

# Lowering the Barrier to Entry in Online Group Communication

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Software Engineering & Internet Computing**

eingereicht von

**Jakob Lahmer**

Matrikelnummer 0626313

an der  
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Purgathofer

Wien, 17.04.2015

\_\_\_\_\_  
(Unterschrift Verfasser)

\_\_\_\_\_  
(Unterschrift Betreuung)



# Lowering the Barrier to Entry in Online Group Communication

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Software Engineering & Internet Computing**

by

**Jakob Lahmer**

Registration Number 0626313

to the Faculty of Informatics  
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Purgathofer

Vienna, 17.04.2015

\_\_\_\_\_  
(Signature of Author)

\_\_\_\_\_  
(Signature of Advisor)





# Erklärung zur Verfassung der Arbeit

Jakob Lahmer  
Pfarrhofgasse 11, 3384 Haunoldstein

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

---

(Ort, Datum)

---

(Unterschrift Verfasser)



# Abstract

New technologies are accepted very slowly. Humans are creatures of habit and new technologies often require the adoption of everyday routines. Especially in the field of online group communication these adjustments not only affect an individual user but also his / her social environment. This fact results in a high barrier to entry for new communication technologies.

In contrast, new technologies can be very helpful and usually provide an additional value compared to established solutions. Due to the high barrier to entry, the usage of new communication solutions poses the risk of excluding individual persons with less computer knowledge. The latter is especially drastic when it comes to communication within voluntary associations, which are generally comprised of very diverse members with dissimilar technological background.

The goal of this work is to facilitate online group communication while keeping the barrier to entry as low as possible. This is done by using a technique from information visualization that is applied to the visual structure of a computer-mediated communication (CMC) solution. The field of information visualization offers various techniques for displaying large data sets in a way users can easily understand and explore. The latter is a main problem in online group communication: Participants of an online discussion usually refer to a specific previous statement (*focus*), but should also keep the overall trend of the discussion in mind (*context*).

The hypothesis is that by applying a technique from information visualization to the visual structure of an online group communication system it becomes easier to follow a discussion. To this end, existing information visualization techniques and types of CMC are analyzed and assessed through literature review. Subsequently a CMC solution with a low barrier to entry is selected (*e-mail*) and an appropriate information visualization technique (*Fisheye View*) is applied to its visual structure. Using the principle of *progressive enhancement* the resulting application can be used either only by *e-mail* or using the enhanced visual structure provided in a web interface.

The evaluation conducted by a user test and open interviews revealed that the developed application has the potential to improve asynchronous online group communication while keeping the barrier to entry very low, but is not applicable for real-time communication.



# Kurzfassung

Neue Technologien werden nur sehr langsam angenommen. Der Mensch ist ein Gewohnheitstier und neue Technologien erfordern oft eine Anpassung von alltäglichen Routinen. Besonders im Bereich der Online-Kommunikation betreffen diese Anpassungen nicht nur einzelne Benutzer und Benutzerinnen sondern auch sein bzw. ihr soziales Umfeld. Dieser Umstand führt zu einer hohen Hemmschwelle neue Kommunikationstechnologien einzusetzen.

Im Gegensatz dazu können neue Technologien sehr hilfreich sein und bieten im Vergleich zu gängigen Lösungen üblicherweise einen deutlichen Mehrwert. Aufgrund der hohen Zugangshemmnis birgt der Einsatz von neuen Kommunikationslösungen das Risiko Personen mit weniger Computerkenntnissen auszugrenzen. Dies hält vor allem Vereine mit unterschiedlichsten Mitgliedern davon ab neue Kommunikationslösungen zu verwenden.

Das Ziel dieser Arbeit ist es die Online-Gruppenkommunikation zu vereinfachen und gleichzeitig die Zugangshemmnis so gering wie möglich zu halten. Zur Vereinfachung von Online-Gruppenkommunikation wird eine Technik der Informationsvisualisierung auf die visuelle Struktur einer Computer-mediated Communication (CMC) Lösung angewandt. Der Bereich der Informationsvisualisierung bietet verschiedene Techniken um großen Datensätze für Benutzer und Benutzerinnen verständlich und leicht durchsuchbar anzuzeigen, was ein großes Problem der Online-Gruppenkommunikation ist: Teilnehmer und Teilnehmerinnen einer Online-Diskussion beziehen sich üblicherweise auf eine spezifische vorherige Aussage (*focus*), sollen aber auch einen Überblick über den generellen Verlauf der Diskussion haben (*context*).

Die Hypothese ist, dass durch die Anwendung einer Technik der Informationsvisualisierung auf die visuelle Struktur einer Online-Gruppenkommunikationslösung einer Diskussion leichter gefolgt werden kann. Im Rahmen der Diplomarbeit werden dazu zunächst bestehende Techniken der Informationsvisualisierung und verschiedene Typen von CMC durch Literaturrecherche analysiert und im gegebenen Kontext bewertet. Anschließend wird eine CMC Lösung mit einer geringen Zugangshemmnis (*E-Mail*) ausgewählt und eine passende Technik der Informationsvisualisierung (*Fisheye View*) auf die visuelle Struktur angewandt. Durch die Verwendung des Prinzips der *progressiven Verbesserung* kann die daraus resultierende Anwendung sowohl nur per E-Mail, als auch über ein Web-Interface mit der verbesserten visuellen Struktur genutzt werden.

Die Evaluierung durch einen Usertest und offene Interviews hat ergeben, dass die entwickelte Anwendung das Potential hat asynchrone Online-Gruppenkommunikation zu verbessern ohne die Zugangshemmnis zu erhöhen, aber nicht für die Echtzeitkommunikation anwendbar ist.

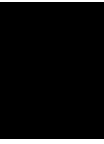


# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation and Problem Description . . . . .	1
1.2	Aim of the Work . . . . .	2
1.3	Structure and Approach . . . . .	3
<b>2</b>	<b>Information Visualization</b>	<b>5</b>
2.1	Zooming . . . . .	6
2.2	Overview+Detail . . . . .	9
2.3	Focus+Context . . . . .	13
<b>3</b>	<b>Computer-mediated Communication</b>	<b>23</b>
3.1	Definition . . . . .	23
3.2	Types of CMC . . . . .	26
3.3	Barriers to Entry . . . . .	41
<b>4</b>	<b>Context Evaluation</b>	<b>45</b>
4.1	Application Context . . . . .	45
4.2	Selection of Technologies . . . . .	50
<b>5</b>	<b>Design and Implementation</b>	<b>57</b>
5.1	Application Design . . . . .	57
5.2	Development Process . . . . .	69
<b>6</b>	<b>Evaluation</b>	<b>81</b>
6.1	Approach . . . . .	81
6.2	Results and Interpretation . . . . .	82
6.3	Linear Layout versus Fisheye View Layout . . . . .	85
<b>7</b>	<b>Outlook</b>	<b>89</b>
7.1	Further Development . . . . .	89
7.2	Scope of Further Development . . . . .	93
<b>8</b>	<b>Conclusion</b>	<b>97</b>
8.1	Summary . . . . .	97

8.2 Future Work . . . . .	99
<b>Bibliography</b>	<b>101</b>





# Introduction

## 1.1 Motivation and Problem Description

In the field of graphical information visualization a common problem is displaying a large data set in a way people can easily understand and explore [1]. Furthermore people often need the ability to focus on a particular detail in the data set without losing the surrounding context information. For example navigating on a map can be simplified by providing both a detailed view and an overview of the current location [2].

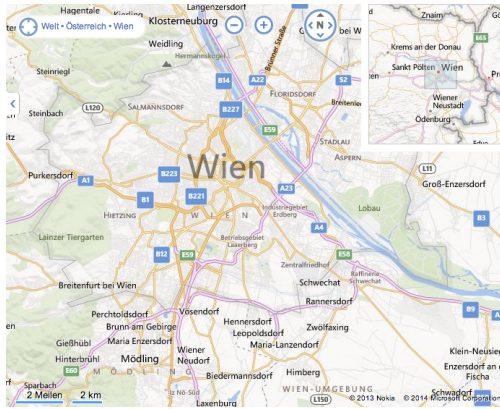
Information visualization offers various techniques to fulfill these requirements. The most common ones are:

**Zooming:** Users can zoom in to focus on an information subset and zoom out for more context information. Using the *Zooming* technique it is not possible to see both the focused and the contextual view simultaneously [2].

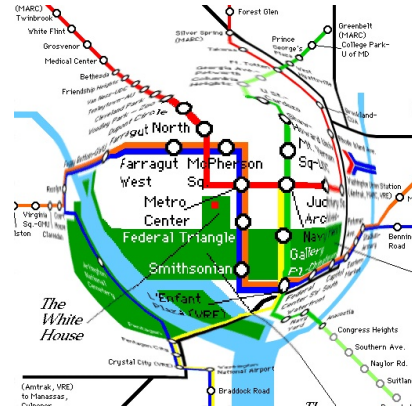
**Overview+Detail:** The interface provides a detailed view of an information subset and a spatially separated overview. In contrast to *Focus+Context* the two views are completely separated, although a user action in one view is often reflected directly in the other [3]. *Overview+Detail* is often used by web mapping applications like Google Maps [4] or Bing Maps [5], as shown in Figure 1.1a.

**Focus+Context:** The best-known *Focus+Context* technique is the *Fisheye View*. It combines the detail and the context view by magnifying the current point of interest and compressing the information further away, as seen in Figure 1.1b.

The same requirements - focusing on a detail while keeping the context in mind - can be applied to online group communication. When arguing in an online discussion, participants often refer to a specific previous statement (*focus*), but should also keep the overall trend of



(a) Example for *Overview+Detail*. The overview is shown in the upper right corner whereas the detail view covers the majority of the screen [5].



(b) The underground map of Washington displayed using the *Fisheye View* technique. The current point of interest is shown in full size, while the information further away is displayed distorted [6].

Figure 1.1: Examples for information visualization techniques.

the discussion in mind (*context*). Especially when communication takes place by e-mail the overview can be lost easily [7].

Although the variety of online communication tools seems to be immeasurable, e-mail remains the most common communication tool in business space. Particularly voluntary associations, which often can not afford a business communication tool, rely on e-mail for their internal communication [8]. Another reason for the wide-spread usage of e-mail is its low barrier to entry and the ease of use. The latter is important for voluntary associations because their members potentially have very different levels of computer skills.

An e-mail conversation is structured as a tree. Each e-mail has exactly one ancestor (except the initial message). Today's e-mail clients offer many features but mostly lack the ability to represent e-mails differently than as a structured list. Applying the techniques of graphical information visualization to an e-mail based communication system has the potential to facilitate online group communication without introducing more complexity to the user.

## 1.2 Aim of the Work

The object of this thesis is to simplify online group communication while keeping the barrier to entry as low as possible. In particular the visualization of large communication threads should be enhanced. Although information visualization offers a variety of techniques for visualizing large data sets, commonly used web-based communication tools like *e-mail*, *newsgroups* and *Internet forums* stick to the structure of an ordered list. Within the resulting "wall of content" the context information of previous messages often gets lost.

This thesis investigates the characteristics of both information visualization techniques and web-based communication tools. The aim is to find a presentation form for large communica-

tion threads that allows the user to focus on a specific message while providing more context information for helping the user to keep the overview. Especially in long e-mail conversations with multiple participants the overview is lost very quickly and questions like “What was the initial purpose of the message?” and “What has Alice previously said about this?” arise.

The results of the investigations are evaluated in the context of the task to design an e-mail based communication solution for voluntary associations. Voluntary associations are generally comprised of very diverse members with dissimilar technological background. Therefore they often rely on e-mail for the internal communication [8] and have experience with large e-mail threads. Especially more technophilic members are anxious to facilitate the communication and are open for new solutions. The goal of this thesis is a universally usable solution, which facilitates online group communication without excluding persons with less computer knowledge.

The central question this thesis wants to answer within the mentioned context is: Can online group communication be facilitated by applying a technique of graphical information visualization to an e-mail based communication system?

### **1.3 Structure and Approach**

Chapter 2 and chapter 3 present the state of the art in the fields of information visualization and computer-mediated communication through literature review. Chapter 2 lists and explains the most important techniques of graphical information visualization. For each technique a brief history is given and based on examples their benefits and disadvantages are discussed.

Computer-mediated-communication is described and classified in chapter 3. Furthermore the most popular types are introduced and the barriers to entry are discussed.

The context of the developed application and the evaluation of the presented information visualization techniques and types of computer-mediated communication are covered in chapter 4. First the context of the application, the status quo and the requirements for the developed application are described. Afterwards the decision for an optimal visualization technique and the selection of a computer-mediated communication type are discussed. Furthermore different e-mail thread visualization techniques are presented and discussed.

Chapter 5 concentrates on the application’s design and the development process. The application’s design based on the evaluation of the context in chapter 4, its general structure and the core features are described at first. Afterwards the development process and the technical structure are discussed, the first prototype is introduced and the productive application is presented.

The evaluation of the designed and developed application is covered in chapter 6. The results are presented and interpreted and the new visual structure is compared to a classic linear layout.

Chapter 7 provides an outlook for future work and potential changes. In addition a brief overview of general opportunities for continuing a scientific project when the scientific work ended is given.

In chapter 8 the results are presented and a short summary of the work is given.



# Information Visualization

“The goal is to support the exploration of unknown data and to reveal features and underlying structures. The main challenges in information visualization are the presentation of very large data sets and the optimal exploitation of the display area.” (Schumann 2004, [9]).

Information visualization deals with the visual representation of abstract data. The objective is to exploit the enormous efficiency of the human visual system to explore large amounts of information and discover insights, that are limited to human imagination and creativity. Furthermore information visualization attempts to offer an easy to understand presentation of an abstract data set. The latter should facilitate the interpretation process and saves users the acquisition of sophisticated interpretation methods.

To achieve this goal Shneiderman introduced the “Visual Information Seeking Mantra”: “*Overview first, zoom and filter, then details-on-demand*” [10]. This mantra describes a fundamental process for information visualization. In a first step an *overview* of the data is needed. The *overview* serves as orientation and the user can acquaint him- or herself with the data set. In a next step the focus can be set to a partition of the initial data set using *zooming* techniques. In addition the data can be restricted by applying *filters*. This helps focusing on the current task and provides a more specific view. Once the data set is reduced to a manageable amount of elements (according to Shneiderman this are a few dozen elements [10, p. 96]), it should be fairly easy to query details about a single element or the selected group of elements (*details-on-demand*).

The central issue in information visualization is the presentation of large data sets on limited presentation space. While the size of data sets is constantly increasing [11], the display size stays predominantly the same. In the past decades several techniques were introduced which try to meet this issue. The fundamental concept used by all techniques is the separation of views. While the *context view* shows an overview of the data set and provides surrounding meta information the *detail view* focuses on a specific part of the data set and shows it as close up. The following sections will describe the most common approaches and compare them to each

other keeping in mind that the objective is to find a suitable visualization technique for an e-mail based communication system.

## 2.1 Zooming

### Introduction

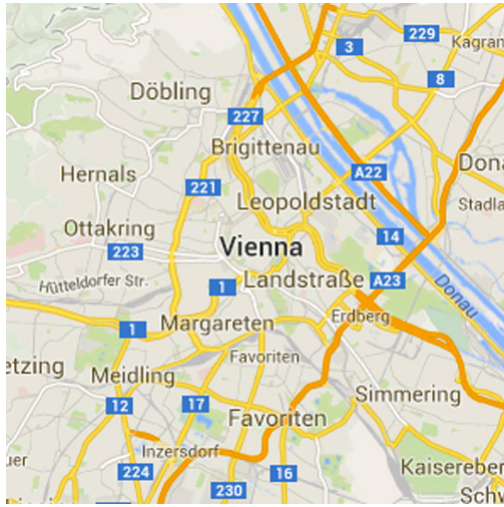
A fundamental information visualization technique is *zooming*. It supports both *contextual* and *detail views*, but can not present both simultaneously. *Zooming* uses *time* as a parameter to separate the views. Due to this limitation only one view of the current information space is shown at any time. Users can magnify the information space to focus on a specific part or demagnify it to get more context information. Furthermore *zoomable interfaces*, or also called *zoomable user interfaces*, often support panning. Besides the description of various *zoomable interfaces* this section also discusses the transition between zoom states. For instance animating the transition between two views is a widely-used technique, as described in section 2.1.

Basically there are two types of *zoomable interfaces*:

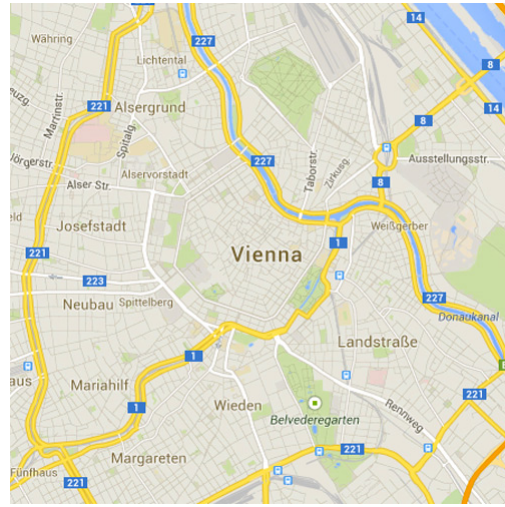
- The more common *geometric zoom* scales the information space linearly. The closer one zooms in the bigger the focused elements are displayed - there is no abstraction or additional transformation of the displayed elements.
- In contrast the *semantical zoom* allows elements to have a more complex relation between the current scaling and their appearance [12]. For example additional features can be exposed depending on the zoom state. The most common use case for *semantical zoom* are maps. The same area on a map is shown with a different level of detail depending on the current zoom state, as seen in Figure 2.1.

### Navigation and Transitions

Because *zoomable interfaces* only present one view at a time, a sophisticated navigation mechanism in the information space is essential. Most applications provide widgets in the toolbar to magnify or demagnify the information space. Those widgets are mostly symbolized by a magnifying glass and are well understood by the user [2]. Another approach to ease the navigation through different zoom states is to provide keyboard shortcuts. For instance double-clicking with the left button on the information space in Google Maps [4] triggers a zoom-in. Although this navigation practice is very intuitive it raises an important question: How can this action be reverted without using the widgets? The common-known *undo* action is not applicable for this use case because it is reserved for actions that modify the data and not their representation. In Google Maps the reverse action is double-clicking the right button. An informal study by Cockburn et al. showed, that even when a user is shown the zoom-in feature, it takes him or her many attempts to find the reverse action [2, p. 10]. As a conclusion the study by Cockburn et al. [2] points out that it is important to keep the reverse action in mind when providing alternate navigation methods.



(a) Map of Vienna providing more context information.



(b) Map of Vienna in a closer zoom state with more details.

Figure 2.1: Map of Vienna in two different zoom states. Figure 2.1a provides more context information, while in Figure 2.1b more details are presented [4].

Another important aspect of *zoomable interfaces* is the transition between zoom states. Basically there are two different techniques for zoom states: *Continuous zooming* facilitates the stageless scaling of the information space. The user can explicitly choose a scaling factor the information space should be presented with. In contrast *discrete zooming* has predefined zoom states a user can choose from.

Regardless of the used technique each *zoomable interface* has to adopt its presentation depending on the current zoom state. One method to respond to a change of the scaling factor is to immediately present the new view. Although this obviously is the most efficient solution animations are frequently used to provide a smooth transition. An animated transition between two zoom states helps the user to assimilate the relationship between the states [13]. As a study by Shanmugasundaram and Irani shows [14] users are as accurate in performing tasks with animated transitions as without, but are almost twice as fast and need less initial familiarization using an animated zooming interface. The used animation speed does not influence this result, which was also shown by Klein and Bederson [15], but a suitable animation speed is essential to reveal the intended meta-information without slowing down the users work flow. According to Shanmugasundaram and Irani a transition time of  $\frac{1}{4}$  of a second is sufficient for the user to recognize and classify the change of the information space [14].

In addition to animate the transition of two zoom states van Wijk and Nuji introduced pan-zooming animations to “offer a smooth animation from one close-up on the map to another” [16]. Therefore they developed a set of equations to reduce the optical flow during pan-zooming. Van Wijks formula assumes that the target element is known and therefore is not applicable for combining *zooming* and *panning* to explore a data set. According to Cockburn [2] combining *zooming* and *panning* techniques to explore a data set is likely to overtax the users spatial

perception.

## Applications using Zooming

The first zoomable desktop environment was Pad and was introduced by Perlin and Fox in 1993 [17]. It was presented as an “infinite two dimensional information plane [...], where objects are organized geographically”. Pad uses *semantic zooming* (described in section 2.1) and supports the concept of *portals*. *Portals* are, as the name already suggests, views on other parts of the Pad surface. Besides allowing the user to peer into and pan over the surface, they can also show an overview or a close up of the workspace (as seen in Figure 2.2) and are even capable to recursively look into other *portals*.

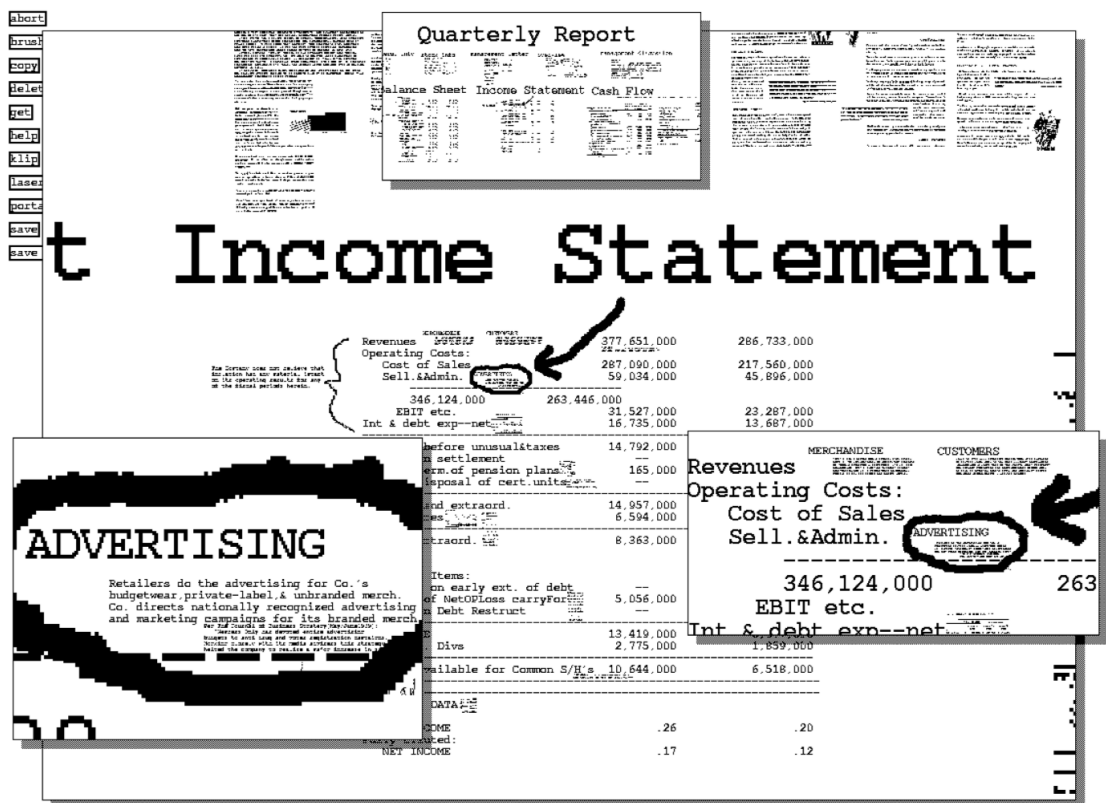


Figure 2.2: A financial document displayed on the Pad surface. At the top a small *portal* gives an overview of the document, while the other *portals* show close ups of specific parts [17].

In the following years various toolkits evolved from extensive research concerning *zooming*. For example *Pad++* (Bederson and Hollan in 1994 [18]), *Jazz* (Bederson et al. in 2000 [12]) and the *Zoomable Visual Transformation Machine (ZVTM)*, a Java based *zoomable user interface* toolkit, were introduced (Pietriga in 2005 [19]).

Nowadays *zooming* can be found in a variety of human computer interfaces. Many standard computer programs (e.g. Adobe Photoshop, Microsoft Word and so on) offer a *zooming* func-



tionality. Further enhancements and modifications mostly try to either display the *contextual* and the *detail view* simultaneously or to combine them in a different way. The umbrella terms for these techniques are *Overview+Detail* and *Focus+Context*. Both are described in detail in the following sections.

## 2.2 Overview+Detail

### Introduction

The *Overview+Detail* technique is capable of showing the *contextual* and the *detail view* simultaneously by using two separated presentation spaces. One view presents an overview of the data set for orientation (*context*), the other one shows a detailed view of the data the user currently works with (*focus*).

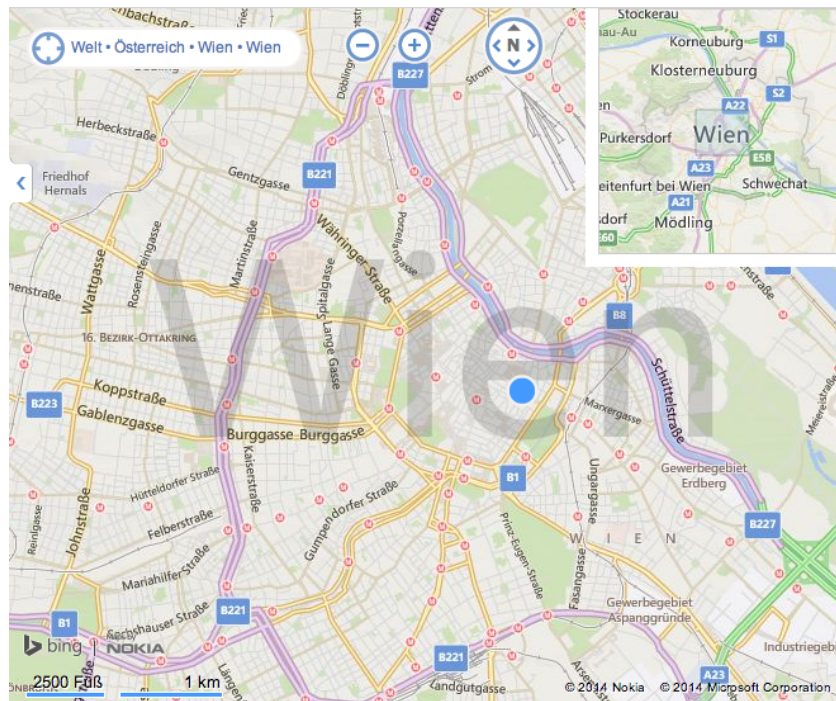


Figure 2.3: Bing Maps [5] uses an *Overview+Detail* technique. While in the upper right corner an *contextual view* provides an overview of the current focused location, the bigger part of the screen shows the separated *detail view*.

A typical use case is navigation on a map as seen in Figure 2.3. The *contextual view* (located in the upper right corner) serves as orientation for the user. The interactive rectangular subregion within the *contextual view* (highlighted using a blue background) corresponds to the area shown in the *detail view*. Tidwell calls this a “You are here” sign which users can use to tell at a glance where they are currently focusing on [3]. The *detail view* shows an enlarged projection of the *contextual view* and allows the user to inspect the details. The two views should be synchronized,

so that changing one view is directly reflected in the representation of the other one. For instance scaling the *detail view* should result in an assimilation of the *contextual view* according to the new scale factor.

Referring to Hornbæk et al. the usage of *Overview+Detail* techniques has three major benefits [20]:

- First of all *navigation is faster and more efficient* because users can use the overview window for navigation.
- Another benefit is that the contextual view eases *keeping track of the current position* and may provide task relevant information itself.
- Third the overview gives the user a *feeling of control*.

A possible drawback is performance because two separated views must be generated. In addition more screen space is needed than for interfaces without an overview.

## Lenses

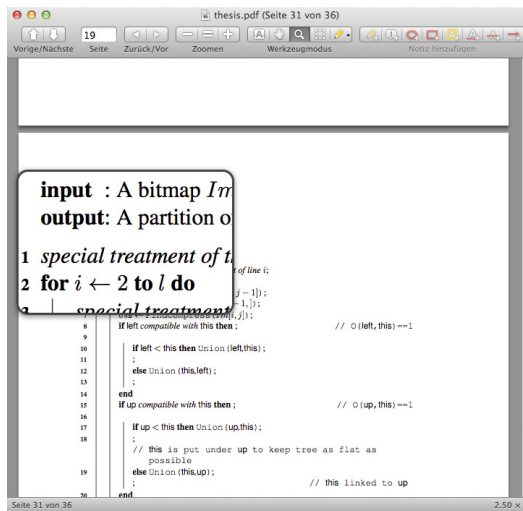
Common *Overview+Detail* interfaces separate the two views on a two dimensional surface - they are splitted using the x and y coordinates. Another technique to spatially divide the views is to use the z-coordinate, e.g. overlaying the *contextual view* with a *detail view* (so called *lenses*) or vice versa. The lense follows the users mouse movements and shows the background in more detail.

According to Cockburn et al. lenses are movable display regions that overlay the default window [2, p. 8]. They are very similar to *fisheye views* (described in section 2.3) but use a completely separated region to display information and do not use distortion techniques. As seen in Figure 2.4 this technique can be used to provide a more detailed view by zooming the information in the background (Figure 2.4a displays the *detail view* on top), or to provide an *overview* of the data, as seen in Figure 2.4b.

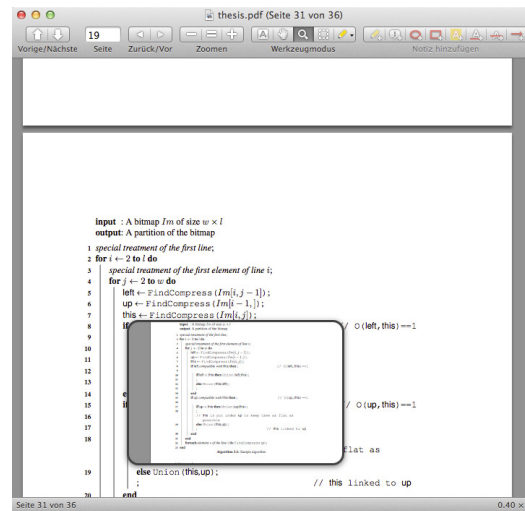
## Applications using Overview+Detail

Nowadays many standard applications use *Overview+Detail* techniques to enrich the user experience. A very popular use case is to show thumbnails besides a document for providing more overview. Almost all applications dealing with documents (e.g. text documents or presentations) have the possibility to show a thumbnail preview of the document (e.g. Microsoft PowerPoint, Adobe Reader or Skim). Thumbnails provide a faster way to navigate a document and aid the user to orient, but also has some drawbacks: the higher the scale ratio of thumbnails is the more thumbnails can be displayed without scrolling, but using a lower scale ratio, more context information of each thumbnail is visible for the user. Many applications solve this trade-off by allowing the user to manually chose the scale factor. Furthermore thumbnails can usually be completely hidden because not all users need the additional context information.

Further already mentioned use cases are mapping applications like Google Maps or Bing Maps and *lenses*, which are used in a variety of applications to zoom a spatial region (e.g. in PDF viewers as seen in Figure 2.4).



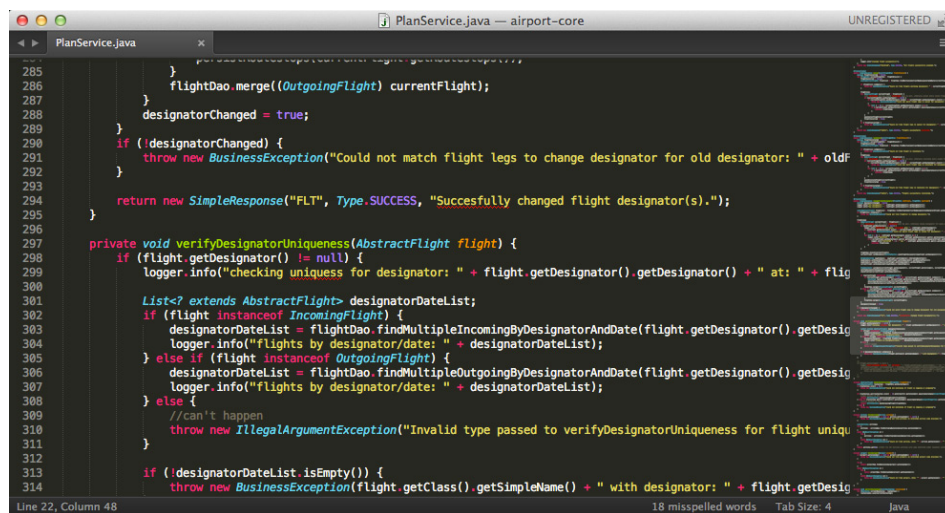
(a) A *lense* used to show a spatial region in more detail seen on the example of Skim.



(b) *Lenses* can also be used to provide an overview of the currently focused information.

Figure 2.4: Skim [22], a PDF reader for Mac OS X, uses *lenses* for both displaying the *contextual* and the *detail view*.

Another use case of *Overview+Detail* techniques is the reading and writing of digital documents. Especially when reading or writing long documents an overview of the entire document can be very useful. As seen in Figure 2.5 the content of the document is shown like in classic linear interfaces (as a linear sequence of text and images) but in addition an overview pane covering the entire document is shown on the right side. The white rectangle in the overview pane indicates the current position of the detail view in the document and can be dragged to change the current position. Hornbæk and Frøkjær investigated the usability of a *linear*, a *Fisheye* and an *Overview+Detail* interface for reading and writing electronic documents [21]. In their experiment all but one subject preferred the *Overview+Detail* interface and Hornbæk and Frøkjær recommend using such an interface for creating electronic documents.



```
285 }
286 flightDao.merge((OutgoingFlight) currentFlight);
287 }
288 designatorChanged = true;
289 }
290 if (!designatorChanged) {
291     throw new BusinessException("Could not match flight legs to change designator for old designator: " + oldF
292 }
293 }
294 return new SimpleResponse("FLT", Type.SUCCESS, "Successfully changed flight designator(s).");
295 }
296
297 private void verifyDesignatorUniqueness(AbstractFlight flight) {
298     if (flight.getDesignator() != null) {
299         logger.info("checking uniqueness for designator: " + flight.getDesignator().getDesignator() + " at: " + flig
300     }
301     List<? extends AbstractFlight> designatorDateList;
302     if (flight instanceof IncomingFlight) {
303         designatorDateList = flightDao.findMultipleIncomingByDesignatorAndDate(flight.getDesignator().getDesignator(), flight.getDate());
304         logger.info("flights by designator/date: " + designatorDateList);
305     } else if (flight instanceof OutgoingFlight) {
306         designatorDateList = flightDao.findMultipleOutgoingByDesignatorAndDate(flight.getDesignator().getDesignator(), flight.getDate());
307         logger.info("flights by designator/date: " + designatorDateList);
308     } else {
309         //can't happen
310         throw new IllegalArgumentException("Invalid type passed to verifyDesignatorUniqueness for flight uniqueness");
311     }
312
313     if (!designatorDateList.isEmpty()) {
314         throw new BusinessException(flight.getClass().getSimpleName() + " with designator: " + flight.getDesignator() + " already exists");
315     }
316 }
```

Line 22, Column 48 18 misspelled words Tab Size: 4 Java

Figure 2.5: The Sublime Text editor provides an overview pane per default [23].

## 2.3 Focus+Context

### Introduction

In contrast to previous described techniques the *Focus+Context technique* does not separate the *detail* and the *contextual view* in different screens. The two views are seamlessly connected on one screen by applying different scale or distortion functions. The main advantage is that users can focus on a special point of interest while keeping the overview of the complete data set. A well-known example of a *Focus+Context* interface is the Mac OS X Dock (Figure 2.6). An icon in the icon panel expands when the user focuses it by moving the mouse over it. Adjacent icons are emphasized too, whereas icons further away are not changed. This effect is very popular and became a symbol for the user experience Apple wants to offer their users. When moving the cursor over the Mac OS X Dock the icons “come closer” to the user and “welcome” the user to the menu. This is comparable to somebody entering a room: present people stand up and welcome the newcomer.

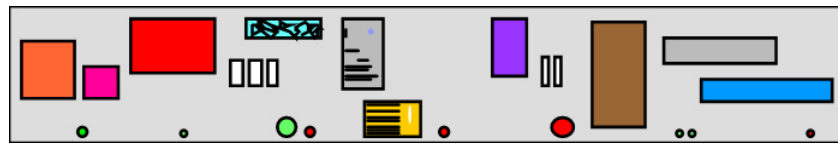


Figure 2.6: The Mac OS X Dock uses a *Focus+Context* interface to highlight the currently focused icon.

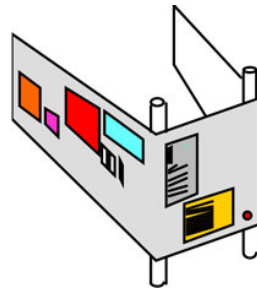
### Techniques

In 1982 Spence and Apperley introduced the *Bifocal Display*, the first *Focus+Context* technique [24]. The idea of the *Bifocal Display* is to take an information space, which can contain documents, articles, e-mails, images, etc. (Figure 2.7a), and wrap it around two vertical sticks (Figure 2.7b). The resulting information space is divided in three regions (two context regions surrounding one focus region). The frontal view of the information space displays the focused part undistorted and stretches the not focused area (Figure 2.7c). Besides stretching and distorting an important concept of the *Bifocal Display* is the continuity between the focus and the context region of the information space. By simply scrolling the user can change the focused region and the previous point of interest is moved to the context region.

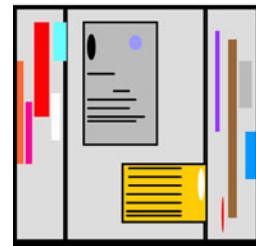
Leung extended the concept of the *Bifocal Display* in 1989 to a two-dimensional form and applied it on the map of the London Underground [26]. He increased the number of regions to nine (as seen in Figure 2.8). The central focus region is surrounded with 8 regions which are demagnified and distorted according to their position. Because the corner regions are distorted in both *X* and *Y* direction they are not distorted if the same scaling factor for *X* and *Y* is chosen. In that case the corner regions are not distorted but only scaled down.



(a) The initial information space used for the *Bifocal Display*.



(b) The information space wrapped around two sticks.



(c) Frontal view of the information space, when wrapped around two sticks. The user sees the screen from this perspective.

Figure 2.7: The transformation of the information space to create a *Bifocal Display* interface [25].

Demagnification in both X and Y dimensions	Demagnification in Y dimension	Demagnification in both X and Y dimensions
Demagnification in X dimension	<b>Central 'Focus' Region</b>  no demagnification	Demagnification in X dimension
Demagnification in both X and Y dimensions	Demagnification in Y dimension	Demagnification in both X and Y dimensions

Figure 2.8: Schematic design of a two-dimensional *Bifocal Display* [27].



Purgathofer designed and developed a visitor guide and entertainment system for the Ars Electronica Center located in Linz within his postdoctoral lecture qualification [28]. An important feature was the presentation of text on a limited available space. Therefore a vertical *Bifocal Display* was chosen to offer a user addition context information about the current position in a long text. As seen in Figure 2.9 an excerpt of the text is shown in full size while the remaining text is shown compressed.

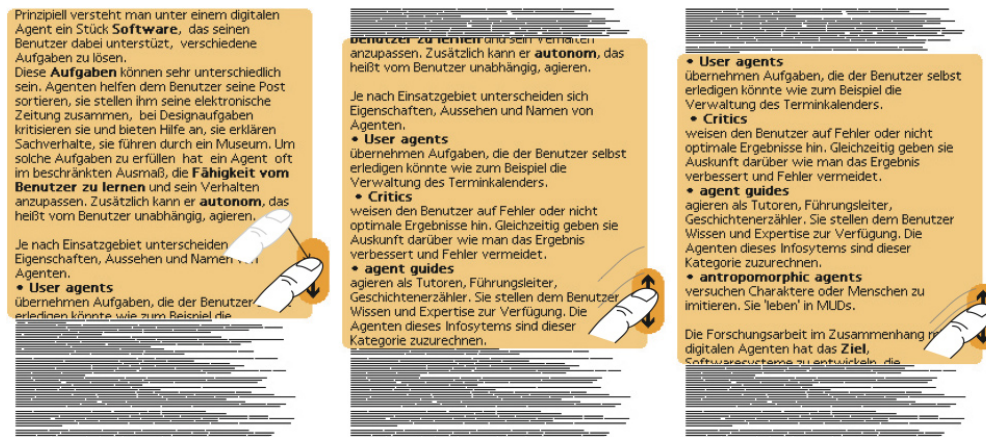


Figure 2.9: Purgathofer uses a *Bifocal Display* in his postdoctoral lecture qualification [28] for showing text on a limited space. Using the control panel a user can scroll up and down within a text.

Another descendant of the *Bifocal Display* is the *Perspective Wall* which was introduced by Mackinlay in 1991 [29]. The main difference to the *Bifocal Display* is that the *Perspective Wall* uses an increasing rate to demagnify the contextual regions, while the *Bifocal Display* uses a constant function. This results in a 3D feeling the *Bifocal Display* can not offer. The drawback is that display space is wasted in the corner regions.

At the same time as the *Bifocal Display* was introduced George Furnas was investigating a similar problem. He was working on how to provide more context information to programmers. For instance it can be very useful for programmers to see the declaration of a variable although it is several pages back. As a result Furnas introduced the *Fisheye View* for structured files in 1981 in an internal memo at Bell Labs [30]. He continued his work with publishing *Generalized Fisheye Views* in 1986 [31].

Furnas' fundamental idea for "providing a balance of local and global context" [31, p. 16] to the user was to apply the well-known concept of the *Fisheye lens* to the users workspace. The *Fisheye lens* is an ultra wide-angle lens and is used in photography to produce highly distorted images. The angle of an *Fisheye lens* can be up to 180 degrees and therefore can capture the full field of view of a human, which is almost 180 degrees. Images captured using a *Fisheye lens* show the focus point in great detail while the surrounding context information is presented distorted (Figure 2.10).

In order to apply the *Fisheye lens* concept to a user interface Furnas introduced the "Degree of Interest" (DOI) function. The DOI function assigns a number  $i$  to each element in the dis-



Figure 2.10: Picture of a BMW using a *Fisheye lens*. The hood is the current point of focus and shown in great detail, the rest of the picture is shown distorted [33].

played structure. Depending on the current task  $i$  indicates how interested the user is in seeing the element.

The most general form of the DOI function takes an *a priori importance* and the *distance* between two elements into account. The resulting DOI function is:

$$DOI_{fe(x|.=y)} = F(API_{(x)}, D_{(x,y)}) \quad (2.1)$$

$DOI_{fe(x|.=y)}$ , the “Degree of Interest” according to the *Fisheye View*, for the element  $x$ , when the current focus is on element  $y$ , is defined by a function  $F$  combining the *API* (*a priori importance*) of  $x$  and the distance between  $x$  and  $y$  ( $D_{(x,y)}$ ). As stated by Furnas [31] the usefulness of the DOI function depends on a suitable definition of  $F$ .

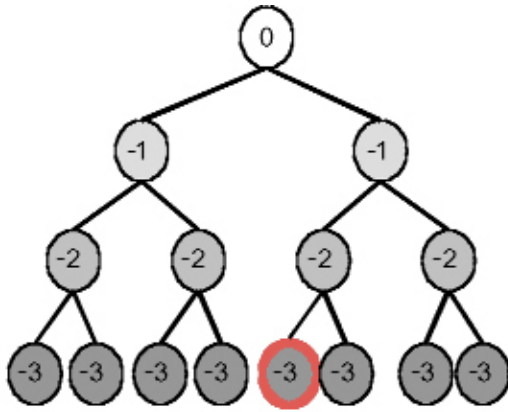
Furnas demonstrates the functionality of the general DOI function on the example of a simple tree. The distance function  $D_{(x,y)}$  is naturally given by the path length between two nodes  $x$  and  $y$ . Assuming that elements closer to the root are more important the *a priori importance* can be defined as  $API_{(x)} = -D_{(x,root)}$  and the combining function  $F$  is set to  $API_{(x)} - D_{(x,y)}$  - the greater the distance the less important an element is:

$$DOI_{fe(tree)(x|.=y)} = -D_{(x,root)} - D_{(x,y)} \quad (2.2)$$

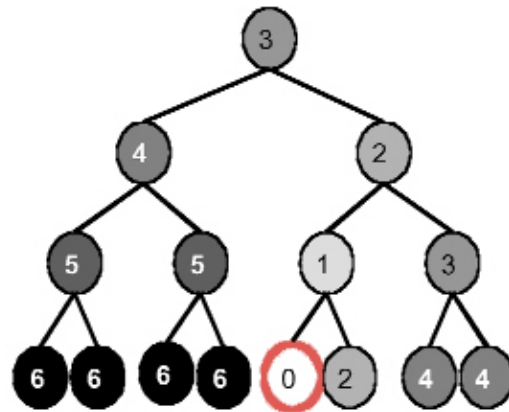
Furnas also defines a threshold  $c$  and only displays nodes where  $DOI_{fe(tree)(x)} \geq c$ . By using a threshold Furnas reduces the displayed elements and is able to optimally exploit the available workspace. The example shown in Figure 2.11 visualizes the computation steps of the  $DOI_{fe(tree)}$  function. In 2.11a the *API* value is computed using the negative distance to the root node, 2.11b shows the computed distance values starting at the current point of interest (red node). The resulting  $DOI_{fe(tree)}$  function values are shown in 2.11c. By setting the threshold to  $c = -5$  the final tree is generated (Figure 2.11d).

In his subsequent research Furnas focuses on *what* and *how* information is presented [1]. He states that it is more important *what* information is shown, because the user needs the information to perform a specific task, than *how* this information is presented. Among other things he

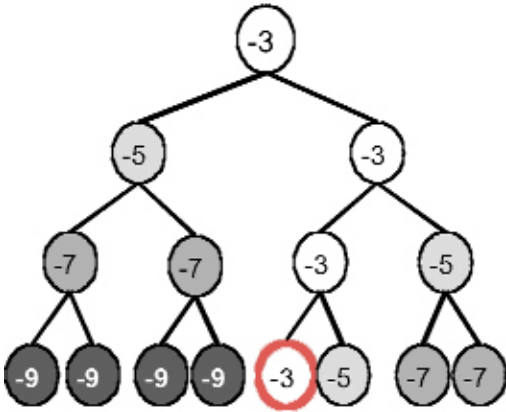




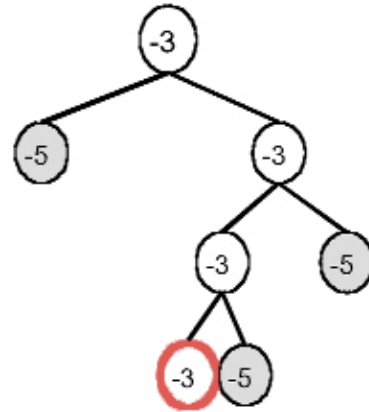
(a) Tree with listed *a priori* importance values ( $API_{(x)} - D_{(x,y)}$ ).



(b) The computed distance values ( $D_{(x,y)}$ ) starting at the current point of interest (marked red).



(c) The resulting  $DOI_{fe(tree)}$  values of each node.



(d) Setting the threshold to  $c = -5$  and only displaying nodes with  $DOI_{fe(tree)(x)} \geq c$  the tree using the *Fisheye View* is produced.

Figure 2.11: The different computation steps while applying the  $DOI_{fe(tree)}$  function to a tree. The current point of interest is marked red [25].

describes the difference between semantically filtering content and visually distorting it on the example of a list of letters (Figure 2.12). According to Furnas not only explicitly filtering and deciding *what* content is shown, but also choosing *how* it is presented (e.g. scaling the representation size through visual distortion) is a filtering function. The most obvious example is that an element is filtered out when its size gets smaller than the pixel size of the display. Because it is very likely that a user does not recognize the true significance of a very small element the user can not perceive an element correctly long before it is actually filtered out.

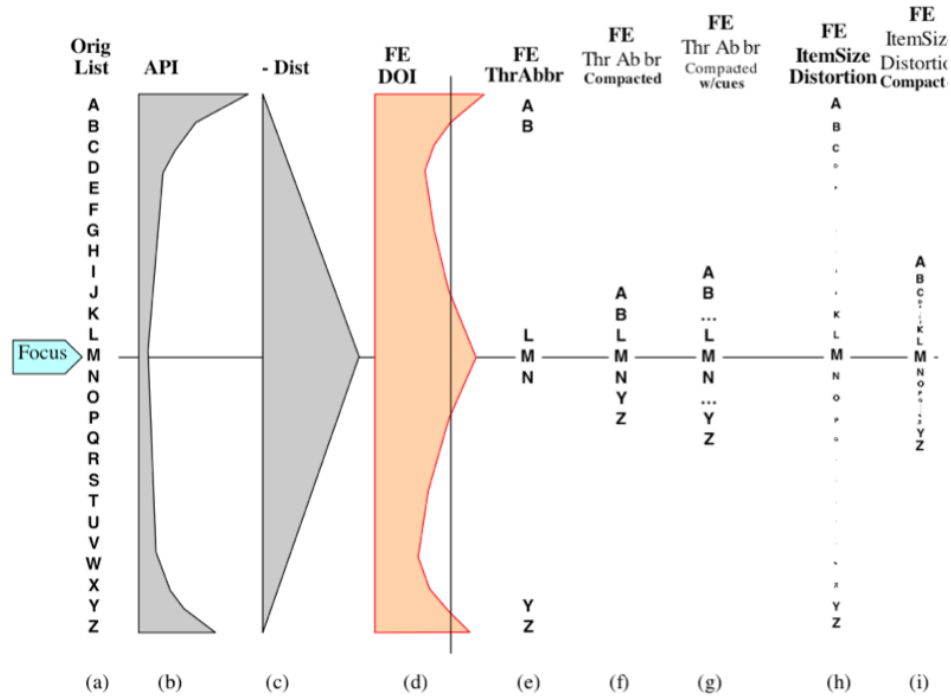


Figure 2.12: Content filtering vs. visually distortion in the Fisheye View of a list of letters. Figure (a) displays an ordered list of letters, Figure (b) shows the visual representation of an *a priori importance* function, assuming the first and the last letters are more significant. The distance from the current point of interest (the letter *M*) to the other letters is pictured in Figure (c). Figure (d) shows the resulting Fisheye *DOI* function, calculated by adding up the *API* and the distance. Furthermore a threshold is defined to generate a Fisheye *DOI* subset. In Figure (e) the resulting subset after applying the *DOI* function and the threshold is shown, Figure (f) shows the same subset using a distorted geometry to decrease the needed space, but loses distance information. Figure (g) tries to reintroduce the lost distance information by adding the elision marks "...". In contrast to the previous Figures, Figure (h) shows the letters visually distorted: Depending on the *DOI* functions value the letters have a different size. Figure (i) removes the blank space to make the list more compact [1].

In addition Furnas shows that all available methods of *how* to display the Fisheye *DOI* subset have specific disadvantages. It depends on the given task to choose the most appropriate technique. Furnas lists the following techniques and their disadvantages [1]:

- **Visual distortion techniques** are able to present a complete data set simultaneously but introduce geometric complexity. The user has to keep in mind that there are distortions and positions may differ from the original data set.
- **Zooming techniques** do not introduce distortion complexity but present only parts of the data set at once. This parts are spread out over time, which makes it very hard to define an appropriate Focus+Context rendering.
- **Overview+Detail or View+Closeup techniques** present the information simultaneously but are not able to provide continuity on the edges of the view. This raises questions like “Where does this road from the closeup continue in the overview?”, which are not easy to answer.
- **The Focus+Context screen** shows the information simultaneously and without any discontinuity at the edges. The Focus+Context screen is a special display device consisting of one or more high-resolution “focus screens” embedded into a larger “context screen”. Content is displayed across both display regions - the scaling of the content is preserved, while the resolution varies between the displays. Focus+Context screens are applicable to all tasks using Fisheye like views. In contrast to normal screens, Focus+Context screens provide a non-distorted view of the given task [34]. The installation of a Focus+Context screen is expensive and mostly does not pay off for the given task.

A widely-used visualization algorithm for *Fisheye Views* was introduced by Sarkar and Brown in 1992 [35]. In their article about “Graphical Fisheye Views of Graphs” they present geometric transformations to produce visual distortion of images and maps. They use the Euclidean distance between a point and the users current point of interest to transform each points size, position and the detail of presentation.

Another *Focus+Context* technique to display hierarchical data is the *Hyperbolic Tree* and was presented by Lamping et al. in 1995 [36]. Because tree representations of hierarchical data in Euclidean space can quickly become a clutter the *Hyperbolic Tree* uses the hyperbolic space. The entire tree is shown at once within a unit disk. By using a *Fisheye lens* showing the current node of interest with more detail than nodes farther away is possible. The currently focused node is given more space and is shown in the center of the unit disk while nodes farther away are compressed.

Various further techniques to display hierarchical data using the *Focus+Context* concept exist nowadays. E.g. Plaisant et al. presented the *SpaceTree* technique in 2002 [37]. The *SpaceTree* technique uses extensive animations when changing the current point of interest, so the user does not get lost. In addition it uses dynamic rescaling of branches to optimally exploit the available display space. Another system specialized on hierarchical data is the *TreeJuxtaposer*, introduced by Munzner et al. in 2003 [38]. The *TreeJuxtaposer* concentrates on the optimization of large tree navigation and comparison. It provides algorithms and methods to efficiently

navigate trees with more than 500.000 nodes and allows the structural comparison of trees with more than 100.000 nodes.

According to Cockburn et al. [2] *Focus+Context* techniques, especially the *Fisheye View*, have two main drawbacks caused by the visual distortion of the information space: first, data can be misinterpreted, e.g. the North-South and the East-West grid lines are distorted when a *Fisheye View* is applied on a map. The user has to keep the distortion in mind when reading the map. Another drawback is that elements are displaced away from their actual position when an adjacent element is focused. This behavior is necessary to display the current point of interest in more detail but can confuse users. A good example is the Mac OS X Dock as seen in Figure 2.6. When an icon is focused it is magnified and needs more display space, which implies that the adjacent icons need to be displaced.

### Applications using Focus+Context

The presented *Focus+Context* techniques are used in various software applications. As aforementioned a well-known example is the Mac OS X Dock, as seen in Figure 2.6. As soon as the user focuses an icon in the panel by moving the mouse over it the focused icon expands. Adjacent icons are emphasized too whereas icons farther away are not changed. Another application of the *Fisheye View* technique on a list, as the Mac OS X Dock is in the broadest sense, are Fisheye Menus presented by Bederson in 2000 [39]. Fisheye Menus were developed to ease selecting a specific item from a long list, e.g. selecting a country name from a dropdown list. To present as many items as possible without scrolling the items are displayed using an extremely small font size. The font size of an element is increased when the user moves the mouse over an element. Like in the Mac OS X Dock adjacent items are enlarged too. Both the Mac OS X Dock panel and the Fisheye Menus have the same drawback: due to the distortion of the information space and the subsequent displacement of items small mouse movements result in a change of the currently focused element. Because all items must be selectable the distance the mouse must move to focus the next element is equal to the smallest font size within the whole menu. Bederson solves this issue by offering a “Focus Lock Mode”, which is enabled by moving the cursor to the right side of the menu. The “Focus Lock Mode” locks the focus mode (bigger font size) on the currently selected item. Moving the cursor up and down the focused item does not change (the font size does not change dynamically). This facilitates the selection of an adjacent item extremely. Apple solves this issue by enlarging the whole Dock panel - this is possible since the Dock size is limited by the number of containing items and the Dock panel increases or decreases its size accordingly.

Another application area for *Focus+Context* techniques are text files, especially source code files of programs. Furnas initial description of the *Fisheye View* [30] focused on the presentation of structured files, e.g. program regions are hidden when the *DOI* value for this region is below a specific threshold. Manually hiding code or text blocks (also called code folding) is a standard feature of many IDEs (Integrated Development Environments) and text editors nowadays. IDEs, e.g. *Eclipse*, *Netbeans* and *Microsoft Visual Studio*, but also text editors such as *Sublime Text*, *Text Mate* and *Atom* support manual code folding out of the box, as seen in Figure 2.13. Automatic code folding using a *DOI* function, as contemplated by Furnas, is only used in research applications. Kersten and Murhpy introduced a “Degree of Interest Model for IDEs” called *My-*

lyn in 2005 [40]. *Mylyn*, formerly called *Mylar*, uses a *DOI* model to capture the context of the current task by monitoring the programmers activities. Depending on the current task *Mylyn* offers a different view of the IDE to the programmer.

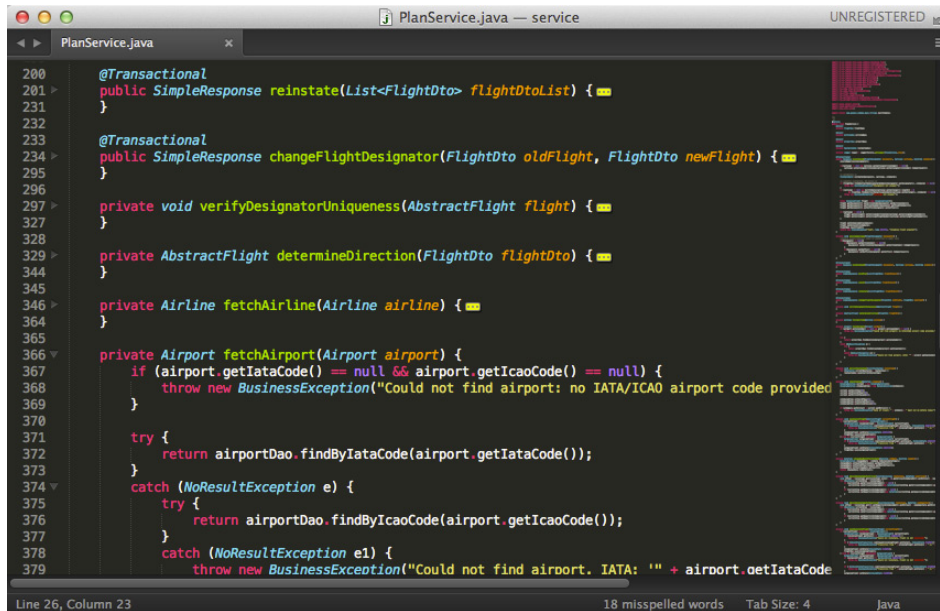


Figure 2.13: The Sublime Text editor offers the user to hide code blocks. The carets next to the line numbers indicate hidden blocks, clicking them unfolds the code block again [23].

*TableLens*, introduced by Rao and Card in 1994 [41], was the first interactive implementation of a *Focus+Context* technique for tables. *TableLens* encodes all entries as small bars and is capable of presenting an overview of large data sets. In addition a *Fisheye View* is available to expand specific rows. In 2004 Bederson et al. extended the concept of *TableLens* and introduced “*DateLens*: A Fisheye Calendar Interface for PDAs” [42]. *DateLens* is designed to fit on small devices and supports the visualization of different time-spans, as seen on Figure 2.14. User studies compared *DateLens* with classic calendar applications. The tested users completed tasks significantly faster using the *DateLens* interface and rated *DateLens* as being easier to use.

Little known applications of *Focus+Context* techniques are the *Document Lens* [43], which allows users to browse a complete document at once, and the *Focus+Context screen* [34], a special display device consisting of one ore more high-resolution “focus screens” embedded into a larger “context screen”.

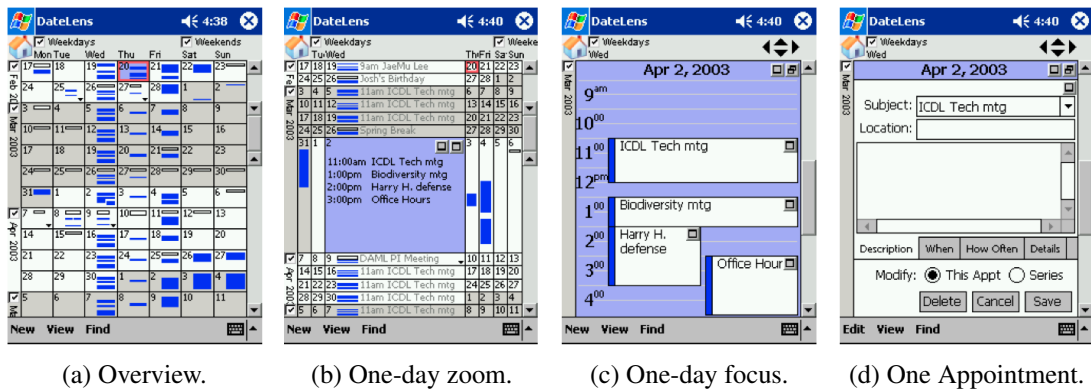


Figure 2.14: The *DateLens* interface configured to show twelve weeks. The transition between the different levels of detail are animated [2].

# Computer-mediated Communication

This chapter presents and describes basic definitions and concepts of computer-mediated communication (CMC). Because the research area of computer-mediated communication is very large, only the most relevant branches are covered in this chapter.

First various definitions of CMC are presented and a method for classifying CMC systems is shown. Afterwards the most important types of CMC are compared to each other and the barrier to entry a new CMC system is discussed.

## 3.1 Definition

Because researchers from various fields use the umbrella term computer-mediated communication (CMC) to classify their research, there are different definitions for CMC. One of the more general definitions is by McQuail and specifies CMC as *any communication that occurs through the use of two or more electronic devices* [44]. While the term CMC originally only referred to written natural language messages transferred via a computer connection, it is applied to all kinds of communication that depends on electronic devices nowadays.

Thurlow et al. use another strategy to identify the key principles of CMC [45, p. 17ff]. The term computer-mediated communication is divided into its constituents, which are defined separately:

**Communication:** The term *communication* is derived from the Latin word “communicare”, which means “to share, to divide” [46]. Compatibly to this translation Burkart defines communication as a form of social interaction [47, p. 170]. This interaction is initiated on purpose and the goal is the mediation of meanings, opinions, ideas, feelings, etc. To transport a message from the sender to the receiver communication depends on a medium, e.g. human speech or writing. In addition Burkart states that communication is interactive and therefore implies reciprocity, i.e. communication participants respond to each others actions with their own acts.

Besides the formal definition of communication, various models try to explain human communication by visualizing it. One of the most simple and popular models is the *Shannon and*

*Weaver Model*, which was introduced by Shannon in 1948 [48] and is shown in Figure 3.1. First the sender uses a transmitter to encode the message (e.g. spoken voice is encoded into wave signals). Afterwards the message is transported using an available channel and decoded by the recipient. If the recipient is not able to decode the message (e.g. because he is deaf), or can not receive the exact message (e.g. because “noise” distorted the message) the effective communication between sender and receiver is affected. The distraction can affect the communication flow and probably leads to misconceptions between sender and receiver.

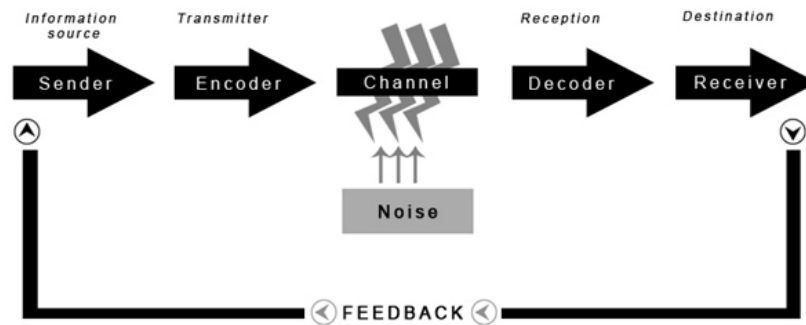


Figure 3.1: The Shannon and Weaver Model of communication [49]. The sender encodes a message using a transmitter and transports it through a channel to the receiver. The receiver decodes the message and can give feedback to the sender.

This simple model, often also referred to as *standard view of communication*, is widely used to describe human communication. Although the model is generally well accepted it is also criticized. Thurlow et al. [45, p. 17] argues that the meaning of a message does not reside in words, but is dependent on the current context, e.g. the location, the participants and the current mood of the participants. Other models like the *interactive model* and *transactional model* are built on top of the *Shannon and Weaver Model* and are not part of this thesis. A more thorough introduction in the field of communication and an explanation of further communication models can be found in Rob Anderson and Veronica Ross’s *Questions of Communication* [50].

**Mediated:** As already mentioned all communication needs a medium or a channel, as stated by Shannon [48], to transmit a message. A medium is defined as the *means of communication* and therefore is something through which an effect is produced, e.g. people are influenced. Thurlow et al. conclude that “mediation is simply the process or means by which something is transmitted” [45, p. 18]. It does not matter whether a message, a feeling or any kind of media data is transferred. Because we indicated that the meaning of a message depends on the content and the situational context it is communicated in, communication is, besides verbal or nonverbal channels, also mediated through the interaction with people.

Basically there are three *contextual channels* which influence (or mediate) communication: The *psychological* channel (our perceptions and prototypes), the *social* channel (our relationships, our experiences and stereotypes) and the *cultural* channel (the ideologies and attitudes of the society we live in) effect communication. CMC adds another more material layer of mediation: The *technological* channel. Although communication has been technologically me-



diated for centuries, e.g. when writing using a pen or by using the telephone, CMC usually restricts what it means with technology to “machinery designed, built and used for the purpose of information exchange and communication” [45, p. 19], or in other words to *Information and Communication Technologies* (ICTs).

**Computer:** Nowadays computers are indispensable for our everyday live. They are present when using our phone, driving the car, watching television, and so on. Thurlow et al. state that “almost everything we do [...] is mediated by computers” [45, p. 19]. By using modern communication technologies, e.g. video conferencing, computers bring us nearer to the primary art of communication: The face-to-face communication. By the omnipresence of these technologies we get more and more used to them and the computerization of almost all areas of live becomes more and more invisible.

While some scholars like Ferris proposed that the definition of computers in CMC should be broad enough to include Computer Supported Cooperative Work (CSCW) and Computer Assisted Learning (CAL) [51], the general position nowadays is to focus on the communication technology and not to centralize various fields of research [45].

## Classification

Because CMC covers a wide range of communication options, a very common method for classification is to consider the context of use of a CMC system. Johansen [52] presented the widely-used CSCW matrix (Computer Supported Cooperative Work), as seen in Figure 3.2, for classifying CSCW systems in 1988. However, the CSCW matrix can be directly applied to categorize CMC systems. It uses two dimensions of work for classifying collaboration: First whether the *place* of work is the same or geographically distributed, and second whether collaboration takes place at the same *time* or asynchronously.

Depending on the parameters *time* and *space* the CSCW matrix, also called time/space matrix, distinguishes the following collaboration modes:

- **Same time / same place:** Collaboration happens by *face to face* interactions using additives like decision rooms, shared tables or single display groupware.
- **Same time / different place:** To bridge the geographical distribution *synchronous remote* interactions, like videoconferencing, real-time groupware or instant messaging are used.
- **Different time / same place:** *Continuous* interactions often rely on team rooms, public displays or pin boards for collaboration.
- **Different time / different place:** *Asynchronous remote* interactions are using techniques like e-mail, online bulletin boards, blogs or version control systems (VCS) for collaborative work.

In the context of this thesis the latter collaboration mode (different time / different place) is the most interesting. Therefore the introduction of various types of CMC systems in the following section will mainly concentrate on representatives of this group.

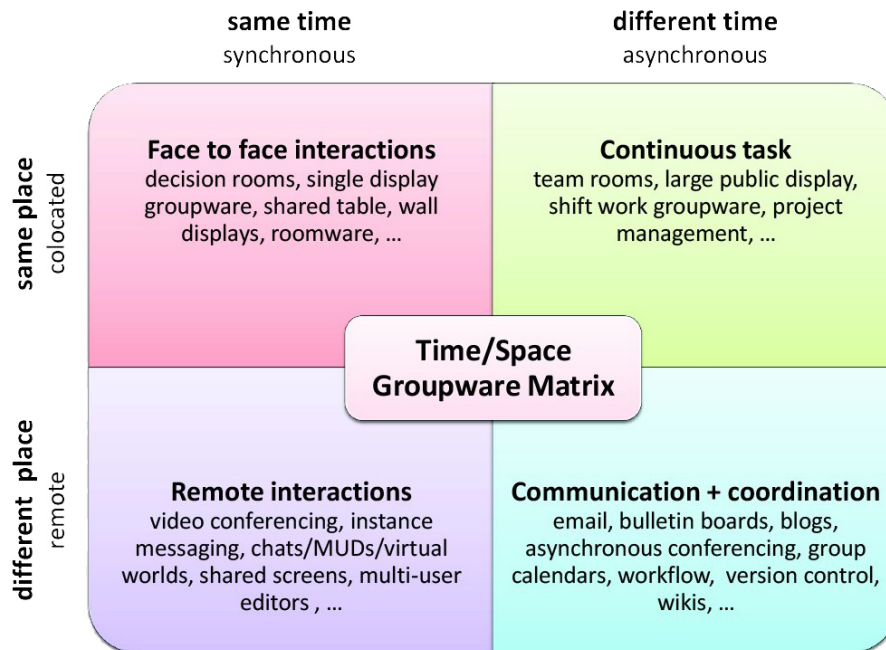


Figure 3.2: The CSCW matrix is used to classify CMC systems. Depending on the parameters *time* (synchronous or asynchronous) and *place* (collocated or distributed) different techniques and CMC systems can be used to facilitate collaboration.

## 3.2 Types of CMC

This chapter outlines the most common and basic CMC systems. Because this thesis primarily deals with e-mail communication, the focus lies on asynchronous remote systems.

### E-mail

E-mail stands for *electronic mail* and is an asynchronous geographically distributed communication form. An e-mail is a digital text message sent from a sender to one or more recipients over a computer network, e.g. the Internet. A typical e-mail conversation is an asynchronous dialog between two persons (one-to-one communication), although all e-mail systems permit multiple recipients nowadays [53]. In contrast to Instant Messaging, e-mails are based on a store-and-forward technique. This does not require the recipient to be online when the sender submits the message. The message is delivered as soon as the recipient is online.

### E-mail Message Format

E-mails use the syntax specified by the *Internet Message Format*, which is defined by the RFC 5322 [54]. According to the *Internet Message Format* an e-mail consists of two major sections:

- The **message header** containing structured meta information about the sender, the recipients and the message.
- The **message body** holding the messages content as unstructured text, like a regular letter's body does.

In detail the *message header* contains informations about the sender of the message (*From* field), the recipient (*To* field), additional recipients (*CC* field, which stands for Carbon Copy), additional hidden recipients (*BCC* field, which stands for Blind Carbon Copy), an e-mail address a reply is sent to (*Reply-To* field), the subject of the message and the date of the message in a structured way. Furthermore the *message header* stores additional meta informations like the *In-Reply-To* field, which indicates if the e-mail is a reply to another message, and the *Content-Type* field, which specifies how the message should be displayed. Some of these fields, like the *To* and the *From* field, are specified to be required when sending an e-mail, while others, e.g. the *CC* and *BCC*, are optional fields and must not be provided by the sender.

In contrast to the structured *message header* the *message body* contains the actual content of the message as unstructured text. Besides plain-text content almost all e-mail clients allow the use of HTML for the message body. HTML e-mails have the ability to add in-line links, to use formatted text and to define a custom layout. The main drawbacks of HTML e-mails are privacy and security concerns because of web bugs and their abuse to spread malicious software. Users sometimes disable HTML e-mails due to these concerns. The *Multipurpose Internet Mail Extensions* solves this issue by allowing a “multipart” message body which contains the content both in plain-text and in HTML format. The user can chose which version should be displayed. To send a “multipart” message the *Content-Type* header field must be set to “multipart/mixed”, which is specified in the RFC 2045 [55]. This *Content-Type* is supported by almost all e-mail clients and is widely-used to transmit e-mails in both formats, as seen in Figure 3.3 on the example of Mozilla Thunderbird.

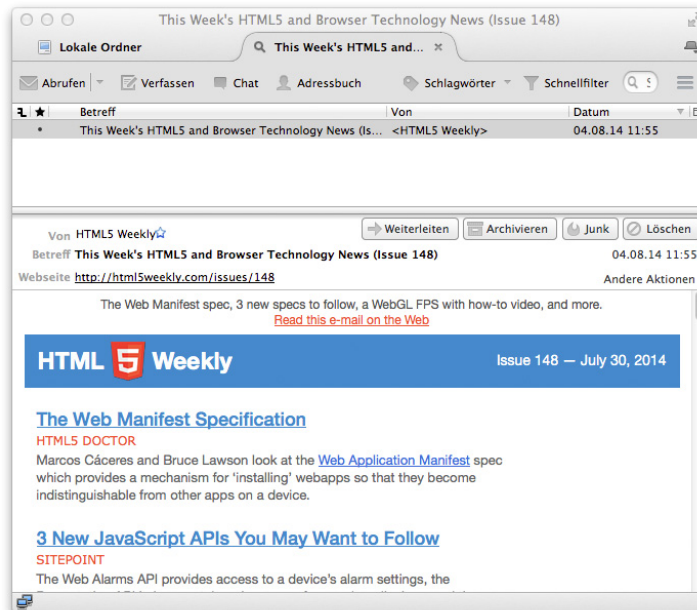
The *Multipurpose Internet Mail Extensions* (MIME) extends the original e-mail protocol to support:

- “Multipart” message bodies (e.g. to send plain-text and HTML e-mails at once).
- Text and header information in different character sets than in ASCII (e.g. UTF-8 and ISO 8859 encoding are supported).
- Binary attachments in e-mails (all kinds of data files can be exchanged).

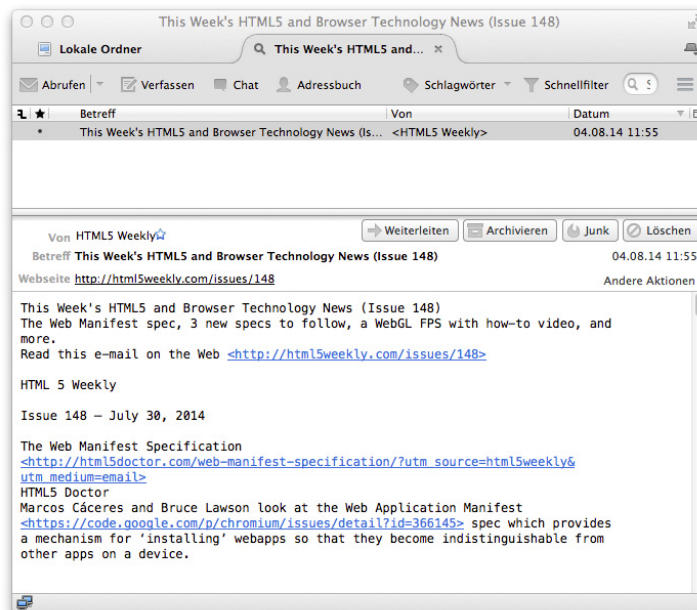
MIME is specified in six different RFC memoranda, the RFC 2045 - 2049 [55], and is jointly responsible for the great success of e-mail.

## History

In the early 1970s Ray Tomlinson, who was working on the ARPANET project for the US Department of Defense, experimented with “SNDMSG” - a system to store messages in files. During these experiments with the inter-system message transport Tomlinson decided to use the



(a) One issue of the well known “HTML5 Weekly” [56] newsletter displayed in HTML mode.



(b) The “HTML5 Weekly” [56] newsletter uses a “multipart” message to provide a plain text version of the HTML content.

Figure 3.3: A “multipart” e-mail displayed both in HTML and in plain text using Mozilla Thunderbird [57], an open source e-mail client.

“@” symbol to separate the users login name from the name of the computer. He sent the first message between two *DEC-10* computers, which were standing next to each other. This feature was included very quickly in the ARPANET and soon e-mail was the most prevalent use of ARPANET [58, 53, 59].

Besides e-mail many other messaging systems, e.g. Mailbox systems, X.25, Novell or BTX were developed. As the ARPANET, and later the Internet, grew these systems became less important and were superseded.

Nowadays e-mail is, next to the World Wide Web, the most important and most widely used service the Internet offers. According to the Email Statistics Report, published by The Radicati Group, the number of registered e-mail accounts will grow from 4.1 billion in 2014 to over 5.2 billion in 2018 [60]. Furthermore the report states that “Email remains the most pervasive form of communication in the business world, while other technologies such as social networking, instant messaging [...] are also taking hold, email remains the most ubiquitous form of business communication” [60, p. 2].

## Instant messaging

In contrast to e-mail instant messaging (IM) is a synchronous form of distributed one-to-one communication. Both the sender and the recipient have to be online at the same time to communicate with each other. Usually only text messages are exchanged, although more advanced IM applications support Voice over IP, video chats and file transfers. The transmitted messages are typically quite short and more casual than e-mails. This probably comes because IM is more often used in private life than e-mail [53, 61].

Depending on the IM protocol, IM applications use either a client-server architecture, where the server forwards the senders message to the receiver, or are built on top of a peer-to-peer architecture, where the messages are transmitted directly from the sender to the receiver (point-to-point).

## History

As networks developed in the late 1980s the messaging protocols spread with the networks. Almost all early IM programs used real-time text. Real-time text does not wait for a user to actively submit a message but transmits messages character by character as soon they are typed. One of the first IM programs was the Unix *talk* command line program using a peer-to-peer connection for transmission. In early versions of *talk*, shown in Figure 3.4, characters from each user were intermingled if both users typed simultaneously. The latter was the case because *talk* did not separate the text from each user. Later versions used a text user interface to separate the messages of each user [62].

GUI based interactive IM clients started their success story in the mid 1990s with *ICQ* (I seek you) and *AIM* (AOL Instant Messenger). Besides these many other companies, e.g. Microsoft, Yahoo! and Ubique, introduced their own proprietary messenger protocols and clients. If users wanted to use more than one of these messaging platforms they had to install and run multiple applications. In 2000 the *Extensible Messaging and Presence Protocol* (XMPP), formerly known as *Jabber*, was introduced to solve this problem. *XMPP* is a messaging protocol

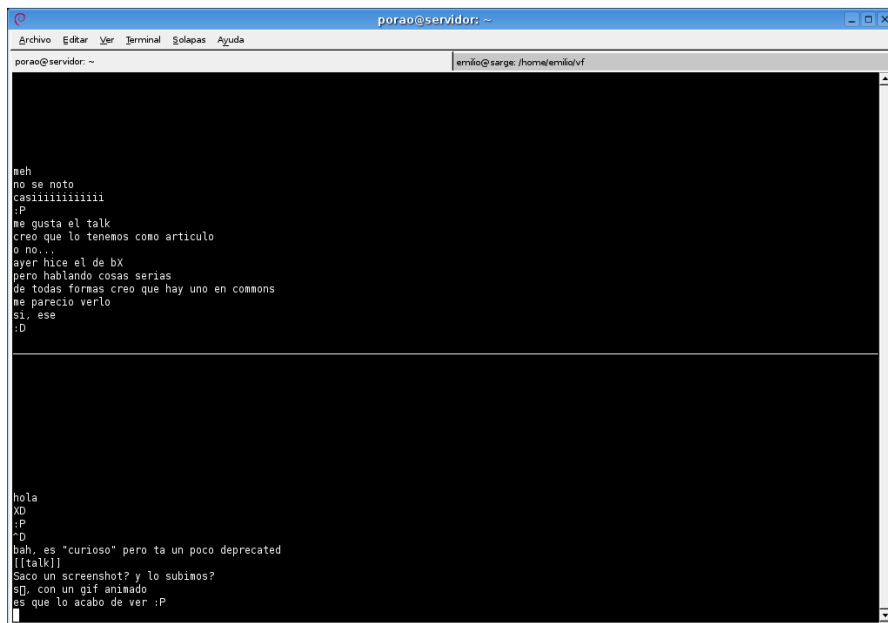


Figure 3.4: The Unix command line instant messaging program *talk* [62].

based on XML acting as gateway for various IM platforms. Therefore there is no need for users to run multiple clients anymore [61].

IM applications are extended consecutively. For instance most IM applications support group communication (one-to-many) nowadays and do not require the receiver to be online at the time the message is sent but use the store-and-forward technique to deliver the message as soon as the recipient starts the IM application. With the rise of smartphones<sup>1</sup> in the mid 2000s mobile instant messaging (MIM) became more and more popular. MIM allows the usage of IM services on mobile devices. Various mobile applications, e.g. the *Blackberry Messenger*, *Skype*, *Threema* and *WhatsApp*, provide MIM features. The functional range is in no way inferior to classic desktop IM applications and covers among other functions the exchange of text, voice and video messages, the transmission of binary data files (e.g. media files) and group conversations. Figure 3.5 shows a current version of “Skype”, an IM client and service by Microsoft which offers many of the latter mentioned features.

## Mailing list

Mailing lists are an asynchronous one-to-many form of communication. They enable the widespread distribution of a message to many Internet users in a very easy way. A mailing list gives a group

<sup>1</sup>Wikipedia describes a smartphone as a phone “with more advanced computing capability and connectivity than basic feature phones” [63]. The fundamental idea behind the smartphone is to combine the features of a basic phone with the functionality of a personal digital assistant (PDA) - a PDA usually provides mobile Internet access and enables the user to do simple tasks while on the way. PDAs are the prototype of nowadays smartphones but are largely considered obsolete with the exceptional popularity of smartphones [64].

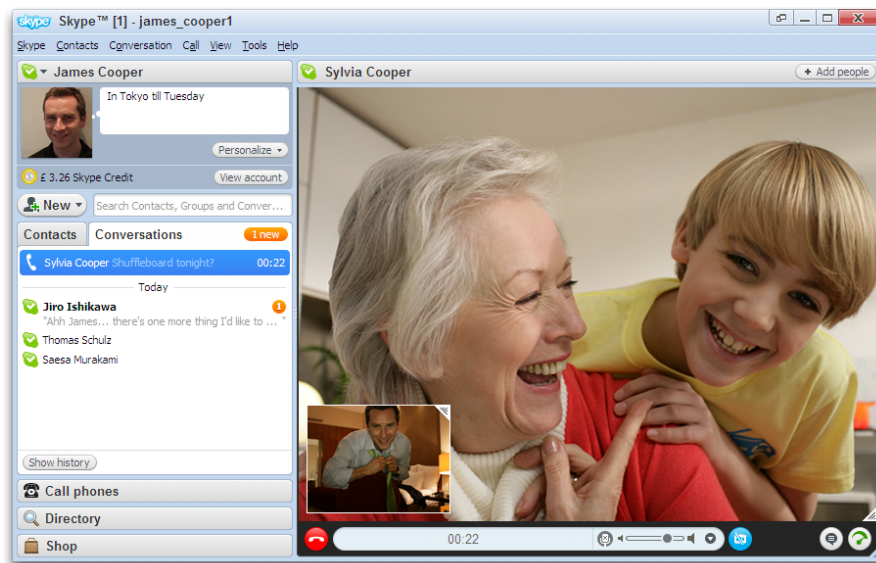


Figure 3.5: The current version of the popular IM client “Skype”. The picture shows the video call feature offered by “Skype” [65].

of people the opportunity to exchange messages without having to know all members of the group. A mailing list can be publicly accessible or only accessible for a closed group of people. Within a mailing list all messages are publicly visible to all members. In contrast to a newsletter the mailing list is multi-directional and lives from the users contributions.

Basically a mailing list is a list of subscribed e-mail addresses with an own e-mail address, called *reflector*. If a message is sent to the reflector e-mail address the message is forwarded to all subscribed users. Therefore all subscribed users can be addressed without knowing their actual e-mail address. Technically a mailing list is a server software like *LISTSERV*, *GNU Mailman*, *Google Groups* or *Yahoo! Groups* which is capable to retrieve and send e-mails. A mailing list software usually provides additional configuration options for administrators like user-depending settings (e.g. changing the authorization for reading and writing posts) and changing the visibility of the mailing list. A big advantage compared to sending an e-mail to multiple recipients is the web archive most mailing list softwares create. The archive stores all messages and mostly provides extended filter and search options [53, 66].

Mailing lists can be moderated or unmoderated. While in unmoderated mailing lists each post is automatically forwarded to the subscribers, each post is more or less restrictive evaluated in a moderated mailing list. Mailing lists are often moderated because of spam. The moderation is done by separate moderators (usually a long time member of the mailing list) or administrators. Besides moderating the discussions on the mailing list the administrators of a moderated mailing list often verify the compliance of the netiquette and have additional privileges like deleting or moving a post and banning or suspending users if their behavior is inappropriate.

## History

Early discussion lists appeared in the early 1980s and found exceptional popularity. In 1986 Eric Thomas introduced *LISTSERV*, which is still one of the most popular mailing list softwares. It provided simple administration tools and therefore helps maintaining mailing lists [53].

Mailing lists can be seen as the prototype of newsgroups and Internet forums. Both are very popular nowadays. The main advantage of mailing lists compared to Internet forums is that messages can be read and written offline. In contrast to a Newsgroup a mailing list can be setup very quickly and no additional account is needed. For participating in newsgroups a Usenet account and a Newsreader<sup>2</sup> is mandatory. Although most Internet users have installed a Newsreader in conjunction with their browser only few users know and use this feature.

## Newsgroups

A newsgroup is usually located within the Usenet and serves as a repository for messages from many users. Newsgroups are an asynchronous one-to-many form of communication and can be seen as public forums. They usually use the Usenet, a distributed Internet discussion system built on top of the UUCP (Unix-to-Unix Copy) protocol. It enables users to read and post messages to newsgroups and therefore is very similar to a mailing list. The main difference to a mailing list is that messages are not forwarded to the users, but have to be retrieved actively. Newsreaders are applications allowing users to connect to the Usenet and are mandatory for using newsgroups. A newsreader works similar to an e-mail client: If it is connected to the Internet and valid credentials are provided the newest posts of the subscribed newsgroups are downloaded and can be read offline. The offline accessibility of posts is the main difference compared to Internet forums, which can be seen as successor of newsgroups [67, 53, 45]. One of the most popular newsreaders is *GRABIT*, which is available for free and can be seen in Figure 3.7.

Technically newsgroups rely on news servers which are usually Usenet servers. A news server hosts selected newsgroups and syncs them at regular intervals with other news servers, as seen in Figure 3.8. A user connects to a news server which offers the requested newsgroup and is able to read and post articles to this newsgroup. The regular sync between news servers guarantees that a message is available on all news servers.

In addition to text messages newsgroups support the exchange of binary data. Posts containing binary data are not allowed in all newsgroups, making it easier for news server administrators to allow or disallow the sync of these traffic intensive newsgroups. Each newsgroup has allocated a certain amount of space for binary data and posts. If the newsgroup runs out of space the oldest posts are deleted. The average time a posts stays in a newsgroup before being deleted is called *retention time* and is usually at least 1000 days for text messages.

Compared to mailing lists very few newsgroups are moderated. Although the creation of a new newsgroup is usually discussed or even has to be approved by a management board, as described in the following section, posting to a newsgroup is usually not moderated. In contrast to the widespread Internet forums newsgroups retained noncommercial and are mostly free of ads.

---

<sup>2</sup>A Newsreader is the application needed to read and write articles on the Usenet. The Newsreader acts as client and connects to a news server to download articles.



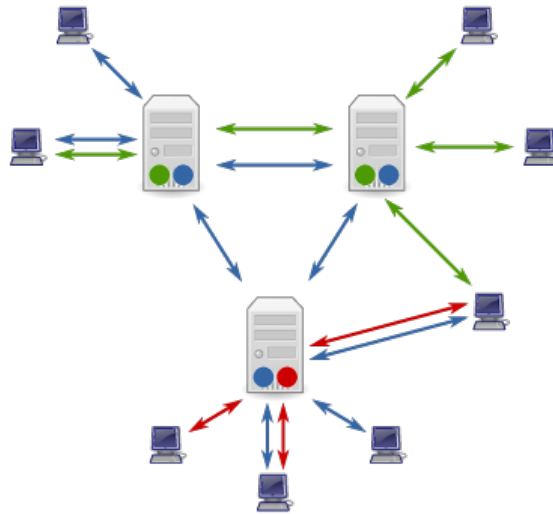


Figure 3.6: This diagram of a newsgroup network shows three news servers hosting different newsgroups (the red, the blue and the green dot). Clients connect to a news server to interact with a newsgroup and news servers sync the newsgroups with each other. [67]

## Hierarchies

A newsgroup is hierarchically organized to facilitate navigation. The hierarchies are separated by using a dot, e.g. *rec.animals.wildlife* is located in the top-level hierarchy *rec*, followed by the hierarchies *animals* and *wildlife*. Today the most commonly known top-level hierarchies are the *Big 8* Usenet hierarchies, formerly known as the *Big 7* because *humanities.\** was added later [67, 68]. They include:

- **comp.\***: All computer-related discussions are located here, e.g. *comp.software*.
- **news.\***: The *news.\** hierarchy is not meant for discussing news events, but deals with the Usenet itself.
- **sci.\***: Science-related topics are discussed here.
- **rec.\***: “rec” is the abbreviation for “recreation and entertainment” and contains groups like games and hobbies.
- **soc.\***: Socialising and discussion of social issues is located here.
- **talk.\***: Controversial topics and discussions of contentious issues such as religion and politics are organized in *talk.\**.
- **misc.\***: All topics which do not fit in any other hierarchy can be found here.
- **humanities.\***: Topics concerning humanities (e.g. literature, philosophy) are discussed here.

The *Big 8* newsgroups are managed by the *Big 8 Management Boards*, which have to approve the creation of new groups. In contrast to the *Big 8* the *alt.\** newsgroup provides more freedom and is not limited to a topic. Anyone can set up a new newsgroup, although new newsgroups usually are discussed in *alt.config* [67].

Besides the *Big 8* and the *alt.\** hierarchies many other hierarchies exist. For instance language and regional specific hierarchies, like *de.\**, *at.\** or *europe.\**, are distributed using news servers.

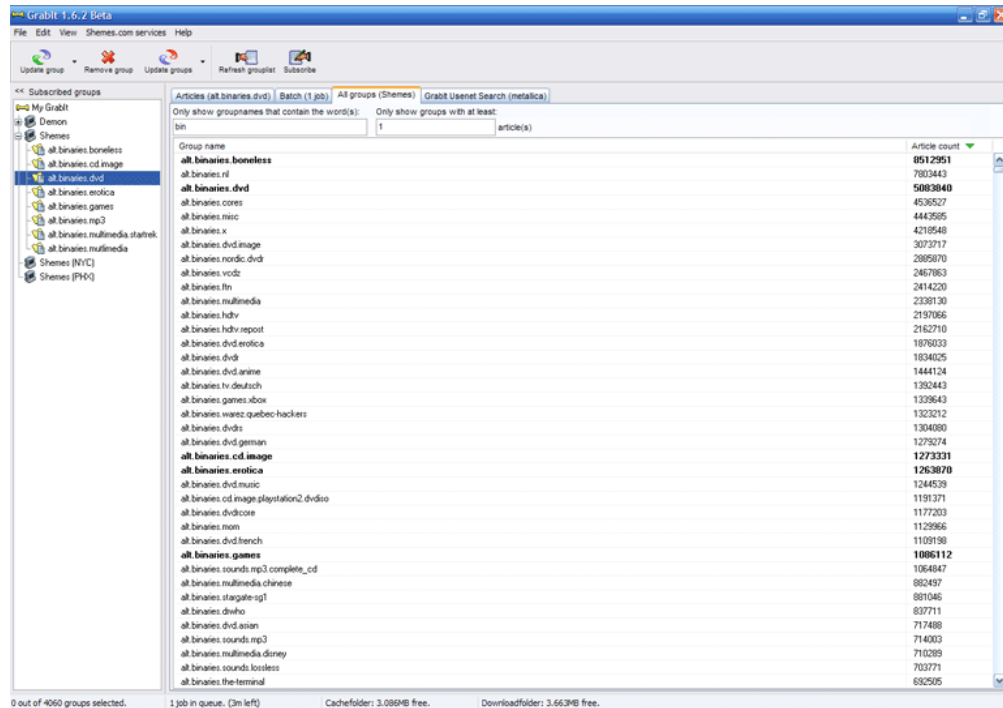


Figure 3.7: The “GRABIT” newsreader is free of charge and provides typical newsreader functionality. This screen shows a filtered list of the available *alt.\** hierarchies [69].

## History

The UUCP protocol, which was originally introduced in 1978 at AT&T Bell Laboratories by Mike Lesk [70], was an important requirement for the development of the Usenet. First experiments with a system, later known as Usenet, developed to replace a local announcement program started in 1979 and were driven by Tom Truscott and Jim Ellis at Duke University. Just one year later, in 1980, Usenet was connected to ARPANET and Michael Horton, who established the connection between Usenet and ARPANET, started to transfer mailing lists into Usenet using the *fa.\** hierarchy. The transition to Usenet newsgroups increased the number of Usenet users enormously and probably is one reason for the following success of newsgroups. In 1986 NNTP (Network News Transfer Protocol) was introduced for distributing Usenet articles over TCP/IP. Since the early 1990s almost all Usenet traffic is transferred using NNTP [67].

Usenet formed the initial Internet community and many important discussions and announcements took place on newsgroups. For instance Tim Berners-Lee announced the launch of the *World Wide Web* in the newsgroup *alt.hypertext*<sup>3</sup> and Linus Torvalds used the *comp.os.minix* newsgroup to introduce *Linux*<sup>4</sup>. Nowadays newsgroups are mostly superseded by Internet forums and more and more news servers are shut down due to increasing costs and low usage.

## Online chat

The main difference compared to instant messaging is that online chats are meant to be a conversation between multiple participants (one-to-many). In contrast instant messaging was mainly designed for communication between a sender and one recipient (one-to-one). As the term “online chat” indicates a chat is primarily an informal conversation in the private surrounding taking place online. Usually the latter is the case because the participants are geographically distributed [53, 71].

Online chats are most commonly organized in *chat rooms*, also called *channels*. Each *chat room* focuses on a specific topic. If users want to participate in an online chat they usually have to register an account and choose a nickname. After login users can join a chat room and participate in the conversation using text messages.

An online chat can be imagined as a public train: Each wagon of the train symbolizes one *chat room*. Passengers sitting in the same wagon can chat with each other, but can not take part on a discussion going on in another wagon - except they move on to the next wagon. From time to time passengers step off and new passengers board in. Everybody who has a ticket can board in and participates more or less in the ongoing conversation.

## History

Before web chats, online chats taking place on a website, became popular applications like *talker* and *IRC* (Internet Relay Chat) were used for online chatting. Both are very early chat programs, use a client-server architecture and require users to connect to a server. To be more exact *IRC* is not a chat application but the application layer protocol used by *IRC* clients to exchange text messages. While most instant messaging systems were developed for one-to-one communication, *IRC* is explicitly designed for group communication and group discussions which take place in separated channels. Although *IRC* is still in active use it lost almost 60% of its users and 50% of its channels between 2003 and 2014 [72].

Since the rise of the World Wide Web in the early 1990s *web chats* become more and more popular. They allow users to communicate in real time using a simple web interface. The main advantage is that no additional software, except a web browser, is needed.

---

<sup>3</sup><https://groups.google.com/d/msg/alt.hypertext/eCTkkOoWTAY/urNMgHnS2gYJ>, accessed 28-December-2014.

<sup>4</sup><https://groups.google.com/d/msg/comp.os.minix/dlNtH7RRrGA/SwRavCzVE7gJ>, accessed 28-December-2014.

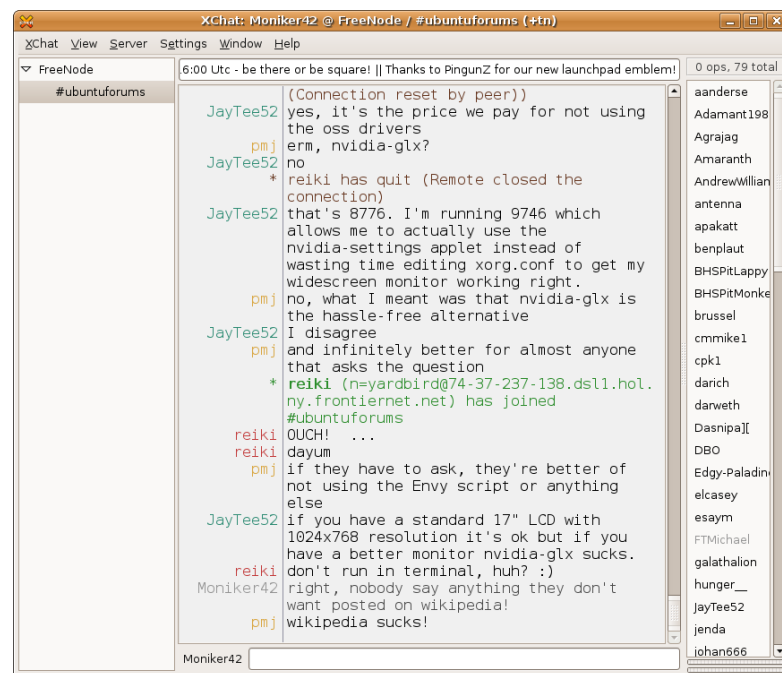


Figure 3.8: An IRC client showing the currently open channels in the left, the chat history of the selected channel in the middle and the logged in users in the right column. Using the text input field on the bottom new messages can be composed [73].

### Instant messaging vs. Online chat

Although both instant messaging and online chat provide a synchronous form of conversation, they are built for different use cases. Instant messages are usually addressed to one or more specific recipients and are more comparable to a phone call: The caller explicitly wants to talk with a specific person. Instant messages usually are stored and forwarded if the recipient was not available at the time of transmission.

In contrast online chats provide open rooms for people interested in the same topic. Users usually do not know each other and the main purpose is to talk about a specific topic. Conversations typically are ongoing even if a user is not logged in and messages a user missed are not stored.

Although instant messaging and online chat are very similar to each other they cover different use cases and are therefore no direct competitors.

### Internet forum

Nowadays a very popular asynchronous form of CMC are Internet forums - also called message boards. Generally speaking Internet forums, also simply called forums or boards, are an online discussion site where people are holding conversations regarding a specific topic. Like in mailing lists and newsgroups the communication concept used by Internet forums is one-to-many:

A sender composes a message and makes it available to many recipients. While early Internet forums were simple versions of existing mailing lists and allowed users to post and reply to messages, nowadays Internet forums offer a variety of features like a privilege based user management, direct messages between members and spam filters. Although there are some free to access forums most Internet forums are only accessible for registered members. While Internet forums, like mailing lists and newsgroups, are also sorted by topic, forums use special terms to describe their structure [53, 74, 75]:

- A **message board**, often simply referred to as **board**, is a rough division into topics and indicates the discussed topics within the board. The term “message board” originates from “bulletin boards” and can refer to a stand-alone Internet forum (e.g. “Informatics Forum”), or a sub-category in an Internet forum (e.g. “PHP board”). In both cases a message board consists either of more specialized boards or an arbitrary number of threads. Within a board threads are usually ordered by the date of the last post in a thread. A common feature of nowadays Internet forums is to make a thread *sticky* which means to pin the thread on top of the list, independently of the last post’s date. This feature is often used for important threads, which rarely receive replies (e.g. threads describing the netiquette, or threads providing general informations).
- Each **thread** covers a very detailed topic within the board. A thread usually has a title describing the topic in detail (e.g. “How to use PHP to print the directory structure”) and an *original post* starting the thread. Within the ongoing discussion the *original post* is often referred to as *OP*. A thread contains multiple posts and usually lists them from the oldest to the newest, although nowadays Internet forums allow the user to order posts according to personal preferences.
- A **post** represents the message a forum member wants to exchange. It is comparable to an e-mail in a mailing list and usually consists of a subject, the message, the user’s details, a date and a time. A post has exactly one predecessor (except it is the threads original post) and, depending on the Internet forum, one or more ancestors. Although technically a post has exactly one predecessor its content often refers to various previous posts. Mostly the content of a post has a maximum character limit ensuring that a single post does not get too long. Usually the author of a post can edit or remove the post for a certain time after its creation.

One of the first forum sites available was Delphi Forums [76]. Delphi Forums started in 1983 and is still in active use today. Over the last two decades about four million members posted 300 million posts in 26 million threads in the Delphi Forums. While Delphi Forums offer boards for almost all conceivable topics (ranging from lifestyle over religion to games), most Internet forums specialize on a specific topic. For instance forums are often used in conjunction with online multiplayer games. Players can share experiences, ask for help or discover hidden

features. Figure 3.9 shows the Internet forum for *World of Warcraft*, a very popular MMORPG<sup>5</sup>, which is in active use and is separated in an numerousness of boards.

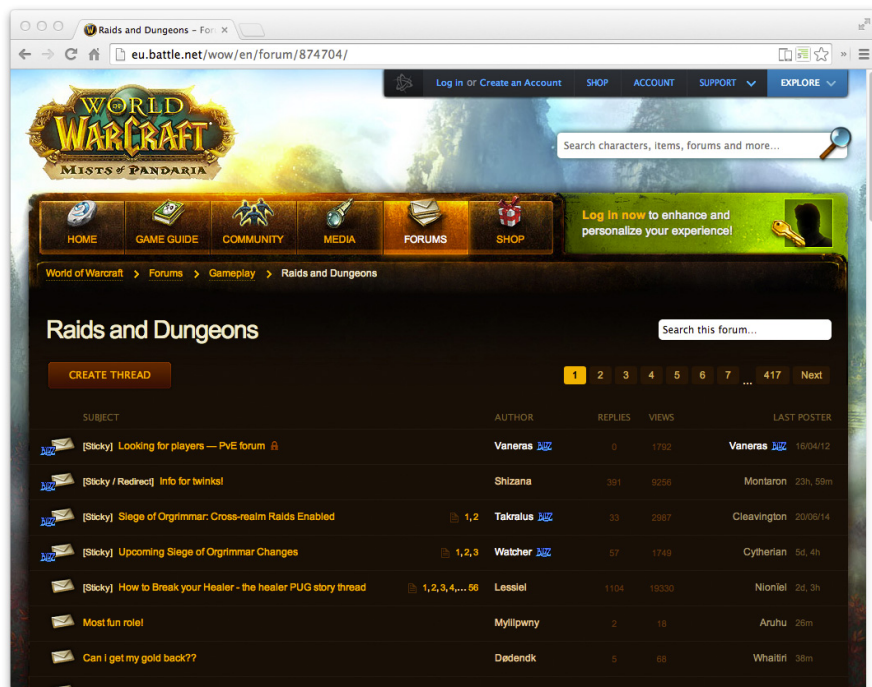


Figure 3.9: The “Raids and Dungeons” board of the *World of Warcraft* Internet forum [78].

## Moderation and administration

Like newsgroups Internet forums are often moderated. In the context of forums this means that some users (moderators) are granted additional privileges and are among other things responsible to moderate discussions, enforcing the compliance of the netiquette and keeping the forum free from spam. Moderators are usually well-known forum members, who actively contributed to the forum for a long time and are known to comply with the netiquette. To fulfill their responsibilities moderators are granted additional privileges like deleting or moving posts and threads, and to block, suspend, ban or warn forum members [74, 67].

While moderators typically follow the discussions and manage the day-to-day business within a forum, administrators are responsible for the technical parts the operation of an Internet forum involves. This includes upgrading the Internet forums software, ensuring the security and integrity of the database, creating and deleting boards or sub-boards and the promotion or demotion of members to moderators or other user roles. In addition administrators decide about the forums appearance, the discussed topics and usually define the netiquette.

<sup>5</sup> A MMORPG (massively multiplayer online role-playing games) is an online role-playing game, where a very large number of players participate at the same time and can interact with each other. *World of Warcraft* (WoW) is one of the most popular MMORPGs and counts about 7,8 million subscribers at the end of the year 2013[77].

## reddit

A rather new form of CMC is *reddit*. reddit was founded in 2005 and is a social news aggregator. Registered users can post links or text messages, which can be *upvoted* (positive feedback is given) or *downvoted* (negative feedback) by other users. The term reddit is a portmanteau of “read” and “edit” and indicates that read posts can be edited by up or downvoting, but also is a play word for “I read it on reddit”. All posts can be commented and comments themselves can be upvoted or downvoted by users. Postings are categorized into areas of interest, called *subreddits*. Users can create new subreddits about any topic of their choice. All users interested in the new topic can subscribe to the subreddit. Messages from subscribed subreddits are added to the users front page. A subreddit is always prefixed with “/r/” and thereby easily recognized as a subreddit, regardless of the context it is used in.

Each subreddit has its own front page where all entries are listed ordered by their popularity. The front page of the reddit site itself, as seen in Figure 3.11, shows an aggregation of the most popular entries of all subscribed subreddits. The popularity of a post is calculated using an algorithm taking the *up and downvotes*, the *age of the post* and the users *karma* into account. A users *karma* indicates the reputation within the reddit community and is mainly calculated on the basis of the up and downvotes of the users posts. The users karma is divided into “link-karma”, the karma a user earned for posts containing a link, and “comment-karma”, the karma awarded for the users comments. So called self-posts which only contain text and do not refer to an external source are not taken into account at all [79].

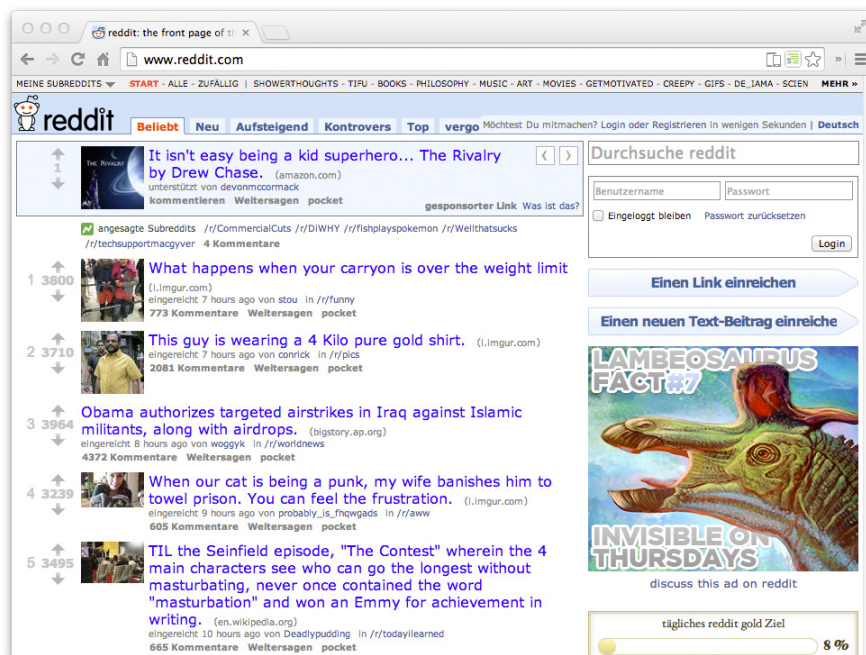


Figure 3.10: The front page of reddit showing the top-rated postings of all subscribed subreddits [80].

Because of the very dynamic algorithm for calculating a posts page rank many dozens of entries appear on the front page throughout the day. This is probably one main reason for reddit's success. According to *reddits about page* [81] reddit had more than 115 million unique visitors in July 2014 with more than three million logged in users a day, who contributed to almost eight thousand subreddits and up or downvoted posts more than 22 million times. According to *reddit metrics*<sup>6</sup>, an independent tracking tool for reddit there are more than 460 thousand subreddits and each day about 500 new subreddits are created. Each of the top nine subreddits has more than six million subscribers.

The most popular subreddits include “/r/funny”, “/r/pics”, “/r/worldnews” and “/r/Music”. A special case is the very famous “/r/IAmA” (“I Am A”) subreddit, where the slogan is *AMA* (Ask Me Anything). A user may prompt others to ask any question by posting an *AMA*, or *AMAA* (Ask Me Almost Anything) entry. Usually these entries provide additional informations about the user like “I am musician, performance artist, blogger, writer, street performer and weirdo Amanda Palmer. AMA.”<sup>7</sup>, and a proof to verify the users identity. The latter is usually achieved by posting a photo. The “/r/IAmA” subreddit uses reddit's comment function for both questions and answers and became famous as more and more celebrities including *Barack Obama*<sup>8</sup>, *Madonna*<sup>9</sup>, *Bill Gates*<sup>10</sup> and many more participated and posted an *AMA* prompt. *Barack Obama* initiated one of the most popular *AMAs* starting with “I am Barack Obama, President of the United States – AMA”<sup>8</sup>. In a very short time more than 13 thousand comments were posted resulting in a total black-out of reddit [82].

It is not uncommon that a single post on reddit, especially in the very popular subreddits, has several thousand comments. To manage this amount of comments reddit provides various sorting options for users (e.g. “Best”, “Hot”, “New” and “Controversial”). Per default reddit uses the “Best” option which, similar to the algorithm for ordering entries on the front-page orders the comments depending on their up and downvotes. To be exact reddit uses the Wilson score interval to calculate the most probable fraction of positive ratings and uses this for ordering, as explained by Even Miller [83].

The open nature and the ability to use reddit for both very popular and niche topics created one of the largest Internet communities. A main factor of success definitely is the used social rating and page-rank algorithm. The latter is a good way to supply the users with personally relevant and interesting updates. In addition the variety and openness of the covered topics created a very diverse and extensible community. If a topic a user is interested in is not already covered on reddit it is easy to create a new subreddit for it.

---

<sup>6</sup><http://redditmetrics.com/>

<sup>7</sup>[http://www.reddit.com/r/IAmA/comments/1l75br/i\\_am\\_musician\\_performance\\_artist\\_blogger\\_writer/](http://www.reddit.com/r/IAmA/comments/1l75br/i_am_musician_performance_artist_blogger_writer/)

<sup>8</sup>[http://www.reddit.com/r/IAmA/comments/z1c9z/i\\_am\\_barack\\_obama\\_president\\_of\\_the\\_united\\_states/sort=new](http://www.reddit.com/r/IAmA/comments/z1c9z/i_am_barack_obama_president_of_the_united_states/sort=new)

<sup>9</sup>[http://www.reddit.com/r/IAmA/comments/1mj1yl/amaa\\_ask\\_madonna\\_almost\\_anything/](http://www.reddit.com/r/IAmA/comments/1mj1yl/amaa_ask_madonna_almost_anything/)

<sup>10</sup>[http://www.reddit.com/r/IAmA/comments/18bhme/im\\_bill\\_gates\\_cochair\\_of\\_the\\_bill\\_melinda\\_gates/](http://www.reddit.com/r/IAmA/comments/18bhme/im_bill_gates_cochair_of_the_bill_melinda_gates/)



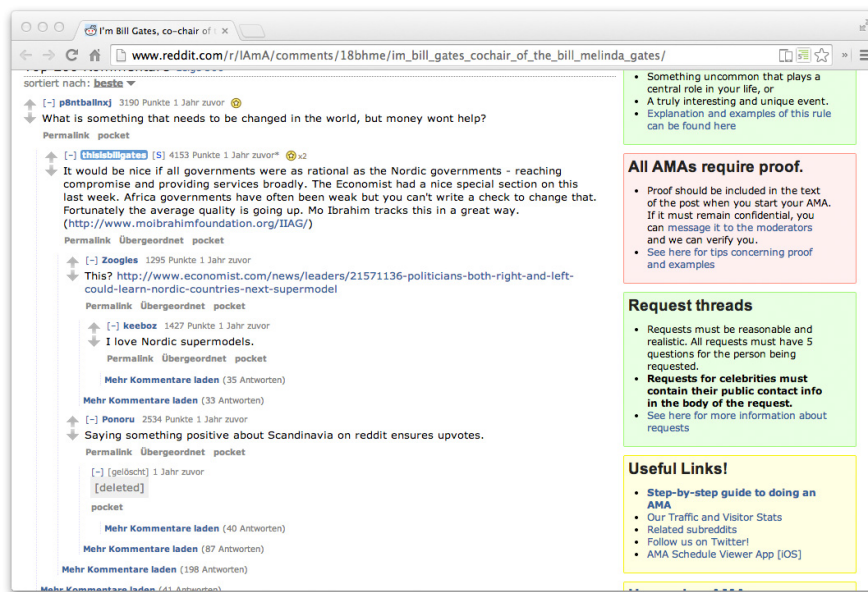


Figure 3.11: Reddits comment functionality using the the example of the famous “/r/IAmA” subreddit. The picture shows the AMA prompted by Bill Gates, which received more than 25 thousand comments [80].

### 3.3 Barriers to Entry

“Working with new technologies is often difficult, simply because they are new, and because individual routines and social routines have to be established in using them.” (Bromme et al. 2005, [84, p. 13])

As Bromme et al. state [84, p. 13] new technologies are usually adopted slowly. The causes for the slow adoption, especially in the field of CMC, are manifold and are not easy to subsume. Besides that most people have confirmed habits they don’t easily give up, the social setting plays an important role. For instance someone is more amenable to try a new technology if friends already use it. This behavior can be observed in social networks, e.g. Facebook or Twitter, but also in other types of CMC. Especially synchronous communication types like instant messaging and online chats require the participants to use the same software. This is the case because the various softwares protocols are usually incompatible to each other. More details about instant messaging can be found in section 3.2.

Especially when focusing on the field of CMC the social setting is very important. Although, or actually because, the variety of online communication software seems to be immeasurable, people tend to use the same tools over a long period. Apart from that, various other more or less obvious technical reasons cause a slow adoption of a new technology, respectively a new type of CMC system. For example the user interface should be usable intuitively, i.e. an average computer user should be able to understand an application without instructions. According to Nielsen this includes, amongst others, the *visibility of the system status*, the *match between*

*system and the real world, consistency and standards* and the *error prevention*. More guidelines can be found in Nielsen's "10 Usability Heuristics for User Interface Design" [85].

Within the various CMC types described in Section 3.2 the inhibition threshold for its usage differs. For instance the usage of e-mail is naturally nowadays. On the one hand this is because e-mail is very similar to mail (a "letter" is exchanged between a sender and a recipient) and therefore easily understandable. On the other hand e-mail is besides the World Wide Web (WWW) the most popular Internet service and required to sign up for most Internet services, including social networks and instant messaging [60, 86]. The success of e-mail lies in its simplicity. Most e-mail clients are very simple to operate and therefore accessible to almost all people capable of using a computer. Another reason for the success of e-mail is that both e-mail accounts, which provide a unique e-mail address and an inbox for the user's e-mails, and e-mail clients, the optional software which stores the user's e-mails for offline access, are available for free.

In comparison to e-mail newsgroups have a rather high barrier to entry. Additional software is required to access newsgroups and although the hierarchical structure facilitates the navigation newcomers are often overwhelmed by the numerousness of hierarchies and sub-hierarchies. In addition access to newsgroups is usually not available for free but a monthly fee is mostly inevitable.

The barrier to entry is lower for Internet forums than it is for newsgroups. On the one hand because Internet forums usually are available for free and do not require specific software, except an Internet browser. On the other hand because Internet forums are mostly specialized on a specific topic and therefore easier to navigate. Furthermore posts within an Internet forum are often readable without being registered. The latter is particularly interesting because Shirky states that "the general form of a power law distribution appears in social settings when some set of items - users, pictures, tags - is ranked by frequency of occurrence." [87, p. 125]. Or when using the *Pareto principle*<sup>11</sup> to describe this observation: 20 percent of the users are responsible for 80 percent of the content. As seen in Figure 3.12 the *Pareto principle* uses a power law distribution for visualization.

This observation indicates that there is a *core* of users (about 20 percent) who actively post to the ongoing discussion, and a *long tail*, as called by the power law, of users (the remaining 80 percent) who participate occasionally, or not at all, but follow the discussion. According to Shirky [87] this behavior can be observed in Internet forums, mailing lists and other online discussion systems.

reddit is also known to have many "lurkers". "Lurkers" are users who follow a discussion but do not actively participate. Probably this is because most parts of reddit are accessible without registration. Regardless this openness the barrier to entry reddit is pretty high. Although people used to the Internet are accustomed to similar rating algorithms, reddit's page-rank algorithm can be confusing to new users. The latter is the case because entries change their position very often and usually link to an external page. In addition the variety of covered topics seems to be

---

<sup>11</sup> According to the *Pareto principle*, also known as the *80-20 rule*, 80 percent of the consequences come from 20 percent of the causes. The *Pareto principle* is used to describe social, scientific, and many other types of observable phenomena. For instance in business "80 percent of your sales come from 20 percent of your clients", or in computer science "20 percent of the code has 80 percent of the errors" are common rules of thumb [88].

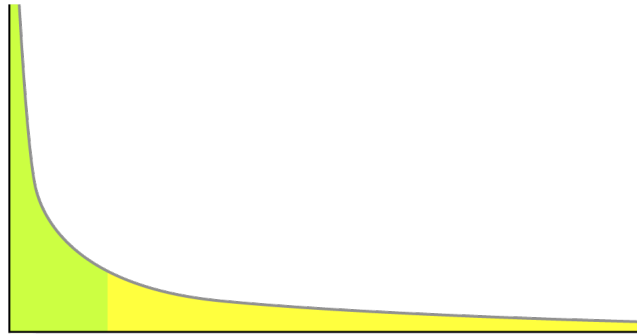


Figure 3.12: The power-law graph visualizing the *80-20 rule* shows a long tail to the right covering only 20 percent of the mass and a small region on the left covering 80 percent.

unmanageable at first glance.

Nevertheless reddit grows from day to day and especially the diversity and openness encourages users to join. reddit fosters a responsive and respectful interaction with each other and supports a feeling of solidarity. reddit's "karma" ranking, which expresses the users reputation within the community, encourages users to contribute constructively and reduces the amount of unnecessarily provocative posts.

All in all e-mail has the lowest barrier to entry for various reasons. Particularly in comparison with other types of CMC e-mail convinces by its simplicity and ease to use. The simplicity of e-mail is important for new users but also results in some drawbacks:

- Conversations can not be categorized like in newsgroups or Internet Forums.
- Most e-mail clients present a conversation only as a structured list and lack the possibility to visualize the connections between the messages.
- In long e-mail conversations with multiple participants the overview is lost very quickly.
- Only text messages (with an optional attachment) are supported.

After evaluating the context of the application to be developed in the following chapter, chapter 5 will present an approach to reduce the drawbacks of e-mail conversation while keeping the barrier to entry as low as possible.



# Context Evaluation

The context of the application to be developed and the used information visualization technique are described in this chapter. First the requirements and goals of the application are discussed. Afterwards the decision-process towards an information visualization technique and the selection of a CMC system are explained.

## 4.1 Application Context

The motivation for a new computer-mediated communication solution arose during conversations about the currently used communication channels at the “Elternverein Wiedner Gymnasium” (the parents’ association for the grammar school called *Wiedner Gymnasium* located in Wieden, Vienna). The “Elternverein Wiedner Gymnasium”, in the following referred to as EV, at the *Wiedner Gymnasium* has very engaged members who communicate regularly with each other. On the one hand this can be attributed to the commitment of the members, on the other hand this is because the *Sir Karl Popper Schule* is integrated in the *Wiedner Gymnasium*. The *Sir Karl Popper Schule* is “a special branch [within the Wiedner Gymnasium] with experimental status for highly-gifted pupils comprising grades 9 to 12”<sup>1</sup> and requires more participation from parents than an average Austrian grammar school. Due to their communication needs and organizational structure the EV of the *Wiedner Gymnasium* is comparable to a voluntary association.

This chapter lists the EV’s currently used communication techniques and inspects their problems. Furthermore the requirements for facilitating communication are identified and their feasibility is investigated. The goal of this chapter is to provide an overview of the current problems and to list the requirements for a new communication solution. The presented information were collected in seven interviews with different members of the EV. Most interviews took place in a cafe or the interview partners’ office.

---

<sup>1</sup>Sir Karl Popper Schule - <https://www.popperschule.at>

## Organizational Structure

The EV of the *Wiedner Gymnasium* is divided in three separated units:

- The **EV of the *Wiedner Gymnasium*** is the most general parents' association and is responsible for organizing the prom, managing the finances and arranging parents-teachers meeting.
- The **EV *Begabungsförderung*** (talent promotion) organizes additional courses for all pupils, e.g. computer courses or touch system courses.
- The **EV *Popperverein*** manages the concerns regarding the pupils of the *Sir Karl Popper Schule*. This unit is more separated because it only affects a fraction of the schools' pupils.

Because the problems and requirements are strongly overlapping the separation will be ignored for the rest of this work. Besides the separation into units the EV contains different working groups. A working group usually consists of three to eight members and is responsible for a specific topic. In addition to the separated working groups the EV has a management board responsible for holding the annual meeting and managing the day-to-day business. Besides others there are working groups for the following areas:

**Events:** The *events* working group organizes and coordinates all kinds of school related events, e.g. the prom or field trips.

**Teaching aids:** This working group is responsible for the approval of additional teaching material and has its own budget for extraordinary expenses for disposal.

**Communication:** As the name says, the *communication* working group is responsible for the internal and external communication of the EV.

Within each class one parent is elected to be "spokesparent". The spokesparent is responsible for forwarding information from the EV to all parents of a class and hands over requests from parents to the EVs' management board or directly to a working group. The EV has about 50 active members including the spokesparents. The EVs managing board has six members and organizes six EV committee meetings each year. Besides the committee meetings, which are open to attend for all parents, the working groups meet each other approximately once a month.

The main communication and the distribution of notes and meeting protocols is operated by the secretary of the management board. The secretary has an overview of the EVs members and administers the member-lists of the working groups. Furthermore the secretary sends information to the classes spokesparent. The spokesparents are responsible for forwarding necessary information to all parents within their class. Therefore each spokesparent manages the contact information of the parents of the class. The information forwarding does not always work as pictured and it happens that some parents do not receive the information at all. Although this does not occur regularly this is a major problem the EV wants to be solved. Parents are displeased because of the lack of information and miss the opportunity to directly communicate with the EV's management board.

As many voluntary associations [8] the EV of the *Wiedner Gymnasium* heavily relies on e-mails for their internal communication. Depending on the actively discussed topics the amount of received messages ranges from several messages a day to a single message in two weeks. The EV does not use software solutions (e.g. mailing lists) for managing the recipient list of e-mails but simply sends e-mails to multiple hand picked recipients. Depending on who should be addressed, e.g. the working group *communication*, different recipients are added to the “To”-field. The latter makes the sender responsible for ensuring that nobody is forgotten. The usage of e-mail as a *one-to-many* communication tool - e-mail is designed for *one-to-one* communication as described in Section 3.2 - causes various problems:

- **A recipient can easily be forgotten** and therefore probably misses important information. Each sender is responsible to address the correct recipients. This can be managed by either creating local recipient groups or by manually adding the recipients. Both cases are error-prone and can displease a forgotten recipient.
- **Conversations are intransparent** for not involved persons. This can be problematic when new people join the EV or an ongoing discussion should be shared with a larger group of people. Another problem related to the transparency is the reporting. Usually the working groups operate independently. Nevertheless the management board wishes to be involved more often into ongoing discussions because it could give good input. The lack of transparency within the EV makes it almost impossible for the management board to keep an overview of ongoing discussions.
- Neither discussions nor their results are **archived or persisted** in a globally accessible location. This makes it impossible to look up previous decisions. In addition new members do not have the opportunity to incorporate. If the discussions are only stored in e-mails new members either have to be introduced manually or have to filter out the important information from a large number of e-mails. In addition the only way to refer to a discussions outcome is to search one’s mailbox and hope that the discussion thread was not deleted.
- Although most e-mail clients, both web based and desktop applications, are capable of displaying related e-mails in a threaded view, as seen in Figure 4.1, **e-mail conversations usually quickly get confusing**. Especially when several people participate in a conversation it is not always obvious which statement an e-mail refers to, or to whom the sender responds.

In addition to e-mails the members of the EV use task specific applications (e.g. Doodle<sup>2</sup> for arranging personals meeting). Over the time the EV has tried out various different communication solutions. For instance an Internet forum was used to connect the parents of a class more closely. The Internet forum was initiated by the EV for a specific class after a conflict arose. While the conflict existed the Internet forum was heavily used and was widely accepted. But as soon as the conflict was settled and the class returned to normality the Internet forum was used

---

<sup>2</sup><http://doodle.com/>, accessed 28-December-2014.

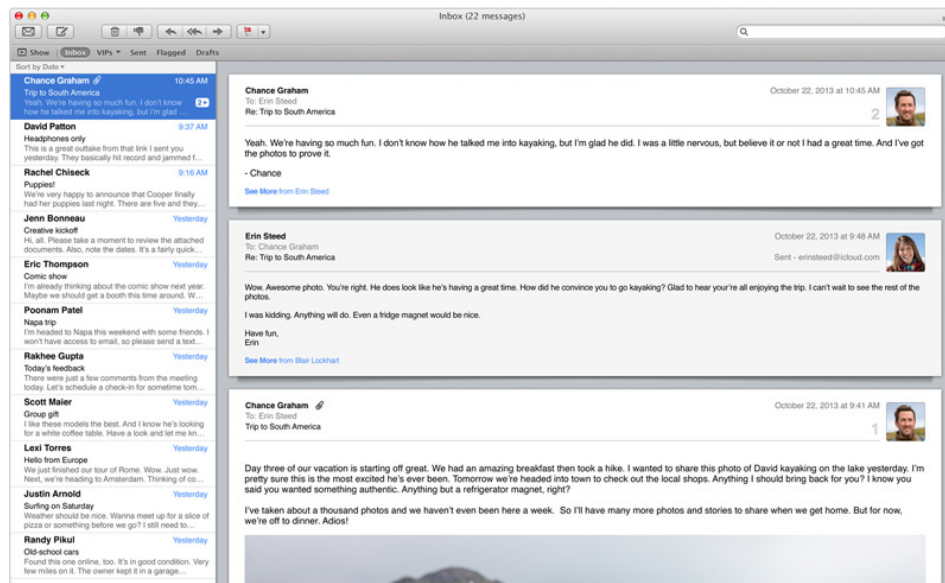


Figure 4.1: Related e-mails are displayed in a conversation view by the example of Apple Mail [89]. The related e-mails are displayed among each other and numbered starting with the first e-mail.

less often - until nobody used it anymore. Although some parents tried to revitalize the Internet forum a major drawback of Internet forums circumvented the success: Users typically have to actively visit the forum to see if there is something new and does not get informed about new postings. Although nowadays Internet forums provide features for notifications via e-mail they are usually not activated per default. The latter is because notification e-mails usually annoy users very quickly - especially when they receive a single e-mail for each new post.

Another approach of the EV was called “Lauffeuer” (which can be translated as “spread like wildfire”). The goal of the Lauffeuer was to inform parents of current events, appointments, the latest changes within the EV and to keep the parents up to date. The Lauffeuer was implemented using e-mail and is comparable to a newsletter but was sent as normal e-mail. Although the Lauffeuer was widely accepted it was not continued. Collecting and preprocessing the needed information is very time consuming and became too much work for a single person.

## Requirements

The previous section covered some major drawbacks of the current communication process within the EV. These drawbacks influence the productivity and the motivation of parents to participate. This section describes the goals and requirements to meet the mentioned drawbacks to improve the communication process.

A very popular request is that parents of a class can compare notes with each other. Especially in the lower grades parents are keen to exchange experiences, opinions and feelings about school related topics with each other. The topics parents are interested in include the school it-



self, the schools' teachers, the pupils homework, the given exams and many other school related topics. Because the topics may be critical regarding a teacher or the school, it is very important that the conversations are confidential and not accessible for teachers and school staff.

Although most conversations usually affect and are interesting for the parents of one specific class, there may be topics that concern the whole level of education or even all lower grade classes. These topics may include across classes study trips, school events or the behavior of a teacher. In summary besides the ability to discuss class related topics with the class' parents it should also be possible to discuss topics regarding a broader audience.

In addition it is very important to keep the barrier to entry as low as possible. Although many people actively use the Internet and have several accounts at Internet services, a new platform always has a barrier to entry. Both people used to the Internet and people not as familiar with the Internet should be able to receive information from the EV and to participate in discussions concerning their children. It should be guaranteed that nobody is excluded. Another argument to keep the entrance barrier as low as possible is that according to a study of the OCG, the Austrian Computer Association, people often overestimate their Internet skills, as shown in Figure 4.2 [90].

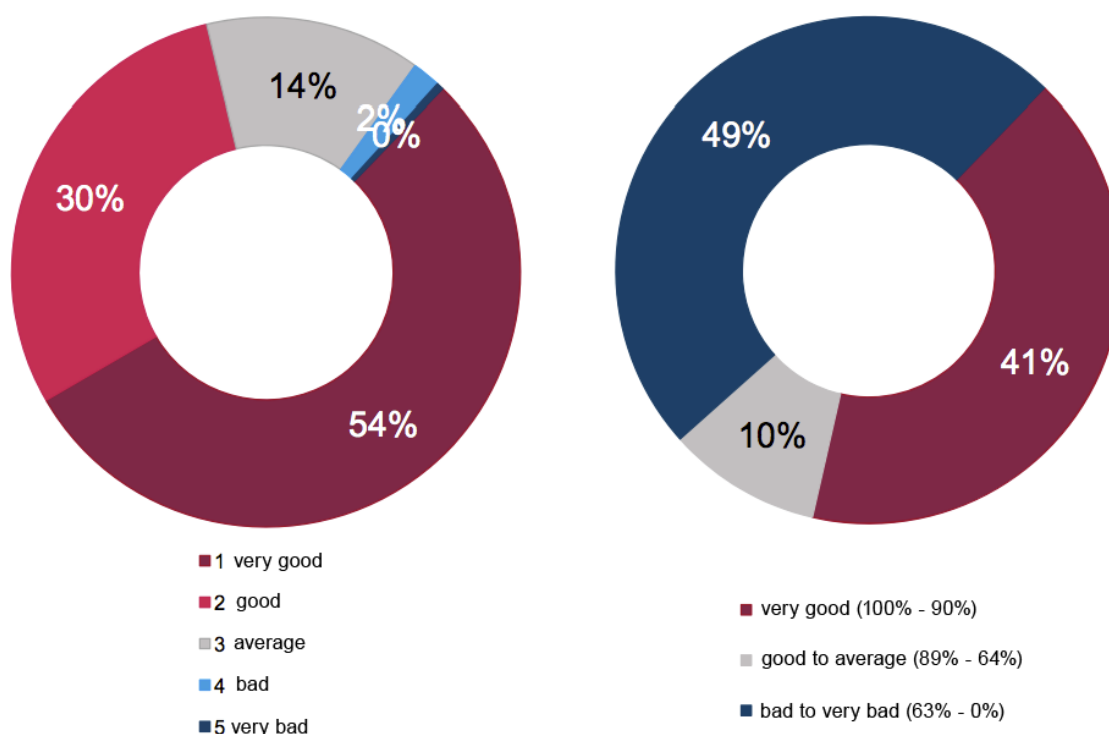


Figure 4.2: The left diagram shows the participants' self-assessment about their Internet skills. While only 16% of the participants estimated their Internet skills to be average or below the test results, shown in the right diagram, reveal that 59% of the participants have average or below Internet skills [90].

Another requirement for a communication solution is, that conversations are persisted and

can be looked up at a later time. This accompanies the creation of a knowledge database, where decisions and their formation can be reproduced. To facilitate this, conversations have to be archived in a globally accessible location. A conversation archive brings several advantages: Discussions and the course of conversations are *transparent* and can easily be *reproduced*, all information is stored in one location and it is easy to involve new people.

The latter implicitly introduces the next requirement: More people should be reached and motivated to participate within the EV. Although the EV of the *Wiedner Gymnasium* is very active compared to other EVs, it is important that as many parents as possible are involved in decision making processes. Decisions have a better foundation when supported by a majority.

Because the distribution of information and announcements using the spokesparents is complex and introduces delays and uncertainties, a further requirement is to unify the information distribution. For instance this could be accomplished by providing a standardized way for spokesparents to manage the contact information of a class (e.g. in a unified platform). This enables the EV (to be more detailed the management board of the EV) to contact an arbitrary large fraction of parents at once and without knowing their personal contact information (e.g. the contact information are stored in the platform but not accessible to the EV). This reduces the delay and ensures that parents receive all necessary information.

Besides these particularly expressed requirements additional requests, which are rarely related to a CMC system, were remarked. E.g. different post types as social media sites provide them could be used for a better categorization of a post. Besides text posts these types could include polls, announcements, invitations or data posts containing digital files. Another request was a shared calendar for each class. Such a calendar could contain appointments regarding a single class, e.g. exam dates or study trips.

In conclusion the following requirements for a new communication system were identified:

- A *very low barrier to entry* to not exclude somebody.
- Separated *areas for parents* in various compositions (e.g. for a single class, for a whole level of education or a custom composition).
- The *archiving of conversations* for the purpose of look-up.
- The unified *contact management* for spokesparents.

The following chapters will describe the design and implementation of a communication system addressing these requirements.

## 4.2 Selection of Technologies

This section describes the decision process for an appropriate CMC system by evaluating the available alternatives with regard to the requirements defined in the previous section. Furthermore an applicable information visualization technique for the chosen CMC system is selected.

## CMC System

After identifying the most important requirements in the previous section, this section evaluates existing CMC applications taking the identified requirements into account. Because of the users' diversity and their different levels of computer skills the most important requirement is to keep the *barrier to entry* as low as possible. While evaluating the various types of CMC (in detail described in section 3.2) it turned out that e-mail has a very low barrier to entry. Especially when compared to newer types of CMC, e.g. Internet forums where a signup process has to be completed, this becomes obvious quickly.

The low barrier to entry is one of the main reasons for the great success of e-mail and results from various reasons: On the one hand e-mail uses the same concept as mail and therefore is easy to understand, on the other hand e-mail is one of the oldest and most popular Internet services. More information about the barriers to enter a new CMC system are given in section 3.3.

Compared to other CMC types, e.g. newsgroups or Internet forums, e-mail has another main advantage: An e-mail is sent to the user while newsgroups and Internet forums usually require the user to visit the Forum, or to search the Newsgroup for updates. Of course using e-mail requires a user to open the e-mail client or to log in at the e-mail providers site to receive new e-mails. Since e-mail is very popular this recurring step is done regularly. Many people, especially when e-mail is required for their work, tend to have the e-mail client opened at any time. Therefore an incoming e-mail is comparable to a notification.

E-mail seems to be the perfect fit for the first requirement but does not meet the requirement to provide *separated areas* for users. These separated areas are meant to encapsulate messages within a group of recipients, as a mailing list does. Although a mailing list works similar (one sender posts a message to the mailing list, which is forwarded to all subscribed users), it only partially solves the problem. The problem of mailing lists is that they do not support "overlapping" messages. E.g. mailing lists are not capable of handling nested groups and a post can not belong to multiple groups. A main requirement for the application to be developed is that users are able to post the same message to multiple groups and to lead a shared discussion across members of multiple groups. The latter is neither supported by other CMC systems like Internet forums or mailing lists, nor is it possible using mailing lists. Posting a message to multiple mailing lists results in separated discussions but does not enable subscribers of different mailing lists to exchange messages.

Furthermore the requirement for *archiving conversations* in a globally accessible location is not supported by e-mails out of the box. E-mail is a user dependent CMC. The latter means that exchanged messages are stored in the user's inbox and can only be accessed by the user. In contrast newsgroups and Internet forums store the messages in a globally accessible space. This space is either publicly available or requires a user to authenticate. Mailing lists often provide an online archive. This usually publicly available archive stores all messages and is searchable. Although e-mails are not meant to be stored in a globally accessible location, it is possible and meaningful in some contexts as mailing lists show.

Although Internet forums and newsgroups fulfill some requirements slightly better than e-mail, e.g. the *archiving of conversations*, e-mail has the advantage of providing a very low barrier to entry. None of the evaluated CMC systems is as easy to understand and widely adopted as e-mail. Because a low barrier to entry is the most important requirement and an application with

a low inhibition threshold is more likely to be adopted the application to be developed relies on e-mails. The drawbacks of using e-mail, e.g. the not available global archive, are compensated by its above mentioned advantages.

## Information Visualization Technique

In order to choose a suitable information visualization technique the characteristics of e-mail conversations are inspected in detail. The focus of this section lies on the visualization of an e-mail conversation with multiple participants, a basic description of e-mails and the used protocols can be found in section 3.2.

### E-mail Thread Visualization

All e-mail conversations are threaded, e.g. they are a chain of messages referring to each other. Generally a thread can be defined as “a collection of individual messages related to each other by the reply function in email” (Kerr 2003, [91, p. 1]). The first message is called the *root message* and is the parent of all following messages. A thread can be visualized using a Tree Diagram: Nodes represent messages and arcs describe their reply-to relationship. Therefore each message within a thread has a *depth* defined by the number of reply-to relationships the message is away from the root message. Amongst others a thread can be visualized using Thread Arcs, Tree Diagrams and a Tree Table [91]. The latter three methods are shown in Figure 4.3 and visualize a thread containing 1 to 6 messages. Kerr argues that visualizing threads provides more context information to the reader of a message and helps navigating within a conversation.

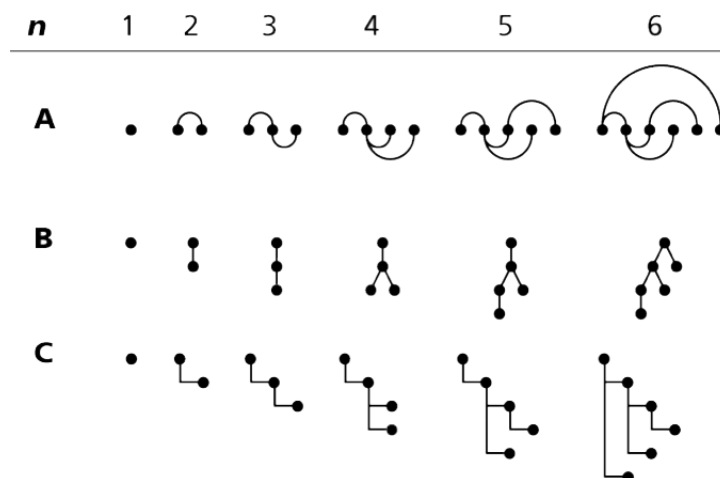


Figure 4.3: Three different types to visualize a thread as it grows from 1 to 6 messages: (A) shows the visualization using Thread Arcs, (B) uses a Tree Diagram and (C) uses a Tree Table. [91].

A thread can either be **narrow** (also called single-threaded) or **bushy** (also called multi-threaded). The former is used to describe a thread where each message has only one direct reply, e.g. when two participants mutually reply to each other. The latter describes a thread having

two or more replies per thread. This usually happens when several participants take part in a discussion and refer to each other. The difference between the two types can easily be seen in a Thread Arcs visualization, as described by Kerr [91] and shown in Figure 4.4.

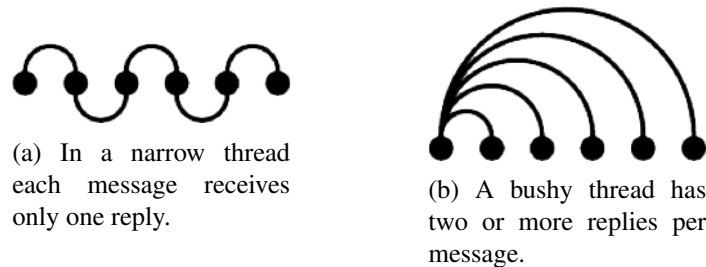


Figure 4.4: A narrow and bushy thread visualized using Thread Arcs [91]. Messages are represented using dots, while their reply-to relationship is represented by arcs.

While narrow threads usually are easy to follow, bushy threads can easily become confusing. Theoretically users can reply to each message within a thread by opening the corresponding message and clicking the reply-to button. This creates the correct reply-to relation within the conversation and integrates the message meaningful in the thread. Practically users tend to either reply to the *root message* or to the *last received message* - regardless what message they refer to. The latter was investigated by Kerr [91].

Both cases have drawbacks and make it difficult to follow the discussion for other users: Replying to the *root message* makes it very hard to classify the message correctly within a thread. No context information about the integration within a thread is given and recipients have to classify the content themselves. This can lead to misunderstandings and can be confusing. Using the *last received message* to reply to a thread is a common practice but poses the same problems as replying to the root message. In addition recipients can be even more confused because when replying to the last message the message usually contains the content of all previous messages as cite.

Therefore the goal of the information visualization technique is to encourage users to build an insightful thread by replying to the correct messages. To reach this goal the structure of an e-mail thread must be more present to the user. If a user understands the structure behind an e-mail thread, it is more likely that he or she replies to the semantically correct message. The resulting thread will probably be even more bushy but will also be easier to follow. To achieve this goal it is necessary to provide more context information about the thread to the user. Ideally this additional information should not disturb users in their intended action and should be unobtrusive. Mainly the intended actions are to read or to write a message. Therefore the context information should be coexisting with the detail view of a message.

### Selecting an Information Visualization Technique

In chapter 2 some information visualization techniques were described. The goal is to find an information visualization technique that provides more context information about the structure of the thread to the user. While concentrating on a single message the user should be able to

keep the overall structure in mind. Thereby the user is encouraged to reply to the corresponding message. The latter helps recipients to correctly classify the message within the conversation.

The first presented technique is *zooming*: *Zooming* provides both a contextual and a detail view. A drawback of *zooming* is that it is not capable of displaying both views simultaneously because time is used as a parameter to separate the views. An abstract type of *zooming* is used in nowadays e-mail clients: While the contextual view provides an overview of the latest received messages, as seen in Figure 4.5, the detail view, shown when an e-mail is selected, shows the selected e-mail in detail, but hides the contextual view. Because a user ideally has both the contextual and the detailed information available at the same time *zooming* is not the best choice.

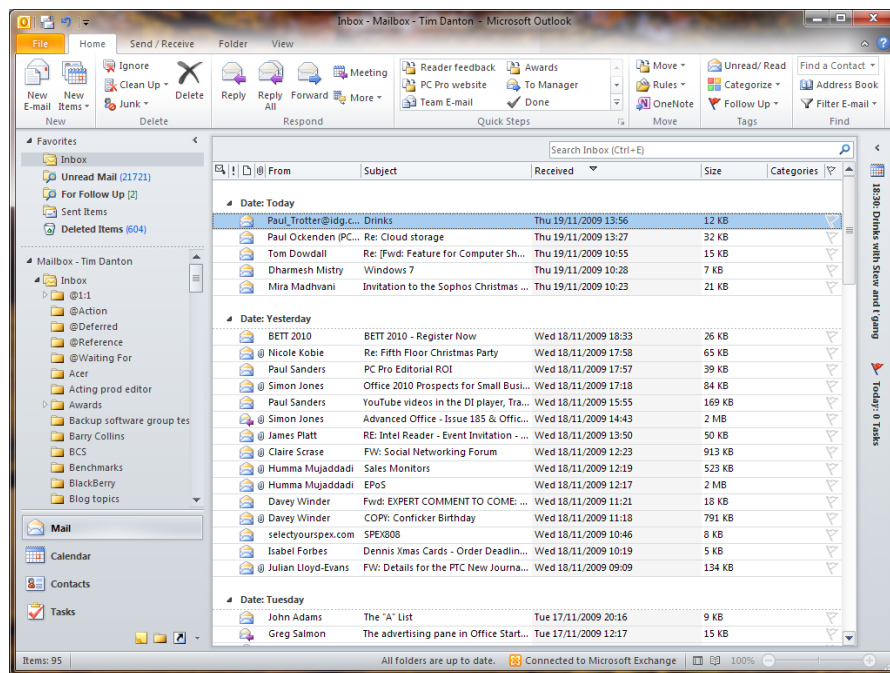


Figure 4.5: A screenshot of Microsoft Outlook 2010 showing an overview of the last received messages. Selecting a message results in a change of the view: The overview (the contextual view) is hidden and the selected e-mail is shown in detail<sup>3</sup>.

Both other presented information visualization techniques, *Overview+Detail* as well as *Focus+Context*, are capable of displaying a contextual and a detail view at the same time. The main difference between them is that *Overview+Detail* uses a separated view to display the overview while *Focus+Context* combines the two views in one. A more detailed description of both techniques can be found in section 3.2.

On the one hand some applications, e.g. mapping applications, benefit heavily from an *Overview+Detail* technique. The provided context information (in the case of mapping applications this usually is a more general view of the selected area) is easy to understand and helps the

<sup>3</sup><http://www.pcpro.co.uk/blogs/2009/11/19/microsoft-outlook-2010-screenshots/>

user to keep oriented. In addition the separated view clarifies the independence of the detailed view (although the views usually are synchronized and affect each other). On the other hand this technique introduces additional complexity - especially in communication systems. Kerr used an *Overview+Detail* technique already back in 2003 to visualize e-mail threads using Thread Arcs [91, 92]. Amongst other features the implemented prototypical e-mail client, called “ReMail”, provided a separated Thread Arc visualization and a list of all participants of a thread, as seen in Figure 4.6. Among the tested persons, who were all working in the field of software engineering, “ReMail” was well adopted and all but one participant would like to have a thread visualization in an e-mail client. In spite of the good test results the Thread Arc visualization introduces a complexity an average users will probably not understand without explanation. Especially when keeping the requirement to *keep the entrance barrier as low as possible* in mind this technique becomes problematic.



Figure 4.6: Screenshot of ReMail, a prototypical implementation of an e-mail client using Thread Arcs for visualizing e-mail threads [91]. Both the e-mail preview (A) and the separated “Thread View” view (B) show a Thread Arc visualization of the current thread. While the Thread Arc visualization within the e-mail shows the region around the currently selected e-mail, the “Thread View” visualizes the complete thread.

In contrast *Focus+Context* techniques do not introduce a separate view but combine the

detail and the contextual view. The most popular *Focus+Context* technique is the *Fisheye View*. The latter shows the currently focused area in detail while visually distorting the surrounded context information. Thus a user can concentrate on a focused area while keeping an overview of the complete data set. As previously stated the detail view, which usually is the detailed view of a single message, is important to easily understand the content of the message. Whereas the context information is helpful for classifying the content correctly.

Because the *Fisheye View*, like any other *Focus+Context* technique, does not divide a data set but shows information “further away” visually distorted, the structure of the whole data set is apparent to the user at any time. Therefore the *Fisheye View* covers all previously stated requirements to visualize an e-mail based CMC system:

- The user can **focus on a single message** by using the *Fishye Views* focus point.
- Users can navigate within the data set using the provided **context information**, which is shown at any time. The *context information* implicitly explains the structure of the thread to the user and facilitates navigating the thread.

The *Fisheye View* heavily relies on visual distortion for reducing the information density within the context area, as described in section 2.3. This approach works well for graphical data sets, but can not be easily applied to text- and content-heavy visualizations, e.g. as the visualization of a thread structure is. The latter is the case because text may not be readable after the distortion or the relation between the messages may not be clearly visible anymore.

A better approach to reduce the information density in text based data sets is *content filtering*: Specific parts of the data set can be shown simplified or can even be hidden because they are not relevant in the context area. For instance the body of an e-mail message can be reduced, the senders e-mail address can be hidden and even the subject can be trimmed. Depending on the target the context region should meet it may be sufficient to show an icon to symbolize that there is more information.

In the context of the application to be developed the context area should give the user the following information: “There are messages replying to the message I am currently reading”. When also taking the thread’s depth into account the needed information is: “The current message has replies which have replies themselves”. The following chapter tries to design an application providing this context information implicitly to the user.



# Design and Implementation

This chapter describes the design and implementation of an actual software application by applying an information visualization technique to the visual structure of a CMC system. First the design of the application, its general structure and the implemented features are described. Afterwards the development process, the first prototype and the productive version are introduced.

## 5.1 Application Design

This chapter describes the design of the application to be developed. As previously stated the application will be based on e-mails and a *Focus+Context* technique is used to facilitate the software interaction. As a working name the application will be called *fisheye* because it applies the *Fisheye View* concept to a CMC system.

### General Structure

Because of previously argued reasons it was decided to build the application on top of e-mail communication. To be more specific the target to be met is that **the application is usable in a meaningful fashion to people only familiar with e-mail communication**. This implies, that all communication takes place via e-mail and per default all messages are sent via e-mail. Using e-mails ensures that the barrier to entry is very low and nobody is excluded.

To fulfill the requirement for making all messages globally accessible the application is implemented as a web application. This has several additional advantages for the users:

- In addition to an Internet browser, which is per default installed on almost all computers nowadays, no native software application is needed to use the application.
- Due to the autonomy of special client software the application can be accessed from all computers having access to the Internet. The user does not depend on his or her personal computer but can use the application from wherever he or she is.

- The application’s code is located in a central spot - the application’s web server. This reduces the costs for changes or updates and makes them immediately available to all users.

The biggest challenge designing the application was to find an appropriate representation for an e-mail thread. As Holzkorn describes in his thesis about an asynchronous online discussion system [93] a linear view of a reply tree is only meaningful when the reply tree of a message is either broad and shallow or very narrow and deep. The former is the case when almost all replies refer to the original post and do not reference other comments. The latter is comparable to a face to face or instant messaging conversation: Each message refers to the directly preceding message. Therefore the thread is extremely narrow but becomes deep very fast. Both types can be represented in a linear view where messages are listed one after the other.

The aim to make the structure of an e-mail thread more apparent to the users can not be met using a linear view. As previously described in section 4.2 an e-mail thread with multiple participants becomes bushy very quickly and can be both broad and deep. Holzkorn suggests the usage of *Miller Columns* where multiple horizontally aligned columns represent different levels within an hierarchy. The most popular usage of *Miller Columns* is the “Column View” in the Mac OS X Finder, shown in Figure 5.1.

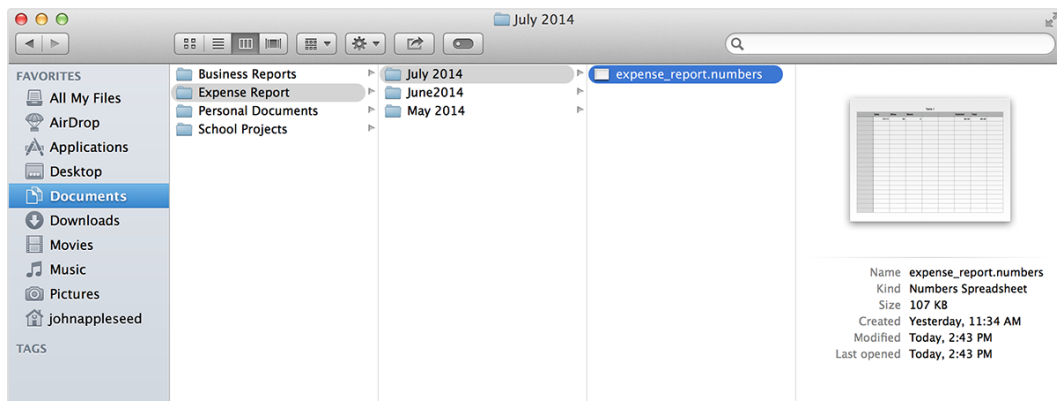


Figure 5.1: The Mac OS X Finder uses *Miller Columns* to facilitate the navigation within the file system. The farther a column is to the left the higher it is in the hierarchy. In addition to the available columns for navigation the outer right column already presents a preview of the selected file.

Using the concept of *Miller Columns* helps to structure the presented content and provides the opportunity to display a thread tree in a meaningful way. In addition to showing multiple hierarchy levels at once it is important that users can concentrate on a single message while keeping the overall structure in mind. Therefore the *Fisheye View* was chosen as an information visualization technique for being able to both emphasize a single message and to show the structure of the thread. When applying the *Fisheye View* technique to the *Miller Columns* the current point of interest can easily be emphasized by resizing the presented columns, as seen in Figure

5.2. The column containing the currently focused message holds the majority of the available space while the columns being further away become smaller and smaller.

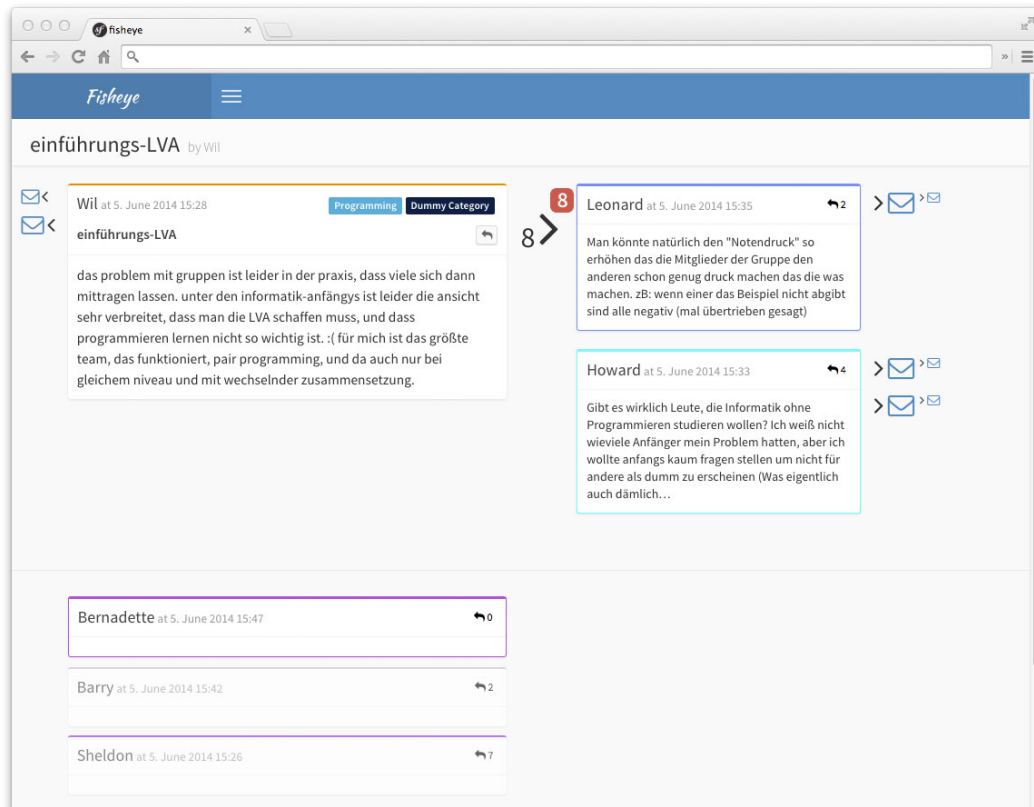


Figure 5.2: The thread view of the *fisheye* application. The outer left column (showing two envelopes) facilitates the navigation to hierarchically higher levels. The second column shows the currently focused message. On the bottom of the page but in the same column, the siblings of the focused message are shown collapsed (the body is hidden). The following column shows the two direct replies to the focused message with an preview of the content. The last column showing small envelopes, symbolizes the following replies. Already read messages are grayed out. The shown conversation evolved from a user experience test.

In addition to the adjustments of the columns width the presented information of a single message varies from column to column. While the focused message is presented in detail (the sender, the date, the subject and the body are shown in full length and size), the content of replies is filtered. For instance the body is not shown completely but expands when the user moves the mouse over it. The sender and the title are shown simplified and the message's subject is only shown when it differs from the original message. Messages in a deeper level are only symbolized using an envelope icon. This helps the user to estimate the number of subsequent replies without having to open the message.

The concept of *Miller Columns* reveals another interesting user interface detail: While the

conventional scrolling direction of nowadays computer applications is vertical the available screen space is more expandable in the horizontal direction. The *Miller Columns* make perfect use of this and applications like the Mac OS X Finder uses horizontal scrolling to increase the number of accessible columns. Although horizontal scrolling becomes more convenient due to modern track pads horizontal scrolling is especially in the context of a Web application not widely accepted nowadays. Due to this consideration the *fisheye* application does not use horizontal scrolling but uses the whole available horizontal space to present as much context information as possible to the user. As known from most Web applications vertical scrolling is used to show long content. Nevertheless the majority of the navigation and context information elements should be reachable without scrolling.

The *fisheye* application uses four *Miller Columns*, which are resized applying the *Fisheye View* concept. In Figure 5.3 the following columns are accented in green:

- Column 1:** The first column enables the user to navigate to hierarchically higher levels within the thread tree. Each envelope represents a level in the hierarchy and links to the according message. The more levels a message is away the smaller the envelope is shown. The shown example indicates that there are (including the root message) two higher levels. The root message is always shown on top. The messages are shown simplified because usually the user already read these messages on the way to the currently focused message. Moving the mouse over an envelope results in a tooltip being shown. This tooltip provides more information about the message to the user.
- Column 2:** The second column shows the currently focused message in detail. The header contains the senders' name and the date of the message in the top left corner. Below this meta information the message's subject is shown. On the right the "tags" the message is posted to and a reply button are shown. Using "tags" the messages can be categorized - a more detailed explanation can be found in the following section. Besides the header information the message's body is shown in full length below the header. On the bottom of *Column 2* the messages on the same hierarchical level as the focused message (the messages' siblings) are shown in a simplified representation. Their body is shown collapsed but unfolds when the user moves the mouse over a message. The hierarchical level of the focused message is given indirectly through the number of envelopes in *Column 1*. Therefore the focused message in Figure 5.3 shows a message on the third level. To give the user an overview of the thread's remaining size the sum of all replies and sub-replies to the focused message are shown on the right. While the first number indicates the total number of replies the superscripted highlighted number reflects the number of unread replies. In the given example the user has not read any reply which results in both numbers being equal.
- Column 3:** The direct replies to the focused message are shown in the third column. Each reply is shown separated and in a simplified way: The "tags" are hidden (because they usually are identical to the focused message ones), the subject is

only shown when it differs from the focused message and the message body is only shown partly. In the top right corner a reply button facilitates the direct reply to a message. In contrast to the reply button of the focused message this reply button contains a number which indicates the number of existing replies to the message. This number, as the reply count in *Column 3*, sums up all replies and sub-replies to the message and helps the user to estimate the size of the subtree. The third column shows the messages one hierarchical level below the focused message shown in *Column 2*. In the example this is equal to the forth hierarchy level.

**Column 4:** The forth column combines all further hierarchy levels. On the right side of each message in *Column 3* the subsequent replies to the according message are shown. As in *Column 1* the messages are simplified to envelopes which reveal more informations when the user moves the mouse over it. To symbolize the structure of the thread even better messages replying directly to a message from *Column 3* are shown among themselves and all subsequent replies are shown abreast. In addition direct replies are shown enlarged. On the example for Figure 5.3 the first message of *Column 3* has two replies, but only one direct reply. The second message has two direct replies and four replies taking the sub-replies into account. Although this structure is not easy to describe, the visual representation of the number of replies gives implicit context information to the user. Usually a user is not that much interested in the actual number of replies but wants to know at a glance whether there are few or many replies. Even if the user does not understand the meaning at once the user recognizes the different number of envelopes and knows there is an difference.

These four columns are used to present both the detail and the contextual view. As already mentioned the *Miller Columns* are a visualization technique for presenting a tree structure. As the Mac OS X Finder and almost all known use cases of *Miller Columns* the *fisheye* application orders the columns from the left to the right: The outer left column shows the highest hierarchical level (the root message), the second column the messages below and so on. Due to the combination of the *Fisheye View* technique and the *Miller Columns* the first and the last column of the *fisheye application*, *Column 1* and *Column 4* in Figure 5.3, are special cases: Both usually contain messages from multiple hierarchy levels. The first column contains all messages on the direct route from the currently focused message to the root message and the last column subsumes all grandchildren and subsequent messages of the focused message. This results in a very efficient way the available horizontal space is used while providing the needed context information to the user.

A very important requirement of the *fisheye* application is the low barrier to entry. Therefore each message is represented by an e-mail. The application itself is capable of parsing and sending e-mail messages. In detail this means, that a message written within the application is forwarded to all recipients as an e-mail. By simply clicking the reply button a recipient can reply to the received message. The message is sent to the *fisheye* application, which parses the e-mail, saves the message on the correct position within the thread tree and forwards it to the

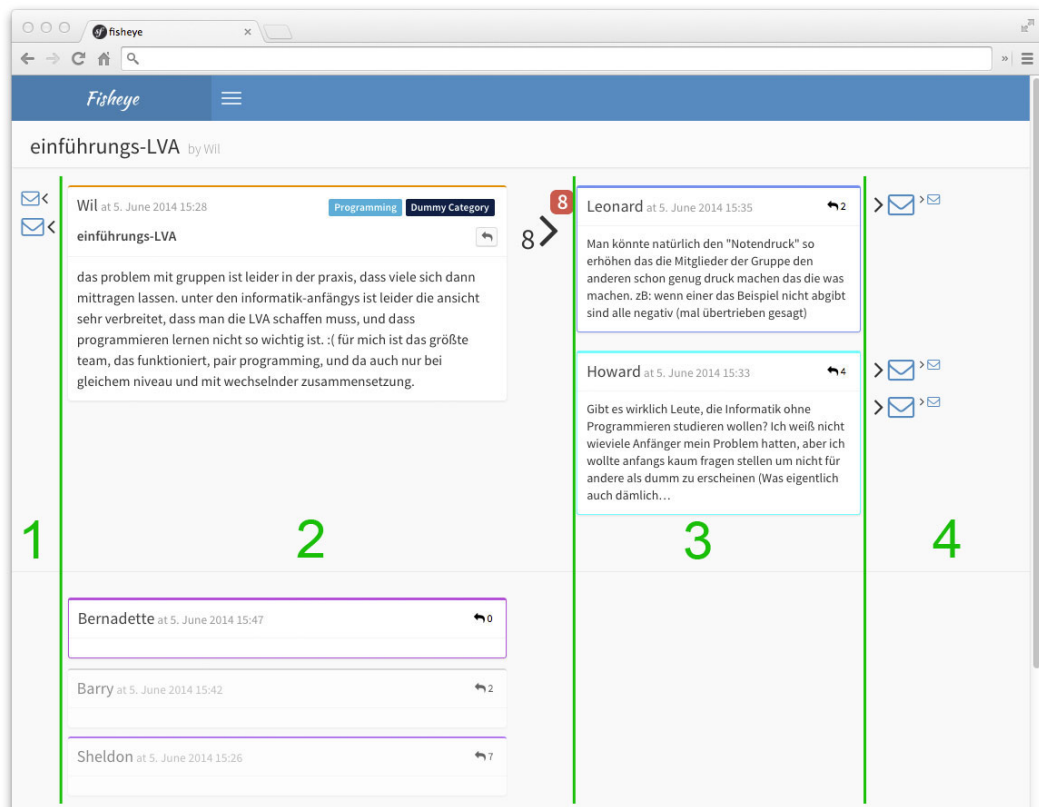


Figure 5.3: The *fisheye* application with highlighted columns. *Column 1* provides the upwards navigation back to the root, *Column 2* shows the currently focused message, *Column 3* shows the direct replies and *Column 4* gives an overview of the following replies.

recipients via e-mail. This functionality is comparable to a mailing list, which provides a similar functionality, but lacks the applications interface. The *fisheye* application offers both the ability to use the additional features provided by the application and also to use the application only via e-mails, therefore not excluding anybody.

### Additional Features

In addition to the general design decisions for the *fisheye* application specific features are outlined here. These features are directly related to the main requirements and should help users to keep the overview while navigating the thread tree.

#### Categorize Messages using Tags

A very important requirement is to offer *separated areas for more or less independent groups of people*. For instance it should be possible to post a message only visible for a very small group of people - e.g. the parents of a class, or for the members of a working group of the EV.

In addition to these very small groups it should be possible to post a message to a more general selection like the parents of a whole level of education (which usually consists of four classes), all members of the EV, or even the parents of all lower grade pupils.

On one side the goal of this categorization is to prevent that users receive messages they are not interested in, on the other side some messages are not meant to be read by all users but are addressed to a subset. The former is important because too many uninteresting messages annoy users and really important messages would disappear in the flood of messages. The latter is important because recipients usually want to know who they are addressing. Knowing who can read a message enables confidential communication and makes it easier to talk about awkward topics, e.g. the behavior of a teacher or other class internal problems. In addition it is very important that messages are not accessible for external persons, especially for teachers and school staff.

To meet this requirement the *fisheye* application uses tags to categorize a thread. In the context of the *fisheye* application the tags of a message are represented by the recipients of a message. In detail this means that the sender of the original message (the root message of the thread) selects a tag in the messages *To* field instead of listing all recipients individually. Users can subscribe to specific tags and afterwards receive messages addressed to these tags. This functionality is similar to mailing lists: A message sent to the mailing lists *reflector* address (the mailing lists e-mail address) is forwarded to all subscribers of the mailing list. While messages can only be sent to one mailing list the *fisheye* application allows a message to contain multiple tags. This enables users from different groups to share a thread without having to subscribe both tags.

A practical use case of this feature could be as follows: *class A* and *class C* of the third grade plan a mutual school excursion. To reduce the costs parents are asked to form traveling communities. Using the *fisheye* application it is possible to start a thread with the tags “Class A” and “Class C”. This results in a shared thread between the subscribers of the tags “Class A” and “Class C”. Although parents of pupils in *class A*, or *class C*, are only subscribed to the tag “Class A”, or “Class C”, both groups can access the shared thread - without having to subscribe to an additional tag.

Like the structure of a school’s classes the used tags are nested. A school contains several classes, which can be divided into a lower grade and an upper grade. Both the lower and the upper grade can be divided further into the single levels of education and each level consists of multiple classes. This structure suggests a tag structure as shown in Figure 5.4: A “school” tag subsumes the lower and the upper grades and therefore contains all classes. The “lower grades” and “upper grades” tags divides the “school” tag into two areas and contain the according levels of education. Each level of education has its own tag and the smallest groups are the individual classes.

In addition to the school related tags there are independent tags for the EV. These tags are nested like the school tags, but only contain tags for each working group and a tag for the management board of the EV.

Although users can post messages to all tags, users are not able to subscribe all tags. For instance users are only able to subscribe tags reserved for the EV if an administrator unlocks them. In addition parents are only allowed to subscribe to the class tags their children are

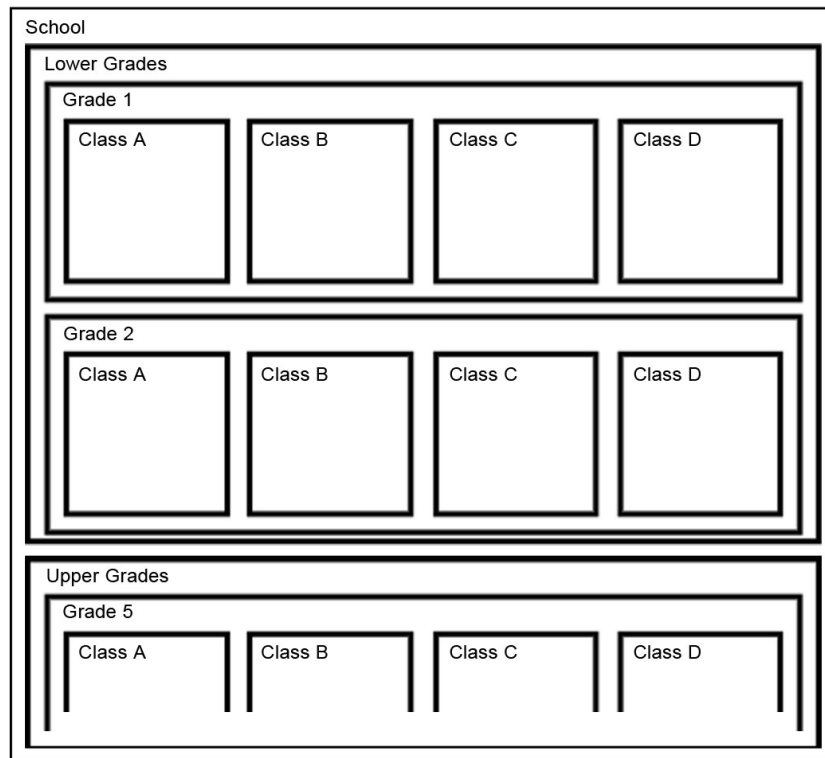


Figure 5.4: The *fisheye* application provides tags to address different recipient groups. The most general tag is the “school” tag, which is divided in an “lower grades” and “upper grades” tag. Within the tags for each level of education each class has its own tag.

visiting. The latter ensures that class related topics are not eavesdropped and discussions stay private within a class.

Because tags categorize threads they are also used for the overview of all threads. As seen in Figure 5.5 the overview page shows three tags on one page. Pagination, located in the top right corner, is used to navigate additional tags. While Internet forums usually order topics vertically the tags are shown in horizontal columns. This has the advantage that threads from various tags are presented on top of the page while ordering them among each other would only show threads from one tag without scrolling.

### Reply without losing Context

A common problem of e-mails is that during composing a reply the context of the initial message is lost. This can happen especially when the initial message is very long or covers various topics. Nowadays e-mail clients try to solve this issue by appending the received message to the bottom of the reply. Therefore users are able to browse the initial message but constantly have to scroll between the initial message and the reply they are composing. The *fisheye* application tries to solve this problem by arranging the initial message and the reply box side by side.



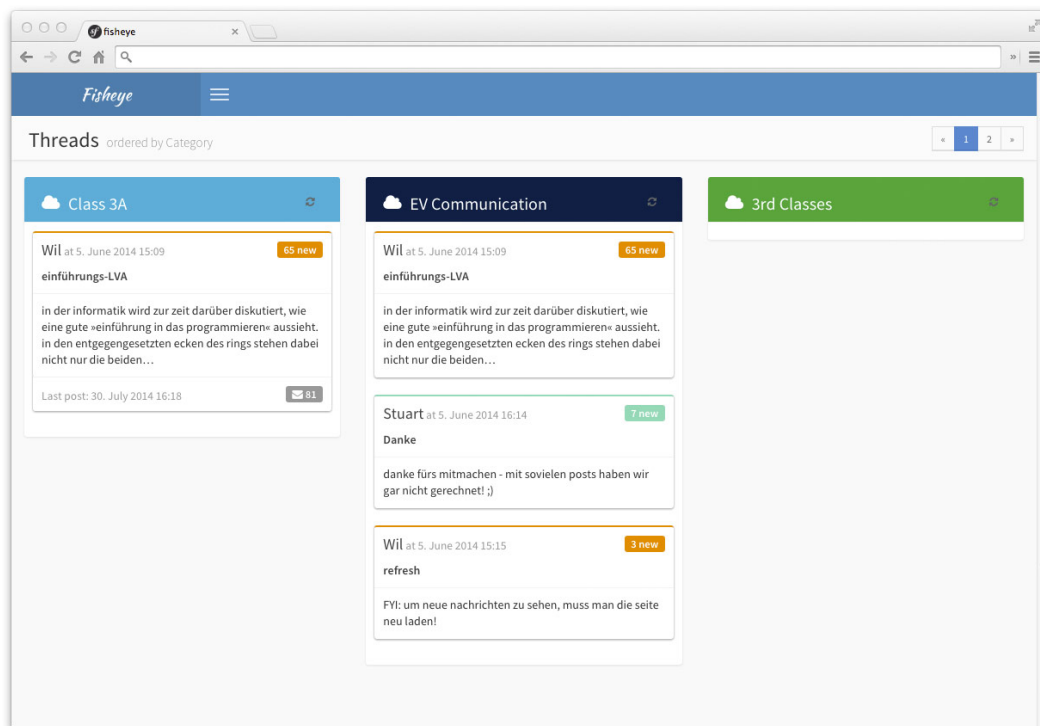


Figure 5.5: The overview page shows the threads categorized by their tags.

The *fisheye* application offers three different reply buttons: The most dominant reply button is located in the top right corner of the currently focused message. Clicking this reply button results in a reply box shown next to the detailed message and the user can reply directly to the focused message, as seen in Figure 5.6. The already available replies are moved to the bottom of the reply box.

The second button enables the user to directly reply to a reply of the focused message and is located in the top right corner of each reply. Replying to a reply results in an adjustment of the columns: The first column is hidden completely, the second and third column are moved to the left and their width is reduced. The fourth column is hidden and replaced by a reply box, as seen in Figure 5.7. Therefore users see the messages of two hierarchy levels in full length and can compose their reply keeping the previous messages in mind. The third option allows a user to directly reply to a sibling of the focused message. By clicking the reply button within a sibling the reply box is shown to the right of the according sibling.

Because the *fisheye* application handles all messages as e-mails, the reply box is graphically oriented on the “Compose New Message” user interface used by nowadays e-mail clients: Vertically aligned input fields for the recipients (*To* field), the subject and the message body have to be filled out before the user can send a message. In contrast to classical e-mail clients the user can not edit the list of recipients and additional text formatting options are shown on top of the body. The former is because the author of the original message has the task and the privilege to determine the list of recipients (by addressing one or more tags), the latter is a feature most users

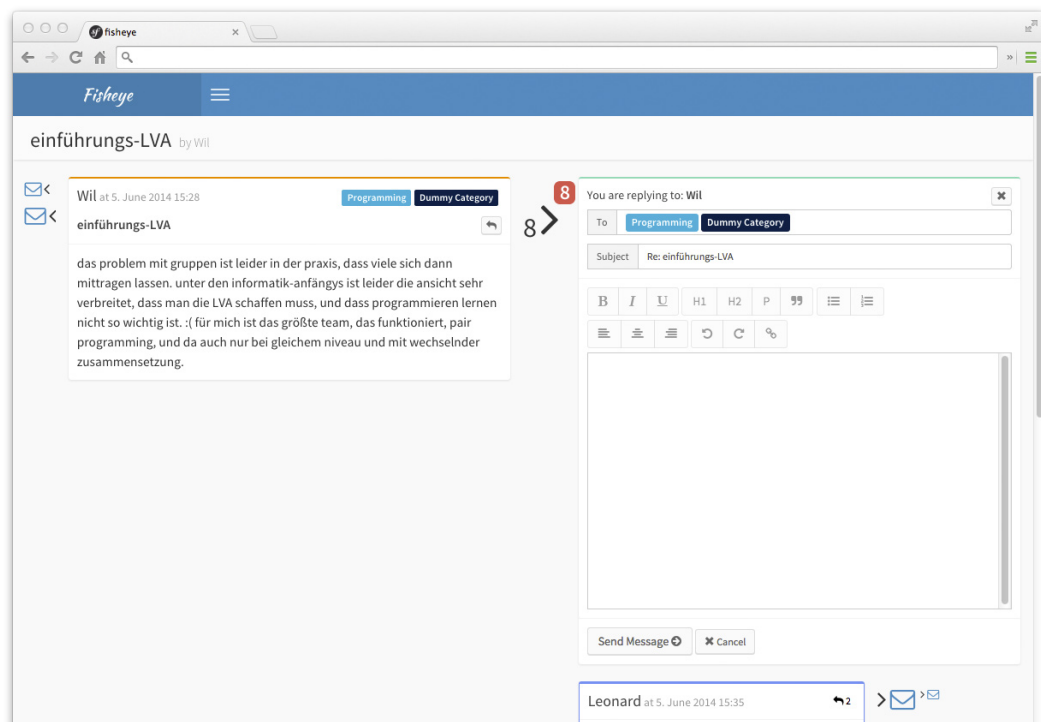


Figure 5.6: When replying to the currently focused message the message and the reply box are shown side by side. Therefore users can compose a reply without losing the context of the focused message.

know from other CMC systems, e.g. Internet forums, and gives the user the chance to enrich their content.

### Message Preview

The symbolization of messages using envelopes (used in the first and the last column) provide context information for the user without introducing the complexity of showing a complete message. This symbolization of messages gives the user an overview of the threads' structure, but lacks the ability to preview a hidden message without opening it. For users a preview can be very helpful to get along within a thread. Although the structure of the thread may be clear, bushy threads can easily become confusing. This is particularly the case when searching for a specific message.

To facilitate navigation within a thread the *fisheye* application shows a preview of a message when the user moves the mouse over an envelope. The preview of a message is implemented as a tooltip and shows the information directly above the envelope. It contains the name of the author, the number of replies and an abstract of the body of the message. The latter is important to classify a message.

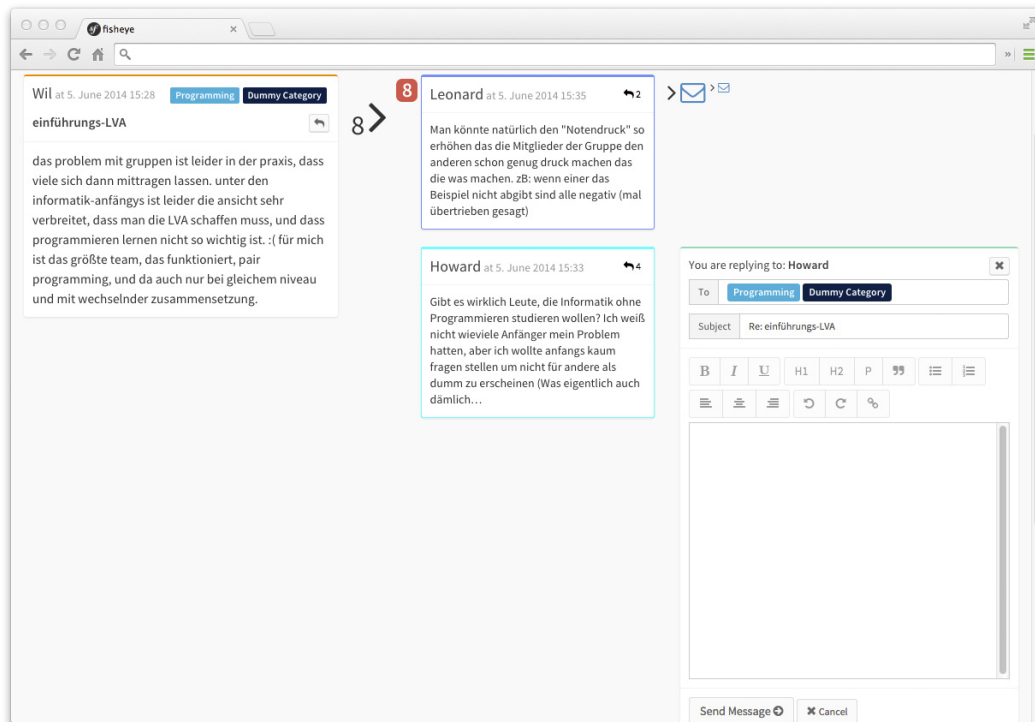


Figure 5.7: Users are able to directly reply to a reply of the currently focused message. The *fisheye* application shows messages of two hierarchies to achieve this: In the outer left column the focused message is shown, followed by the message directly replying and the reply box.

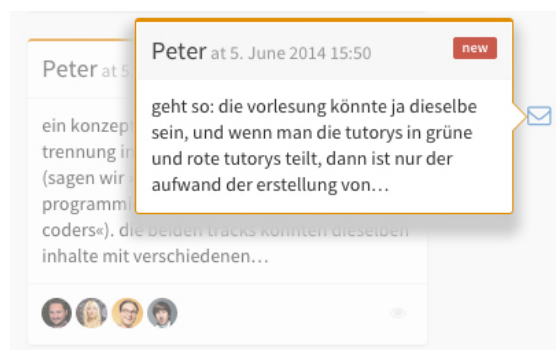
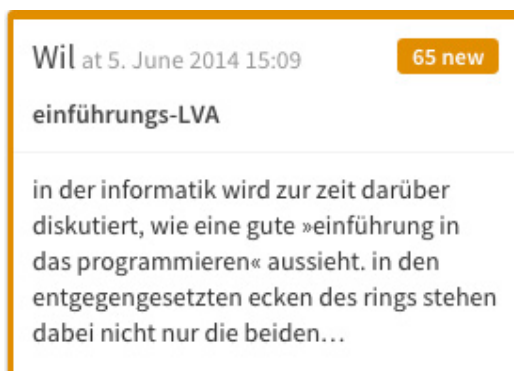


Figure 5.8: The message preview as tooltip in the *fisheye* application.

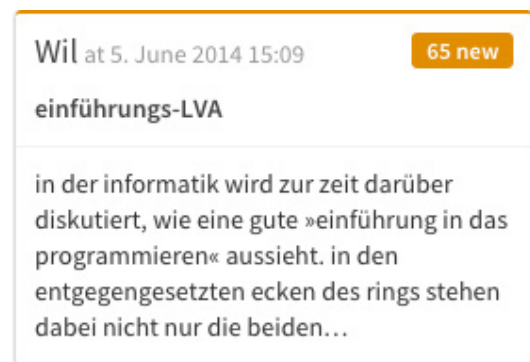
## Message Highlighting

As each tag has its own color, every user has assigned a unique color. This color is used to highlight messages composed by the user. Therefore the author of a message is quickly recognized without even reading the authors' name. In addition to highlighting the messages of the user, threads the user initiated are also highlighted using the users unique color.

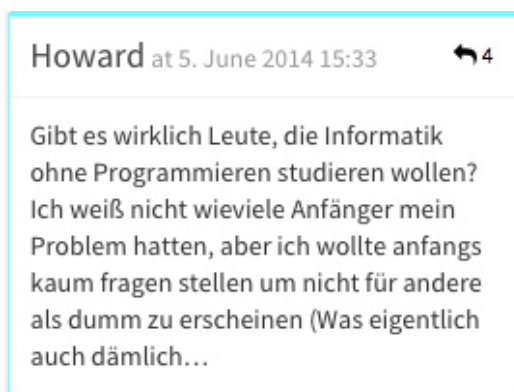
Both read and unread messages (and threads) are marked by a colored border on top of the message. Unread messages and threads are more highlighted by using a surrounding border in the according color, as seen in Figure 5.9. While an already read thread is not shown grayed out already read posts are grayed out to accent new message even more.



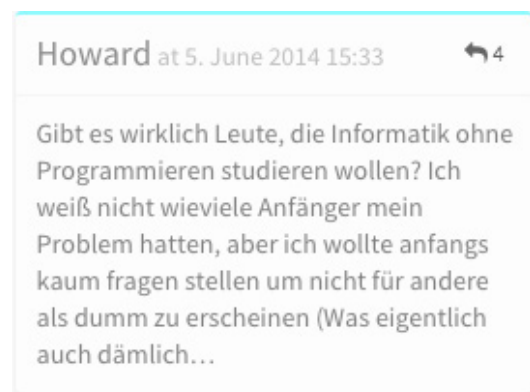
(a) An unread thread is accented using a thick colored border. In addition the number of unread messages is shown in the top right corner.



(b) An already read thread shows the color of the original post's author only on top of the box.



(c) In contrast to the representation of a new thread, new messages have a thinner colored border.



(d) In addition to only showing a border on top of the box, already read messages are grayed out to accent unread messages even more.

Figure 5.9: The different representation modes of messages and threads. While new messages are surrounded with an colored border, already read messages only have a border on top of the message. The used border color is unique for each author.

At the first login each user gets assigned a random color. Because a white background is used in the *fisheye* application very bright colors are excluded. Users are able to change their color in the user settings.

## 5.2 Development Process

This chapter describes the used technologies and the development process from the first prototype to the final application.

### Technical Structure

Both the prototype and the productive version are built using a *LAMP* Stack [94]. *LAMP* is an acronym for *Linux*, *Apache*, *MySQL* and *PHP* and describes the typical stack of a PHP based web application. PHP was selected because it is an important requirement that the application can be run on a typical web hosting service. In addition PHP is a very flexible and well-supported programming language.

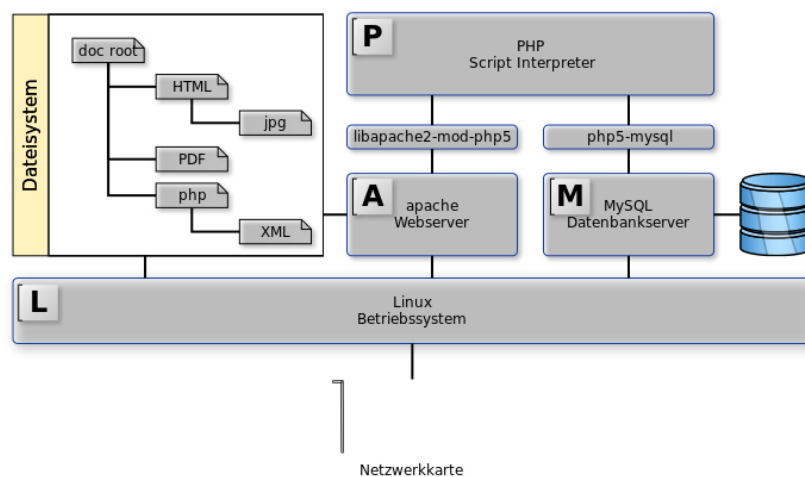


Figure 5.10: The *fisheye* application is built using an *LAMP* stack [94].

As described in Figure 5.10 the *LAMP* stack processes a request as follows:

1. The network card receives a request.
2. The operating system forwards the request to the defined Webserver.
3. The Webserver (usually an *Apache* or *nginx* Webserver)
  - a) decrypts the received request,
  - b) loads and parses the according file from the file system
  - c) and calls the PHP interpreter.

4. The PHP interpreter opens the database connection.
5. The MySQL server retrieves the requested database entries.
6. The PHP interpreter processes the retrieved datasets and closes the database connection.
7. The Webserver prepares the response.
8. The operating system returns the response to the network card, which forwards the response to the client.

The front end layout of the *fisheye* application is built on top of *Bootstrap* [95]. *Bootstrap* is a HTML, CSS and JavaScript framework for developing web applications. It provides pre-defined stylings for specific front end components like buttons, form elements and labels. In addition the front end uses *AngularJS* [96]. *AngularJS* is a JavaScript framework for dynamic web applications. It heavily relies on data binding (by binding functionality to a DOM element) and dependency injection to minimize dependencies. The *fisheye* application uses *AngularJS* mainly for the communication with the *REST* API, the dynamic DOM elements and as templating engine.

## Symfony 2

On top of the *LAMP* stack the PHP framework *Symfony 2* [97] is used. *Symfony 2* is a well-known PHP framework for web projects. Like many frameworks *Symfony 2* heavily uses the patterns of *SoC* (Separation of Concerns) and *IoC* (Inversion of Control) for separating a program into distinct sections. Each section addresses and encapsulates the logic for a specific concern and provides a reusable module.

The *Symfony 2* community already provides many reusable *bundles* (*Symfony 2* calls a directory that has a well-defined structure and contains all source code files to fulfill a specific concern, a *bundle*), which can be used to facilitate the development of a web application. In addition to the *Symfony 2* core bundles the *fisheye* application uses the external bundles listed in Table 5.1.

The *fisheye* application extends the functionality of the used bundles by providing its own bundles. The bundles and their purpose are described in Table 5.2.

## REST

As the provided bundle list in Table 5.2 reveals, the *fisheye* application uses a *RESTful* API for client-server communication. *REST* is the abbreviation for *Representational state transfer* and demands a webserver to produce the same result, when an *URI* (Unique Resource Identifier)

<sup>1</sup><https://github.com/FriendsOfSymfony/FOSUserBundle>, accessed 27-December-2014.

<sup>2</sup><https://github.com/FriendsOfSymfony/FOSRestBundle>, accessed 27-December-2014.

<sup>3</sup><https://github.com/FriendsOfSymfony/FOSOAuthServerBundle>, accessed 27-December-2014.

<sup>4</sup><https://github.com/schmittjoh/JMSSerializerBundle>, accessed 27-December-2014.

<sup>5</sup><https://github.com/haydenk/AbimusFetchBundle>, accessed 27-December-2014.

Name	Description
FOSUserBundle <sup>1</sup>	Provides a basic user authentication and user management system.
FOSRestBundle <sup>2</sup>	Adds basic REST support for <i>Symfony 2</i> applications (e.g. routes are defined and resources can be specified).
FOSOAuthServerBundle <sup>3</sup>	Adds the ability to create an own OAuth Server component, which is used to authenticate with the <i>RESTful</i> API.
JMSSerializerBundle <sup>4</sup>	Adds advanced serialization options for objects to improve client-server communication.
AbimusFetchBundle <sup>5</sup>	Adds basic IMAP functionality for retrieving and sending e-mails via IMAP.

Table 5.1: Used external *Symfony 2* bundles.

Name	Description
JLMFisheyeCoreBundle	The CoreBundle is an application-bundle and is not meant for reuse. It is the entry point for the application and exposes the RESTful API.
JLMFisheyeAuthBundle	Implements the OAuth 2 Service Provider and the OAuth 2 authentication - e.g. this bundle is responsible for creating and renewing authentication keys (also called Tokens).
JLMFisheyeEmailBundle	The EmailBundle handles all actions concerning e-mails, e.g. parsing and sending e-mails to the correct user group.
JLMFisheyeFrontendBundle	The FrontendBundle is also an application-bundle and contains all resources needed by the client, e.g. the user interface and the <i>REST</i> client, and is completely independent from the other bundles.

Table 5.2: The *Symfony 2* bundles implementing the *fisheye* application.

is called multiple times. For instance the URI *https://www.test.com/user/1* always returns the user object of the user associated with the identifier *1*. *REST* uses the standard HTTP request methods [98] for communication. The basic HTTP request methods are defined as described in Table 5.3. While the server provides the *RESTful* API, the client communicates with the server using *AngularJS* [96].

## OAuth 2

For securing and controlling access to the provided *RESTful* API *OAuth 2* [99] was selected. *OAuth 2* is an open protocol and a common way to secure *REST* APIs. For instance *Google*, *Facebook*, *GitHub* and *Twitter* use *OAuth 2* to grant *Consumers* access to their API. *OAuth 2* distinguishes the following roles within an authentication process:

**Consumer:** This is the application requesting access for the informations of the user.

Method	Description
GET	Requests the specified resource from the server. The GET method has no side-effects, which means that the server state is not modified (e.g. no entities are created or modified)
POST	Creates a new (sub)resource. The response contains the URI to the newly created resource.
PUT	Modifies the addressed resource. If the given resource is not available on the server, it is created.
DELETE	Deletes the given resource.

Table 5.3: The used HTTP request methods and their meaning.

This can be a website, a desktop application or any service with Internet access.

**User:** The user is the information owner. *OAuth 2* allows him to manage the *Consumers* which can access his private informations.

**Service Provider:** This is the web service holding the user's informations. After a user granted access to a *Consumer*, the *Consumer* can access the user's informations from the *Service Provider*.

**Token:** A token is an arbitrary string containing characters and numbers. *Tokens* are used instead of the user's credentials to access the user's informations.

The *OAuth 2* authorization process works as follows and is symbolized in Figure 5.11:

1. The *User* requests a protected resource from an application (*Consumer*).
2. The *Consumer* redirects the *User* to the Authorization Server (*Service Provider*).
3. The *Service Provider* returns a login page to the *User*.
4. The *User* logs in and grants access for the *Consumer*.
5. The *Service Provider* validates the users credentials, appends an "authentication code" to the response and forwards the user to the application (*Consumer*).
6. The *Consumer* requests a *Token* from the *Service Provider* using the "authentication code".
7. The *Service Provider* validates the "authentication code" and returns

a) Access Token - used to access the informations on the *Service Provider*

<sup>1</sup>[https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/images/thumb/8/8e/OAuth\\_20.jpg/500px-OAuth\\_20.jpg](https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/images/thumb/8/8e/OAuth_20.jpg/500px-OAuth_20.jpg), accessed 27-December-2014.



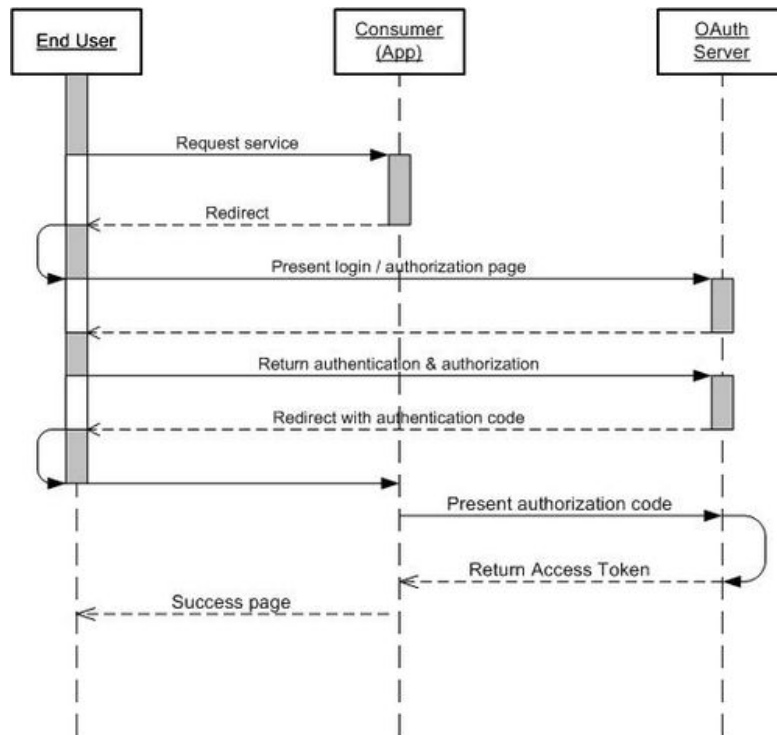


Figure 5.11: The *OAuth 2* authorization process<sup>1</sup>.

b) Refresh Token - used to refresh the Access Token, when it is expired

The *fisheye* application uses *OAuth 2* both on the server by providing an own *Service Provider* and on the client side for authentication. The *Service Provider* is needed to make the applications login independent from external *Service Providers*, which is an important requirement. This makes it possible for users to either authenticate using an external *Service Provider*, e.g. their *Google* or *Facebook* account, or to connect to the internal *OAuth 2 Service Provider*.

## First Prototype

The first prototype was developed from April to June 2014. The layout was already divided into four columns as described in Section 5.1, and the *Fisheye View* was already applied to them. The focused message had more space and was even more structured like an e-mail than it was in further developments. The number of replies was only shown as an overall sum next to the focused message, as seen in Figure 5.12.

In order to receive feedback to the used structure as early as possible a online user test was conducted in late June 2014. Twelve participants were asked to discuss the restructuring of the “Einführung ins Programmieren” (Introduction to Programming) course. Most of the participants were either students themselves, scientific staff members or affiliated to the *Human Computer Interaction Group*. The results of the conducted user test can be found in detail in chapter 6.

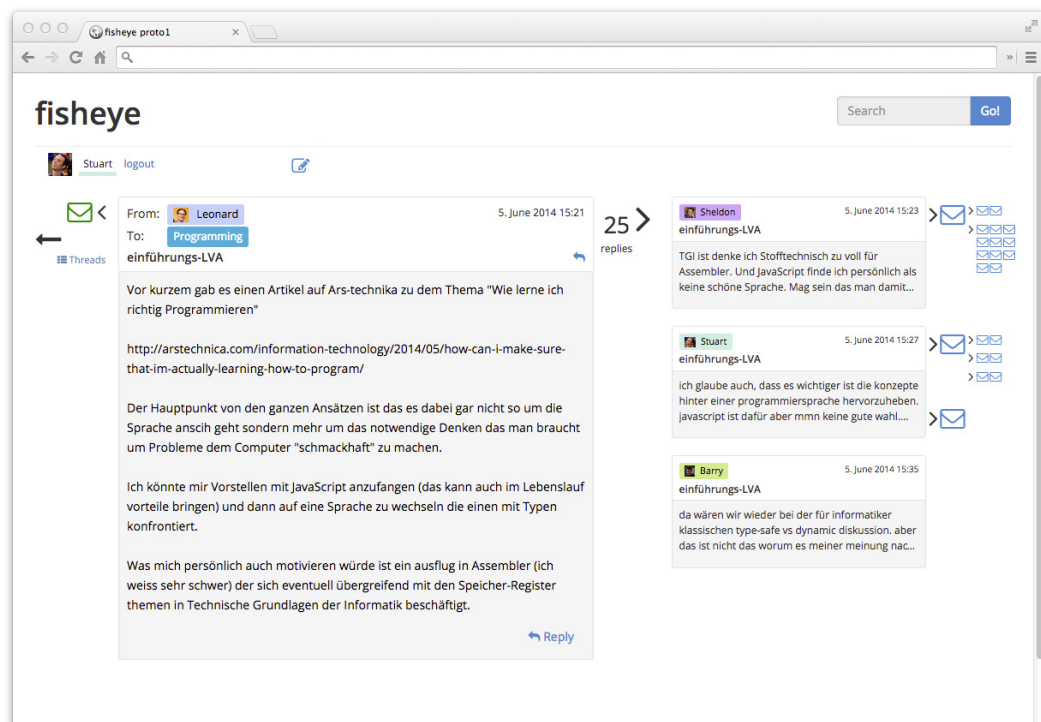


Figure 5.12: The detail view of a message on the second hierarchy level in an early prototype version. The shown discussion resulted from a user test in June 2014.

The goal of the first prototype was to gain insights regarding the optimal ratio between detail and context information. Almost 50% of the available horizontal width was used for the currently focused message, whereas the context information, especially the forth column, got very little space. As long as the hierarchy of the thread was not too deep the provided space was sufficient, but as a thread got broader and deeper the messages symbolized by envelopes overlapped and it was no longer clear where they belong to.

Participants were asked for feedback once the test was over. In detail the participants were asked to answer questions about the usability of the *fisheye* application and to point out missing features. In summery the feedback revealed the following issues:

1. Due to having to navigate many hierarchy levels the orientation and context within the thread gets lost.
2. The ordering of the sub-posts is confusing.
3. Navigating the thread and replying to a message requires very many mouse clicks.

Besides the mentioned issues participants also gave positive feedback. For instance it was noted that the correlation and overview between messages was better than in an Internet Forum and the discussion was more active and easier to follow. Furthermore some participants pointed

out that the user test could be completed without any technical issues or bugs. A detailed evaluation of the user test can be found in chapter 6.

## Productive Version

The development of the productive version started in July 2014. A first usable version was available in September 2014 but development is still in progress. The focus of the productive version was to improve the user interface and to incorporate the feedback of the conducted user test. The following section describes the improvements in comparison to the first prototype. A detailed description of all implemented features is given in Section 5.1.

The first notable difference compared to the first prototype is the layout. Although the structure and the used technologies have remained the same, the front end has been completely redesigned. The source code is more structured and a *Bootstrap* template<sup>1</sup> is used to make the front end layout more uniform.

## Landing Page

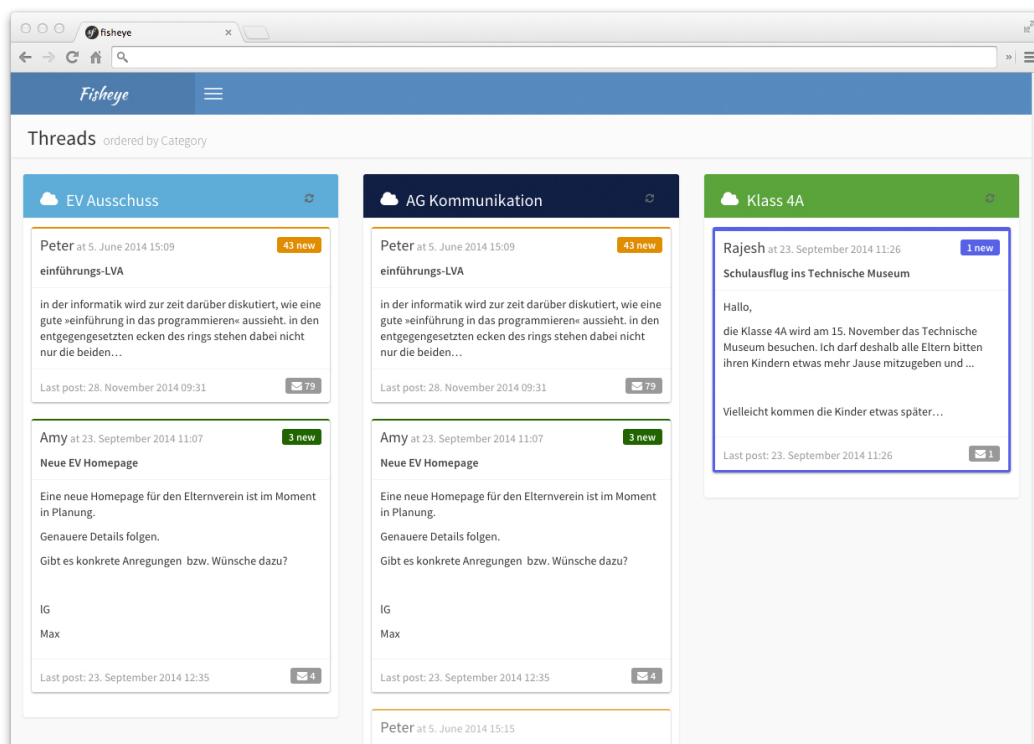


Figure 5.13: The landing page of the productive version shows all available threads. The threads are categorized according to their related tags.

<sup>1</sup><http://almsaeedstudio.com/AdminLTE/>, accessed 27-December-2014.

As seen in Figure 5.13 the landing page provides an overview of all visible threads. Threads are categorized using their associated tags and are ordered by the date of the last contribution, so that a user sees the recently active threads at a glance. Because only three tags are shown on a page, the tags are paginated. Currently the position of a tag is fixed (they are ordered by their database index). In further development it is planned to let the user choose the position of a tag.

Because the amount of threads per tag can vary only the first five threads per tag are shown. By scrolling the page older threads are loaded asynchronously. To improve the usability a search bar is considered for further development. Using the search bar it will also be possible to filter the threads by tag and / or user.

An improvement compared to the first prototype is the representation of a thread. As in Section 5.1 described in detail new threads are highlighted using a colored border. The color represents the author of the initial post. In Figure 5.13 the outer right thread by “Rajesh” is marked as new because the user has not opened it yet. Furthermore the thread preview shows the date of the last contribution and the total number of messages within the thread in the bottom.

## Thread View

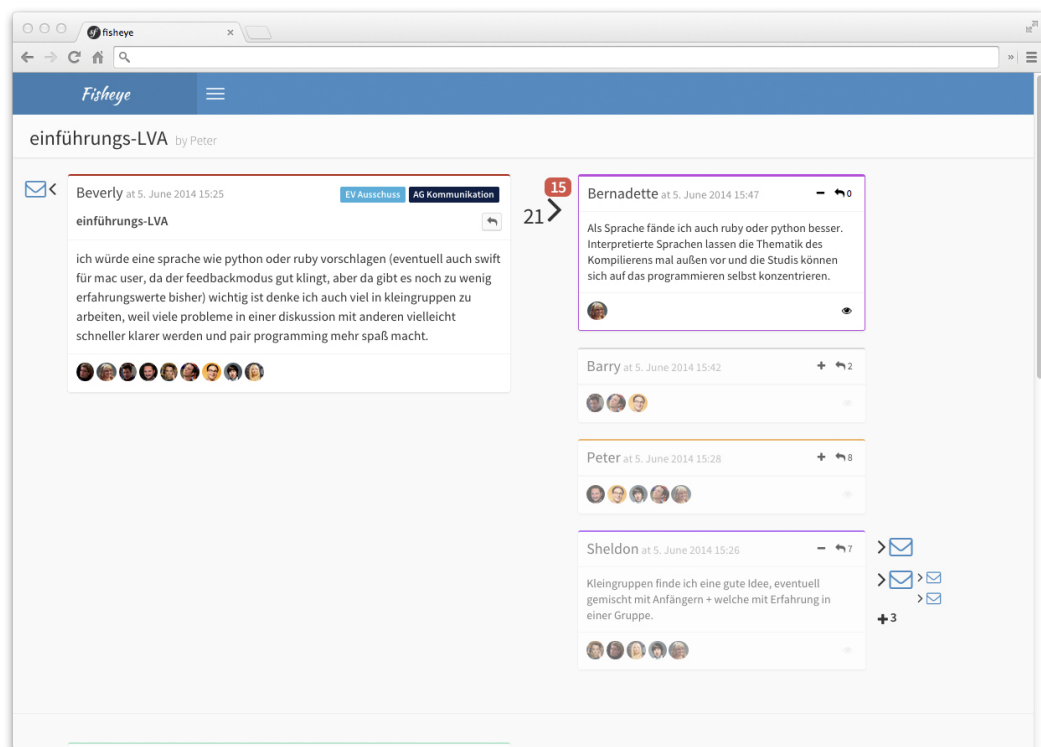


Figure 5.14: The presentation of a thread within the *fisheye* application. The screenshot shows a message on the second hierarchy level. The grayed out messages have already been read by the user.

Like in the first prototype the thread view is still divided into four columns. The currently focused message is centered and provides more information than the preview messages on each side. In the bottom of each message the avatar of all users participating in the sub-thread are shown. Therefore the user sees all participants at a glance and can estimate the importance of the sub-thread: The more people participate in a sub-thread the more likely it is that the topic is controversial and needs further discussion.

Another difference compared to the first prototype is the presentation of replies. Whereas it was not possible to interact with a reply without opening it in the first prototype the user can manipulate the reply messages directly now:

- The minus / plus symbol in the top right corner collapses / expands the complete message. Thereby opening each reply to see the full content is not mandatory any more. This enormously simplifies navigating between messages and makes the context more clear.
- The *reply* symbol in the top right corner directly opens the reply dialog, as seen in Figure 5.15. The user sees two hierarchy levels in full length and has more context information when composing the reply. This feature is described in detail in Section 5.1.
- Clicking the *eye* symbol in the bottom right corner marks a message as read / unread.

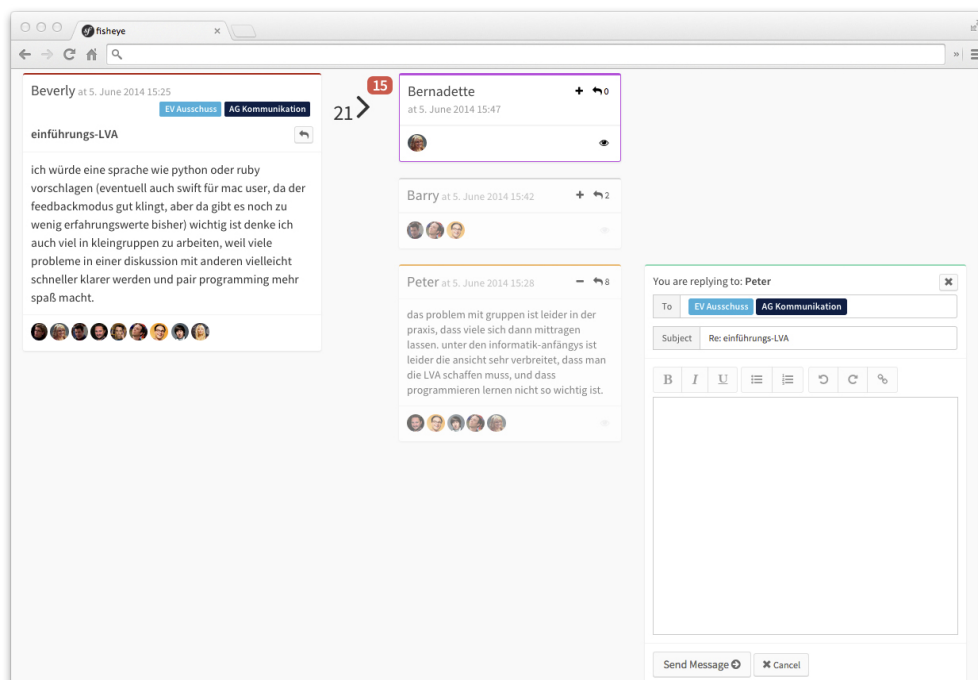


Figure 5.15: In the productive version it is possible to show messages of two hierarchies in full length while composing a reply message.

The user test of the first prototype revealed the complex navigation as the most critical issue. According to some participants too many clicks were needed to navigate the thread. To meet this issue, in addition to the improved presentation of reply messages as described above, the siblings of the currently focused message are shown below the replies. The latter is described in detail Section 5.1 and shown in Figure 5.16.

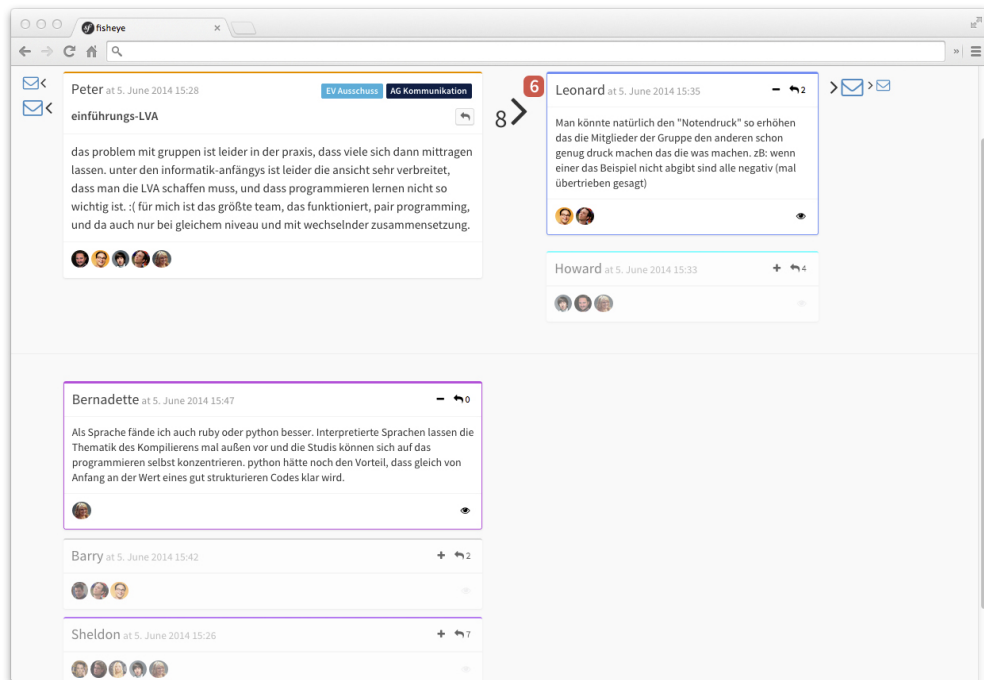


Figure 5.16: To facilitate the navigation within a thread the siblings of the focused message are shown below the replies.

A further usability improvement is that already read messages are grayed out, while new messages are highlighted with a colored border.

## E-mail Handling

While in the first prototype it was not possible to use the application via e-mail, the productive version allows users to interact with the application only by e-mail. This was achieved by developing the “JLMFisheyeEmailBundle”. An exception endures to the current state of development: New threads can only be created in the *fisheye* web application. When a new thread is created within the *fisheye* application all users, who are subscribed to a tag the thread was posted to, are notified via e-mail. A special *Reply-To* e-mail address (see Section 3.2 for detailed information about the e-mail header fields) is set to the sent notifications. Therefore a recipient can reply directly to a received notification and the *fisheye* application is able to parse the message and put it in the correct context. The *fisheye* applications “JLMFisheyeEmailBundle” listens to an IMAP e-mail account and parses all messages from the inbox using a cronjob.

A generated *Reply-To* e-mail address has the following format:

$$[local - part] + [resource - hash] - [notification - hash]@[global - part] \quad (5.1)$$

Resulting in a typical *Reply-To* e-mail looking like:

$$fisheye + 03091d - 9ac2f8782bbd2f589bec5652acd2d14b@fisheye.com \quad (5.2)$$

The “local-part” part makes the e-mail address unique within the “global-part” and is well-known from normal e-mail addresses. The “global-part” consists of three parts: The *host name*, a *dot* and a *Top Level Domain* name (for instance “gmail.com”). After the “local-part” a “+” character follows, which is not part of the “local-part” and informs the mail provider to ignore all following characters. The “resource-hash” identifies the resource a message is sent to and indicates where the message should be appended within the thread. Usually this is the parent e-mail message. The “notification-hash” is separated by an “-” character and specifies the notification a user replies to. Because the notification entry also refers to the targeted resource the “resource-hash” would not be needed, but introduces an additional security check: Only if the “resource-hash” and the retrieved notification refer to the same resource the message is pushed to the *fisheye* application.





# Evaluation

The following chapter describes the evaluation of the developed *fisheye* application introduced in the previous chapter.

## 6.1 Approach

The *fisheye* application was tested in an early state to retrieve feedback to the new visual structure as early as possible. Therefore a user test for a prototype version was conducted in late June 2014. Some essential features for a productive application were missing but the goal of the user test was not to test the usability of the final application but to gain an estimation of the adoption of the new visual structure. The known missing features included:

- The highlighting of read and unread messages.
- Notifying the user about new messages within the currently opened conversation.
- The automatic update of the user interface when a new message was posted to the conversation.

The user test was conducted as an online test for various reasons. The most obvious is that testing a group communication tool with a single test user contradicts the purpose of a group communication tool. Especially testing a group communication tool without receiving responses from multiple participants does not correspond to the usage within a productive environment.

Therefore twelve participants were asked to use the *fisheye* application for discussing the restructuring of the “Einführung ins Programmieren” (Introduction to Programming) course. The topic was chosen because it fulfilled the requirement to choose a controversial topic. The latter was seen as important because the *fisheye* application is particularly designed to facilitate the navigation and orientation in bushy threads. A controversial topic is likely to result in a bushy thread. Most of the twelve participants were either students themselves, scientific staff members or affiliated to the *Humen Computer Interaction Group*.

For the participants the user test started by receiving the URL to the application and the required login credentials via e-mail. Prof. Purgathofer also participated in the user test and started the discussion by posting the initial message. Besides the thread about restructuring the “Einführung ins Programmieren” course all participants had access to a thread providing meta information about the user test. These informations included a note on how to give feedback after the test and to refresh the page regularly because new messages are not automatically pushed to the user interface. A more detailed description of the application was not provided. The user test was scheduled to last approximately one hour.

To the end of receiving interpretable results the participants were asked to answer prepared questions once the user test was over. The questions were targeted to receive feedback regarding the new visual structure. Especially the usability and orientation within the discussion was considered as being most interesting. To receive even more detailed feedback about the visual structure three participants of the user test were asked to participate in personal interviews after the user test. These interviews were held as open interviews and the participants were asked to describe the user test from their point of view.

After the most critical issues which evolved from the user test were implemented, the *fish-eye* application was presented and discussed with end users from the parents’ association of the Wiedner Gymnasium. Furthermore the resulting application was discussed with Thomas Pam-minger, the founder of *wollzelle*<sup>1</sup> - an digital agency located in Vienna, multiple times. Thomas Pam-minger has lots of experience in the field of user experience and communication tools and gave important feedback on how to improve the usability of the new visual structure.

## 6.2 Results and Interpretation

In the course of the conducted user test more than 80 messages were composed in approximately one hour. The resulting conversation reached a depth of nine and a breadth of eight messages (both values are measured on the broadest and deepest location of the conversation). Therefore the resulting structure of the conversation is bushy as was expected. The resulting bushiness of the thread indicates that the test users generally understood the new visual structure. The latter is confirmed when analyzing the messages for their classification: Almost all messages are replies to an appropriate message and are classified in the correct hierarchy level. Thereby the conversation is clustered into multiple sub-threads where different areas of the general topic are covered. For instance the different possible programming languages for an introduction to programming course were discussed in separate sub-threads.

An exception is the second hierarchy level (the first replies to the initial message). Three of the eight first level replies referred to the existing programming experiences of students. This overlapping is most likely caused by the contemporaneous usage of the *fish-eye* application by all participants and the missing notifications about new posts. After Prof. Purgathofer initiated the discussion no replies or sub-threads were available. Therefore most participants started to compose a first level reply. Because of the missing notifications for new messages participants composing a reply didn’t see other messages before submitting. A way to see replies composed

---

<sup>1</sup><http://www.wollzelle.com/>, accessed 28-December-2014.

in the meantime would have been to refresh the page before submitting the own reply. The latter was not seen as a reasonable approach because this would have disrupted the current user interaction completely. Therefore this solution was not suggested to the participants. The result was that replies were composed nearly simultaneously and without the awareness of other replies. This is also confirmed when investigating the date of the submitted posts: Six of the eight first level replies were composed within nine minutes, four of them even within five minutes. The resulting thread shows that especially the second hierarchy level covers the same topic in multiple sub-threads. Although this behavior was also observed in deeper hierarchy levels it confused participants especially in upper hierarchy levels.

This observation was confirmed during the personal interviews and led to the conclusion that the *fisheye* application and its new visual structure is not applicable for real-time group communication. Like e-mail communication the *fisheye* application is built for asynchronous group communication and does not satisfy the requirements for real-time communication. Notifications for new messages and an automatic update of the active conversation would probably have de-esclated the occurred problems. Nevertheless the *fisheye* application is not designed for real-time group communication. Other types of CMC, e.g. Instant Messaging, are more applicable for this use case, as described in section 3.2. Due to the setting of the user test (in detail described in section 6.1) the conversation was more a real-time discussion than an asynchronous one in the beginning. The longer the user test lasted and the more the conversation scattered into sub-threads, the more asynchronous it became. This is confirmed considering the date of the posted messages within the sub-threads. In contrast to upper hierarchy levels, e.g. the second hierarchy level, the period of time between submitted messages is significantly longer on deeper levels. The latter corresponds better to the daily usage of a group communication tool.

Besides this observations the user test revealed the following issues:

1. Due to having to navigate many hierarchy levels the orientation within the thread gets lost.
2. The ordering of the sub-posts (the direct replies to the currently focused message) was confusing.
3. Navigating the thread and replying to a message requires very many mouse clicks.
4. Unread and new messages should be highlighted.
5. Users asked for a view showing only their own posts or to highlight their posts within the thread view.
6. Users asked for an edit function after sending a message.
7. Some user interface elements, e.g. the “back to Threads” button, were misinterpreted.

The mentioned issues are ordered by their negative impact on the usability of the overall application. The following steps were taken to meet them:

**Issue 1:** The orientation and navigation within a conversation is considered to be the most critical issue. This directly affects the usability of the application and frustrated

participants. As participants already suggested this issue can probably be met by providing more information within the preview of a message. Especially the tooltip preview of the hidden messages, which are represented by envelopes, did not provide enough information to classify the message correctly. In addition the forth column requires more space and the envelopes should be arranged in a more structured way.

- Issue 2:** This issue arose because sub-posts were ordered taking the date of the last post within the sub-thread into account. As a result it happened that newer sub-threads were listed below older ones. Furthermore the positions changed as soon as a new reply was posted within a sub-thread. In conjunction with not highlighting new messages this was very confusing for almost all participants. We decided to order sub-threads using the date of the sub-threads' root message and not the date of the latest post within the sub-thread.
- Issue 3:** An attempt to correct the issues originating from the complex navigation was made by providing additional routes within the application. For instance the siblings of the focused message are added below the focused message and therefore are directly accessible. Furthermore it should be possible to directly reply to a reply of the currently focused message. Also the "upwards" navigation to upper hierarchy levels in column 1 has to be revised. The meaning of the vertically aligned envelopes was not intuitive enough and confused many participants.
- Issue 4-7:** These issues cover features that were not intended for the prototype version but will be considered in further development. Especially issue (7) referring to the inconsistent user interface is a well-known fact that will be addressed in the productive version.

Besides the mentioned issues participants also gave positive feedback. For instance the question "Was it easy to navigate within the discussion?" was answered positively by more than 70% of the participants, the question "Were you able to follow the discussion?" even by 100%. The open interviews revealed that the structure was easy to understand and has the potential to facilitate the navigation in bushy conversations. In contrast the question comparing the visual structure to an Internet Forum ("Was it easier to navigate compared to an classic Internet Forum?") was answered negatively by all participants. The subsequent interviews revealed that the main drawback of the new visual structure was the missing highlighting of read and unread messages. While in a linear layout one can simple scroll over an already read message the *fisheye* application requires a user to actively change the focus and open the next message. The latter is especially irritating when opening a message again and again because it is not marked as *already read*.

Whereas the navigation was criticized mainly for the previous described reason the correlation of messages and the overview of the conversation's structure was seen as a significant improvement compared to an Internet Forum. Furthermore the discussion was generally seen as more active and easier to follow compared to an Internet Forum. The interviewed participants strengthened the general feedback and consented to the list of most critical issues. In addition

some participants pointed out that using the application felt comfortable. All in all the user test was encouraging and important learnings evolved from the participants feedback.

### 6.3 Linear Layout versus Fisheye View Layout

In the course of the subsequent interviews the resulting conversation was presented in both a linear layout (Figure 6.1) and the new visual structure used by the *fisheye* application (Figure 6.2). The linear view uses the layout of a classic Internet Forum. Messages are ordered by their date and indentations are used to indicate the according hierarchy level.

It has to be noted that the two views are not comparable directly for many reasons. For instance the linear view does not offer any user interaction elements, it uses a very basic styling and messages are shown in full length while the *Fisheye View* makes some message only available via mouse-over and other user interactions.

Nevertheless the participants were asked for their opinion in regard to overview and orientation. Although the linear layout allows a user to scroll the complete conversation on a single page the orientation within the conversation is lost easily. Especially in deeper hierarchies it is not always clear which message is the parent and what message is referred to. The *fisheye* application solves this problem by presenting the messages in the current context. Only one message can be focused and replies are always shown next to the focused message. All interviewed participants were in favor of the *Fisheye View* in the scope of overview and orientation. Generally the main advantage of the linear view is seen in the navigation which is much easier and more intuitive than in the *Fisheye View* (scrolling versus clicking).

A participant had the idea of introducing keyboard shortcuts to facilitate the navigation within the *Fisheye View*. This seems like a good idea because it reduces the number of required clicks significantly. Furthermore the definition of the shortcuts should be straight forward: A conversation is a directed graph without loops and therefore traversable with a few commands. The question remains if shortcuts reduce the complexity of the navigation for all users because shortcuts are usually only used by advanced users. Usually shortcuts confuse normal users more than helping them. Nevertheless the idea is considered in further development.

on of the user test in a linear view. In addition to the initial marked via indentations, 28 of the more than 80 messages

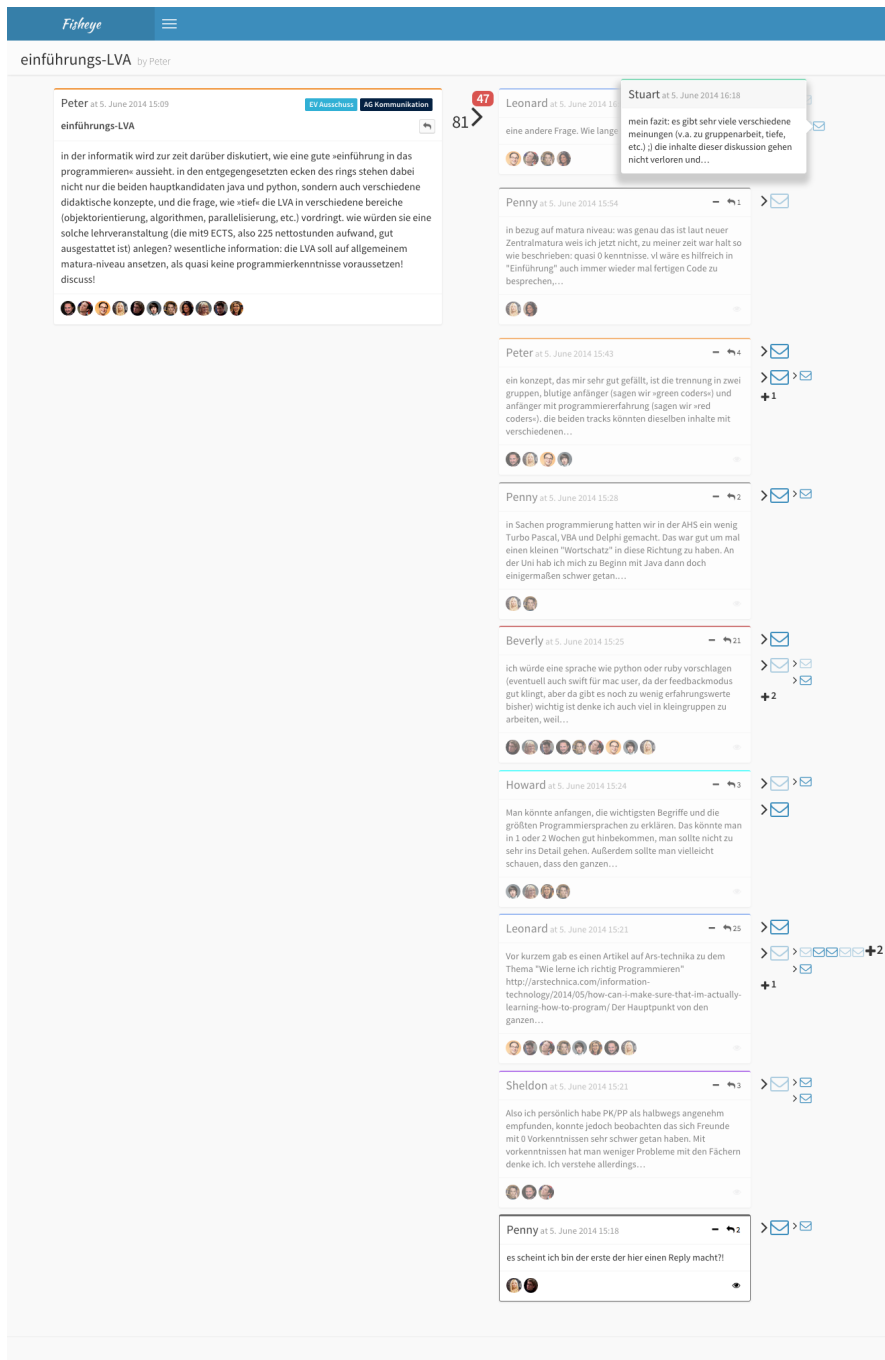


Figure 6.2: The resulting conversation of the user test using the new visual layout. In contrast to the linear view the third and subsequent hierarchy levels are not shown per default but accessible via mouse-over, as seen in the top right.





# CHAPTER 7

## Outlook

Besides listing further development steps for the *fisheye* application this chapter shows what prospects for further development are generally available for scientifically started software projects. The latter is a recurring problem because the developers of scientific software usually are students and rarely start a scientific career where they would be able to pursue the started project.

### 7.1 Further Development

Although the productive version is a great leap forward compared to the first prototype, development is still in progress. This section describes the further planned development steps. The development steps are separated into three sections: First the *Extensions* section describes ideas that could extend the *fisheye* application in general, then the *Front End* section describes planned improvements for the user interface and the user experience, finally the *Back End* section describes functional improvements.

#### Extensions

Due to the decoupled design of the *fisheye* application it can be easily extended. As a result of conversations with various EV members and usability experts the idea of providing different thread types arose. At the moment only one thread type, the *plain text* type, is available. This type is ideal for discussions regarding topics like “How can I support my children to get better grades?” or “What is your opinion on shutting down the cafeteria?”. In contrast if a decision should be made a plain text thread will become quickly very complex. For instance the topic “Should we spend 1.000€ on new school books?” usually only requires a “yes” or “no” answer from the users. The more people are involved in such a discussion the more complex it becomes. Answers are hidden in sub-threads and have to be summed up manually by the original poster for retrieving an overall result.

The general idea is to provide different thread types for varying tasks, like *Facebook*<sup>1</sup> offers

---

<sup>1</sup><https://www.facebook.com/>, accessed 28-December-2014.

different post types for various activities, as seen in Figure 7.1.

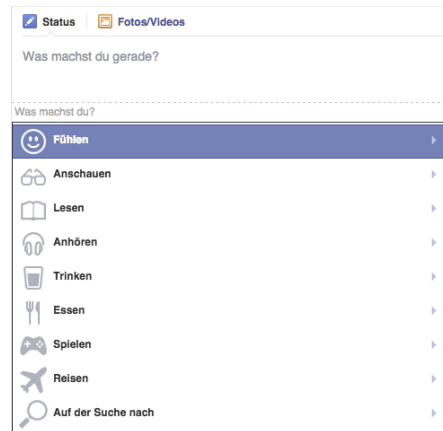


Figure 7.1: Facebook offers many different post types for various situations<sup>1</sup>.

Possible thread types are:

**Text:** Text messages can be posted without any limitations (this is the currently implemented thread type).

**Poll / Decision:** Using a *poll type* users can select a predefined answer for a given question and write an optional comment. In addition users may add custom answers or suggest additional answers to the original poster. The original poster can mark a poll thread as “closed” which closes the poll and shows the poll’s result both next to the original poster’s message and the thread preview on the landing page. The latter improves usability enormously because an closed poll thread does not have to be opened to see its overall result.

**Appointment:** The *appointment type* allows users to find a mutually agreed date. The functionality of the *appointment type* is not defined in detail yet but it may be implemented using the API of an existing scheduling service, like *Doodle*<sup>2</sup> provides it.

**Document / Versioning:** The *document type* allows sharing and discussing binary files. Some EV members requested this type for sharing reports. It not only includes the ability to share files but also to version specific files. Therefore a member list can be kept up to date in a single thread. As the *appointment type* the *document type* is not defined in detail yet.

A major requirement for the *fisheye* application is the low barrier of entry. Therefore the e-mail compatibility has to be kept in mind during the development of additional thread types.

<sup>2</sup><http://doodle.com/features/api>, accessed 28-December-2014.

If a thread type can not be used by e-mail it has to be evaluated thoroughly if the thread type is implemented anyway.

Another idea for extending the *fisheye* application are mobile applications. More and more people use mobile devices for communication [60]. Although the *fisheye* application heavily relies on e-mails and therefore is accessible and usable on mobile devices a mobile application has some major benefits: Users can be informed about updates via notifications, the user interface can be designed to perfectly fit the needs of the users and advanced features of the web application, which are not accessible via e-mail, can be implemented. Because the *fisheye* application is developed using a *REST* API, as described in Section 5.2, a mobile application can directly access the API and only the front end needs to be developed. The back end and the used *REST* API are not affected when developing a mobile application.

## Front End

As previously noted the productive version is a work in progress. Most conversation partners find the basic layout appealing. Nevertheless there is always space for improvements. This section lists the planned front end changes.

A very important feature for a modern web application is a *Responsive Web Design* (RWD). A site designed using RWD adopts the layout to the users viewing environment. For instance the content of a web application is shown in four columns when viewed on a desktop computer. Opening the same application on a smart phone shows the application in a single column. Because a smart phone usually has a smaller screen and four columns would not fit on the phones display, the user experience is better when the content is only shown in one column, as seen in Figure 7.2<sup>3</sup>. Technically there are various methods to achieve this adoption of the layout. The most common one are *CSS Media Queries*<sup>4</sup>, which allow developers to define specific stylings only applied when the application is shown on smaller screens. Although the used *Bootstrap* framework supports RWD, the *fisheye* application does not use RWD at the moment. Therefore a major goal for the further front end development is to include a RWD.



Figure 7.2: Illustration for responsive web design technique. Flexible & responsive website for desktop, netbook, tablet and smartphone screen<sup>3</sup>.

<sup>3</sup>[http://de.wikipedia.org/wiki/Datei:Responsive\\_Web\\_Design.png](http://de.wikipedia.org/wiki/Datei:Responsive_Web_Design.png), Muhammad Rafizeldi, accessed 28-December-2014.

<sup>4</sup>[https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media\\_queries](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries), accessed 28-December-2014.

In addition to adding a RWD some layout improvements are planned in order to simplify the usability and to improve the user experience. As the user test has shown and various conversations verified the used envelope icons are misleading. Especially the icons leading to hierarchically deeper messages in the forth column (as seen in Figure 5.3) are difficult to understand. To make the meaning more clear a consistent layout element for showing the amount of read and unread messages within a sub-thread has to be developed. The layout element, in the following referred to as *message-status-symbol*, has to illustrate the amount of messages in a sub-thread and how many of them are unread. The *message-status-symbol* will be used throughout the *fish-eye* application. To be more specific it will be used on the *Landing Page* for each thread preview and in the *Thread View* next to the focused message (Column 2 in Figure 5.3) and in the forth column for representing hierarchically deeper messages. A first idea for the *message-status-symbol* is to show an arrow containing both values.

Another layout element will be introduced in the *Thread View*. The *conversation-status-symbol* will be used on the same positions as the *message-status-symbol* and should illustrate the number of sub-threads of a message. A first idea is to use overlapping speech bubbles to illustrate the bushiness of a sub-thread.

Furthermore the following minor improvements are planned:

- Like in the first prototype the avatar of a user will be placed next to the user's name in the *Thread View*.
- Both users and tags are associated with unique colors at the moment. Because this can be misleading the unique color property for one of them may be removed.
- The back navigation in Column 1, as seen in Figure 5.3, will be revised.

## Back End

The further development of the back end concentrates on introducing the most essential features. These include the development of a user management, a tag management and an administration interface.

The user management is needed because it is currently not possible to create new user accounts or edit their settings. The *fish-eye* application is used by a closed community and it is an important requirement that it is not possible to register without an invitation. The *fish-eye* application's target groups are voluntary association or similar closed communities having similar requirements as the parents' associations.

On the example of the parents' association the invitation process could work as follows: An authorized user, in the following referred to as *admin*, has the privilege to create new user accounts. At the beginning of a school year each class elects a "spokesparent", who represents the parents of a class externally. The "spokesparent" collects the contact informations of all parents within the class and asks an *admin* (usually *admins* are members of the parents' association) to create the accounts according to the collected contact informations. The *admin* uses the "create new users" interface of the *fish-eye* application to create the user accounts. For creating a new

user only the name and an e-mail address are mandatory. After the *admin* created the user accounts, each parent receives an invitation e-mail containing a link to a registration form. The parent completes the registration form and can login to the *fisheye* application from now on.

An *admin* also has the privilege to delete users. This is necessary when a pupil leaves or completes the school. Normal users should be able to edit their profile and settings. For instance a user can change the selected color, the avatar and the e-mail notification settings.

The back end must also provide a tag management system. Using the tag management it is possible to create new tags, edit tags and delete tags. If a tag is deleted the associated threads are either deleted or moved to a new tag. A user can be associated with an arbitrary number of tags. If a user is associated with a tag the user receives all messages addressed to the tag and can post new messages to the tag. This mechanism is similar to newsgroups - described in detail in Section 3.2.

A field for improvements in the back end is the handling of e-mail messages. Although parsing incoming e-mails basically works it can be improved by cutting off quoted text and additional encoding checks. Furthermore it is currently not possible to create a new thread by e-mail. In contrast to replying to a message creating a new thread requires the user to manually enter a predefined e-mail address. Also the tags must be specified either in the e-mail address or in the subject of the message. To define and implement an appropriate method is part of the further development process.

The field of e-mail handling also affects sending notifications for new messages. At the moment the notification message only contains the content of the new message. To motivate a recipient to participate in a discussion and the provide more context information the notification message could contain additional meta informations. For instance the total amount of messages in the thread, the number of unread messages and the names of the participants could be included. Another very important part of the notification message is the backlink to the message in the *fisheye* application and the information that the user can directly reply to the message via e-mail. *GitHub*<sup>5</sup> combines the two informations by writing “Reply to this email directly or view it on GitHub.” on the bottom of each message (“view it on GitHub” is linked to the according *GitHub* page).

## 7.2 Scope of Further Development

A recurring problem of scientific software projects is the further development and maintenance when the scientific work ended. The developers of scientific projects usually are students supervised by a professor. Furthermore the project is often their last task before leaving university. Sometimes the project can be continued in the scope of a subsequent scientific work. Beside this possibility this section gives a brief overview of opportunities for students who affiliated with their scientific project and want to continue their work. Depending on the scope of the scientific project not all presented opportunities are suitable.

---

<sup>5</sup><https://github.com/>, accessed 28-December-2014.

## Open Source Software

If the legal preconditions allow it the project can be published as *Open Source Software* (OSS). The *Open Source Initiative* (OSI) defines OSS as follows: “Open source software is software that can be freely used, changed, and shared (in modified or unmodified form) by anyone.”<sup>6</sup>. The big advantage of OSS is that anybody who is interested in the project can contribute and use it. The requirements are that the source code is publicly available, the source code is allowed to be modified, copied, shared and modified versions of the source code are allowed to be shared.

The development of OSS usually takes place on web services provided for free, like *GitHub*<sup>5</sup>, *launchpad*<sup>7</sup> or *bitbucket*<sup>8</sup>. This approach makes the development process publicly visible and encourages interested people to contribute.

Although OSS is given away for no charge there are various models for funding the development. One approach is to offer the software for free and to charge for support, training lessons and customizations. For instance *SensioLabs*<sup>9</sup>, the main developers of the *Symfony* 2 framework, and *Canonical*<sup>10</sup>, the company behind the well-known Linux distribution *Ubuntu*, develop their software as OSS and offer chargeable training lessons and technical support for their products. Another approach for funding an OOS is to sell licenses for proprietary add-ons. A rather new procedure for funding an OOS is *crowdfunding*, which is described in the next section.

## Crowdfunding

The *Oxford Dictionaries* defines *crowdfunding* as “The practice of funding a project or venture by raising many small amounts of money from a large number of people, typically via the Internet.”<sup>11</sup>. Crowdfunding is a rather new approach of funding all kind of projects. The first platform for crowdfunding, *ArtistShare*, launched in 2003 [100]. In the following years many platforms like *IndieGoGo* (2008), *Kickstarter* (2009) and *RocketHub* (2009) followed. While *ArtistShare* only hosts art related projects the other sites host also funding campaigns for social causes, entrepreneurs and small businesses. According to Freedman and Nutting *Kickstarter* hosted from its launch in 2009 through September 2014 more than 180.000 funding campaigns. 40% of them were successful. The successful campaigns raised a total amount of \$1.355 billions from more than 7.1 million backers.

Generally the *crowdfunding* platforms distinguish between “all-or-nothing” and “keep-it-all” campaigns: While in “all-or-nothing” campaigns the project only receives the fund if the stated funding goal is reached, projects may keep the funds even if the goal is not reached in “keep-it-all” campaigns. The model of *crowdfunding* has three types of actors: the project initiator, who proposes the idea and wants to be funded, the crowd, usually individuals who want to support the idea, and the platform, which brings the project initiator and the crowd together [101].

---

<sup>6</sup><http://opensource.org/>, Open Source Initiative, accessed 28-December-2014.

<sup>7</sup><https://launchpad.net/>, accessed 28-December-2014.

<sup>8</sup><https://bitbucket.org/>, accessed 28-December-2014.

<sup>9</sup><http://sensiolabs.de/>, accessed 28-December-2014.

<sup>10</sup><http://www.canonical.com/>, accessed 28-December-2014.

<sup>11</sup>[http://www.oxforddictionaries.com/us/definition/american\\_english/crowdfunding](http://www.oxforddictionaries.com/us/definition/american_english/crowdfunding), accessed 28-December-2014.

Most projects on *crowdfunding* platforms are “rewards-based” *crowdfundings*. The latter means that they offer a specific reward for a given pledge. For instance the *Pebble*<sup>12</sup> campaign rewarded a pledge of \$99 or more with a *Pebble* watch, a pledge of \$220 or more with two watches and so on. Although the reward definitely is an incentive for backers, Freedman and Nutting state that “most backers pledge amounts less than the minimum required to get a reward” [100]. Freedman and Nutting conclude that most people simply want to support the project and are not up to a reward.

There are special *crowdfunding* platforms for scientific research projects, e.g. *experiment*<sup>13</sup>. These platforms usually do not reward backers with material goods but with insights into the research and development process of the project.

A *crowdfunding* campaign can have many benefits for a project initiator. Among other things a successful campaign provides access to capital, serves as a free marketing tool and gives early feedback for a project idea. Furthermore using an “all-or-nothing” campaign usually is for free if the stated funding goal is not reached.

## Product Development

Another opportunity for continuing a scientific project is the scope of product development. Not all scientific projects can be converted into a stand-alone product. The *Lean startup* provides a method for developing products and testing their market relevance in an early state. A core concept of the *Lean startup* method, introduced by Ries in 2011, is the *Minimum Viable Product* (MVP). A MVP is the “version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort.” [102]. The idea of the MVP is to test a product idea as soon as possible so that the team receives feedback quickly and can react to it.

The *Lean startup* method provides more core principles:

- Use continuous deployment to deploy changes immediately into production.
- Use split testing (A/B testing), which means that different versions of a product are offered to customers at the same time, and measure the difference of the two groups.
- Use actionable metrics for business decisions.
- Correct the course of your business idea to test new fundamental hypothesis about the product (“Pivot”).
- Loop the process of turning ideas into products, measuring the customers’ reactions against the built product and learn whether preserve the idea or pivot it (“Build-Measure-Learn”).

All in all the *Lean startup* method is only one method among others for testing a product idea for market relevance.

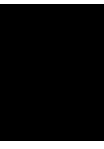
---

<sup>12</sup><https://getpebble.com/>, accessed 28-December-2014.

<sup>13</sup><https://experiment.com/>, accessed 28-December-2014.







## Conclusion

This chapter gives a brief summary of this work and provides an overview of open research questions and future work.

### 8.1 Summary

Especially in the field of online group communication new technologies are adopted very slowly. This is due to the fact that a new group communication technology usually requires adjustments of everyday routines and habits, which not only affect an individual user but also his / her social environment. This fact results in a high barrier to entry for new group communication technologies.

In addition the usage of new communication technologies poses the risk of excluding individual persons with less computer knowledge. This becomes especially clear when focusing on the communication within voluntary associations, which are generally comprised of very diverse members with dissimilar technological background. Nevertheless the usage of new technologies usually facilitates the communication and provides an additional value compared to established solutions.

This thesis has shown how to facilitate online group communication while keeping the barrier to entry as low as possible. The former was achieved by applying an information visualization technique to the visual structure of a CMC system. A main problem of online group communication is the presentation of long and bushy conversations (a discussion with multiple replies per message). While the common approach (presenting conversations in the form of an ordered list and using indentations for child messages) works great for narrow conversations, bushy conversations become confusing very quickly. The field of information visualization offers a variety of techniques to present a large data set in a way users can easily understand and explore. Therefore the hypothesis was that it becomes easier to follow a discussion by applying one of these techniques to the visual structure of a CMC system. With the objective of keeping the barrier to entry as low as possible an easy to use and wide-spread CMC system had to be selected.

To get an overview of both fields literature review was used at the beginning. In the following techniques from both fields were selected and a web application with the aim to facilitate online group communication was designed and implemented.

Because of its very low barrier to entry and its wide-spread usage e-mail was selected as a communication tool. E-mail complies with the requirement to use a well-known standard and is the most common used online communication tool. To keep the low barrier to entry of e-mail, the resulting application uses the principle of “progressive enhancement”: The applications’ basic features can be used only by e-mail and more advanced features are available in the web application. In the context of the application this means that users can participate in conversations without using the web application. To this end all messages are sent via e-mail and the application correctly classifies replies sent by e-mail in the active conversation. On the one hand this results in a very low barrier to entry because an e-mail account is sufficient to participate in discussions. On the other hand advanced users are able to perform more complex actions within the web application.

Considering the structure of an e-mail conversation the *Fisheye View* technique was chosen as information visualization technique and applied to the visual structure of the web application. The *Fisheye View* technique shows the current point of interest emphasized and displays information “further away” visually distorted. In the context of the developed application this allows a user to focus on a single message while preserving the needed context information to correctly classify the message within a bushy conversation. Instead of visually distorting information further away the information density of previous and subsequent messages is reduced. Therefore the *Fisheye View* technique perfectly satisfies the required mixture of detail and context information without introducing additional complexity.

In order to receive feedback to the new visual structure early a user test was conducted with a prototype version. With the goal to create a bushy conversation thread the participants were asked to discuss a rather controversial topic (the restructuring of the “Introduction to Programming” course). As the user test revealed the intended effect of the new visual structure has been achieved with some exceptions: A minority of users did not understand the new structure immediately and were confused by the horizontal arrangement of the messages initially. In subsequent open interviews the confusion could mainly be traced back to misleading user interface elements and the known fact that some features, e.g. the highlighting of already read messages, were missing in the prototype version. The results from the user test were considered in the following development steps. With the objective of evaluating the new visual structure in detail the application was discussed regularly with potential end users and experts in the field of User Experience during the development process. These discussions produced very valuable feedback and were important for the further development.

Altogether the evaluation revealed that the new visual structure has the potential to facilitate asynchronous online group communication. Due to the new structure the relation between messages is recognized at first glance and conversations are clustered into various sub-threads, where different areas of the general topic are discussed. Besides the new visual structure the low barrier to entry was seen as the most important value added. The usage of e-mail drastically reduces the risk of excluding individual persons with less computer knowledge, which in turn facilitates the introduction of a new technology particularly for voluntary associations. In con-

trast the evaluation has shown that the application is not applicable for real-time communication and some user interface elements are misleading and need to be standardized.

## 8.2 Future Work

This thesis has shown how to combine two techniques from different fields of informatics to enhance the outcome. However, this is only a single use case for combining techniques in an interdisciplinary way. The number of research fields of informatics is immeasurable and each one is dedicated to a limited number of tasks. Combining techniques from various research fields has the potential to improve the outcomes for each research field.

An unconsidered aspect in this thesis is the meaningfulness of applying an information visualization technique to other fields than computer-mediated communication. Especially large database sets or long dropdown lists appear to be suitable for usability improvements using an information visualization technique.

Originally the *Fisheye View* technique uses visually distortion for displaying information further away from the current point of interest. Instead of using visual distortion the information density was reduced within this thesis. It would be interesting if visually distorting, or combining visual distortion and the reduction of information density, would have produced comparable results.

Before applying the *Fisheye View* technique to the visual structure of a conversation, *Miller Columns* are used to horizontally align a conversation thread according to its hierarchy levels. An interesting question would be whether the *Fisheye View* technique can be applied to other presentation forms of a conversation, e.g. a tree view, too.

Another important question, that remains unanswered, is if a different information visualization technique would have been a better choice for restructuring the appearance of a group communication tool. Although the *Fisheye View* technique was selected after close examination and to the best knowledge and belief, the result can not be compared with other techniques. It would be of great interest how comparable implementations - with different visualization technique for the visual appearance - perform to each other. Especially the difference to an *Overview+Detail* technique would be interesting, because it separates the detail and the contextual view.

In contrast the *Fisheye View* technique could be applied to a different CMC system. E.g. instant messaging, where often many messages are received in a short time period, would be an interesting selection.

The evaluation within this thesis mainly investigates the users behavior when participating in a discussion. Other interesting questions for evaluating the application, e.g. “Is it easy to look up a specific information?” or “Can a conversation be skimmed through faster compared to an Internet forum?”, are not discussed. Especially the comparison with Internet forums in the context of skimming through a conversation would be of great interest.



# Bibliography

- [1] G. W. Furnas, “A fisheye follow-up: Further reflections on focus + context,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’06, (New York, NY, USA), pp. 999–1008, ACM, 2006.
- [2] A. Cockburn, A. Karlson, and B. B. Bederson, “A review of overview+detail, zooming, and focus+context interfaces,” *ACM COMPUT. SURV*, pp. 1–31, 2008.
- [3] J. Tidwell, *Designing Interfaces*. O’Reilly Media, Inc., 2005.
- [4] Google, “Google maps.” <https://maps.google.com>, 2014. [Online; accessed 15-June-2014].
- [5] Microsoft, “Bing maps.” <https://www.bing.com/maps/>, 2014. [Online; accessed 20-June-2014].
- [6] T. Cassie, “Fisheye strategy,” 2002. [Online; accessed 28-December-2014].
- [7] G. D. Venolia and C. Neustaedter, “Understanding sequence and reply relationships within email conversations: A mixed-model visualization,” Tech. Rep. MSR-TR-2002-102, Microsoft Research, January 2003.
- [8] S. Misell, “Marketing for voluntary associations,” [Online; accessed 4-August-2014].
- [9] H. Schumann and W. M. 0004, “Informationsvisualisierung: Methoden und perspektiven.,” *it - Information Technology*, vol. 46, no. 3, pp. 135–141, 2004.
- [10] B. Shneiderman, “The eyes have it: A task by data type taxonomy for information visualizations,” in *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL ’96, (Washington, DC, USA), pp. 336–, IEEE Computer Society, 1996.
- [11] D. A. Keim, “Datenvisualisierung und data mining.,” *Datenbank-Spektrum*, vol. 2, pp. 30–39, 2002.
- [12] B. B. Bederson, J. Meyer, and L. Good, “Jazz: An extensible zoomable user interface graphics toolkit in java,” in *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST ’00, (New York, NY, USA), pp. 171–180, ACM, 2000.

- [13] B. B. Bederson and A. Boltman, "Does animation help users build mental maps of spatial information?," in *Proceedings of the 1999 IEEE Symposium on Information Visualization, INFOVIS '99*, (Washington, DC, USA), pp. 28–, IEEE Computer Society, 1999.
- [14] M. Shanmugasundaram and P. Irani, "The effect of animated transitions in zooming interfaces," in *Proceedings of the Working Conference on Advanced Visual Interfaces, AVI '08*, (New York, NY, USA), pp. 396–399, ACM, 2008.
- [15] C. Klein and B. B. Bederson, "Benefits of animated scrolling," in *CHI '05 Extended Abstracts on Human Factors in Computing Systems, CHI EA '05*, (New York, NY, USA), pp. 1965–1968, ACM, 2005.
- [16] J. J. Van Wijk and W. A. A. Nuij, "Smooth and efficient zooming and panning," in *Proceedings of the Ninth Annual IEEE Conference on Information Visualization, INFOVIS'03*, (Washington, DC, USA), pp. 15–22, IEEE Computer Society, 2003.
- [17] K. Perlin and D. Fox, "Pad: An alternative approach to the computer interface," in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '93*, (New York, NY, USA), pp. 57–64, ACM, 1993.
- [18] B. B. Bederson and J. D. Hollan, "Pad++: A zooming graphical interface for exploring alternate interface physics," in *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology, UIST '94*, (New York, NY, USA), pp. 17–26, ACM, 1994.
- [19] E. Pietriga, "A toolkit for addressing hci issues in visual language environments.," in *VL/HCC*, pp. 145–152, IEEE Computer Society, 2005.
- [20] K. Hornbæk, B. B. Bederson, and C. Plaisant, "Navigation patterns and usability of zoomable user interfaces with and without an overview," *ACM Trans. Comput.-Hum. Interact.*, vol. 9, pp. 362–389, Dec. 2002.
- [21] K. Hornbæk and E. Frøkjær, "Reading of electronic documents: The usability of linear, fisheye, and overview+detail interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '01*, (New York, NY, USA), pp. 293–300, ACM, 2001.
- [22] C. Hofman, A. Maxwell, and M. McCracken, "Skim." <http://skim-app.sourceforge.net/>, 2014. [Online; accessed 20-June-2014].
- [23] J. Skinner, "Sublime text." <http://www.sublimetext.com/>, 2014. [Online; accessed 20-June-2014].
- [24] R. Spence and M. Apperley, "Data base navigation: an office environment for the professional," *Behaviour & Information Technology*, vol. 1, no. 1, pp. 43–54, 1982.
- [25] R. Spence and M. Apperley, *Bifocal Display*. Aarhus, Denmark: The Interaction Design Foundation, 2013.

- [26] Y. K. Leung, "Human-computer interface techniques for map based diagrams," in *Proceedings of the Third International Conference on Human-computer Interaction on Designing and Using Human-computer Interfaces and Knowledge Based Systems (2Nd Ed.)*, (New York, NY, USA), pp. 361–368, Elsevier Science Inc., 1989.
- [27] Y. K. Leung and M. D. Apperley, "A review and taxonomy of distortion-oriented presentation techniques," *ACM Trans. Comput.-Hum. Interact.*, vol. 1, pp. 126–160, June 1994.
- [28] P. Purgathofer, *designlehren - zur gestaltung interaktiver systeme*. habilitation, Technische Universität Wien, 2004.
- [29] J. D. Mackinlay, G. G. Robertson, and S. K. Card, "The perspective wall: Detail and context smoothly integrated," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, (New York, NY, USA), pp. 173–176, ACM, 1991.
- [30] G. W. Furnas, "The fisheye view: A new look at structured files," tech. rep., 1981.
- [31] G. W. Furnas, "Generalized fisheye views," *SIGCHI Bull.*, vol. 17, pp. 16–23, Apr. 1986.
- [32] Wikipedia, "Fisheye lens — Wikipedia, the free encyclopedia." [http://en.wikipedia.org/wiki/Fisheye\\_lens](http://en.wikipedia.org/wiki/Fisheye_lens). [Online; accessed 12-June-2014].
- [33] Mjposner, "Car fisheye." [http://en.wikipedia.org/wiki/File:Car\\_Fisheye.jpg](http://en.wikipedia.org/wiki/File:Car_Fisheye.jpg), 2012. [Online; accessed 14-June-2014].
- [34] P. Baudisch, N. Good, and P. Stewart, "Focus plus context screens: Combining display technology with visualization techniques," in *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, (New York, NY, USA), pp. 31–40, ACM, 2001.
- [35] M. Sarkar and M. H. Brown, "Graphical fisheye views of graphs," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, (New York, NY, USA), pp. 83–91, ACM, 1992.
- [36] J. Lamping, R. Rao, and P. Pirolli, "A focus+context technique based on hyperbolic geometry for visualizing large hierarchies," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, (New York, NY, USA), pp. 401–408, ACM Press/Addison-Wesley Publishing Co., 1995.
- [37] C. Plaisant, J. Grosjean, and B. Bederson, "Spacetree: supporting exploration in large node link tree, design evolution and empirical evaluation," in *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, pp. 57–64, 2002.
- [38] T. Munzner, F. Guimbretière, S. Tasiran, L. Zhang, and Y. Zhou, "Treejuxtaposer: Scalable tree comparison using focus+context with guaranteed visibility," *ACM Trans. Graph.*, vol. 22, pp. 453–462, July 2003.

- [39] B. B. Bederson, "Fisheye menus," in *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST '00, (New York, NY, USA), pp. 217–225, ACM, 2000.
- [40] M. Kersten and G. C. Murphy, "Mylar: A degree-of-interest model for ides," in *Proceedings of the 4th International Conference on Aspect-oriented Software Development*, AOSD '05, (New York, NY, USA), pp. 159–168, ACM, 2005.
- [41] R. Rao and S. K. Card, "The table lens: Merging graphical and symbolic representations in an interactive focus + context visualization for tabular information," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, (New York, NY, USA), pp. 318–322, ACM, 1994.
- [42] B. B. Bederson, A. Clamage, M. P. Czerwinski, and G. G. Robertson, "Datelens: A fisheye calendar interface for pdas," *ACM Trans. Comput.-Hum. Interact.*, vol. 11, pp. 90–119, Mar. 2004.
- [43] G. G. Robertson and J. D. Mackinlay, "The document lens," in *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*, UIST '93, (New York, NY, USA), pp. 101–108, ACM, 1993.
- [44] D. McQuail, *Mcquail's mass communication theory*. Thousand Oaks, CA: SAGE Publications, 6th. ed., 2010.
- [45] C. Thurlow, L. Lengel, and A. Tomic, *Computer Mediated Communication*. SAGE Publications, 2004.
- [46] D. Harper, "Online etymology dictionary." <http://www.etymonline.com/>, 2014. [Online; accessed 18-July-2014].
- [47] R. Burkart, "Kommunikationstheorien," in *Öffentliche Kommunikation* (G. Bentele, H.-B. Brosius, and O. Jarren, eds.), Studienbücher zur Kommunikations- und Medienwissenschaft, pp. 169–192, VS Verlag für Sozialwissenschaften, 2003.
- [48] C. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, July, October 1948.
- [49] C. Theory, "All about theories for communication.." <http://communicationtheory.org/>, 2014. [Online; accessed 18-July-2014].
- [50] R. Anderson and V. Ross, *Questions of Communication: A Practical Introduction to Theory*. Bedford/St. Martin's, 2002.
- [51] P. Ferris, "What is cmc? an overview of scholarly definitions." <http://www.december.com/cmc/mag/1997/jan/ferris.html>, 2014. [Online; accessed 21-July-2014].



- [52] R. Johansen, *GroupWare: Computer Support for Business Teams*. New York, NY, USA: The Free Press, 1988.
- [53] N. S. Baron, *Language of the Internet*, pp. 59–127. Stanford, CA, USA: CSLI publications, 2003.
- [54] E. P. Resnick, “Rfc 5322 - internet message format.” <http://tools.ietf.org/html/rfc5322>, 2014. [Online; accessed 22-July-2014].
- [55] N. Freed, Innosoft, and N. Borenstein, “Rfc 2045 - multipurpose internet mail extensions (mime) part one: Format of internet message bodies.” <http://tools.ietf.org/html/rfc2045>, 2014. [Online; accessed 22-July-2014].
- [56] P. Cooper, “Html5 weekly.” <http://html5weekly.com/>, 2014. [Online; accessed 4-August-2014].
- [57] Mozilla, “Thunderbird.” <https://www.mozilla.org/en-US/thunderbird/>, 2014. [Online; accessed 4-August-2014].
- [58] T. Campbell, “The first e-mail message.” <http://www.cs.umd.edu/class/spring2002/cmsc434-0101/MUIseum/applications/firstemail.html>, 2014. [Online; accessed 22-July-2014].
- [59] Wikipedia, “Email — Wikipedia, the free encyclopedia,” 2014. [Online; accessed 22-July-2014].
- [60] S. Radicati, “Email statistics report, 2014-2018,” 2014. [Online; accessed 22-July-2014].
- [61] Wikipedia, “Instant messaging — Wikipedia, the free encyclopedia,” 2014. [Online; accessed 22-July-2014].
- [62] Wikipedia, “talk (software) — Wikipedia, the free encyclopedia,” 2014. [Online; accessed 22-July-2014].
- [63] Wikipedia, “Smartphone — Wikipedia, the free encyclopedia,” 2014. [Online; accessed 22-July-2014].
- [64] A. Smith and F. Wempen, *CompTIA Strata Study Guide Authorized Courseware: Exams FC0-U41, FC0-U11, and FC0-U21*. CourseSmart, Wiley, 2011.
- [65] Microsoft, “Skype.” <http://www.skype.com/en/>, 2014. [Online; accessed 4-August-2014].
- [66] Wikipedia, “Electronic mailing list — Wikipedia, the free encyclopedia,” 2014. [Online; accessed 22-July-2014].
- [67] Wikipedia, “Usenet newsgroup — Wikipedia, the free encyclopedia,” 2014. [Online; accessed 24-July-2014].

- [68] Big-8 Management Board, “Big-8 usenet,” 2014. [Online; accessed 24-July-2014].
- [69] Shemes, “Grabit.” <http://www.shemes.com/>, 2014. [Online; accessed 4-August-2014].
- [70] Carnegie Mellon School of Computer Science, “Store and forward communication: Uucp and fidonet,” 2012. [Online; accessed 22-July-2014].
- [71] K. Beck, *Computervermittelte Kommunikation im Internet*. Lehr- und Handbücher der Kommunikationswissenschaft, München: Oldenbourg, 2006.
- [72] Pingdom, “Irc is dead, long live irc,” 2014. [Online; accessed 22-July-2014].
- [73] Wikipedia, “Internet relay chat — Wikipedia, the free encyclopedia,” 2014. [Online; accessed 25-July-2014].
- [74] Wikipedia, “Internet forum — Wikipedia, the free encyclopedia,” 2014. [Online; accessed 24-July-2014].
- [75] A. Weinberger and H. Mandl, “Computer-mediated knowledge communication,” 2003.
- [76] “Delphi forums.” <http://www.delphiforums.com/>, 2014. [Online; accessed 4-August-2014].
- [77] E. Makuch, “World of warcraft subscriptions on the rise,” 2014. [Online; accessed 24-July-2014].
- [78] Blizzard, “World of warcraft subscriptions on the rise,” 2014. [Online; accessed 05-August-2014].
- [79] reddit, “Frequently asked questions - reddit.” <http://www.reddit.com/wiki/faq>, 2014. [Online; accessed 4-August-2014].
- [80] reddit, “reddit - the front page of the internet.” <http://www.reddit.com/>, 2014. [Online; accessed 4-August-2014].
- [81] reddit, “reddit - about reddit.” <http://www.reddit.com/about/>, 2014. [Online; accessed 4-August-2014].
- [82] A. C. MADRIGAL, “Ama: How a weird internet thing became a mainstream delight.” <http://www.theatlantic.com/technology/archive/2014/01/ama-how-a-weird-internet-thing-became-a-mainstream-delight/282860/>.
- [83] E. Miller, “How not to sort by average rating.” <http://www.evanmiller.org/how-not-to-sort-by-average-rating.html>, 2014. [Online; accessed 11-August-2014].

- [84] R. Bromme, F. Hesse, and H. Spada, *Barriers and Biases in Computer-Mediated Knowledge Communication: And How They May Be Overcome*. Computer-Supported Collaborative Learning Series, Springer, 2005.
- [85] J. Nielsen, “10 usability heuristics for user interface design,” 2014. [Online; accessed 06-August-2014].
- [86] Pew Research Center, “Pew global attitudes survey,” 2014. [Online; accessed 22-July-2014].
- [87] C. Shirky, *Here Comes Everybody: The Power of Organizing Without Organizations*. Penguin Press, 2008.
- [88] N. Bunkley, “Joseph juran, 103, pioneer in quality control, dies.” [http://www.nytimes.com/2008/03/03/business/03juran.html?\\_r=0](http://www.nytimes.com/2008/03/03/business/03juran.html?_r=0), 2014. [Online; accessed 4-August-2014].
- [89] Apple, “Mail.” <http://www.apple.com/de/osx/apps/>, 2014. [Online; accessed 12-August-2014].
- [90] Österreichischen Computer Gesellschaft OCG, “Computerkenntnisse der österreicherinnen,” 2014. [Online; accessed 22-July-2014].
- [91] B. Kerr, “Thread arcs: An email thread visualization.,” in *INFOVIS*, pp. 211–218, IEEE Computer Society, 2003.
- [92] S. L. Rohall, D. Gruen, P. Moody, M. Wattenberg, M. Stern, B. Kerr, B. Stachel, K. Dave, R. Armes, and E. Wilcox, “Remail: a reinvented email prototype.,” in *CHI Extended Abstracts* (E. Dykstra-Erickson and M. Tscheligi, eds.), pp. 791–792, ACM, 2004.
- [93] P. Holzkorn, “Discuss. new designs for asynchronous online discussion for e-learning in higher education,” Master’s thesis, Technische Universität Wien, 2011.
- [94] Wikipedia, “Lamp (software bundle) — Wikipedia, the free encyclopedia,” 2014. [Online; accessed 27-Dezember-2014].
- [95] M. Otto, J. Thornton, and B. contributors, “Bootstrap - the world’s most popular mobile-first and responsive front-end framework.” [Online; accessed 27-Dezember-2014].
- [96] Google, *AngularJS - Superheroic JavaScript MVW Framework*, 2014. [Online; accessed 27-Dezember-2014].
- [97] SensioLabs, *Symfony, High Performance PHP Framework for Web Development*, 2014. [Online; accessed 27-Dezember-2014].
- [98] Wikipedia, “Hypertext transfer protocol - request methods — Wikipedia, the free encyclopedia,” 2014. [Online; accessed 27-Dezember-2014].

- [99] E. D. Hardt, "Rfc 6749 - the oauth 2.0 authorization framework." <http://tools.ietf.org/html/rfc6749>, 2014. [Online; accessed 27-Dezember-2014].
- [100] D. M. Freedman and M. R. Nutting, "A brief history of crowdfunding," 2014.
- [101] A. Ordanini, L. Miceli, M. Pizzetti, and A. Parasuraman, "Crowd-funding : transforming customers into investors through innovative service platforms," 2011.
- [102] E. Ries, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Business, 2011.