continuing its success ever since the earliest presentations of SpreadVector. On the other hand, it also became apparent that some users felt this approach encouraged a shallow form of collaboration, based solely around the exchange of quotes. To some degree, this may also have been caused by the test scenarios, which only revolved around fact research. Future studies will have to evaluate the highlighting solution in more diverse scenarios and possibly against other common forms of local interaction, such as real-time telepointers. The same applies to CoopFox's result collection feature, which currently only allows collaborative sorting, but no complex arrangement or editing. In general, it can be said that CoopFox has great potential to be used as a framework for future experimentation with many more sophisticated and possibly even web site specific collaboration support features.

The overarching goal of my thesis was to explore whether the established principles for real-time collaborative browsing could be adapted in a form that is more appealing to the current every-day internet users, who are often engaged in spontaneous and highly dynamic interactions, rather than planned long-term collaboration. CoopFox has shown that this is not only possible, but also has a high demand among web users. In fact, one of the most common questions which came up in the debriefing discussions during the user study was whether CoopFox was already available for download.

In particular, my implementation shows that many of the technical problems which impeded the usability of earlier projects have since been overcome. Browser extensions (especially for Firefox) now provide a very powerful development environment for rich client applications with sophisticated collaborative features and easy handling. XMPP has evolved into a powerful open communication standard, which could be used for a variety of collaborative tools, without the need for specialized server software. Future versions of CoopFox should expand even further on the use of such open standards, for instance by implementing the Jingle protocol for peer-to-peer data transfers [Ludwig et al. 2009] or the Web Real-Time Communication standard to capture live audio/video streams [Hazaël-Massieux et al. 2011]. This would open up possibilities for many more advanced features, such as integrated audio/video conferencing, on-demand screen-sharing, and real-time collaborative editing for both results and web forms.

Some technical challenges also remain for the current implementation, such as better compatibility with various web sites and XMPP services. These aspects were not evaluated in this thesis, since my work focussed largely on concepts. First tests indicate that my highlighting approach is still unreliable on pages which include DHTML elements such as content carousels, and that both Facebook and Google XMPP services do not forward all of my custom stanzas correctly. However, finding workarounds for these problems should only be a matter of engineering effort.

Finally, beyond the scope of collaborative browsing, I also hope that my work can be of inspiration to the development of real-time distributed groupware in general. Not just most of my related work but most of all my references stem from the 1990s or even 80s. Still, very few of the prototypes and proposed solutions for synchronous collaboration have ever translated into widespread practical applications. This “groupware barrier” has been pointed out by several authors from the field of CSCW [Schmidt 2011, pp. 356-357] [Gutwin et al. 2008, pp. 1411-1412] [Grudin & Palen 1995, p. 276] [Grudin 1988, p. 88]. It is often related to overly specialized solutions, critical mass issues and technical challenges. CoopFox demonstrates how added groupware value can be created by combining existing open technologies and simple generic interface functions. Given the growing importance of interactive online activities and the fast improving communication infrastructure, it seems that the groupware barrier could be broken now more than ever.

# References


Annett, J. Hierarchical Task Analysis. In Hollnagel, E., ed. Handbook of Cognitive Task Design. Lawrence Erlbaum, Mahwah, NJ, USA, 2003, pp. 17-35.

Arantes, F., de Moraes, C.R., Silva, S.H., de Mattos Fortes, R.P. and da Graça Campos Pimentel, M. Where and with Whom Do You Wanna Meet?: Session-Based Collaborative Work. In Proceedings of the 16th Brazilian Symposium on Multimedia and the Web – WebMedia 2010 (Oct. 5-8, Belo Horizonte, BR). ACM, New York, NY, USA, 2010, pp. 123-130.

Atterer, R., Schmidt, A. and Wnuk, M. A Proxy-Based Infrastructure for Web Application Sharing and Remote Collaboration on Web Pages. In Proceedings of the 11th IFIP TC 13 International Conference – INTERACT 2007 (Sept. 10-14, Rio de Janeiro, BR). Springer-Verlag, Berlin, DE, 2007, pp. 74-87.

Baker, K., Greenberg, S. and Gutwin, C. Heuristic Evaluation of Groupware Based on the Mechanics of Collaboration. In Little, M. and Nigay, L., eds. Engineering for Human-Computer Interaction. Springer-Verlag, Berlin, DE, 2001, pp. 123-139.

Baltes, B.B., Dickson, M.W., Sherman, M.P., Bauer, C.C. and LaGanke, J.S. Computer-Mediated Communication and Group Decision Making: A Meta-Analysis. Organizational Behavior and Human Decision Processes 87, 1 (Jan. 2002). pp. 156-179.

Bamberg, W. Add-on SDK. Mozilla Developer Network, Mozilla Foundation, http://developer.mozilla.org/en-US/Add-ons/SDK, 2013. (Accessed 28/12/2013).

Bardram, J., Bunde-Pedersen, J. and Soegaard, M. Support for Activity-Based Computing in a Personal Computing Operating System. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems – CHI 2006 (Apr. 22-27, Montreal, CA). ACM, New York, NY, USA, 2006, pp. 211-220.

Bernheim Brush, A.J., Bargeron, D., Gupta, A. and Cadiz, J.J. Robust Annotation Positioning in Digital Documents. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems – CHI 2001 (Mar. 31 - Apr. 5, Seattle, WA, USA). ACM, New York, NY, USA, 2001, pp. 285-292.

Bernstein, A. How Can Cooperative Work Tools Support Dynamic Group Process? Bridging the Specificity Frontier. In Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work – CSCW 2000 (Dec. 2-6, Philadelphia, PA, USA). ACM, New York, NY, USA, 2000, pp. 279-288.

Borghoff, U.M. and Schlichter, J.H. Computer-Supported Cooperative Work: Introduction to Distributed Applications. Springer-Verlag, Berlin, DE, 2000.

Brooke, J. SUS: A 'Quick and Dirty' Usability Scale. In Jordan, P.W., Thomas, B., Weerdmeester, B.A. and Mcclelland, I.L., eds. Usability Evaluation in Industry. Taylor & Francis, London, UK, 1996, pp. 189-194.

Cabri, G., Leonardi, L. and Zambonelli, F. Supporting Cooperative WWW Browsing: A Proxy-Based Approach. In Proceedings of the Seventh Euromicro Workshop on Parallel and Distributed Processing – PDP 1999 (Feb. 3-5 Funchail, PT). IEEE Computer Society, Los Alamitos, CA, USA, 1999, pp. 138-145.

Capra, R., Velasco-Martin, J. and Sams, B. Collaborative Information Seeking by the Numbers. In Proceedings of the 3rd International Workshop on Collaborative Information Retrieval – CIR 2011 (Oct. 28, Glasgow, UK). ACM, New York, NY, USA, 2011, pp. 7-10.

Card, S.K., Moran, T.P. and Newell, A. The Keystroke-Level Model for User Performance Time with Interactive Systems. Communications of the ACM 23, 7 (July 1980). pp. 396-410.

Carroll, J.M., Neale, D.C., Isenhour, P.L., Rosson, M.B. and McCrickard, D.S. Notification and Awareness: Synchronizing Task-Oriented Collaborative Activity. International Journal of Human-Computer Studies 58, 5 (May 2003). pp. 605-632.

Carroll, J.M. and Rosson, M.B. Paradox of the Active User. In Carroll, J.M., ed. Interfacing Thought: Cognitive Aspects of Human-Computer Interaction. MIT Press, Cambridge, MA, USA, 1987, pp. 80-111.

Cheng, L.-T., Patterson, J., Rohall, S.L., Hupfer, S. and Ross, S. Weaving a Social Fabric into Existing Software. In Proceedings of the 4th International Conference on Aspect-Oriented Software Development – AOSD 2005 (Mar. 14-18, Chicago, IL, USA). ACM, New York, NY, USA, 2005, pp. 147-158.

Chong, N.S.T. and Sakauchi, M. E-CoBROWSE: co-Navigating the Web with Chat-Pointers and Add-Ins – Problems and Promises. In Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems – PDCS 2000 (Nov. 6-9, Las Vegas, NV, USA). ACTA Press, Anaheim, CA, USA, 2000, pp. 803-808.

Clark, J. and DeRose, S. XML Path Language (XPath) Version 1.0. World Wide Web Consortium (W3C), http://www.w3.org/TR/xpath/, 1999. (Accessed 28/12/2013).

Cockburn, A. and Greenberg, S. Making Contact: Getting the Group Communicating with Groupware. In Proceedings of the Conference on Organizational Computing Systems – COOCS 1993 (Nov. 1-4, Milpitas, CA, USA). ACM, New York, NY, USA, 1993, pp. 31-41.

Dangoor, K. CommonJS Modules. The CommonJS Project, http://www.commonjs.org/specs/modules/1.0/, 2009. (Accessed 28/12/2013).

Dasgupta, S., Granger, M. and McGarry, N. User Acceptance of E-Collaboration Technology: An Extension of the Technology Acceptance Model. Group Decision and Negotiation 11, 2 (Mar. 2002). pp. 87-100.

Davis, F.D. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS quarterly 13, 3 (Sept. 1989). pp. 319-340.

Denning, P.J. and Yaholkovsky, P. Getting to "We". Communications of the ACM 51, 4 (Apr. 2008). pp. 19-24.

der Kinderen, G. Openfire Server. Igniterealtime, Jive Software, http://www.igniterealtime.org/projects/openfire/index.jsp, 2007. (Accessed 28/12/2013).

Dewan, P. and Choudhary, R. Coupling the User Interfaces of a Multiuser Program. ACM Transactions on Computer-Human Interaction 2, 1 (Mar. 1995). pp. 1-39.

Díaz, O. Understanding Web Augmentation. In Grossniklaus, M. and Wimmer, M., eds. Current Trends in Web Engineering. Springer-Verlag, Berlin, DE, 2012, pp. 79-80.

Dourish, P. and Bellotti, V. Awareness and Coordination in Shared Workspaces. In Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work – CSCW 1992 (Oct. 31 - Nov. 4, Toronto, ON, CA). ACM, New York, NY, USA, 1992, pp. 107-114.

Edwards, W.K. Session Management for Collaborative Applications. In Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work – CSCW 1994 (Oct. 22-26, Chapel Hill, NC, USA). ACM, New York, NY, USA, 1994, pp. 323-330.

Ellis, C.A. and Gibbs, S.J. Concurrency Control in Groupware Systems. In Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data – SIGMOD 1989 (May 31 - June 2, Portland, OR, USA). ACM, New York, NY, USA, pp. 399-407.

Ellis, C.A., Gibbs, S.J. and Rein, G. Groupware: Some Issues and Experiences. Communications of the ACM 34, 1 (Jan. 1991). pp. 39-58.

Endsley, M.R. Design and Evaluation for Situation Awareness Enhancement. In Proceedings of the Human Factors and Ergonomics Society 32nd Annual Meeting – HFES 1988 (Oct. 24-28, Anaheim, CA, USA). SAGE Publications, Thousand Oaks, CA, USA, 1988, pp. 97-101.

Evans, B.M. and Chi, E.H. Towards a Model of Understanding Social Search. In Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work – CSCW 2008 (Nov. 8-12, San Diego, CA, USA). ACM, New York, NY, USA, 2008, pp. 485-494.

Facebook Chat API. Facebook Developers, Facebook Inc., http://developers.facebook.com/docs/chat/, 2010. (Accessed 28/12/2013).

Forsyth, D.R. Group Dynamics. Cengage Learning, Belmont, CA, USA, 2009.

Gaver, W.W. Sound Support for Collaboration. In Proceedings of the Second European Conference on Computer-Supported Cooperative Work – ECSCW 1991 (Sept. 24-27, Amsterdam, NL). Kluwer Academic Publishers, Dordrecht, NL, 1991, pp. 293-308.

Gianoutsos, S. and Grundy, J. Collaborative Work with the World Wide Web: Adding CSCW Support to a Web Browser. In Proceedings of the 1996 Australian Symposium on Computer Supported Cooperative Work – OzCSCW 1996 (Aug. 30, Brisbane, AU). IEEE Computer Society, Los Alamitos, CA, USA, 1996, pp. 14-21.

Golovchinsky, G., Qvarfordt, P. and Pickens, J. Collaborative Information Seeking. Computer 42, 3 (Mar. 2009). pp. 47-51.

Google Talk Developer Documentation. Google Developers, Google Inc., http://developers.google.com/talk/talk_developers_home, 2012. (Accessed 28/12/2013).

Greenberg, S. and Roseman, M. GroupWeb: A WWW Browser as Real Time Groupware. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems – CHI 1996 (Apr. 13-18, Vancouver, BC, CA). ACM, New York, NY, USA, 1996, pp. 271-272.

Greenberg, S. and Roseman, M. Using a Room Metaphor to Ease Transitions in Groupware. In Ackerman, M.S., Pipek, V. and Wulf, V., eds. Sharing Expertise: Beyond Knowledge Management. MIT Press, Cambridge, MA, USA, 2003, pp. 203-256.

Gross, T. CSCW3: Transparenz- und Kooperationsunterstützung für das WWW. In Groupware und organisatorische Innovation - Tagungsband der Deutschen Fachtagung zu Computer Supported Cooperative Work – DCSCW 1998 (Sept. 28-30, Dortmund, DE). Vieweg+Teubner Verlag, Berlin, DE, 1998, pp. 37-50.

Gross, T., Stary, C. and Totter, A. User-Centered Awareness in Computer-Supported Cooperative Work-Systems: Structured Embedding of Findings from Social Sciences. International Journal of Human-Computer Interaction 18, 3 (June 2005). pp. 323-360.

Grudin, J. Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces. In Proceedings of the 1988 ACM Conference on Computer Supported Cooperative Work – CSCW 1988 (Sept. 26-28, Portland, OR, USA). ACM, New York, NY, USA, 1988, pp. 85-93.

Grudin, J. Groupware and Social Dynamics: Eight Challenges for Developers. Communications of the ACM 37, 1 (Jan. 1994). pp. 92-105.

Grudin, J. and Palen, L. Why Groupware Succeeds: Discretion or Mandate? In Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work – ECSCW 1995 (Sep. 11-15, Stockholm, SE). Kluwer Academic Publishers, Dordrecht, NL, 1995, pp. 263-278.

Gutwin, C. and Greenberg, S. A Descriptive Framework of Workspace Awareness for Real-Time Groupware. Computer Supported Cooperative Work (CSCW) 11, 3-4 (Sept. 2002). pp. 411-446.

Gutwin, C., Greenberg, S., Blum, R., Dyck, J., Tee, K. and McEwan, G. Supporting Informal Collaboration in Shared-Workspace Groupware. Journal of Universal Computer Science 14, 9 (May 1 2008). pp. 1411-1434.

Hawkey, K. and Inkpen, K. Web Browsing Today: The Impact of Changing Contexts on User Activity. In Extended Abstracts on Human Factors in Computing Systems – CHI 2005 (Apr. 2-4, Portland, OR, USA). ACM, New York, NY, USA, 2005, pp. 1443-1446.

Hazaël-Massieux, D., Alvestrand, H. and Håkansson, S. WebRTC. Web Real-Time Communications Working Group, World Wide Web Consortium, http://www.w3.org/2011/04/webrtc/, 2011. (Accessed 28/12/2013).

Hildebrand, J. and Saint-Andre, P. XEP-0033: Extended Stanza Addressing. XMPP Standards Foundation, http://xmpp.org/extensions/xep-0033.html, 2004. (Accessed 28/12/2013).

Huang, J., Lin, T. and White, R.W. No Search Result Left Behind: Branching Behavior with Browser Tabs. In Proceedings of the Fifth ACM International Conference on Web Search and Data Mining – WSDM 2012 (Feb. 8-12, Seattle, WA, USA). ACM, New York, NY, USA, 2012, pp. 203-212.

Hupfer, S., Cheng, L.-T., Ross, S. and Patterson, J. Introducing Collaboration into an Application Development Environment. In Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work – CSCW 2004 (Nov. 6-10, Chicago, IL, USA). ACM, New York, NY, USA, 2004, pp. 21-24.

Ishii, H. and Miyake, N. Toward an Open Shared Workspace: Computer and Video Fusion Approach of TeamWorkStation. Communications of the ACM 34, 12 (Dec. 1991). pp. 37-50.

Ivov, E. Jitsi. http://jitsi.org/, 2011. (Accessed 28/12/2013).

Jacobs, S., Gebhardt, M., Kethers, S. and Rzasa, W. Filling HTML Forms Simultaneously: CoWeb – Architecture and Functionality. Computer Networks and ISDN Systems 28, 7–11 (May 1996). pp. 1385-1395.

Johansen, R. GroupWare: Computer Support for Business Teams. The Free Press, New York, NY, USA, 1988.

jQuery. The jQuery Foundation, http://jquery.com/, 2006. (Accessed 28/12/2013).

jQueryUI. The jQuery Foundation, http://jqueryui.com/, 2008. (Accessed 28/12/2013).

Kaplinsky, C. TightVNC. The TightVNC Project, GlavSoft LLC, http://www.tightvnc.com/, 2000. (Accessed 28/12/2013).

Kellar, M., Watters, C. and Shepherd, M. A Field Study Characterizing Web-Based Information-Seeking Tasks. Journal of the American Society for Information Science and Technology 58, 7 (May 2007). pp. 999-1018.

Kimbro, L. XPConnect. Mozilla Developer Network, Mozilla Foundation, http://developer.mozilla.org/en-US/docs/Mozilla/XPConnect, 2005. (Accessed 28/12/2013).

King, W.R. and He, J. A Meta-Analysis of the Technology Acceptance Model. Information & Management 43, 6 (Sept. 2006). pp. 740-755.

Kruse, N. Nettalk. http://www.ntalk.de/Nettalk/en/, 2004. (Accessed 28/12/2013).

Kuhlthau, C.C. Inside the Search Process: Information Seeking from the User's Perspective. Journal of the American Society for Information Science 42, 5 (June 1991). pp. 361-371.

Kuutti, K. The Concept of Activity as a Basic Unit of Analysis for CSCW Research. In Proceedings of the Second European Conference on Computer-Supported Cooperative Work – ECSCW 1991 (Sept. 24-27, Amsterdam, NL). Kluwer Academic Publishers, Dordrecht, NL, 1991, pp. 249-264.

Levene, M. An Introduction to Search Engines and Web Navigation. John Wiley & Sons, Hoboken, NJ, USA, 2011.

Lowet, D. and Goergen, D. Co-Browsing Dynamic Web Pages. In Proceedings of the 18th International Conference on World Wide Web – WWW 2009 (Apr. 20-24, Madrid, ES). ACM, New York, NY, USA, 2009, pp. 941-950.

Ludwig, S., Beda, J., Saint-Andre, P., McQueen, R., Egan, S. and Hildebrand, J. XEP-0166: Jingle. XMPP Standards Foundation, http://xmpp.org/extensions/xep-0166.html, 2009. (Accessed 28/12/2013).

Maier, N. MinimizeToTray Revived. http://addons.mozilla.org/de/firefox/addon/minimizetotray-revived/, 2009. (Accessed 28/12/2013).

Mandviwalla, M. and Olfman, L. What Do Groups Need? A Proposed Set of Generic Groupware Requirements. ACM Transactions on Computer-Human Interaction 1, 3 (Sept. 1994). pp. 245-268.

Manohar, N.R. and Prakash, A. Replay by Re-execution: A Paradigm for Asynchronous Collaboration via Record and Replay of Interactive Multimedia Sessions. SIGOIS Bulletin 15, 2 (Dec. 1994). pp. 32-34.

McVittie, S., Saint-Andre, P. and McQueen, R. XEP-0276: Presence Decloaking. XMPP Standards Foundation, http://xmpp.org/extensions/xep-0276.html, 2012. (Accessed 28/12/2013).

Mirra, M. xmpp4moz. https://github.com/bard/xmpp4moz, 2009. (Accessed 28/12/2013).

Morris, M.R. A Survey of Collaborative Web Search Practices. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems – CHI 2008 (Apr. 5-8, Florence, IT). ACM, New York, NY, USA, 2008, pp. 1657-1660.

Morris, M.R. Collaborative Search Revisited. In Proceedings of the 2013 Conference on Computer Supported Cooperative Work – CSCW 2013 (Feb. 23-27, San Antonio, TX, USA). ACM, New York, NY, USA, 2013, pp. 1181-1192.

Morris, M.R. and Horvitz, E. SearchTogether: An Interface for Collaborative Web Search. In Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology – UIST 2007 (Oct. 7-10, Newport, RI, USA). ACM, New York, NY, USA, 2007, pp. 3-12.

Muller, M.J., Geyer, W., Brownholtz, B., Wilcox, E. and Millen, D.R. One-Hundred Days in an Activity-Centric Collaboration Environment Based on Shared Objects. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems – CHI 2004 (Apr. 24-29, Vienna, AT). ACM, New York, NY, USA, 2004, pp. 375-382.

Nardi, B.A., Whittaker, S. and Bradner, E. Interaction and Outeraction: Instant Messaging in Action. In Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work – CSCW 2000 (Dec. 2-6, Philadelphia, PA, USA). ACM, New York, NY, USA, 2000, pp. 79-88.

NCSA. Group Annotations in NCSA Mosaic. National Center for Supercomputing Applications, University of Illinois, IL, USA, http://www.softlab.ntua.gr/facilities/documentation/www/Mosaic-Docs-2.4/group-annotations.html, 1993. (Accessed 28/12/2013).

Niederhausen, M., Pietschmann, S., Ruch, T. and Meißner, K. Web-Based Support by Thin-Client Co-browsing. In Badr, Y., Chbeir, R., Abraham, A. and Hassanien, A.-E., eds. Emergent Web Intelligence: Advanced Semantic Technologies. Springer-Verlag, London, UK, 2010, pp. 395-428.

Nielsen, J. Heuristic Evaluation. In Nielsen, J. and Mack, R.L., eds. Usability Inspection Methods. John Wiley & Sons, New York, NY, USA, 1994, pp. 25-62.

Olson, J.S. and Teasley, S. Groupware in the Wild: Lessons Learned from a Year of Virtual Collocation. In Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work – CSCW 1996 (Nov. 16-20, Boston, MA, USA). ACM, New York, NY, USA, 1996, pp. 419-427.

Paul, S.A. and Morris, M.R. CoSense: Enhancing Sensemaking for Collaborative Web Search. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems – CHI 2009 (Apr. 4-9, Boston, MA, USA). ACM, New York, NY, USA, 2009, pp. 1771-1780.

Pennington, D.C. The Social Psychology of Behaviour in Small Groups. Psychology Press, New York, NY, USA, 2002.

Redko, N. RBTray. http://rbtray.sourceforge.net/, 1998. (Accessed 28/12/2013).

Rodriguez Perez, J.A., Leelanupab, T. and Jose, J.M. CoFox: A Synchronous Collaborative Browser. In Hou, Y., Nie, J.-Y., Sun, L., Wang, B. and Zhang, P., eds. Information Retrieval Technology. Springer-Verlag, Berlin, DE, 2012, pp. 262-274.

Rodriguez Perez, J.A., Whiting, S. and Jose, J.M. CoFox: A Visual Collaborative Browser. In Proceedings of the 3rd International Workshop on Collaborative Information Retrieval – CIR 2011 (Oct. 24-28, Glasgow, UK). ACM, New York, NY, USA, 2011, pp. 29-32.

Röscheisen, M., Mogensen, C. and Winograd, T. Interaction Design for Shared World-Wide Web Annotations. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems – CHI 1995 (May 7-11, Denver, CO, USA). ACM, New York, NY, USA, 1995, pp. 328-329.

Rose, D.E. and Levinson, D. Understanding User Goals in Web Search. In Proceedings of the 13th International Conference on World Wide Web – WWW 2004 (May 17-22, New York, NY, USA). ACM, New York, NY, USA, 2004, pp. 13-19.

Saint-Andre, P. Extensible Messaging and Presence Protocol (XMPP): Core. Internet Engineering Task Force, http://xmpp.org/rfcs/rfc6120.html, 2011a. (Accessed 28/12/2013).

Saint-Andre, P. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. Internet Engineering Task Force, http://xmpp.org/rfcs/rfc6121.html, 2011b. (Accessed 28/12/2013).

Saint-Andre, P. XEP-0045: Multi-User Chat. XMPP Standards Foundation, http://xmpp.org/extensions/xep-0045.html, 2012a. (Accessed 28/12/2013).

Saint-Andre, P. XEP-0071: XHTML-IM. XMPP Standards Foundation, http://xmpp.org/extensions/xep-0071.html, 2012b. (Accessed 28/12/2013).

Sarin, S. and Greif, I. Computer-Based Real-Time Conferencing Systems. Computer 18, 10 (Oct. 1985). pp. 33-45.

Schmidt, K. Cooperative Work and Coordinative Practices. Springer-Verlag, London, UK, 2011.

Schümmer, T. and Lukosch, S. Patterns for Computer-Mediated Interaction. John Wiley & Sons, Chichester, UK, 2007.

Schuster, T. ECMAScript 6 Support in Mozilla. Mozilla Developer Network, Mozilla Foundation, http://developer.mozilla.org/en-US/docs/Web/JavaScript/ECMAScript_6_support_in_Mozilla, 2012. (Accessed 28/12/2013).

Scott, S., Grant, K. and Mandryk, R. System Guidelines for Co-located, Collaborative Work on a Tabletop Display. In Proceedings of the Eighth European Conference on Computer Supported Cooperative Work – ECSCW 2003 (Sept. 14-18, Helsinki, FI). Kluwer Academic Publishers, Dordrecht, NL, 2003, pp. 159-178.

Shen, H. and Sun, C. Highlighting: A Gesturing Communication Tool for Real-Time Collaborative Systems. In Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing – ICA3PP 2002 (Oct. 23-25, Beijing, CN). IEEE Computer Society, Los Alamitos, CA, USA, 2002, pp. 180-187.

Shepherd, E. Bootstrapped Extensions. Mozilla Developer Network, Mozilla Foundation, http://developer.mozilla.org/en-US/Add-ons/Bootstrapped_extensions, 2010a. (Accessed 28/12/2013).

Shepherd, E. Firefox 4 for Developers. Mozilla Developer Network, Mozilla Foundation, http://developer.mozilla.org/en-US/Firefox/Releases/4, 2010b. (Accessed 28/12/2013).

Shepherd, E. New in JavaScript 1.8.5. Mozilla Developer Network, Mozilla Foundation, http://developer.mozilla.org/en-US/docs/Web/JavaScript/New_in_JavaScript/1.8.5, 2010c. (Accessed 28/12/2013).

Soliman, R., Braun, R. and Simoff, S. The Essential Ingredients of Collaboration. In Proceedings of the 2005 International Symposium on Collaborative Technologies and Systems – CTS 2005 (May 15-20, Saint Louis, MO, USA). IEEE Computer Society, Los Alamitos, CA, USA, 2005, pp. 366-373.

Trevor, J., Koch, T. and Woetzel, G. MetaWeb: Bringing Synchronous Groupware to the World Wide Web. In Proceedings of the Fifth European Conference on Computer Supported Cooperative Work – ECSCW 1997 (Sept. 7-11, Lancaster, UK). Kluwer Academic Publishers, Dordrecht, NL, 1997, pp. 65-80.

Twidale, M.B., Nichols, D.M. and Paice, C.D. Browsing is a Collaborative Process. Information Processing & Management 33, 6 (Nov. 1997). pp. 761-783.

Ullrich, D. and Diefenbach, S. INTUI. Exploring the Facets of Intuitive Interaction. In 10. fachübergreifende Konferenz für interaktive und kooperative Medien: Interaktive Kulturen – Mensch & Computer 2010 (Sept. 12-15, Duisburg, DE). Oldenbourg Wissenschaftsverlag, Munich, DE, 2010, pp. 251-260.

van der Veer, G.C., Lenting, B.F. and Bergevoet, B.A.J. GTA: Groupware Task Analysis - Modeling Complexity. Acta Psychologica 91, 3 (Apr. 1996). pp. 297-322.

Van Slyke, C., Ilie, V., Lou, H. and Stafford, T. Perceived Critical Mass and the Adoption of a Communication Technology. European Journal of Information Systems 16, 3 (July 2007). pp. 270-283.

Wasserman, A.I. The Design of 'Idiot-Proof' Interactive Programs. In Proceedings of the 1984 National Computer Conference – AFIPS 1973 (June 4-8, New York, NY, USA). AFIPS Press, Reston, VA, USA, 1973, pp. 34-38.

Wexelblat, A. and Maes, P. Footprints: History-Rich Tools for Information Foraging. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems – CHI 1999 (May 15-20, Pittsburgh, PA, USA). ACM, New York, NY, USA, 1999, pp. 270-277.

Whiteside, J. and Wixon, D. Improving Human-Computer Interaction – A Quest for Cognitive Science. In John, M.C., ed. Interfacing Thought: Cognitive Aspects of Human-Computer Interaction. MIT Press, Cambridge, MA, USA, 1987, pp. 353-365.

Wiltse, H. and Nichols, J. PlayByPlay: Collaborative Web Browsing for Desktop and Mobile Devices. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems – CHI 2009 (Apr. 4-9, Boston, MA, USA). ACM, New York, NY, USA, 2009, pp. 1781-1790.

Xia, S., Sun, D., Sun, C., Chen, D. and Shen, H. Leveraging Single-User Applications for Multi-User Collaboration: The Coword Approach. In Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work – CSCW 2004 (Nov. 6-10, Chicago, IL, USA). ACM, New York, NY, USA, 2004, pp. 162-171.

Yi, M.Y., Jackson, J.D., Park, J.S. and Probst, J.C. Understanding Information Technology Acceptance by Individual Professionals: Toward an Integrative View. Information & Management 43, 3 (Apr. 2006). pp. 350-363.

# Appendix A: User Study Worksheets

# Studie: Unterstützung von verteilter Web-Recherche

Vielen Dank für Ihre Teilnahme an dieser Studie im Rahmen meiner Masterarbeit!
Ziel der folgenden Untersuchung ist es, besser zu verstehen, wie Personen an getrennten Orten gemeinsam im Web recherchieren und welche technischen Hilfsmittel dabei die Arbeit erleichtern können.

Im ersten Teil (etwa 45 Minuten) werden Sie hierzu als Gruppe drei kurze Rechercheaufgaben erledigen, bei denen Sie in unterschiedlichem Umfang technische Unterstützung erhalten. Dabei kommt es nicht auf Qualität oder Geschwindigkeit an, sondern einzig auf Ihre Eindrücke zu Benutzerfreundlichkeit sowie der Arbeitsweise Ihrer Gruppe. Beides bewerten Sie nach jeder Aufgabe in einem kurzen Fragebogen.

Im zweiten Teil (etwa 15 Minuten) werden Sie eine der zuvor gezeigten Lösungen weiter kennenlernen und in einem ausführlichen Fragebogen genauer bewerten.

**Einverständniserklärung:**

- Mir ist bewusst, dass ich an einer wissenschaftlichen Studie teilnehme.
- Mir ist bewusst, dass meine Daten anonym ausgewertet werden.
- Mir ist bewusst, dass meine Angaben sowie auch aufgezeichnete Aussagen im Rahmen wissenschaftlicher Publikationen anonym verwendet werden können.
- Mir ist bewusst, dass meine Aktionen sowohl auf Video wie auch direkt im Programm aufgezeichnet werden. Diese Rohdaten werden jedoch nicht veröffentlicht und nach ihrer Auswertung nicht dauerhaft gespeichert.
- Mir ist bewusst, dass die Teilnahme an dieser Studie nicht verpflichtend ist und dass ich jederzeit aufhören kann.
- Mir ist bewusst, dass meine Bewertung keinen Einfluss auf die Note der betroffenen Masterarbeit hat und dass ehrliche Antworten für die Qualität dieser Studie wichtig sind.

Vollständiger Name (in Druckschrift): _____

**Datum, Unterschrift:** _____

# Studie: Unterstützung von verteilter Web-Recherche

# Teil 1

## Ihre Aufgabe:

Sie wollen gemeinsam einen **Gesundheits-Ratgeber für Studierende** schreiben. Dabei interessieren Sie insbesondere drei häufige Probleme: Verspannung, Stress und schlechte Ernährung.

Im Folgenden werden Sie hierzu **drei kurze Online-Recherchen** ausführen (eine zu jedem Problem, je 5 Minuten). Dabei stehen Ihnen jeweils unterschiedliche technische Mittel zur Verfügung. Bei jeder Recherche sind Ihnen die zu durchsuchenden Webseiten fest vorgegeben.

*Hinweis zur Kommunikation: Stellen Sie sich in allen drei Fällen vor, dass Sie an getrennten Orten arbeiten und außer den jeweiligen Hilfsmitteln nur eine Sprachverbindung (Telefon oder Skype) besitzen. Während keiner der folgenden Recherchen dürfen Sie daher die Bildschirme der anderen sehen.*

*Hinweis zum Zeitlimit: Halten Sie die 5-Minuten-Beschränkung strikt ein. Es kommt bei nicht auf die Qualität oder Anzahl Ihrer Ergebnisse an! Konzentrieren Sie sich stattdessen auf Ihre Eindrücke zur Benutzer-freundlichkeit der Hilfsmittel und auf die Abläufe innerhalb Ihrer Gruppe.*

*(Bitte nicht weiterblättern)*

## Suche 1: Verspannung

Sie werden diese Recherche in getrennten Webbrowsern ausführen.

Dabei dürfen Sie miteinander sprechen und den Text-Chat (Nettalk) benutzen, aber die Bildschirme der anderen Gruppenmitglieder nicht sehen.

Zudem steht jedem von Ihnen ein Texteditor (Wordpad) zur Verfügung, in dem Sie Ihre Ergebnisse sammeln können. Am Ende der Recherche muss mindestens einer von Ihnen über die vollständige Liste verfügen. Wie Sie sich dabei untereinander abstimmen, bleibt Ihnen überlassen.

**Ihr Ziel:** Sammeln Sie Aussagen oder Zitate dazu, wie man als Studierender Verspannungen vermeiden oder lindern kann. Notieren Sie diese in einer Form, die Ihnen später beim Schreiben des Ratgebers nützlich wäre. Sortieren Sie Ihre Ergebnisse, so dass Ihrer gemeinsamen Meinung nach wichtigere Funde weiter vorne stehen.

*(Bitte nicht weiterblättern)*

## Bewertung Suche 1

| | trifft nicht zu | trifft wenig zu | unentschieden | trifft eher zu | trifft völlig zu | |
|---|---|---|---|---|---|---|
| Ich war immer ausreichend darüber informiert, was die anderen Gruppenmitglieder gerade taten. | ☐ | ☐ | ☐ | ☐ | ☐ | L_1 G |
| Es fiel mir leicht, meiner eigenen Arbeit nachzugehen und diese dennoch mit der Gruppe abzustimmen. | ☐ | ☐ | ☐ | ☐ | ☐ | L_2 T |
| Ich musste meine Arbeitsgeschwindigkeit oft an die der Gruppe anpassen. | ☐ | ☐ | ☐ | ☐ | ☐ | L_3 S (R) |
| Es fiel mir leicht, mich mit den anderen Gruppenmitgliedern auf eine Ergebnisliste zu einigen. | ☐ | ☐ | ☐ | ☐ | ☐ | L_4 G |
| Wenn ich durch andere Gruppenmitglieder in meiner Arbeit unterbrochen wurde, konnte ich diese danach schnell wieder fortsetzen. | ☐ | ☐ | ☐ | ☐ | ☐ | L_5 T |
| Individuelles Arbeiten wurde gut unterstützt. | ☐ | ☐ | ☐ | ☐ | ☐ | L_6 S |
| Ich hatte stets einen guten Überblick über den Fortschritt unserer Arbeit. | ☐ | ☐ | ☐ | ☐ | ☐ | L_7 G |
| Wenn ich über eine spezielle Textstelle sprechen wollte, musste ich lange warten, bis die anderen Gruppenmitglieder bereit waren. | ☐ | ☐ | ☐ | ☐ | ☐ | L_8 T (R) |
| Die Aktivität der anderen Gruppenmitglieder hat mich von meiner eigenen Arbeit abgelenkt. | ☐ | ☐ | ☐ | ☐ | ☐ | L_9 S (R) |
| Enge Zusammenarbeit wurde gut unterstützt. | ☐ | ☐ | ☐ | ☐ | ☐ | L_10 G |
| Wenn ich mich alleine auf eine Textstelle konzentrieren wollte, konnte ich danach schnell wieder zur Gruppe aufschließen. | ☐ | ☐ | ☐ | ☐ | ☐ | L_11 T |
| Ich konnte meinen eigenen Arbeitsstil verfolgen. | ☐ | ☐ | ☐ | ☐ | ☐ | L_12 S |
| Ich konnte mich mit meinen anderen Gruppenmitgliedern gut koordinieren. | ☐ | ☐ | ☐ | ☐ | ☐ | L_13 G |
| Der Wechsel zwischen Individual- und Gruppenarbeit wurde gut unterstützt. | ☐ | ☐ | ☐ | ☐ | ☐ | L_14 T |
| Ich fühlte mich oft von schneller arbeiteten Gruppenmitgliedern unter Druck gesetzt. | ☐ | ☐ | ☐ | ☐ | ☐ | L_15 S (R) |
| Ich bin mit der gesammelten Liste von Ergebnissen zufrieden. | ☐ | ☐ | ☐ | ☐ | ☐ | L_16 O |
| Ich war allgemein zufrieden mit der technischen Unterstützung. | ☐ | ☐ | ☐ | ☐ | ☐ | L_17 O |

*(Bitte nicht weiterblättern)*

Auswahl: ☒ Korrektur: ■ ☒

## Suche 2: Stress

Sie werden diese Recherche mit Hilfe eines Fernsteuerungs-Programms ausführen, über das Sie alle stets denselben Bildschirminhalt sehen und somit einen Webbrowser gemeinsam nutzen können.

Dabei dürfen Sie miteinander sprechen und Ihren gemeinsamen Mauszeiger zur Kommunikation verwenden, aber die Bildschirme der anderen Gruppenmitglieder nicht direkt sehen.

Zudem steht Ihnen gemeinsam ein Texteditor (Wordpad) zur Verfügung, in dem Sie Ihre Ergebnisse sammeln können. Wie Sie sich dabei untereinander abstimmen, bleibt Ihnen überlassen.

**Ihr Ziel:** Sammeln Sie Aussagen oder Zitate dazu, wie man als Studierender Stress vermeiden oder bewältigen kann. Notieren Sie diese in einer Form, die Ihnen später beim Schreiben des Ratgebers nützlich wäre. Sortieren Sie Ihre Ergebnisse, so dass Ihrer gemeinsamen Meinung nach wichtigere Funde weiter vorne stehen.

*(Bitte nicht weiterblättern)*

## Bewertung Suche 2

| | trifft nicht zu | trifft wenig zu | unentschieden | trifft eher zu | trifft völlig zu | |
|---|---|---|---|---|---|---|
| Ich war immer ausreichend darüber informiert, was die anderen Gruppenmitglieder gerade taten. | ☐ | ☐ | ☐ | ☐ | ☐ | T_1 G |
| Es fiel mir leicht, meiner eigenen Arbeit nachzugehen und diese dennoch mit der Gruppe abzustimmen. | ☐ | ☐ | ☐ | ☐ | ☐ | T_2 T |
| Ich musste meine Arbeitsgeschwindigkeit oft an die der Gruppe anpassen. | ☐ | ☐ | ☐ | ☐ | ☐ | T_3 S (R) |
| Es fiel mir leicht, mich mit den anderen Gruppenmitgliedern auf eine Ergebnisliste zu einigen. | ☐ | ☐ | ☐ | ☐ | ☐ | T_4 G |
| Wenn ich durch andere Gruppenmitglieder in meiner Arbeit unterbrochen wurde, konnte ich diese danach schnell wieder fortsetzen. | ☐ | ☐ | ☐ | ☐ | ☐ | T_5 T |
| Individuelles Arbeiten wurde gut unterstützt. | ☐ | ☐ | ☐ | ☐ | ☐ | T_6 S |
| Ich hatte stets einen guten Überblick über den Fortschritt unserer Arbeit. | ☐ | ☐ | ☐ | ☐ | ☐ | T_7 G |
| Wenn ich über eine spezielle Textstelle sprechen wollte, musste ich lange warten, bis die anderen Gruppenmitglieder bereit waren. | ☐ | ☐ | ☐ | ☐ | ☐ | T_8 T (R) |
| Die Aktivität der anderen Gruppenmitglieder hat mich von meiner eigenen Arbeit abgelenkt. | ☐ | ☐ | ☐ | ☐ | ☐ | T_9 S (R) |
| Enge Zusammenarbeit wurde gut unterstützt. | ☐ | ☐ | ☐ | ☐ | ☐ | T_10 G |
| Wenn ich mich alleine auf eine Textstelle konzentrieren wollte, konnte ich danach schnell wieder zur Gruppe aufschließen. | ☐ | ☐ | ☐ | ☐ | ☐ | T_11 T |
| Ich konnte meinen eigenen Arbeitsstil verfolgen. | ☐ | ☐ | ☐ | ☐ | ☐ | T_12 S |
| Ich konnte mich mit meinen anderen Gruppenmitgliedern gut koordinieren. | ☐ | ☐ | ☐ | ☐ | ☐ | T_13 G |
| Der Wechsel zwischen Individual- und Gruppenarbeit wurde gut unterstützt. | ☐ | ☐ | ☐ | ☐ | ☐ | T_14 T |
| Ich fühlte mich oft von schneller arbeiteten Gruppenmitgliedern unter Druck gesetzt. | ☐ | ☐ | ☐ | ☐ | ☐ | T_15 S (R) |
| Ich bin mit der gesammelten Liste von Ergebnissen zufrieden. | ☐ | ☐ | ☐ | ☐ | ☐ | T_16 O |
| Ich war allgemein zufrieden mit der technischen Unterstützung. | ☐ | ☐ | ☐ | ☐ | ☐ | T_17 O |

*(Bitte nicht weiterblättern)*

Auswahl: ☒ Korrektur: ■ ☒

## Suche 3: Ernährung

Sie werden diese Recherche mit Hilfe einer Browser-Erweiterung ausführen, über die Sie direkt im Kontext einzelner Webseiten kommunizieren können.

Ein Video erklärt Ihnen die wichtigsten Bedienschritte.
An passenden Stellen haben Sie Gelegenheit, diese zu üben.

Danach weiter auf der Folgeseite…

## Suche 3: Ernährung (Fortsetzung)

Sie dürfen alle Funktionen von CoopFox verwenden und miteinander sprechen, aber die Bildschirme der anderen Gruppenmitglieder nicht sehen.

Nutzen Sie die integrierte Ergebnisliste von CoopFox, um Ihre Empfehlungen zu sammeln und zu sortieren. Wie Sie sich dabei untereinander abstimmen, bleibt Ihnen überlassen.

**Ihr Ziel:** Sammeln Sie Aussagen oder Zitate dazu, wie man sich als Studierender gesund und für geistige Arbeit vorteilhaft ernähren kann. Notieren Sie diese in einer Form, die Ihnen später beim Schreiben des Ratgebers nützlich wäre. Sortieren Sie Ihre Ergebnisse, so dass Ihrer gemeinsamen Meinung nach wichtigere Funde weiter vorne stehen.

*(Bitte nicht weiterblättern)*

Zur Erinnerung

**Nachricht schreiben:** Text unten rechts im Fenster eingeben → Enter / „Send“

**Link zur aktuellen Seite senden:** Klick auf „Send“ ohne zuvor Text einzugeben

**Auf Textstelle Zeigen:** Text markieren wenn andere Person auf selber Seite

**Direkt-Zitat aus Webseite:**
Text markieren → Rechtsklick auf Markierung → „Direct-Quote Selection“

**Auf Nachricht Zeigen:** Einfacher Klick auf Nachricht (außer auf Links)

**Nachricht kommentieren:** Doppelklick auf bestehende Nachricht im CoopChat

**Nachricht / Link / Zitat zu Ergebnissen hinzufügen:**
Rechtsklick auf Nachricht → „Add to Results“

## Bewertung Suche 3

| | trifft nicht zu | trifft wenig zu | unentschieden | trifft eher zu | trifft völlig zu | |
|---|---|---|---|---|---|---|
| Ich war immer ausreichend darüber informiert, was die anderen Gruppenmitglieder gerade taten. | ☐ | ☐ | ☐ | ☐ | ☐ | C_1 G |
| Es fiel mir leicht, meiner eigenen Arbeit nachzugehen und diese dennoch mit der Gruppe abzustimmen. | ☐ | ☐ | ☐ | ☐ | ☐ | C_2 T |
| Ich musste meine Arbeitsgeschwindigkeit oft an die der Gruppe anpassen. | ☐ | ☐ | ☐ | ☐ | ☐ | C_3 S (R) |
| Es fiel mir leicht, mich mit den anderen Gruppenmitgliedern auf eine Ergebnisliste zu einigen. | ☐ | ☐ | ☐ | ☐ | ☐ | C_4 G |
| Wenn ich durch andere Gruppenmitglieder in meiner Arbeit unterbrochen wurde, konnte ich diese danach schnell wieder fortsetzen. | ☐ | ☐ | ☐ | ☐ | ☐ | C_5 T |
| Individuelles Arbeiten wurde gut unterstützt. | ☐ | ☐ | ☐ | ☐ | ☐ | C_6 S |
| Ich hatte stets einen guten Überblick über den Fortschritt unserer Arbeit. | ☐ | ☐ | ☐ | ☐ | ☐ | C_7 G |
| Wenn ich über eine spezielle Textstelle sprechen wollte, musste ich lange warten, bis die anderen Gruppenmitglieder bereit waren. | ☐ | ☐ | ☐ | ☐ | ☐ | C_8 T (R) |
| Die Aktivität der anderen Gruppenmitglieder hat mich von meiner eigenen Arbeit abgelenkt. | ☐ | ☐ | ☐ | ☐ | ☐ | C_9 S (R) |
| Enge Zusammenarbeit wurde gut unterstützt. | ☐ | ☐ | ☐ | ☐ | ☐ | C_10 G |
| Wenn ich mich alleine auf eine Textstelle konzentrieren wollte, konnte ich danach schnell wieder zur Gruppe aufschließen. | ☐ | ☐ | ☐ | ☐ | ☐ | C_11 T |
| Ich konnte meinen eigenen Arbeitsstil verfolgen. | ☐ | ☐ | ☐ | ☐ | ☐ | C_12 S |
| Ich konnte mich mit meinen anderen Gruppenmitgliedern gut koordinieren. | ☐ | ☐ | ☐ | ☐ | ☐ | C_13 G |
| Der Wechsel zwischen Individual- und Gruppenarbeit wurde gut unterstützt. | ☐ | ☐ | ☐ | ☐ | ☐ | C_14 T |
| Ich fühlte mich oft von schneller arbeiteten Gruppenmitgliedern unter Druck gesetzt. | ☐ | ☐ | ☐ | ☐ | ☐ | C_15 S (R) |
| Ich bin mit der gesammelten Liste von Ergebnissen zufrieden. | ☐ | ☐ | ☐ | ☐ | ☐ | C_16 O |
| Ich war allgemein zufrieden mit der technischen Unterstützung. | ☐ | ☐ | ☐ | ☐ | ☐ | C_17 O |

*(Weiter auf Folgeseite)*

Auswahl: ☒ Korrektur: ■ ☒

## **Weitere Anmerkungen zu Teil 1** *(optional, max 3 Minuten)*

Vorteile der Suche in getrennten Browsern:

Nachteile der Suche in getrennten Browsern:

Vorteile der Suche im gemeinsamen Browser:

Nachteile der Suche im gemeinsamen Browser:

Vorteile der Suche mit CoopFox:

Nachteile der Suche mit CoopFox:

*(Bitte nicht weiterblättern)*

# Teil 2

Ein Video erklärt Ihnen, wie Sie in CoopFox
eine neue gemeinsame Recherche beginnen.

## Ihre Aufgabe:

Sie beginnen diesen Teil mit einem eigenen CoopChat. Dieser enthält bereits eine kurze Recherche zu einem der drei Ihnen bekannten Themen. Die Inhalte aller Gruppenmitglieder sind dabei verschieden.

Machen Sie sich zuerst mit Ihren eigenen Inhalten vertraut (max 1 Minute).

Danach weiter auf der Folgeseite…

## Ihre Aufgabe (Fortsetzung):

Vereinigen Sie Ihre CoopChats, indem ein Gruppenmitglied die anderen einlädt. Präsentieren Sie sich dann gegenseitig Ihre Inhalte. Besuchen Sie dabei die betroffenen Webseiten gemeinsam (max 3 Minuten).

*(Bitte nicht weiterblättern)*

Zur Erinnerung

**CoopChats verbinden**

Als einladender:

1. Doppelklick auf Eintrag in Kontaktliste
2. Frage beantworten mit „Invite to Join My CoopChat“

Als eingeladener:

1. Warten auf Einladungs-Benachrichtigung
2. Frage beantworten mit „Merge With My CoopChat“

## Bewertung Teil 2

| Das Zusammenführen meines CoopChats mit denen der andern... | | | | | | | |
|---|---|---|---|---|---|---|---|
| ...war schwer verständlich | ☐ | ☐ | ☐ | ☐ | ☐ | ...war leicht verständlich | Mrg_1_1 |
| ...dauerte mir zu lange | ☐ | ☐ | ☐ | ☐ | ☐ | ...ging mir zu schnell | Mrg_1_2 |
| ...war ein förmlicher Prozess | ☐ | ☐ | ☐ | ☐ | ☐ | ...war ein ungezwungener Prozess | Mrg_1_3 |
| ...erschien mir unlogisch | ☐ | ☐ | ☐ | ☐ | ☐ | ...erschien mir logisch | Mrg_1_4 |
| ...hat mich verunsichert | ☐ | ☐ | ☐ | ☐ | ☐ | ...hat mich nicht verunsichert | Mrg_1_5 |

| Ich konnte die Inhalte neuer Teilnehmer/-innen anhand des CoopChats... | | | | | | | |
|---|---|---|---|---|---|---|---|
| ...leicht nachvollziehen | ☐ | ☐ | ☐ | ☐ | ☐ | ...schwer nachvollziehen | Mrg_2 |

| Meine eigenen Inhalte konnte ich den anderen Gruppenmitgliedern... | | | | | | | |
|---|---|---|---|---|---|---|---|
| ...leicht verständlich machen | ☐ | ☐ | ☐ | ☐ | ☐ | ...schwer verständlich machen | Mrg_3 |

| Die Sortierung des zusammengeführten CoopChats empfand ich als... | | | | | | | |
|---|---|---|---|---|---|---|---|
| ...unlogisch | ☐ | ☐ | ☐ | ☐ | ☐ | ...logisch | Mrg_4 |

| Meine eigenen Inhalte waren im zusammengeführten CoopChat... | | | | | | | |
|---|---|---|---|---|---|---|---|
| ...schlecht nachvollziehbar | ☐ | ☐ | ☐ | ☐ | ☐ | ...gut nachvollziehbar | Mrg_5 |

*(Weiter auf Folgeseite)*

Auswahl: ☒ Korrektur: ■ ☒

## Bewertung CoopFox Allgemein

| Die Recherche mit CoopFox ist im Vergleich zu meiner üblichen Arbeitsweise... | | | | | | | |
|---|---|---|---|---|---|---|---|
| ...langsamer | ☐ | ☐ | ☐ | ☐ | ☐ | ...schneller | CF_1_1 |
| ...komplizierter | ☐ | ☐ | ☐ | ☐ | ☐ | ...unkomplizierter | CF_1_2 |
| ...unorganisierter | ☐ | ☐ | ☐ | ☐ | ☐ | ...organisierter | CF_1_3 |
| ...oberflächlicher | ☐ | ☐ | ☐ | ☐ | ☐ | ...tiefgreifender | CF_1_4 |
| ...im Nachhinein schlechter nachvollziehbar | ☐ | ☐ | ☐ | ☐ | ☐ | ...im Nachhinein besser nachvollziehbar | CF_1_5 |

| Beim Benutzen von CoopFox erreichte ich meine Ziele... | | | | | | | |
|---|---|---|---|---|---|---|---|
| ...nur mit Anstrengung | ☐ | ☐ | ☐ | ☐ | ☐ | ...mit Leichtigkeit | CF_2 |

| Im Nachhinein kann ich mich an die Abläufe der Bedienung... | | | | | | | |
|---|---|---|---|---|---|---|---|
| ...kaum noch erinnern | ☐ | ☐ | ☐ | ☐ | ☐ | ...noch gut erinnern | CF_3 |

| Ohne Anleitung hätte ich die gezeigten Funktionen... | | | | | | | |
|---|---|---|---|---|---|---|---|
| ...nicht verstanden | ☐ | ☐ | ☐ | ☐ | ☐ | ...auch intuitiv verstanden | CF_4 |

| In Zukunft könnte ich das System... | | | | | | | |
|---|---|---|---|---|---|---|---|
| ...nur mit Hilfe nutzen | ☐ | ☐ | ☐ | ☐ | ☐ | ...selbständig nutzen | CF_5 |

| Der Aufwand zum Erlernen von CoopFox im Vergleich zu dessen Nutzen ist... | | | | | | | |
|---|---|---|---|---|---|---|---|
| ...sehr hoch | ☐ | ☐ | ☐ | ☐ | ☐ | ...sehr gering | CF_6 |

| Die Darstellung von Information im CoopChat ist... | | | | | | | |
|---|---|---|---|---|---|---|---|
| ...unübersichtlich | ☐ | ☐ | ☐ | ☐ | ☐ | ...übersichtlich | CF_7 |

*(Weiter auf Folgeseite)*

Auswahl: ☒ Korrektur: ■ ☒

| Wenn ich CoopFox in Zukunft privat einsetzen würde, dann wäre die folgende Funktion / Eigenschaft für ***meine*** Zwecke... | ...entbehrlich | ...teils nützlich | ...nützlich | ...sehr nützlich | ...unentbehrlich | |
|---|---|---|---|---|---|---|
| Verwenden von CoopFox im selben Webbrowser, den ich auch zum alltäglichen Surfen einsetze | ☐ | ☐ | ☐ | ☐ | ☐ | CFF_1 |
| Verwenden meines bereits bestehenden Instant-Messenger-Kontos (z.B. Google Talk) | ☐ | ☐ | ☐ | ☐ | ☐ | CFF_2 |
| Gewöhnlicher Instant Messenger (Private Chat) zusätzlich zum CoopChat | ☐ | ☐ | ☐ | ☐ | ☐ | CFF_3 |
| Nachträgliches Zusammenführen zweier bereits begonnener CoopChats samt ihrer Inhalte | ☐ | ☐ | ☐ | ☐ | ☐ | CFF_4 |
| Gemeinsamer CoopChat mit mehr als einer weiteren Person (3er Gruppe und mehr) | ☐ | ☐ | ☐ | ☐ | ☐ | CFF_5 |

Auswahl: ☒ Korrektur: ■ ☒

Diese drei Funktionen von CoopFox sind für ***meine*** Zwecke am nützlichsten:

1.) _____

2.) _____

3.) _____

Dagegen fehlt mir / stört mich an CoopFox:

_____

_____

_____

_____

*(Weiter auf Folgeseite)*

## Ihre abschließende Meinung

| | trifft nicht zu | trifft wenig zu | unentschieden | trifft eher zu | trifft völlig zu | |
|---|---|---|---|---|---|---|
| Ich würde CoopFox gerne regelmäßig zur gemeinsamen Recherche einsetzen. | ☐ | ☐ | ☐ | ☐ | ☐ | CFO_1 |
| Ich würde CoopFox gerne regelmäßig alleine zur selbständigen Recherche einsetzen. | ☐ | ☐ | ☐ | ☐ | ☐ | CFO_2 |
| Ich würde andere Personen bitten, CoopFox zu nutzen, um mit ihnen zusammen zu arbeiten. | ☐ | ☐ | ☐ | ☐ | ☐ | CFO_3 |

Auswahl: ☒ Korrektur: ■ ☒

## Zu Ihrer Person

Alter: ☐☐ Geschlecht: ☐ männlich ☐ weiblich

Beschäftigung / Studiengang: _____

| Ihre allgemeine Erfahrung im Umgang mit Computern ist… | | | | | | | |
|---|---|---|---|---|---|---|---|
| …sehr hoch | ☐ | ☐ | ☐ | ☐ | ☐ | …sehr gering | Cmp_Lit |

| Wie häufig suchen Sie im Internet nach Informationen? | | | | | |
|---|---|---|---|---|---|
| ☐ mehrmals täglich | ☐ täglich | ☐ wöchentlich | ☐ seltener | ☐ nie | WebSrch |

| Wie häufig arbeiten Sie dabei mit anderen Personen zusammen? | | | | | |
|---|---|---|---|---|---|
| ☐ mehrmals täglich | ☐ täglich | ☐ wöchentlich | ☐ seltener | ☐ nie | CoSrch |

| Wie häufig kommunizieren Sie dabei in Echtzeit (Chat, Telefon, in Person…)? | | | | | |
|---|---|---|---|---|---|
| ☐ mehrmals täglich | ☐ täglich | ☐ wöchentlich | ☐ seltener | ☐ nie | RTCoSrch |

Auswahl: ☒ Korrektur: ■ ☒

# Studie: Unterstützung von verteilter Web-Recherche

**Hintergrund:** Beobachtungen an Internet-Nutzern und –Nutzerinnen zeigen bereits seit Jahren, dass viele von ihnen regelmäßig gemeinsam im Web surfen. Dies geschieht bisher relativ umständlich. Etwa müssen sich mehrere Personen vor einem Bildschirm versammeln, oder es werden unentwegt einzelne Links per Instant Messenger zwischen den Teilnehmern hin und her gesendet.

Schon seit Mitte der 1990er Jahre gibt es daher in der Forschung immer wieder Entwürfe für Mehrpersonen-Webbrowser. Diese waren jedoch in der Regel auf komplexe Aufgaben und sehr enge Kooperation unter den Teilnehmern ausgelegt, so dass sie für den alltäglichen Gebrauch nur schwer brauchbar waren.

**Ziel meiner Arbeit:** CoopFox wurde von mir im Rahmen meiner Masterarbeit entwickelt, um einen kooperativen Webbrowser zu schaffen, der sich auch für spontane und zwanglose Zusammenarbeit eignet. Besonders wichtig war mir dabei der schnelle Wechsel zwischen ungehinderter Einzelarbeit und enger Zusammenarbeit, genau dann wenn diese benötigt wird.

**Ihr Beitrag in dieser Studie:** Um die geplanten Vorzüge von CoopFox in der Praxis zu bestätigen, sollten Sie zum Vergleich zwei weitere häufige Arten des gemeinsamen Surfens testen: Auf der einen Seite die Arbeit in mehreren getrennten Browsern, welche Ihre getrennte Bewegungsfreiheit nicht behindert, aber dafür kaum Möglichkeiten zur effizienten Zusammenarbeit bietet. Auf der anderen Seite Screen-Sharing, das sehr enge Zusammenarbeit ermöglicht, aber dafür nicht erlaubt, sich ungehindert voneinander zu bewegen.

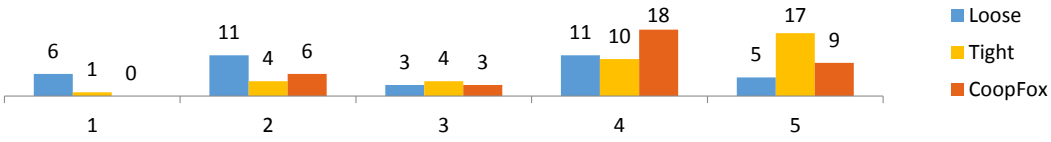
Das von Ihnen im zweiten Teil getestete Vereinigen mehrerer CoopChats ist ebenfalls ein wichtiger Teil meines Konzepts. Die meisten bisherigen Systeme zur unterstützten Gruppenarbeit erfordern, dass sich die Teilnehmer vorab zur Zusammenarbeit verabreden und diese gemeinsam beginnen. Dagegen erlaubt es Ihnen CoopFox, zuerst ungestört alleine zu arbeiten und Ihre Ergebnisse später in eine Gruppenarbeit zu überführen, sobald Sie dies für sinnvoll erachten.

**Hinweis:** Da noch zahlreiche weitere Studierende an meiner Studie teilnehmen, bitte ich Sie, die Inhalte dieser Seite nicht öffentlich zu diskutieren.

**Weiteren Fragen?** Sie erreichen Sie mich unter: masterarbeit@coopfox.net

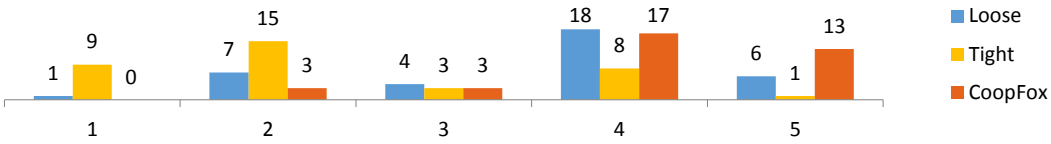
**Noch einmal vielen Dank für Ihre Teilnahme!**

# Appendix B: User Study Results



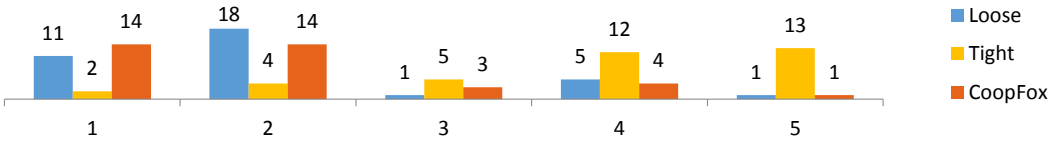

**L/T/C_1: I always knew what the other group members were doing (← no / yes →).**

Mean: 2.94 / 4.06 / 3.83
SD: 1.37 / 1.15 / 1.00
SE: 0.23 / 0.19 / 0.17



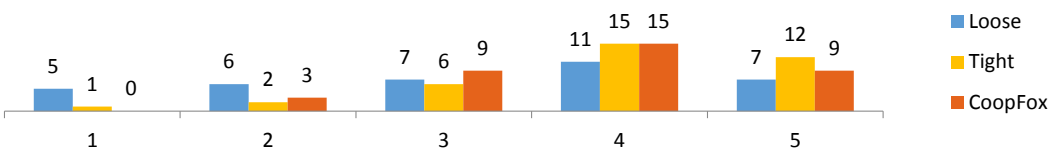

**L/T/C_2: It was easy to handle both my individual work as well as group coordination at the same time (← no / yes →).**

Mean: 3.58 / 2.36 / 4.11
SD: 1.08 / 1.17 / 0.89
SE: 0.18 / 0.20 / 0.15



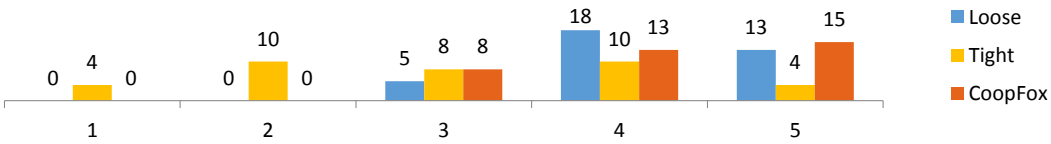

**L/T/C_3: I often had to adapt my working speed to that of other group members (← no / yes →).**

Mean: 2.08 / 3.83 / 2.00
SD: 1.08 / 1.21 / 1.10
SE: 0.18 / 0.20 / 0.18



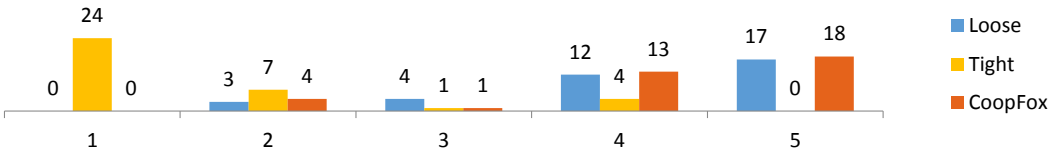

**L/T/C_4: It was easy to agree on a shared result list (← no / yes →).**

Mean: 3.25 / 3.97 / 3.83
SD: 1.34 / 1.00 / 0.91
SE: 0.22 / 0.17 / 0.15



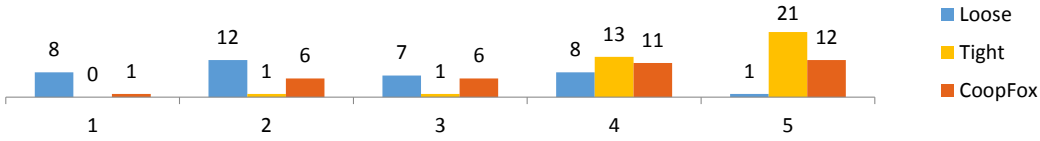

**L/T/C_5: If I was disrupted in my own work, I was quickly able to resume it afterwards (← no / yes →).**

Mean: 4.22 / 3.00 / 4.19
SD: 0.68 / 1.22 / 0.79
SE: 0.11 / 0.20 / 0.13



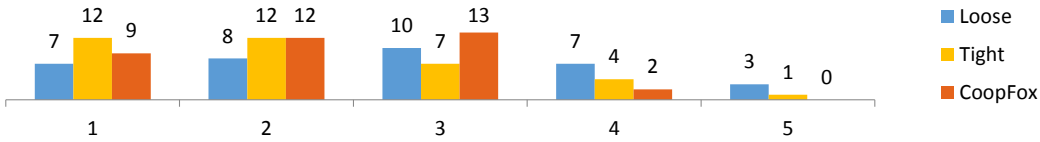

**L/T/C_6: Individual work was supported well (← no / yes →).**

Mean: 4.19 / 1.58 / 4.25
SD: 0.95 / 1.00 / 0.97
SE: 0.16 / 0.17 / 0.16



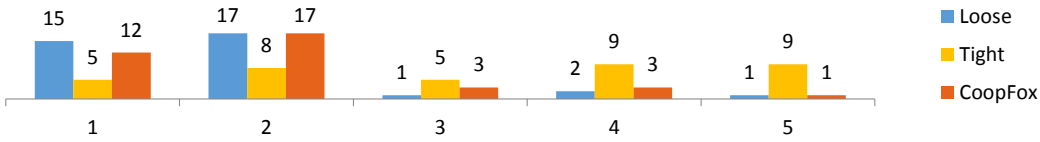

**L/T/C_7: I was always aware of the overall progress of our collaborative work (← no / yes →).**

Mean: 2.50 / 4.50 / 3.75
SD: 1.16 / 0.70 / 1.18
SE: 0.19 / 0.12 / 0.20



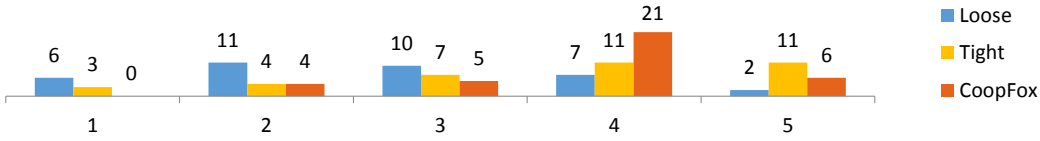

**L/T/C_8: If I wanted to discuss a particular text passage, I had do wait long for the other's attention (← no / yes →).**

Mean: 2.74 / 2.17 / 2.22
SD: 1.24 / 1.11 / 0.90
SE: 0.21 / 0.18 / 0.15



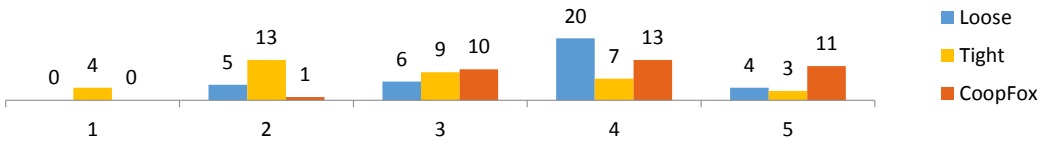

**L/T/C_9: The activity of other group members distracted me from my work (← no / yes →).**

Mean: 1.81 / 3.25 / 2.00
SD: 0.95 / 1.42 / 1.01
SE: 0.16 / 0.24 / 0.17



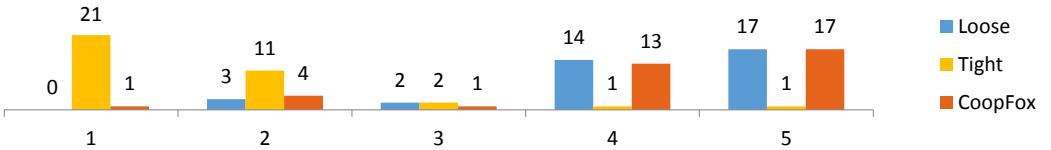

**L/T/C_10: Tight collaboration was supported well (← no / yes →) .**

Mean: 2.67 / 3.64 / 3.81
SD: 1.15 / 1.27 / 0.86
SE: 0.19 / 0.21 / 0.14



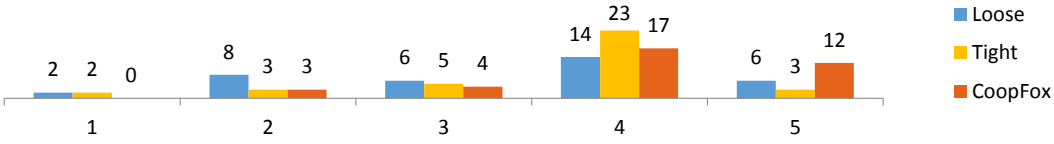

**L/T/C_11: If I wanted to focus on a particular text part alone, I was quickly able to catch up with the group again afterwards (← no / yes →).**

Mean: 3.66 / 2.78 / 3.97
SD: 0.87 / 1.15 / 0.86
SE: 0.15 / 0.19 / 0.14



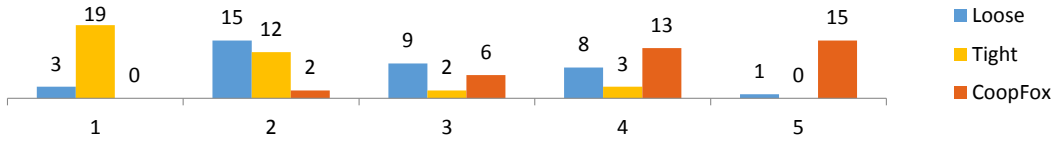

**L/T/C_12: I was able to pursue my own work style (← no / yes →).**

Mean: 4.25 / 1.61 / 4.14
SD: 0.91 / 0.93 / 1.10
SE: 0.15 / 0.16 / 0.18



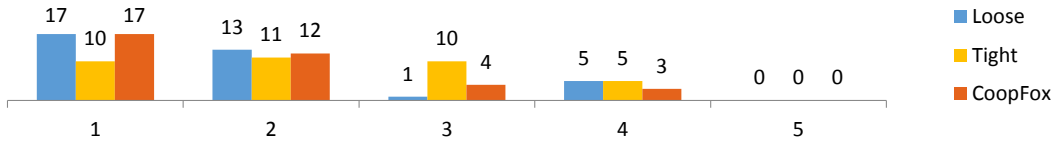

**L/T/C_13: Coordinating with the other group members was easy (← no / yes →).**

Mean: 3.39 / 3.61 / 4.06
SD: 1.18 / 0.96 / 0.89
SE: 0.20 / 0.16 / 0.15



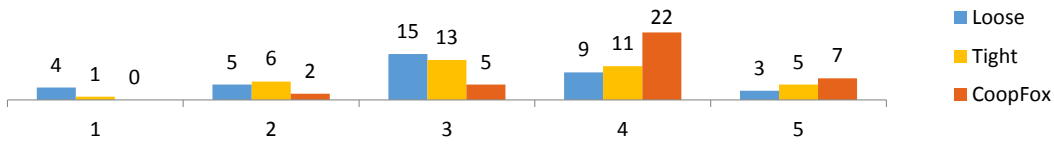

**L/T/C_14: Transitioning between individual and group work was supported well (← no / yes →).**

Mean: 2.69 / 1.69 / 4.14
SD: 1.01 / 0.92 / 0.90
SE: 0.17 / 0.15 / 0.15



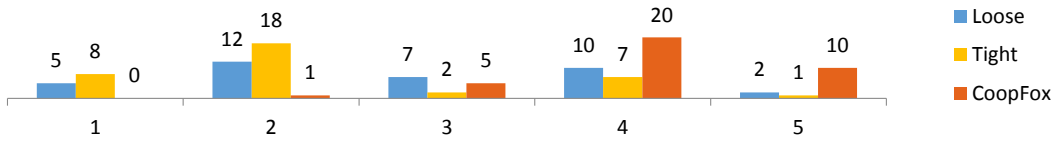

**L/T/C_15: I often felt pressured by faster working group members (← no / yes →).**

Mean: 1.83 / 2.28 / 1.81
SD: 1.03 / 1.03 / 0.95
SE: 0.17 / 0.17 / 0.16



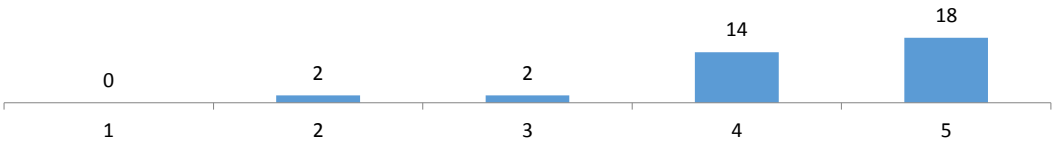

**L/T/C_16: I am satisfied with the collected results (← no / yes →).**

Mean: 3.06 / 3.36 / 3.94
SD: 1.09 / 1.02 / 0.75
SE: 0.18 / 0.17 / 0.13

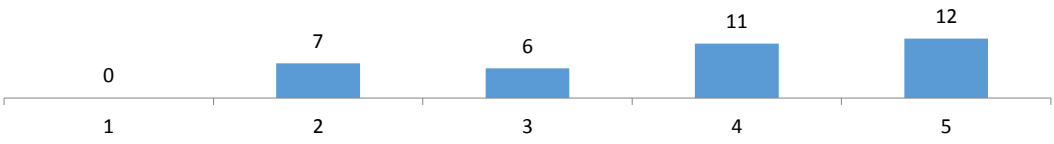



**L/T/C_17: I was satisfied with the technical support (← no / yes →).**

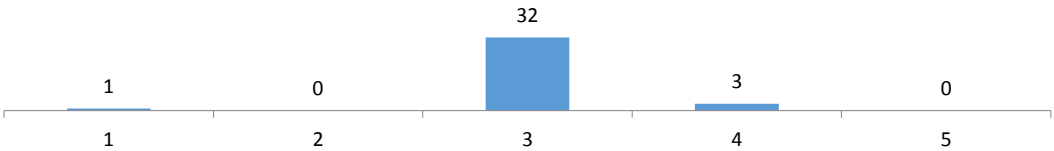
Mean: 2.78 / 2.31 / 4.08
SD: 1.17 / 1.12 / 0.73
SE: 0.20 / 0.19 / 0.12



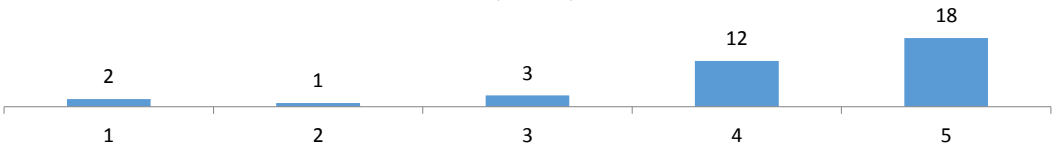

**Mrg_1_1: Session merging was (← hard / easy →) to comprehend.**
Mean 4.33, SD 0.83, SE 0.13



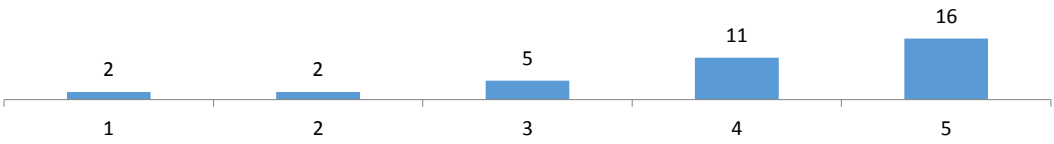

**Mrg_1_3: Session merging was a (← formal / informal →) process.**
Mean 3.78, SD 1.12, SE 0.19



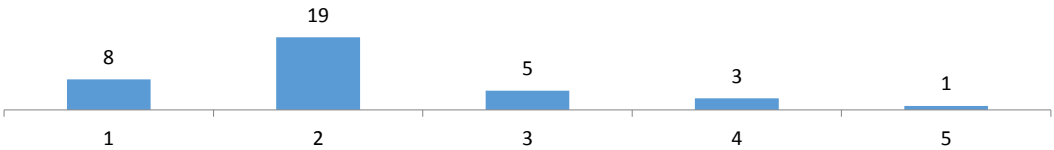

**Mrg_1_2: Session merging (← took too long / went too quick →).**
Mean 3.03, SD 0.45, SE 0.07



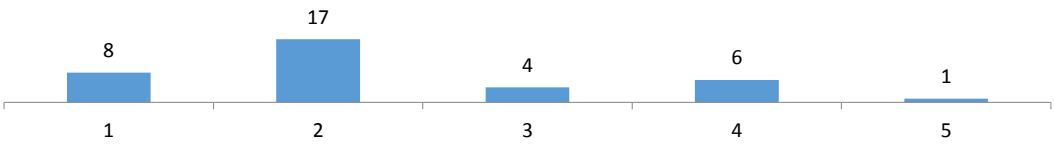

**Mrg_1_4: Session merging felt (← illogical / logical →).**
Mean 4.19, SD 1.09, SE 0.18



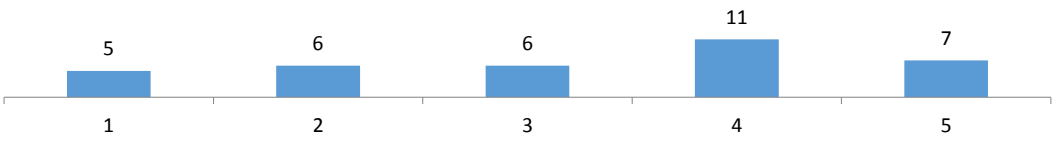

**Mrg_1_5: Session merging (← did / did not →) confuse me.**
Mean 4.03, SD 1.16, SE 0.19



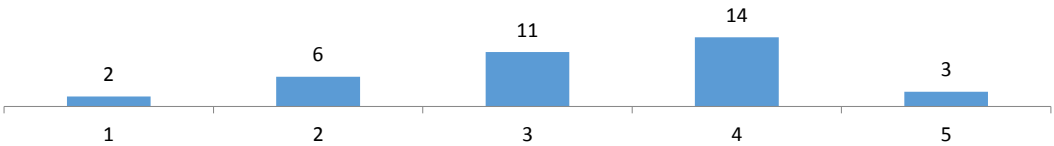

**Mrg_2: Foreign merged content was (← easy / hard →) to comprehend.**
Mean 2.17, SD 0.97, SE 0.16



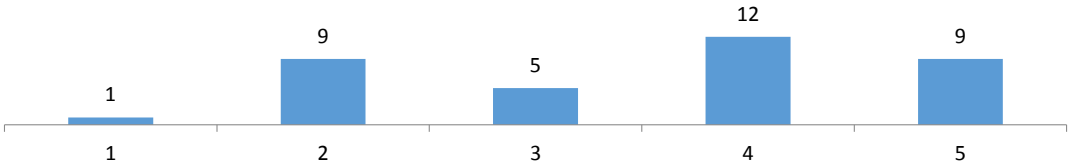

**Mrg_3: My merged content was (← easy / hard →) to explain to others.**
Mean 2.31, SD 1.09, SE 0.18



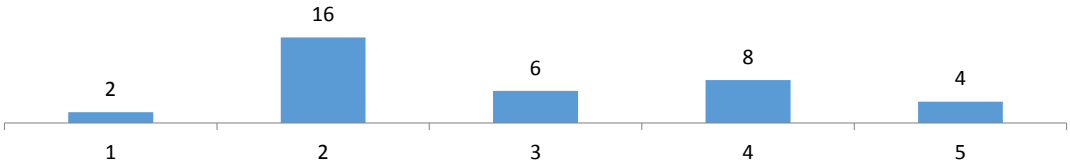

**Mrg_4: The sorting of the merged CoopChat felt (← illogical / logical →).**
Mean 3.26, SD 1.36, SE 0.23



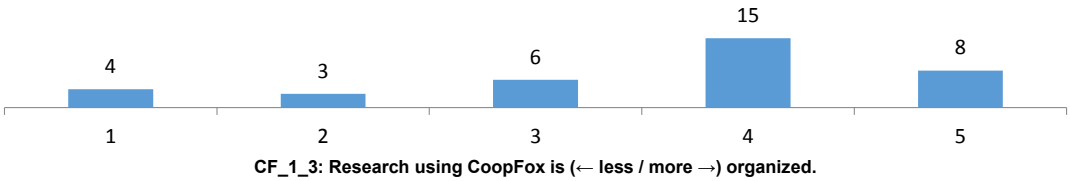

**Mrg_5: My content was (← hard / easy →) to comprehend after merging.**
Mean 3.28, SD 1.03, SE 0.17



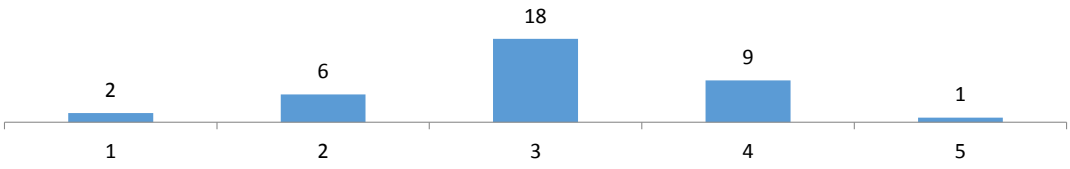

**CF_1_1: Research using CoopFox is (← slower / faster →) .**
Mean 3.53, SD 1.21, SE 0.20



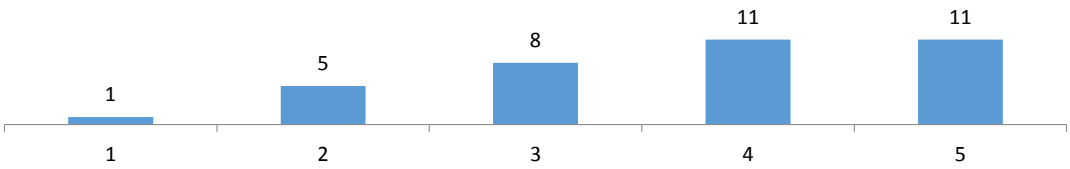

**CF_1_2: Research using CoopFox is (← more / less →) complicated.**
Mean 2.89, SD 1.17, SE 0.19



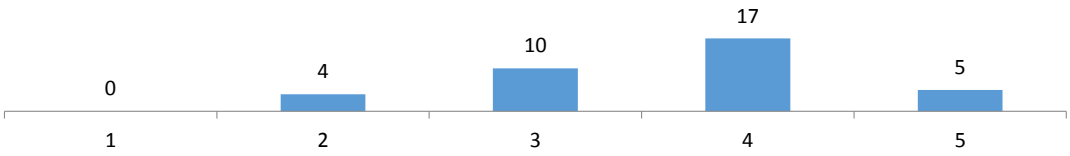

**CF_1_3: Research using CoopFox is (← less / more →) organized.**
Mean 3.56, SD 1.25, SE 0.21



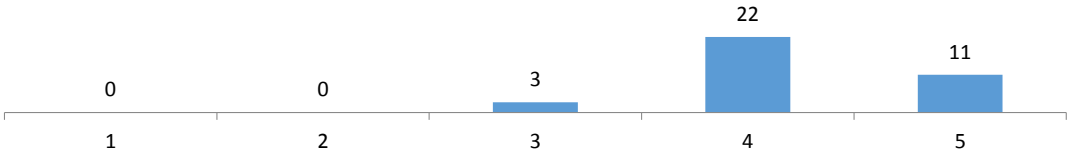

**CF_1_4: Research using CoopFox is more (← superficial / profound →).**
Mean 3.03, SD 0.88, SE 0.15



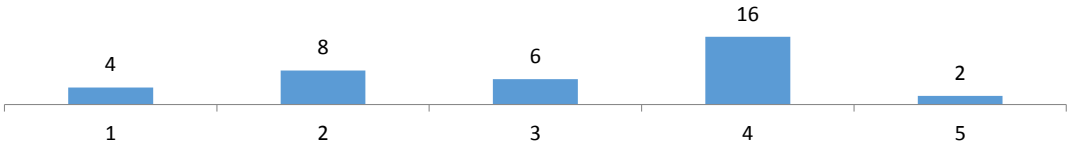

**CF_1_5: Results in CoopFox are (← less / more →) comprehensible.**
Mean 3.72, SD 1.14, SE 0.19



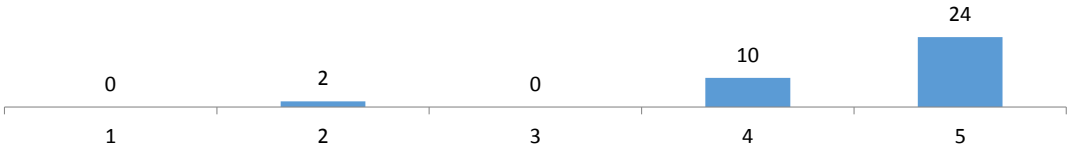

**CF_2: Reaching my goals with CooFox required (← lots of / little →) effort.**
Mean 3.64, SD 0.87, SE 0.14



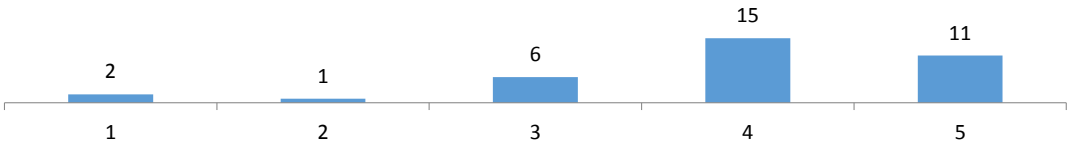

**CF_3: Recalling the operating steps in retrospect is (← difficult / easy →).**
Mean 4.22, SD 0.59, SE 0.10



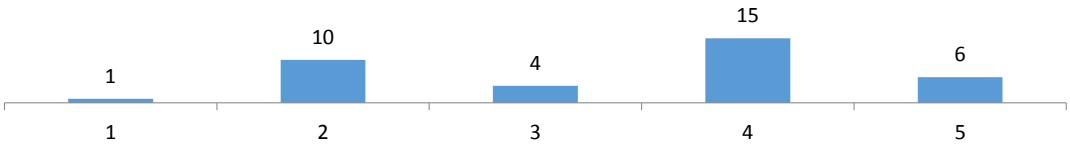

**CF_4: I would have understood CoopFox without instructions (← no / yes →).**
Mean 3.11, SD 1.17, SE 0.19



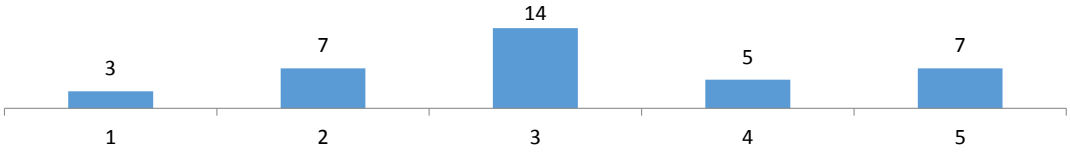

**CF_5: I could now use the sytem (← only with help / on my own →).**
Mean 4.56, SD 0.77, SE 0.13



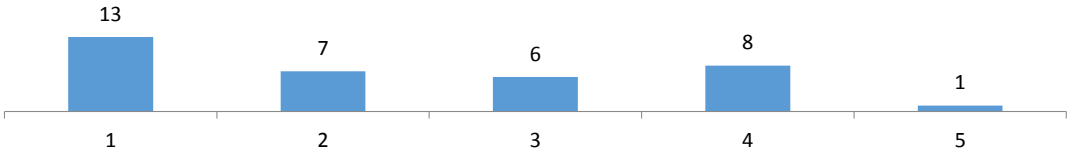

**CF_6: The ratio between learning effort and usefulness is (← high / low →).**
Mean 3.91, SD 1.07, SE 0.18



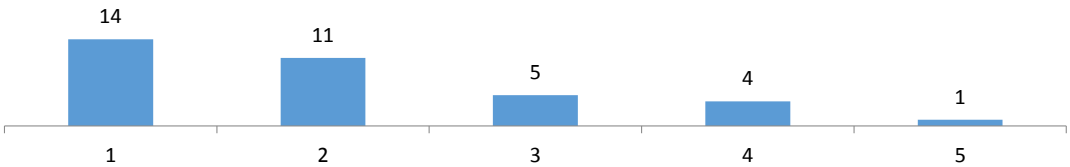

**CF_7: The information in the CoopChat (← is not / is →) well arranged.**
Mean 3.42, SD 1.16, SE 0.19



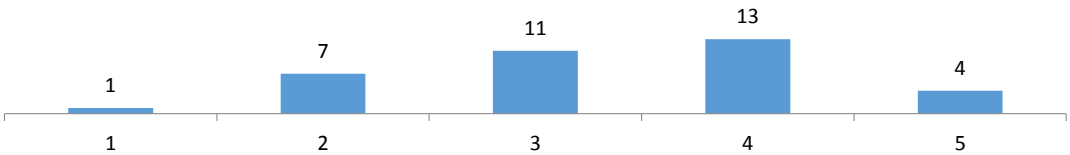

**CFF_1: Using CoopFox in my usual browser (← is not / is →) important.**
Mean 3.17, SD 1.21, SE 0.20



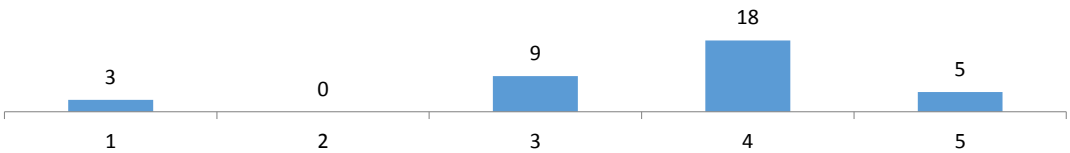

**CFF_2: Loggin in with my existing IM account (← is not / is →) important.**
Mean 2.34, SD 1.28, SE 0.21



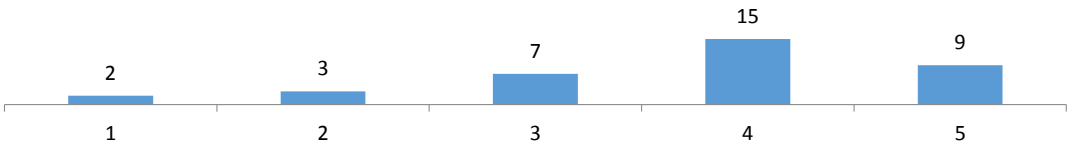

**CFF_3: The separate Private Chat function (← is not / is →) important.**
Mean 2.06, SD 1.14, SE 0.19



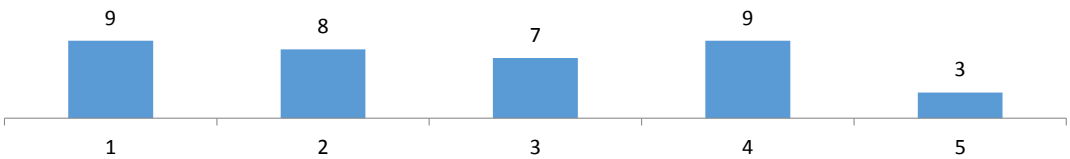

**CFF_4: The session merging function (← is not / is →) important.**
Mean 3.33, SD 1.01, SE 0.17



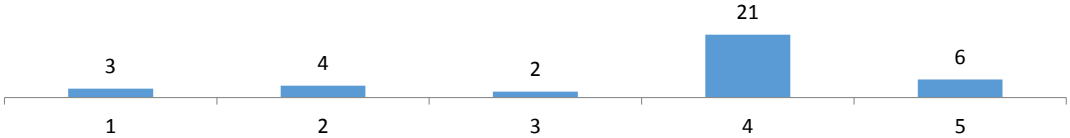

**CFF_5: Sessions with three or more people (← are not / are →) important.**
Mean 3.63, SD 1.03, SE 0.17




**CFO_1: I would like to use CoopFox for group research (← no / yes →) .**
Mean 3.72, SD 1.11, SE 0.17




**CFO_2: I would like to use CoopFox for solitary research (← no / yes →) .**
Mean 2.96, SD 1.33, SE 0.22




**CFO_3: I would ask other people to use CoopFox with me (← no / yes →) .**
Mean 3.64, SD 1.15, SE 0.19

# Appendix C: XMPP Code Samples

```
<?xml version="1.0" ?>
<stream:stream xmlns:stream="http://etherx.jabber.org/streams" from="coopfox.net"
xml:lang="en" version="1.0">

 …

<!-- Juliet logs in with her CoopFox client. -->
<presence from="juliet@coopfox.net/coopfox" to="romeo@coopfox.net/coopfox">
   <c xmlns="http://jabber.org/protocol/caps" node="http://coopfox.net"
   hash="sha-1" ver="dJrM+4YlwrbFgg+Cvf0HTraag7w="/>
</presence>

<!-- Romeo invites Juliet to join his session. -->
<message to="coopfox.net" from="romeo@coopfox.net/coopfox" type="chat" id="m1">
   <coopfox xmlns="http://coopfox.net/xmpp/namespace" timestamp="1388963118773">
      <participant jid="juliet@coopfox.net" action="join"/>
   </coopfox>
   <addresses xmlns="http://jabber.org/protocol/address">
      <address type="to" jid="juliet@coopfox.net/coopfox"/>
   </addresses>
   <thread>t1</thread>
</message>

<!-- (Implicit) Juliet does not reject. -->

<!-- Romeo polls Juliet's messages (fast-forward from his history version). -->
<iq to="juliet@coopfox.net/coopfox" from="romeo@coopfox.net/coopfox" type="get" id="i1">
   <query xmlns="http://coopfox.net/xmpp/namespace/sync" thread="t1" mode="fast-forward">
      <!-- Checksum of Romeo's message history content. -->
      <version>9da9a0b65ff55380f4d0408e4dd9dc8e</version>
   </query>
</iq>

<!-- Juliet has no messages, returns empty result. -->
<iq to="romeo@coopfox.net/coopfox" from="juliet@coopfox.net/coopfox" type="result" id="i1">
   <query xmlns="http://coopfox.net/xmpp/namespace/sync" thread="t1" mode="fast-forward">
      <version>40be2d0195dd41e7f0236d3c6bbbe728</version>
   </query>
</iq>

…

<!-- Juliet polls Romeo's messages (fast-forward from her history version). -->
<iq to="romeo@coopfox.net/coopfox" from="juliet@coopfox.net/coopfox" type="get" id="i2">
   <query xmlns="http://coopfox.net/xmpp/namespace/sync" thread="t1" mode="fast-forward">
      <version>40be2d0195dd41e7f0236d3c6bbbe728</version>
   </query>
</iq>

<!-- Romeo does not recognize Juliet's version, cannot send diff. -->
<iq to="juliet@coopfox.net/coopfox" from="romeo@coopfox.net/coopfox" type="result" id="i2">
   <query xmlns="http://coopfox.net/xmpp/namespace/sync" thread="t1" mode="fast-forward">
      <version>9da9a0b65ff55380f4d0408e4dd9dc8e</version>
   </query>
</iq>
```

```
<!-- Juliet polls Romeo's complete message history. -->
<iq to="romeo@coopfox.net/coopfox" from="juliet@coopfox.net/coopfox" type="get" id="i3">
  <query xmlns="http://coopfox.net/xmpp/namespace/sync" thread="t1" mode="complete"/>
</iq>

<!-- Romeo replies with all his stored messages. -->
<iq to="juliet@coopfox.net/coopfox" type="result" from="romeo@coopfox.net/coopfox" id="i3">
  <query xmlns="http://coopfox.net/xmpp/namespace/sync" thread="t1"
  mode="complete" timestamp="1388963121521">
    <version>9da9a0b65ff55380f4d0408e4dd9dc8e</version>
    <diff>
      <message to="coopfox.net" type="chat" from="romeo@coopfox.net/coopfox" id="m0">
        <body>O, I am fortune's fool!</body>
        <coopfox xmlns="http://coopfox.net/xmpp/namespace" timestamp="1388963098116">
          <location url="http://en.wikipedia.org/wiki/Romeo_and_Juliet"
          urlhash="81e5c5cb0c38556dd5163117dc0b2900"
          icon="http://bits.wikimedia.org/favicon/wikipedia.ico" title="Romeo
          and Juliet - Wikipedia, the free encyclopedia" source="page"/>
        </coopfox>
        <thread>t1</thread>
        <delay xmlns="urn:xmpp:delay" from="romeo@coopfox.net"
        stamp="2014-01-05T23:04:58.116Z">Received</delay>
      </message>
    </diff>
  </query>
</iq>

<!-- Juliet imports Romeo's messages. Both now have the same version.
Sync complete. -->

<!-- Juliet sends her current location. -->
<message to="coopfox.net" from="juliet@coopfox.net/coopfox" type="headline" id="m2">
  <coopfox xmlns="http://coopfox.net/xmpp/namespace" timestamp="1388963121984">
    <location url="http://en.wikipedia.org/wiki/Romeo_and_Juliet"
    urlhash="81e5c5cb0c38556dd5163117dc0b2900"
    icon="http://bits.wikimedia.org/favicon/wikipedia.ico" title="Romeo
    andJuliet - Wikipedia, the free encyclopedia" source="page"/>
  </coopfox>
  <addresses xmlns="http://jabber.org/protocol/address">
    <address type="to" jid="romeo@coopfox.net/coopfox"/>
  </addresses>
  <thread>t1</thread>
</message>

…

<!-- Romeo sends a transient highlight (text selection). -->
<message to="coopfox.net" from="romeo@coopfox.net/coopfox" type="headline" id="m4">
  <coopfox xmlns="http://coopfox.net/xmpp/namespace" timestamp="1388963122255">
    <highlight url="http://en.wikipedia.org/wiki/Romeo_and_Juliet">
      <text context="48e1a8ab" xpath="1d06ca87" precontext="8794c5e7"
      postcontext="b765372f" pretext="b15835f1" posttext="f7139b3a">O brawling
      love, O loving hate</text>
    </highlight>
  </coopfox>
  <addresses xmlns="http://jabber.org/protocol/address">
    <address type="to" jid="juliet@coopfox.net/coopfox"/>
  </addresses>
  <thread>t1</thread>
</message>
```

```
<!-- Romeo posts a link to his current page. -->
<message to="coopfox.net" from="romeo@coopfox.net/coopfox" type="chat" id="m5">
  <body>http://en.wikipedia.org/wiki/Romeo_and_Juliet</body>
  <active xmlns="http://jabber.org/protocol/chatstates"/>
  <addresses xmlns="http://jabber.org/protocol/address">
    <address type="to" jid="juliet@coopfox.net/coopfox"/>
  </addresses>
  <coopfox xmlns="http://coopfox.net/xmpp/namespace" timestamp="1388963130795">
    <location url="http://en.wikipedia.org/wiki/Romeo_and_Juliet"
    urlhash="81e5c5cb0c38556dd5163117dc0b2900"
    icon="http://bits.wikimedia.org/favicon/wikipedia.ico" title="Romeo and
    Juliet - Wikipedia, the free encyclopedia" source="page"/>
  </coopfox>
  <thread>t1</thread>
</message>

<!-- Romeo sends a persistent hightlight (Direct-Quote). -->
<message to="coopfox.net" from="romeo@coopfox.net/coopfox" type="chat" id="m6">
  <coopfox xmlns="http://coopfox.net/xmpp/namespace" timestamp="1388963150925">
    <highlight type="insert" url="http://en.wikipedia.org/wiki/Romeo_and_Juliet">
      <text context="05230099" xpath="25064c43" precontext="72902b46"
      postcontext="db67acd5" pretext="c22695c7" posttext="049005ad">If I profane
      with my unworthiest hand This holy shrine</text>
    </highlight>
    <location url="http://en.wikipedia.org/wiki/Romeo_and_Juliet"
    urlhash="81e5c5cb0c38556dd5163117dc0b2900"
    icon="http://bits.wikimedia.org/favicon/wikipedia.ico" title="Romeo and
    Juliet - Wikipedia, the free encyclopedia" source="page"/>
  </coopfox>
  <addresses xmlns="http://jabber.org/protocol/address">
    <address type="to" jid="juliet@coopfox.net/coopfox"/>
  </addresses>
  <thread>t1</thread>
</message>

<!-- Romeo points to his Direct-Quote. -->
<message to="coopfox.net" from="romeo@coopfox.net/coopfox" type="headline" id="m7">
  <coopfox xmlns="http://coopfox.net/xmpp/namespace" timestamp="1388963260224">
    <chat action="select" id="m6"/>
  </coopfox>
  <addresses xmlns="http://jabber.org/protocol/address">
    <address type="to" jid="juliet@coopfox.net/coopfox"/>
  </addresses>
  <thread>t1</thread>
</message>

…

<!-- Juliet comments on Romeo's former Direct-Quote. -->
<message to="coopfox.net" from="juliet@coopfox.net/coopfox" type="chat" id="m8">
  <body>For saints have hands that pilgrims' hands do touch</body>
  <thread parent="t1">m6</thread>
  <active xmlns="http://jabber.org/protocol/chatstates"/>
  <coopfox xmlns="http://coopfox.net/xmpp/namespace" timestamp="1388963201605"/>
  <addresses xmlns="http://jabber.org/protocol/address">
    <address type="to" jid="romeo@coopfox.net/coopfox"/>
  </addresses>
</message>
```

```
<!-- Juliet deletes her comment. -->
<message to="coopfox.net" type="chat" from="juliet@coopfox.net/coopfox" id="m9">
  <coopfox xmlns="http://coopfox.net/xmpp/namespace" timestamp="1388963251649">
    <chat action="delete" id="m8"/>
  </coopfox>
  <active xmlns="http://jabber.org/protocol/chatstates"/>
  <thread>t1</thread>
  <addresses xmlns="http://jabber.org/protocol/address">
    <address type="to" jid="romeo@coopfox.net/coopfox"/>
  </addresses>
</message>

<!-- Romeo sends a text message. -->
<message to="coopfox.net" from="romeo@coopfox.net/coopfox" type="chat" id="m10">
  <body>My lips, two blushing pilgrims, ready stand</body>
  <active xmlns="http://jabber.org/protocol/chatstates"/>
  <coopfox xmlns="http://coopfox.net/xmpp/namespace" timestamp="1388963288763">
    <location url="http://en.wikipedia.org/wiki/William_Shakespeare"
    urlhash="d669863c846b59b3704260b8f3f99d30"
    icon="http://bits.wikimedia.org/favicon/wikipedia.ico" title="William
    Shakespeare - Wikipedia, the free encyclopedia" source="page"/>
  </coopfox>
  <thread>t1</thread>
  <addresses xmlns="http://jabber.org/protocol/address">
    <address type="to" jid="juliet@coopfox.net/coopfox"/>
  </addresses>
</message>

<!-- Juliet adds Romeo's message to the results. -->
<message to="coopfox.net" from="juliet@coopfox.net/coopfox" type="chat" id="m11">
  <coopfox xmlns="http://coopfox.net/xmpp/namespace" timestamp="1388963297463">
    <result action="up" id="m10"/>
  </coopfox>
  <addresses xmlns="http://jabber.org/protocol/address">
    <address type="to" jid="romeo@coopfox.net/coopfox"/>
  </addresses>
  <thread>t1</thread>
</message>

<!-- Juliet leaves the session. -->
<message to="coopfox.net" from="juliet@coopfox.net/coopfox" type="headline" id="m12">
  <coopfox xmlns="http://coopfox.net/xmpp/namespace" timestamp="1388965552426">
    <participant action="leave" jid="juliet@coopfox.net"/>
  </coopfox>
  <addresses xmlns="http://jabber.org/protocol/address">
    <address type="to" jid="romeo@coopfox.net/coopfox"/>
  </addresses>
  <thread>t1</thread>
</message>

…

</stream:stream>
```