FAKULTÄT FÜR INFORMATIK
Faculty of Informatics

# Easy To Use Spreadsheet Application

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Medieninformatik**

eingereicht von

**Stefan Csizmazia**
Matrikelnummer 0728096

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer: Ao.Univ.Prof. Mag. Dr. Horst Eidenberger

Wien, 09.08.2014 ______________________ ______________________
(Unterschrift Verfasser) (Unterschrift Betreuer)

# Erklärung zur Verfassung der Arbeit

Stefan Csizmazia, Hütteldorferstraße 249/15, 1140 Wien

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.

Wien, 09.08.2014 ____________________

(Unterschrift Verfasser)

# Danksagungen

Viele Menschen haben diese Diplomarbeit zu dem gemacht was sie ist. Neben meinem Betreuer, Prof. Horst Eidenberger, möchte ich vor allem Claire Smith danken, die diesen Text in bewundernswerter Geschwindigkeit und Genauigkeit lektoriert hat. Besonderer Dank gilt auch all den Personen, die im Design- und Entwicklungsprozess mitgewirkt haben – besonders jenen fünf, die ich beobachten und interviewen durfte. Euer Engagement, euer ehrliches Feedback und eure Ideen waren maßgeblich für den Erfolg dieses Projekts. Zu guter Letzt danke ich auch meiner Familie, meinen Freunden und meiner Freundin Sara, die mich in den Jahren meines Studiums in jeder Hinsicht unterstützt haben und denen ich, vor allem in letzter Zeit, nicht das Maß an Aufmerksamkeit schenken konnte, das sie verdienen.

## Abstract

In the years since their first introduction in 1961, spreadsheets have developed into feature-rich multi-purpose tools, with software such as Microsoft Excel being amongst the most popular computer programs of all time. Spreadsheets are widely used in professional environments for tasks such as, inter alia, reporting and data analysis. Despite their versatility, spreadsheets are mostly used for keeping numeric and textual data in tabular form. Many sophisticated features, such as complex formulas and charts or macros, were found to be used only in a minority of use cases.

The goals of this work were to confirm findings from business-related studies for the private domain and further investigate how users work with existing spreadsheet software, which difficulties they encounter and therefore what the requirements for an improved user experience are. First, an online survey was conducted, which built upon findings of related work and served as a basis for subsequent qualitative research in the form of participant observations and semi-structured interviews with a group of real users. As a result, a web-based prototype was developed and continuously tested and evaluated in collaboration with users.

The presented prototype offers an unlimited canvas workspace instead of the traditional grid and integrates known features from spreadsheet and word processing software. Unique features such as a newly developed formula bar and indicators for formula cells along with the intelligent reduction of features led to a significant reduction of complexity, while keeping the software powerful enough to handle most day-to-day tasks.

# Kurzfassung

Seit der ersten Vorstellung des Konzepts im Jahre 1961 haben sich Tabellenkalkulationen zu einem umfangreichen Werkzeug für diverse Anwendungsgebiete entwickelt. Vertreter wie Microsoft Excel zählen zu den weit verbreitetsten Computerprogrammen überhaupt. Tabellenkalkulationen werden vor allem im professionellen Umfeld, unter anderem für Berichte oder Datenanalysen, eingesetzt. Trotz ihrer Vielseitigkeit werden sie hauptsächlich dafür verwendet, um textuelle und numerische Daten in Form von Tabellen zu organisieren und zu speichern. Viele anspruchsvollere Funktionen, wie zum Beispiel komplizierte Formeln und Diagramme oder Macros, kommen nur in einer kleinen Zahl der Anwendungsfälle zum Einsatz.

Die Ziele dieser Arbeit waren es, Forschungsergebnisse, die für den geschäftlichen Bereich erzielt wurden, für den privaten Bereich zu überprüfen, sowie genauer zu untersuchen, wie Benutzer aktuelle Tabellenkalkulationssoftware verwenden, welche Schwierigkeiten sie dabei haben und wie ein verbessertes Nutzererlebnis aussehen könnte. Aufbauend auf vorhandenen Arbeiten, wurde eine Online-Umfrage durchgeführt, deren Ergebnisse als Basis für die darauffolgenden teilnehmenden Beobachtungen und semi-strukturierten Leitfadeninterviews mit einer kleineren Benutzergruppe dienten. Die daraus gewonnenen Erkenntnisse wurden anschließend in einem benutzerzentrierten iterativen Prozess zu einem webbasierten Prototyp verarbeitet.

Der vorgestellte Prototyp setzt anstelle der traditionellen Rasterstruktur auf einen leinwandartigen Arbeitsbereich, auf dem verschiedene Elemente frei platziert werden können. Neben der Integration bekannter Funktionen aus Tabellenkalkulations- und Textverarbeitungsprogrammen, konnten unter anderem eine von Grund auf neu entwickelte Formel-Funktionsleiste und Kennzeichnungssymbole für Formelzellen in Verbindung mit einer zielgerichteten Reduktion des Funktionsumfangs dazu beitragen, die Komplexität der Software maßgeblich zu reduzieren. Gleichzeitig unterstützt der entwickelte Ansatz die meisten alltäglichen Anforderungen im privaten Bereich.

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

## 1.1 Motivation

The assumption of this work is that a majority of spreadsheet software users only use a minority of the features available and do not know, do not understand and/or do not need the rest of them. This project aims to rethink spreadsheets, make them easier to use and fit features and user interfaces better to peoples' everyday tasks.

Unlike most software products, modern spreadsheet applications essentially look like and function in the same way as their ancestors did, which have been around since the 1970s. While this could be seen as an argument for the perfection of the spreadsheet concept, research has shown that spreadsheets are error-prone and often hard to understand.

Therefore, that which has been in use for over 30 years might still not be the ideal solution for people's everyday work. Due to the vast number of features, modern spreadsheet applications are extremely powerful but also can be complex to use. It can take a long time for users to familiarize themselves with spreadsheet concepts, even if all they want to do, for example, is to organize information in a simple table.

After all, spreadsheets are just one way of storing data in tables – in a way similar to relational databases, except with instant in-place calculation of values, a concept related to the "what you see is what you get" (WYSIWYG) approach known from word processor software. While WYSIWYG in word processors works on a meta-level, i.e. formatting and layout, real-time calculation of cell values in spreadsheets operates directly on the data-level. The core of this concept has not changed much over the years; in fact what mostly changed for the better was computing power and graphics rendering facilities, thus making spreadsheets faster and aesthetically pleasing. Furthermore, many features were added to spreadsheet software packages, most of them supporting highly specialized tasks. [1]

The dominance of Microsoft Excel[1] in the field of spreadsheet applications is closely linked to the business world where it can be seen as a de-facto-standard. This dominance emanated to the private world: people often use software at home which they know from work and are familiar with, even though they might not necessarily be satisfied with it. While this behavioral pattern may still hold true for many cases, things started to change over the last couple of years with the rise of the Web 2.0 and small, easy-to-use apps for mobile devices such as Smartphones and Tablets.

[1] http://office.microsoft.com/en-001/excel/ [Accessed: 08-Aug-2014]

With the Web 2.0 it became easier for users to create content and over the years they got used to being not only consumers, but so-called “prosumers” (producer-consumer). Additionally, web-based tools for content creation focusing on usability for everybody and not only computer experts were developed.

Software programs for traditional computers tended to grow over the years with the aim of providing users with toolkits that were as comprehensive as possible. This approach took a 180-degree turn when it came to mobile apps. They tend to be small, simple, specialized and easy to understand for everybody. Furthermore, they can be found and installed via app stores with just a few clicks. A great number of them are also free, keeping the personal investment (time, effort and money) required of individual users to try out a new application and eventually adopt it extremely low[2]. With my work I tried to exploit this new mindset by providing a fresh approach to spreadsheets with a low entry barrier, hoping for users to give it a try and leave behind whatever they were unhappily using before.

“*Finally, lest there be any doubts: there is still much to learn about how people use spreadsheets, and spreadsheet ecology.*” [2]

The fact that, even after extensive literature review, I believed this statement from 1994 to be still true was another motivation for me to complete my master thesis in this field.

## 1.2 Goals

The primary goal of this work is to find solutions to make spreadsheets easier to use and enable users to accomplish their everyday spreadsheet tasks faster and with less effort. However, this goal definition does not set narrow limits in this broad area of research. Therefore exact goals and sub-goals are described below.

Microsoft Excel, being a de-facto-standard in business spreadsheet applications, is also widely used in the private sector. Competitors like LibreOffice Calc[3], Apple Numbers[4] and Google Spreadsheets[5] differ only insignificantly in terms of features, user-interface and usage concept. As experience has shown, software and infrastructure configurations in businesses are often based on long-term decisions, making it unlikely for this work to bring change in these kinds of usage scenarios. In the private area, on the other hand, people are free to decide on their software of choice, rendering this a great opportunity for a new approach to spreadsheets. This leads me to my first goal:

1. Focus research on and propose solutions for the private sphere.

[2] For example, Google announced to have reached 48 billion app installations via the Google Play Store at its 2014 Google IO conference (http://de.engadget.com/2013/05/15/google-io-900-millionen-android-aktivierungen-48-milliarden-ap/ [Accessed: 08-Aug-2014])

[3] https://www.libreoffice.org/discover/calc/ [Accessed: 08-Aug-2014]

[4] https://www.apple.com/mac/numbers/ [Accessed: 08-Aug-2014]

[5] https://docs.google.com/spreadsheet/ [Accessed: 08-Aug-2014]

In order to find a good solution, one needs to understand the problem first. Therefore my second and third goals are:

2. Identify what tasks people try to accomplish using spreadsheets;
3. Identify if and why they fail accomplishing their tasks.

While the previous two goals focus on a high-level view of user tasks, it is also important to take a closer look at the tools and how users interact with them:

4. Identify which spreadsheet software people use and why;
5. Identify which user interface (UI) elements are used and how.

Based on the results of to the above mentioned goals:

6. Define requirements for a web-based software framework prototype as a basis for small easy-to-use spreadsheet applications;
7. Build a working implementation of said prototype;
8. Continuously evaluate and refine the prototype in collaboration with real users.

I want to emphasize the importance of the last two goals: this work should not serve as another theoretical investigation, but rather aims to create a new practical approach to spreadsheets. End-users are of key importance and therefore they take center stage in this project.

## 1.3 Overview over the Thesis

Following this introduction, Chapter 0 presents the background to this work. It explains what a spreadsheet is, when the concept was invented and how it evolved over the years. Related work is presented as well as the fields of application of modern day spreadsheets along with competitors to the standard spreadsheet model that have emerged in the recent years.

Chapter 0 also describes the scientific method used in this work, mainly a user-centered design process, including quantitative and qualitative studies as well as prototyping.

Chapter 3 gives in an in-depth view of the different steps in the design and development process, i.e. design, implementation and results of the online survey, observations and interviews as well as iterative prototyping with continuous user-testing.

The final product, a web-based working prototype is described in Chapter 4, including screenshots of key features and user interface elements.

The results of a final prototype evaluation – again done using interviews and an online survey – are presented in Chapter 5. This chapter also contains a comparison between the final results of this work with existing spreadsheet solutions.

Chapter 6 concludes and outlines steps to further improve the presented prototype and possibly develop it into a full-fledged product.

# 2 Background

Firstly, this chapter aims to give sufficient background information of the history and development of spreadsheets, as well as important scientific work on that topic in order to understand the potentials and goals of this work better. Secondly it describes the scientific approach I took to tackle the challenge of making spreadsheets more user-friendly.

## 2.1 The Spreadsheet

"*A spreadsheet is an interactive computer application program for organization and analysis of data in tabular form.*" [3]

The word "spreadsheet" is often used for multiple different things. For the rest of this thesis I will differentiate between "spreadsheet" and "spreadsheet program" or "spreadsheet software". Spreadsheet programs provide functionalities and the user interface to create spreadsheets. A spreadsheet on the other hand is the sum of data, structure, multimedia elements, such as charts and images, computation routines (formulas) and formatting. Speaking in the words of Microsoft Office: Excel is a spreadsheet program whereas an `xls`-file is a spreadsheet.

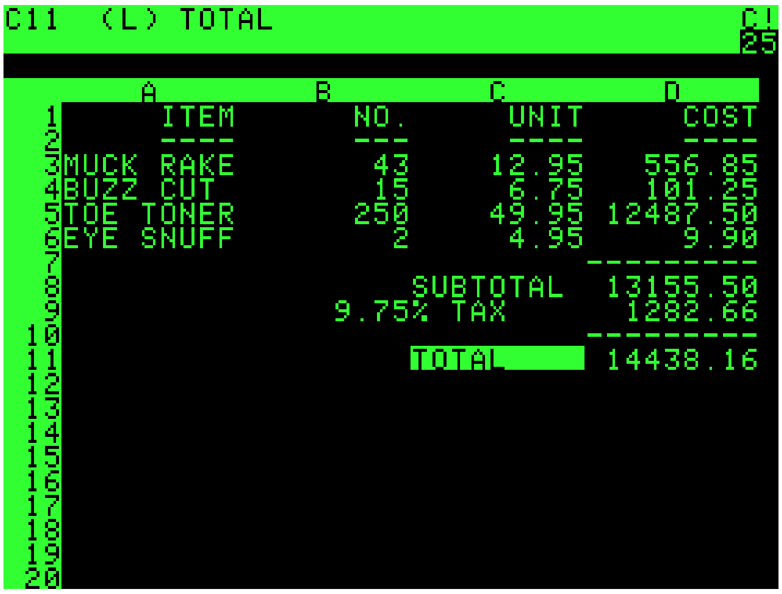
A spreadsheet usually consists of a finite number of cells organized in a grid of rows and columns, where rows are enumerated with numbers (1, 2, 3, 4, …) and columns are enumerated with letters (A, B, C, D, …). Cells may either contain static numeric or textual data or formulas. The latter define the value of a specific cell based on the value of other cells or constant values. Along with basic algebraic operations like addition, subtraction, multiplication, etc. there usually exist a number of functions to be used in formulas, e.g. sum, average, etc. Calculated cell values are automatically updated as soon as a referenced cell changes.

However, there are also other, more formal definitions of what a spreadsheet is, for example the so-called FFR (formulae, formats, relations) model by Sajaniemi. [4]

### 2.1.1 History of Spreadsheets

Computerized spreadsheets were initially developed to make accounting work easier. Therefore the user interface and workflow paradigm tried to mimic the paper worksheets that were used before. The first concept of electronic spreadsheets was proposed by Richard Mattessich in 1961. In the years after that, some implementations for mainframe computer systems were used, mainly at universities and in the financial sector. At this time computers, and therefore spreadsheets, could only be operated and used by a small number of experts.

It was in the late 1970’s when the first program made the concept of spreadsheets available for a broader user base. The software was called “VisiCalc” and it ran on both popular Apple II and IBM PC computers. VisiCalc already employed modern concepts and features such as a WYSIWYG (“what you see is what you get”) UI, formulas with cell references and automatic recalculation of cell values. Figure 1 shows a screenshot of VisiCalc running on an Apple II machine.

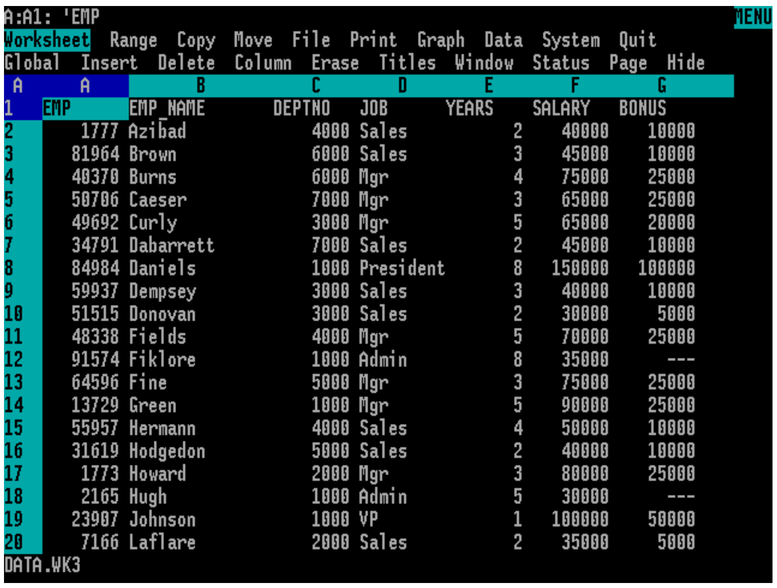


```
C11  (L) TOTAL                                C!
                                              25

          A         B         C         D
 1      ITEM       NO.      UNIT      COST
 2      -----      ---      ----      -----
 3MUCK RAKE         43     12.95    556.85
 4BUZZ CUT          15      6.75    101.25
 5TOE TONER        250     49.95  12487.50
 6EYE SNUFF          2      4.95      9.90
 7                                ---------
 8                      SUBTOTAL  13155.50
 9                 9.75% TAX       1282.66
10                                ---------
11                      TOTAL     14438.16
12
13
14
15
16
17
18
19
20
```

Figure 1: Screenshot of VisiCalc for Apple II[6]

As personal computers became increasingly popular starting in the early 1980’s, other spreadsheet programs were developed and eventually became bestsellers. Lotus 1-2-3 for example, soon became the leading spreadsheet product for MS-DOS. Figure 2 shows a screenshot of Lotus 1-2-3 version 3.0 running on an IBM PC with DOS operating system.

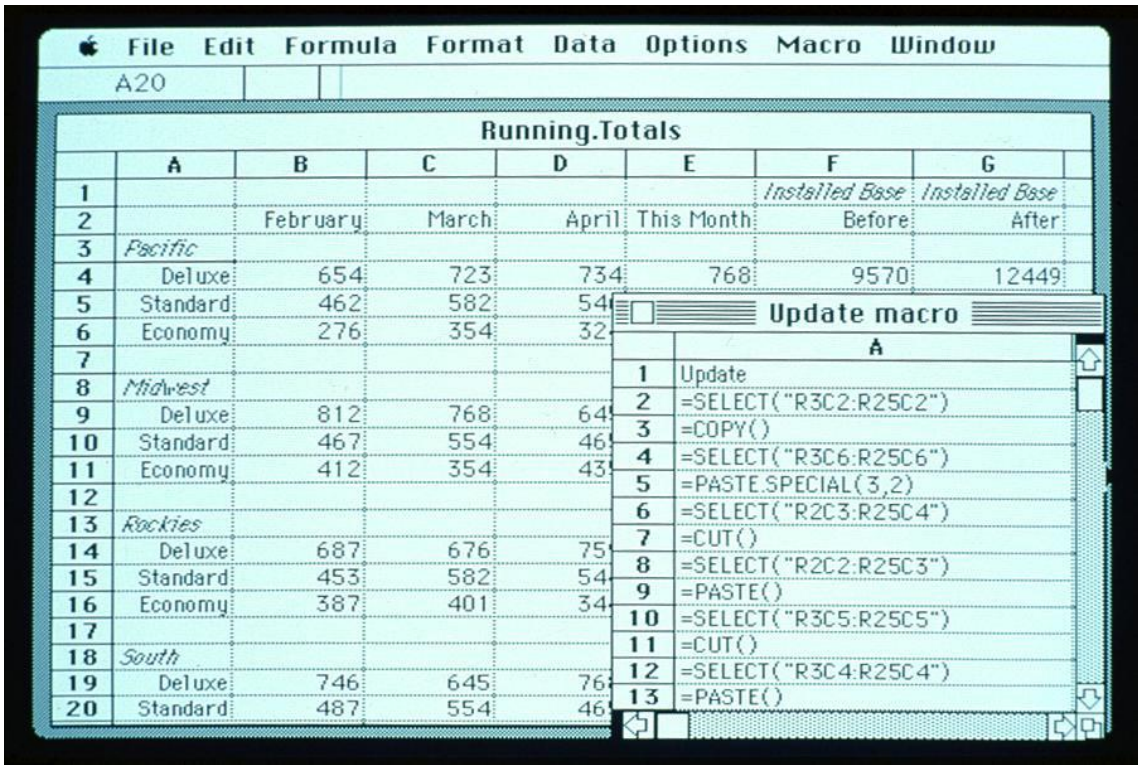


```
A:A1: 'EMP                                                          MENU
Worksheet  Range  Copy  Move  File  Print  Graph  Data  System  Quit
Global  Insert  Delete  Column  Erase  Titles  Window  Status  Page  Hide
A    A          B             C       D          E        F         G
1  EMP      EMP_NAME      DEPTNO   JOB        YEARS   SALARY    BONUS
2       1777 Azibad           4000 Sales          2    40000    10000
3      81964 Brown            6000 Sales          3    45000    10000
4      40370 Burns            6000 Mgr            4    75000    25000
5      50706 Caeser           7000 Mgr            3    65000    25000
6      49692 Curly            3000 Mgr            5    65000    20000
7      34791 Dabarrett        7000 Sales          2    45000    10000
8      84984 Daniels          1000 President      8   150000   100000
9      59937 Dempsey          3000 Sales          3    40000    10000
10     51515 Donovan          3000 Sales          2    30000     5000
11     48338 Fields           4000 Mgr            5    70000    25000
12     91574 Fiklore          1000 Admin          8    35000      ---
13     64596 Fine             5000 Mgr            3    75000    25000
14     13729 Green            1000 Mgr            5    90000    25000
15     55957 Hermann          4000 Sales          4    50000    10000
16     31619 Hodgedon         5000 Sales          2    40000    10000
17      1773 Howard           2000 Mgr            3    80000    25000
18      2165 Hugh             1000 Admin          5    30000      ---
19     23907 Johnson          1000 VP             1   100000    50000
20      7166 Laflare          2000 Sales          2    35000     5000
DATA.WK3
```

Figure 2: Screenshot of Lotus 1-2-3 3.0 for DOS[7]

[6] http://en.wikipedia.org/wiki/VisiCalc#mediaviewer/File:Visicalc.png [Accessed: 08-Aug-2014]

With the advent of graphical user interfaces, Microsoft could gain market shares with their own product called “Excel”. The latter is the most popular spreadsheet application today. Figure 3 shows a screenshot of an early version of Excel running on a Macintosh computer.



| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | *Installed Base* | *Installed Base* |
| 2 | | February | March | April | This Month | Before | After |
| 3 | *Pacific* | | | | | | |
| 4 | Deluxe | 654 | 723 | 734 | 768 | 9570 | 12449 |
| 5 | Standard | 462 | 582 | 54 | | | |
| 6 | Economy | 276 | 354 | 32 | | | |
| 7 | | | | | | | |
| 8 | *Midwest* | | | | | | |
| 9 | Deluxe | 812 | 768 | 64 | | | |
| 10 | Standard | 467 | 554 | 46 | | | |
| 11 | Economy | 412 | 354 | 43 | | | |
| 12 | | | | | | | |
| 13 | *Rockies* | | | | | | |
| 14 | Deluxe | 687 | 676 | 75 | | | |
| 15 | Standard | 453 | 582 | 54 | | | |
| 16 | Economy | 387 | 401 | 34 | | | |
| 17 | | | | | | | |
| 18 | *South* | | | | | | |
| 19 | Deluxe | 746 | 645 | 76 | | | |
| 20 | Standard | 487 | 554 | 46 | | | |

| | A |
|---|---|
| 1 | Update |
| 2 | =SELECT("R3C2:R25C2") |
| 3 | =COPY() |
| 4 | =SELECT("R3C6:R25C6") |
| 5 | =PASTE.SPECIAL(3,2) |
| 6 | =SELECT("R2C3:R25C4") |
| 7 | =CUT() |
| 8 | =SELECT("R2C2:R25C3") |
| 9 | =PASTE() |
| 10 | =SELECT("R3C5:R25C5") |
| 11 | =CUT() |
| 12 | =SELECT("R3C4:R25C4") |
| 13 | =PASTE() |

Figure 3: Screenshot of Microsoft Excel on Macintosh[8]

During the last few years, the Internet has grown in size (in terms of connected computers), users and, with that, attention. The focus on and rapid development of Internet technologies enabled new spreadsheet applications that run on servers and can be used via standard web browsers. Applications such as Google Spreadsheets and Microsoft Office Web Apps contain most of the features of standalone spreadsheet software products, but also introduce new and innovative features, for example collaborative editing.

Spreadsheets were the first “killer-app” in the field of personal computing and a major reason for many people to buy a personal computer. [1] This helped increase both the prevalence of personal computers in private households, as well as the spreadsheet itself because this was the very first way of “computing without coding”, i.e. a way for computer end-users to create their own computing routines without needing to know any programming language. In that way spreadsheets massively increased the everyday value of personal computers for non-expert users, which, in that time, was primarily defined over productive software rather than entertainment.

[7] http://en.wikipedia.org/wiki/Lotus_1-2-3#mediaviewer/File:Lotus-123-3.0-dos.png
[Accessed: 08-Aug-2014]

[8] http://www.flickr.com/photos/microsoftsweden/5394685465/sizes/o/in/photostream/
[Accessed: 08-Aug-2014]

Figure 4 gives an overview of major events in spreadsheet history, as well as the main environments/platforms used at certain times. As mentioned above, the first concept for computerized spreadsheets was introduced in 1961. Nearly 20 years later, in 1979 VisiCalc entered the market; Lotus 1-2-3 (1983) and Microsoft Excel (1985) followed soon after. In the mid-2000s web-based spreadsheet programs, for example Google Spreadsheets were introduced and together with Microsoft Excel they are state-of-the-art today.

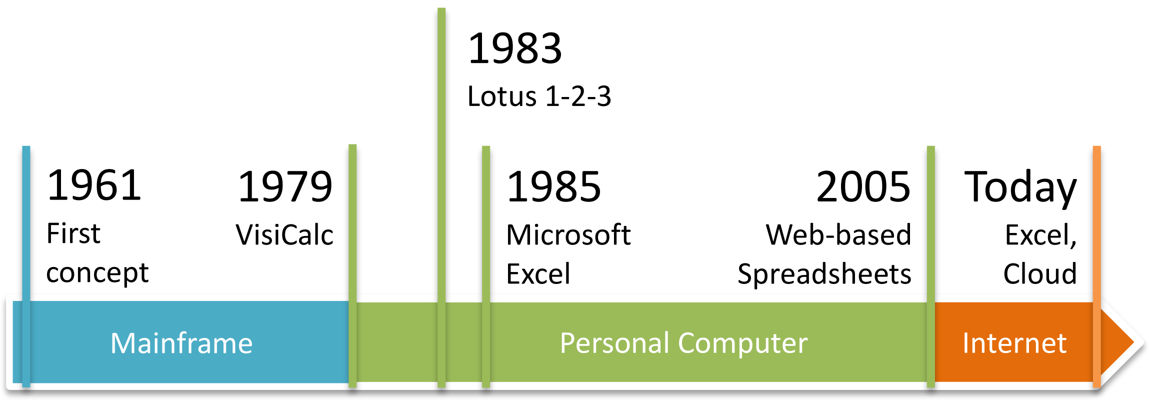


Figure 4: Timeline of spreadsheet history

Table 1 lists major spreadsheet programs along with important features they invented first and other mentionable contributions. The list focuses on programs that played an important role in spreadsheet history and have (or had) a significant market share.

Table 1: Major spreadsheet programs and their important contributions to historical spreadsheet development

| **Spreadsheet Program** | **Major contributions** |
|---|---|
| VisiCalc | • What-you-see-is-what-you-get user interface (WYSIWYG UI)<br>• Formulas with cell references<br>• Automatic recalculation of cell values |
| Lotus 1-2-3 | • Macros (possibility to record keystrokes and replay them automatically)<br>• Charts |
| Microsoft Excel | • First successful spreadsheet software with a graphical user interface (GUI) providing superior user experience compared to predecessors<br>• Still the dominant spreadsheet software today |
| OpenOffice/LibreOffice | • Open source implementation of many Excel features |
| Google Spreadsheets | • Most comprehensive web-based spreadsheet implementation today |

A special aspect of VisiCalc was the developers' approach to usability. Every sold copy of the software was shipped with a paper reference card. Any feature that could not be concisely explained on that card, which had very limited space, would not be included in the software. This approach kept VisiCalc simple, easy to use and free of any feature-bloat. [5]

Compatibility was – and still is – key. In times prior to software patents this led to spreadsheet programs that looked and felt almost the same, in fact VisiCalc, Lotus 1-2-3 and some other competitors even had a nearly identical set of keyboard shortcuts. [5] Considering this was at a time when graphical user interfaces (GUIs) were not invented yet and the keyboard was the only input device, it weighs even heavier. Today compatibility is largely defined and managed by file formats, while spreadsheet software manufacturers try to give their products original features to create unique selling points (USPs).

### 2.1.2 Fields of Application

Shortly after their invention, spreadsheets were praised as being the ideal tool for end-users. It was said that the clear conceptual model of spreadsheets would help users identify what their task is and finish it in the simplest way. Or as Norman put it: "*[…] providing just the right tools for a surprising variety of applications*". [2]

Their usefulness for business applications, e.g. financial reporting and data analysis, made spreadsheets popular. Outside of the financial sector, however, spreadsheets are also widely used in other organizations.

Reporting is a main field of application for spreadsheets. Formatting and charts are of special importance here because data needs to be presented in an appealing way. Once used for one thing, spreadsheets end up playing a major role in management and organization especially in small and mid-size companies. Gable et al. did a case study investigating spreadsheet investment, criticality and control in a Singapore-based company showing that, while the organization and its employees depended heavily on spreadsheets, the level of awareness of that fact and controlling efforts made was low relative to investment and criticality. [6]

Another, rather exotic example for the professional use of spreadsheets is given by Hihn et al. In their paper the authors describe how NASA uses spreadsheets in most of their missions, e.g. for managing parts lists and requirements, monitoring progress and budget planning. They also observed that there is usually very little planning and documentation when it comes to spreadsheet creation. [7]

That spreadsheets could even be part of computer supported cooperative work (CSCW) environments was shown by Ginige et al. who employed an empirical process to gather and specify requirements for a web-based spreadsheet mediated business collaboration system. [8]

In 2010 Chambers et al. [9] programmatically analyzed over 400 spreadsheets and created a statistic regarding which elements were present. In addition they worked through over 200 online forum threads containing questions and answers about Excel. Table 2 lists 5 classes of spreadsheets they could identify.

Table 2: Types of spreadsheets identified by Chambers and Scaffidi [9]

| **Class** | **Description** | **Frequency of occurrence** |
|---|---|---|
| Data entry | Mainly numeric data, only a few formulas and charts, no form elements | 56% |
| Database | Mainly textual data in tabular form | 25% |
| Data form | Used for data input, a lot of form fields and explaining text | 8% |
| Data mix | A mixture of the above three | 7% |
| Data viz | Used for data visualization with charts | 4% |

These results show that a vast majority of spreadsheets created do not include advanced features and are mainly used for managing numeric and textual data in a tabular form.

Other, more specific usage scenarios which also showcase what can be done with spreadsheets are presented in “Rapid development of spreadsheet-based web mashups” and “Social Spreadsheet”. The authors describe ways for building web data mash-ups without the need for programming skills and how spreadsheets can be used for extensive social media data analysis, respectively. [10], [11]

### 2.1.3 Fields of Interest in Spreadsheet Research

Given how great a success they were from very early on, spreadsheets sparked the interest of scientific researchers in the 1980s. Subsequently, a lot of fundamental work was done in the late 1990s and early 2000s. In 2005, Fisher and Rothermel collected and published the EUSES Spreadsheet Corpus, a set of real-world spreadsheets to be used by researchers for standardized testing and evaluation of tools and methodologies. [12]

Figure 5 shows a graphical representation of the distribution of three main topics of interest in spreadsheet research.

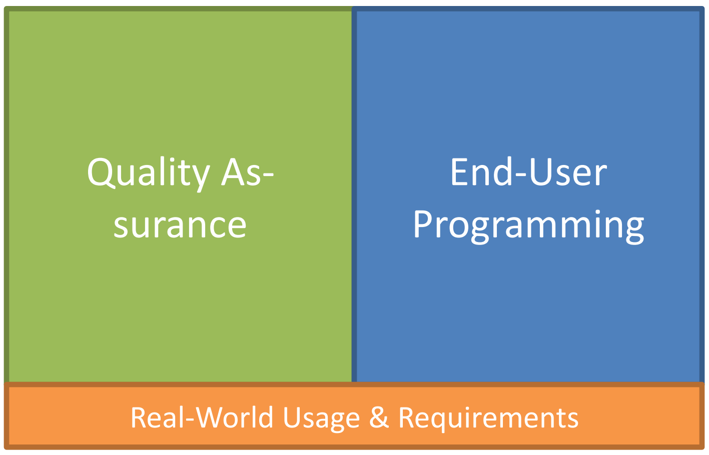


Figure 5: Distribution of main fields of interest in spreadsheet research

Spreadsheet research focuses largely on two topics: quality assurance and end-user programming. Only a small number of papers have been published investigating how spreadsheets are used for day-to-day routines and what the requirements of users are. Hendry and Green criticized the common approach to spreadsheet research in their 1994 paper, saying that most studies they reviewed analyzed the reasons for the success of the spreadsheet paradigm assuming that it actually was a success, but failed to explore weaknesses. [2]

#### 2.1.3.1 End User Programming

To understand what end-user programming is and why it is of such great interest to spreadsheet research, it is important to give some definitions first.

The author of “Cultural Differences and End-User Computing” [13] defined the term “end-user” as a user of end-user-software who possibly does end-user programming (e.g. spreadsheet formulas) to accomplish their tasks. Another definition was given by Sajaniemi [14], where he describes end-users as people who use computers as tools to support their actual task, the usage itself not being their main task. Ko et al. [15] on the other hand, claim that an “end-user” is not solely defined by skillset, training and tools of a person, but by their intent to perform a certain task. For example, a professional software developer who does her private household accounts in Excel might still be regarded end-user for certain tasks according to this definition, even though she has expert knowledge in the field of software engineering.

For 2003 the number of people who used computers at workplaces in the US, thus being potential end-user programmers, was estimated to be around 80 million, showing the importance of this field. [16] Rieman et al. [17] argue that work place software is increasingly developed by end-users themselves. This argument seems valid, especially given the latest developments of Internet technologies: an increasing amount of software today runs inside web-browsers and the online users’ role has evolved from solely being a consumer towards content and information creation.

However, end-user programmers have no interest in software development methodology and best-practices. They do not write specifications for their programs (e.g. spreadsheets) and have no motivation to do so. [18] This of course has positive and negative aspects, as Rosson pointed out in her talk at the OOPSLA’05 conference:

> *“Many people now construct software on their own, building artifacts that range from email filters to spreadsheet simulations to interactive web applications. These individuals are use-developers: they build ad hoc solutions to everyday computing needs. Will use-developers help to resolve the software crisis? Given the right tools, people and groups may be able to rapidly develop custom solutions to many context-specific computing requirements, eliminating the wait for IT professionals to analyze and engineer a solution. Or are these individuals a danger to society? Use-developers are informal programmers with no training in software construction methods or computing paradigms. They have little intrinsic motivation to test their products for even basic concerns like correctness or safety.”* [19]

Professional software engineers usually develop solutions in a process similar to the following:

1. Understand the problem;
2. Build cognitive model of solution and partial solutions;
3. Develop plan for realizing the model.

End-user programming typically does not include the second step because end-users do not have sufficient knowledge in systems design. [20] Also, the discipline of software engineering comes up with a set of guidelines and rules to stick to, in order to minimize errors, while the creation of spreadsheets is often informal and iterative. [21]

The spreadsheet is probably the most famous end-user programming environment. It could be seen as a very "flat" functional programming language with the scheme `variable=formula()`. It does not offer procedures or even object-oriented concepts like other programming languages. In an attempt to make complex computations easier and more reusable, Jones et al. presented an approach where user-defined functions are specified as separate sheets together with input and output definitions. [22] This allows for algorithms more complex than standard formulas, including several intermediate steps. However, the fact that their concept integrates standard spreadsheet functionality with software engineering principles might make it hard to be understood by end-users.

Other exemplary approaches to making software development even easier and less error-prone for end-users are example-driven modeling and literate programming. [23], [24], [25]

#### 2.1.3.2 Quality Assurance

Testing and requirement specification, two main aspects of software quality assurance, are still unsolved problems when it comes to end-user programming. However, having proper quality assurance can be vital, especially in the business domain where erroneous software can have severe consequences.

Humans err. While this is not news, it is nevertheless important to incorporate this fact into system and user interface design. Human error research, a field of cognitive science, knows the base error rate (BER) to be approximately somewhere between 0.6% and 10% for non-trivial cognitive activities of a particular type. BERs are calculated as long-term averages across many people. [26] In spreadsheet research the cell error rate (CER) and – more specifically – the cell error rate of formula cells (CERF) as well as the cell error rate of cells holding static values (CERV) are of interest. Panko et al. [26] could show that CERs found in spreadsheet research are certainly consistent with BERs found in the human error research literature. They conclude that it is rather unlikely for spreadsheets to not contain errors because of two issues:

1. Creating a spreadsheet is a non-trivial cognitive task. Therefore, the CER is likely to be as high as for other such tasks observed in human error research.
2. Spreadsheets typically contain formulas and therefore dependencies between cells could possibly render the result of a formula cell incorrect, even if the formula itself is

correct. Because of such dependencies, a CER of above 0.5% is already critical. [26], [27]

There are two different approaches to tackling the problem of erroneous spreadsheets: a) help users find and correct errors; and b) try to avoid them in the first place.

The most famous approach was developed and refined over several years by Rothermel et al.: “what you see is what you test” (WYSIWYT). Their system’s name was definitely inspired by the widely used “what you see is what you get” paradigm known from, for example, word processors. However, it also contains the major aspect of the tool – error visibility (it is hard to correct an error that you don’t even see). WYSIWYT is a tool to help users debug their own spreadsheets without having to use additional software, e.g. an error console. It adds a small dropdown menu to each cell which lets the user declare the content of a cell to be alright (OK) or faulty (NOK). All rated cells as well as their dependencies are highlighted in different shades of green or red, respectively, depending on how high the confidence is. This color-coding lets the user identify faulty cells easily and also gives a good overview over the test coverage. User tests of this prototype implemented in Excel and Forms/3 have shown that the number of correct spreadsheets was not significantly higher compared to spreadsheets created without WYSIWYT. However, they saw the time needed by test users to create the spreadsheets decreasing. [18], [28], [29], [30]

Other researchers suggested refining the WYSIWYT rating algorithm by adding additional choices to the dropdown menus (e.g. “seems right maybe”, “seems wrong maybe”), encouraging users with less confidence or knowledge to also use the tool. [31]

Since it is rather questionable whether end-user programmers are willing to use tools such as WYSIWYT, some researchers came up with other approaches, for example the automatic generation of spreadsheets from templates. While the templates contain all structure, formulas etc. and are created by professional software engineers, the users are limited to filling in cell data, adding rows and columns and the like. [32] The downside of this is clearly limited flexibility.

### 2.1.4 Modern Spreadsheet Competitors

The classical spreadsheet application runs on a personal computer and is a multi-purpose tool that supports users with many of their tasks, given an adequate amount of training and curiosity. Today, there exist a number of – mostly web-based – applications that cover similar fields of application as classical spreadsheets, most of them specializing in one specific area. With the advent of mobile handheld devices and mobile apps, this specialism even increased, thus we can see a trend from all-in-one multi-purpose tools towards smaller applications that try to master one thing. Examples of modern applications whose functionality could largely be represented as spreadsheets are:

- To-do list (e.g. Any.do[9]);

[9] http://www.any.do/ [Accessed: 08-Aug-2014]

- Travel planning (e.g. Tripwolf[10], Tripcake[11]);
- Finance and expenses (e.g. Splitwise[12]).

Not only are there highly specialized mobile apps, but also the classical spreadsheet paradigm was adapted for handheld devices, introducing a new set of challenges for user interface and interaction designers. [33]

As of today, there are dozens of available spreadsheet products, both web-based and for personal computers. Most of them have in common that they try to mimic user interface and behavior of the market leader Microsoft Excel and also advertise with the support of Microsoft Office file formats.

### 2.1.5 Conclusion

This section gave an overview of the history of the spreadsheet paradigm, how it evolved over the years and which events mutually influenced the development of spreadsheet software. Spreadsheet programs were designed to be multi-purpose tools, thus explaining the plethora of situations in which spreadsheets are used. However, spreadsheets are mainly used to keep numeric data in a tabular form, mostly without complex formulas, charts, etc. A review of related work showed end user programming and quality assurance to be the two main fields of interest in spreadsheet research. The last part of this section therefore explained how modern applications and services try to replace spreadsheets for specific use-cases.

## 2.2 Method

This section describes the scientific method used in this project to fulfil the goals specified in Chapter 1. In my literature review I could identify only a very small amount of work investigating how users work with spreadsheet software and what their problems are. As far as I am aware, none of the research activities focused on the private domain. Confronted with this lack of knowledge, I had to conduct my own user research to be able to define a set of requirements for a new, easy-to-use spreadsheet application.

When I started this thesis, it was clear that this would not be like other classical computer science projects where requirements are more or less well specified and the task is to find a well-performing solution, e.g. a new sorting algorithm for a certain edge case. All I had to work with was the assumption that currently available spreadsheet applications were too complex for the average user. I did not know what the actual problems were, or what possible solutions could look like. Therefore, I decided to employ a user-centered design process to utilize my main resource the best way possible: the actual users.

[10] http://www.tripwolf.com/ [Accessed: 08-Aug-2014]

[11] http://www.tripcake.at/ [Accessed: 08-Aug-2014]

[12] https://www.splitwise.com/ [Accessed: 08-Aug-2014]

### 2.2.1 User-Centered Design Process

User-centered design is a broad term, originally used by researchers at Donald Norman's laboratory at the University of California San Diego, describing a process where end-users influence and contribute to the development of new artifacts. It is not so much of importance *how* users are involved in the design process, but that they *are* involved. The concept does not provide a strict sequence of steps, but includes a variety of tools and methods to be employed during the design and development process. Users could, for example, be involved in the phase of requirements gathering or even work closely together with designers throughout the whole design process. In any case, the goal is to incorporate domain and personal knowledge that only users have into the product design. [34]

Elizabeth Sanders, a social scientist who also worked as a researcher in user-centered design processes from the 1980s onwards, listed three major roles in her article "From User-Centered to Participatory Design Approaches" [35]:

- Designer;
- Social Scientist/Researcher;
- User.

She described the social scientist to serve as an interface between designer and users who are "not really a part of the team". Social scientists were responsible for primary and secondary data gathering while designers interpreted this information and tried to incorporate it into their designs using sketches and scenarios. Those two roles were seen as "distinct, yet interdependent". However, she reported on a changing mindset:

> *"But I can see now, at the end of 1999, that there is a common ground, a new territory being formed by the reciprocal respect between designers and the social scientists. […] In participatory experiences, the roles of the designer and the researcher blur and the user becomes a critical component of the process."* [35]

From my own experience with this thesis and other projects, her prediction holds true: I took on the role of both designer and social researcher and closely integrated a relatively small group of users into my whole development process.

The key to a successful user-centered design process is to acknowledge the distribution of different kinds of expert knowledge. While engineers and designers are experts in their respective fields, end-users, i.e. the people a product or system is built for, have expert knowledge about the environment the artifact is going to be used in. For example, the designer of a TV remote control knows about the plethora of features of the TV set and designs a device that is packed with buttons to support all those features. The TV user on the other hand might only need buttons for turning the device on/off and controlling the volume. She would be overwhelmed by the remote control shipped with her new TV set. This example might be oversimplified, but it brings the main point across: In most cases an end-user would use a system in a completely different way to its designer.

> *"The role of the designer is to facilitate the task for the user and to make sure that the user is able to make use of the product as intended and with a minimum effort to learn how to use it."* [34]

However, this quote from "User-Centered Design" makes the role definition for a designer sound simpler and clearer as it is in reality. Therefore, I want to further analyze the three main parts of above definition:

1. Facilitate the task for the user.

This job already requires plenty of knowledge and understanding. First of all, and this may already be the hardest part, it might not even be clear what the user's task is nor who counts as a user, i.e. who the artifact is to be designed for. The designer needs to narrow down and define the target audience before doing a task-analysis to find out what the problems are that she wants to solve with a particular design.

2. Make sure that the user is able to make use of the product as intended.

Another possible pitfall is hidden in this part of the designer's work because the above sentence could be completed in two different ways: […] make use of the product as intended *by the designer* or […] make use of the product as intended *by the user*. In fact, intentions of users and designers often differ for various reasons, for example domain knowledge, experience and technical knowledge. Within the user-centered design process, designers try to close that gap using interviews, observations and other techniques.

3. With a minimum effort to learn how to use it.

Again, this part of the designer's job requires knowledge about the users which has to be built using interviews, observations, etc. This way, adoption effort can be reduced by incorporating existing knowledge into the new product, for example by using familiar icons and user-interface elements.

The integration of users into a design process can happen at various stages, for example at requirements specification or testing, each requiring different tools and methods. Also, there are different ways to learn from people about their experiences and needs. Among others there are for example:

- Listening to what people say;
- Observing and interpreting what people express;
- Watching what people do;
- Observing what people use.

To choose the right methods and tools it is important to understand that there are different types of knowledge that a user has. First, there is explicit knowledge, i.e. what people are able to express in words. However, while listening to what people say is of course important and can help to gain insight into their habits, it is also necessary to consider that people may only say what they want the researcher to hear. For example, employees of a company who are interviewed about how they use certain software might not tell the full truth because they are scared of consequences from their manager. Second, there is observable knowledge, which can be drawn from observing what people do, for example how they use products as well as their body and facial expression during an interview. Trained researchers may even go deeper and try to find out what people feel in certain situations, which gives the design-team the ability to empathize with the user. The latter is called tacit knowledge, a kind of knowledge which cannot be easily expressed in words. [35]

The following methods are part of typical user-centered design processes and help to build the different types of knowledge mentioned above. The list is not exhaustive, but it includes many popular primary data gathering methods as well as explorative design-generating methods.

- Interviews;
- Observations;
- Personas;
- Scenarios;
- Design Games;
- Prototyping.

All of these methods belong to the group of qualitative research methods, i.e. they are conducted with a relatively small number of participants (compared to large-scale quantitative questionnaires, for example). For each participant, the researcher invests a relatively large amount of time and tries to react and adapt to each person individually in order to gather as much relevant information as possible.

Preece et al. [34] suggested "*ways to involve users in the design and development of a product*" using the above mentioned methods, thus forming the user centered design process:

Table 3: Involving users in the design process [34]

| Method/Technique | Purpose | Process Stage |
|---|---|---|
| Background interviews and questionnaires | Collecting data related to the needs and expectations of users; evaluation of design alternatives, prototypes and the final artifact | At the beginning of the design project |
| Sequence of work interviews and questionnaires | Collecting data related to the sequence of work to be performed with the artifact | Early in the design cycle |
| Focus groups | Include a wide range of stakeholders to discuss issues and requirements | Early in the design cycle |
| On-site observation | Collecting information concerning the environment in which the artifact will be used | Early in the design cycle |
| Role Playing, walkthroughs, and simulations | Evaluation of alternative designs and gaining additional information about user needs and expectations; prototype evaluation | Early and mid-point in the design cycle |
| Usability testing | Collecting quantities data related to measurable usability criteria | Final stage of the design cycle |
| Interviews and questionnaires | Collecting qualitative data related to user satisfaction with the artifact | Final stage of the design cycle |

#### 2.2.1.1 Interviews

Interviews are a classical form of qualitative data gathering, well-known and adopted in social sciences, for example ethnographic research. Different kinds of interviews are used in all kinds of scientific disciplines, including standardized interviews, semi-structured interviews and expert interviews, to name just a few. In user-centered design processes for software products, semi-structured interviews often are the method of choice, and so they were in my thesis.

For semi-structured interviews, a guideline is created by the researcher that contains some opening questions to start the conversation, as well as blocks of questions for each topic/subtopic of interest. The questions do not necessarily have to be formulated as full sentences; a list of keywords can be sufficient, supporting a more natural conversation during which the researcher does not have to read all the questions from paper, but can phrase them as appropriate. Ideally, the guidelines contain opening questions for each topic plus a number of more detailed questions that can be asked in the course of the discussion, depending on how the interviewee responds. Closed questions, i.e. questions that can only be answered with yes or no, as well as suggestive questions should be avoided to support a lively conversation. Using interview guidelines can help the researcher structure the interview as well as the analysis later on while allowing for a relatively open conversation. [36]

#### 2.2.1.2 Observations

As stated above, it is not guaranteed that interviewees will divulge everything on a topic they are asked about in interviews. Also, people do a lot of things automatically (using a well-known computer program or device, for example) and trying to find out about such things solely via interviews may be unsatisfactory. Thus, for the sake of a complete, holistic picture, observable knowledge must also be gathered. [37]

Observations are, as the name implies, the right tool for this task. As with interviews, there are different kinds of observations that have their own characteristics and serve different purposes. Table 4 lists the various kinds along with their advantages and disadvantages. The extent of researcher participation can range from non-participatory to complete participation where the researcher is a member of the group studied.

Due to the large amount of preceding work necessary for active and complete participation observations, they are hardly used in user-centered software development projects. Moderate participation observations provide a good balance between objectivity, i.e. seeing a situation through the eyes of an outsider, and being able to talk to people about ambiguous situations as they occur. The dilemma of maintaining the optimal distance in field research can also be approached by employing two researchers at once, one in the role of “participant-as-observer” and the other in the role of “observer-as-participant”. [42]

Table 4: Types of participant observations [38], [39], [40], [41]

| **Type of observation** | **Level of participation** | **Characteristics** |
|---|---|---|
| Non-participatory | The researcher is not present at all. | Can be achieved with video recording, for example. The researcher is not able to immediately talk to people about specific situations that may need clarification. There is no risk of the researcher altering the interaction, even though people might still behave differently when knowing that they are being filmed. However, filming people without their consent is may be debatable from an ethical standpoint. |
| Passive participation | The researcher is on site but does not actively take part in any interactions. | The researcher is limited in asking questions, as she acts only as a bystander. The ability to move around on site may help to observe more than it is possible with a statically mounted video-camera. |
| Moderate participation | The researcher is on site, stays in the background as much as possible but enters conversations and asks questions as necessary. | The researcher has maximum control over her ambivalent role. Clarifying unclear situations as they occur is feasible. However, the “dilemma of distance” can be an issue. [42] |
| Active participation | The researcher becomes part of the observed group and participates in their activities. | A lot of preparation is necessary for the researcher to integrate into the group and not act like a foreign body. She needs to acquire domain-specific skills and embrace the groups’ habits. On one hand, such tight bonds reduce the chance of observation targets behaving differently during the observation. On the other hand, the risk of the researcher being biased and subjective increases dramatically. [43] |
| Complete participation | The researcher is already a part of the observed group beforehand. | The risk of losing objectivity is extremely high. It is hard to observe “new things”. |

#### 2.2.1.3 Personas and Scenarios

Personas are a way of consolidating knowledge about a target audience into virtual characters. They usually combine features and characteristics of multiple real people which have been discovered using data gathering methods, for example interviews and observations. This explorative method helps creating “typical reference users” that can serve as a benchmark for the design of an artifact. Each persona represents a segment of the potential target user group and typically is described in a compact document including personal details (e.g. name, age, gender, family situation, job, hobbies, daily routine) and details relevant to the particular design (e.g. usage habits, technology problems and needs, potential use cases). Giving such – otherwise rather abstract – data a human face and even a name, can help designers to empathize and put knowledge about users in perspective. [44], [45]

The benefit of personas even increases when used together in conjunction with scenarios. Scenarios describe fictional or real situations in which end-users are confronted with a certain task related to the design. Negative scenarios can help in designing for critical situations. Different levels of detail are possible, ranging from in-depth scenarios describing single user interactions, to wider scenarios describing the user experience during the whole product lifecycle from purchase to replacement. They differ from technical use-case definitions, as they are written as narratives in free text form and technical implementation details are not of interest. Therefore, scenarios (together with personas) provide a good communication base for different stakeholders during the design and development process (e.g. users, designers, developers, marketing experts, managers). [46], [47], [48]

#### 2.2.1.4 Design Games

Another explorative design-generating method is the design game. These are used to approach problems in a playful manner together with end-users and other stakeholders to discover possible usage and interaction difficulties or user preferences that they perhaps were not aware of before playing the game. Therefore, design games are a convenient and inexpensive way of confirming and enriching knowledge gathered with other methods. The shape of these games can vary from self-made board games to card games enriched with digital content or other design mockups to multiplayer online games. Brandt gives various examples of successful design game implementations in her 2006 paper [49].

#### 2.2.1.5 Prototyping

The term “prototyping” describes an iterative process in which a product is designed, developed, tested and refined. In participatory design and other engineering disciplines, limited artifacts produced during those iterations are often called “prototype”. However, “prototype” is a generic term for different related representations, such as:

- Sketches;
- Wireframes;
- Storyboards;
- Mockups;

- Technology probe;
- Working prototypes. [50]

All of those representations are complementary and are best used in different stages of the design process. [51]

*"In fact, a prototype can be anything from a paper-based storyboard through a complex piece of software, from a cardboard mockup to a modeled piece of metal"* [50]

The biggest advantage of creating prototypes is a dramatically decreased "time-to-user" (analogous to the popular term "time-to-market"). Building completely functional systems requires a lot of time, money and effort. Building a whole system at once and only testing it afterwards may lead to a major loss of resources. Building partial systems and evaluating them as early as possible can be a solution for this problem. Errors and misunderstandings can be identified and sorted out at early stages in the development process, thus lowering cost and effort. A prototype can therefore also be seen as a model of a system for testing concepts. [50]

Prototypes serve as visual communication basis for designers, users, developers and other stakeholders. By designers, prototypes are used as tools for exploration, visualization, specification and user testing. Users get a feeling of how the later product will look, feel and behave, giving them a rather realistic sample to criticize and give feedback for (instead of a tedious requirements list, for example). Ideally, prototypes also stimulate interests and desire of users for the future product. [50] , [51]

Distinctive features of prototypical artifacts are:

- Representation: sketch, storyboard, etc. (see above)
- Scope
  - Vertical (deep): demonstrates a small number of possible interactions with the system in depth and detail.
  - Horizontal (broad): presents a big picture of the system's functionality at a shallow level of interaction.
- Executability
  - Non-functional: e.g. sketch on paper
  - Partially functional: e.g. implemented with helper-technologies such as proto.io[13] for mobile device prototypes.
  - Functional prototype: e.g. final product of this thesis (see Chapter 4)
- Maturity/Fidelity: the level of polish should reflect the maturity of the prototype. Low-fidelity prototypes do not contain small details that are insignificant or distracting at a certain stage of the design process (e.g. color of a button in a first sketch). [50]

[13] http://proto.io/ [Accessed: 08-Aug-2014]

##### 2.2.1.5.1 Sketch

Sketches usually are paper-based, non-functional prototypes with low fidelity that are used for communication and documentation in the early stages of the design process, generally for depicting physical aspects of system or interface features. They can be created by hand on an empty canvas or printed device templates[14] as well as with tools such as Balsamiq[15]. [50], [51]

##### 2.2.1.5.2 Wireframe

This prototypical representation is often used to depict the layout and shape of a system or artifact. Therefore, wireframes often consist of the main blocks that form the user interface and shows how they relate to each other. For example, a wireframe for a website would probably contain blocks such as header, logo, navigation, sidebar, content area, footer and the like. It is important to find the right level of fidelity in order to be able to think through and synchronize the understanding of important interactions between designers and other stake-holders while not getting lost in details.

Wireframes often contain annotations and describing text; so-called “standalone wireframes” can even be used as a blueprint for implementation. [51]

##### 2.2.1.5.3 Storyboard

Storyboards are similar to sketches and wireframes in terms of representation (mostly drawn on paper by hand), executability, and maturity. Their scope can be either broad or specific. The former would be used to picture the outward appearance of a system while the latter is more like a sequence of screens showing the process of one particular interaction path. They can serve as graphical complement to scenario narratives, too. [50], [51]

##### 2.2.1.5.4 Mockup

Mockups are enhanced versions of sketches and wireframes and introduce a higher level of fidelity as they include colors and graphics. They are used to test the future look and feel of a product while remaining non-functional. [50], [51]

Tangible prototypes are another kind of mockup that focuses especially on exploring haptic sensations. A prominent example is a piece of wood used by Jeff Hawkins, co-founder of Palm[16] and one of the inventors of the Palm Pilot to see whether it would be feasible to carry around a handheld device which was completely new at that time.

---

[14] E.g. http://www.uistencils.com/ [Accessed: 08-Aug-2014]

[15] http://balsamiq.com/ [Accessed: 08-Aug-2014]

[16] http://www.palm.com/ [Accessed: 08-Aug-2014]

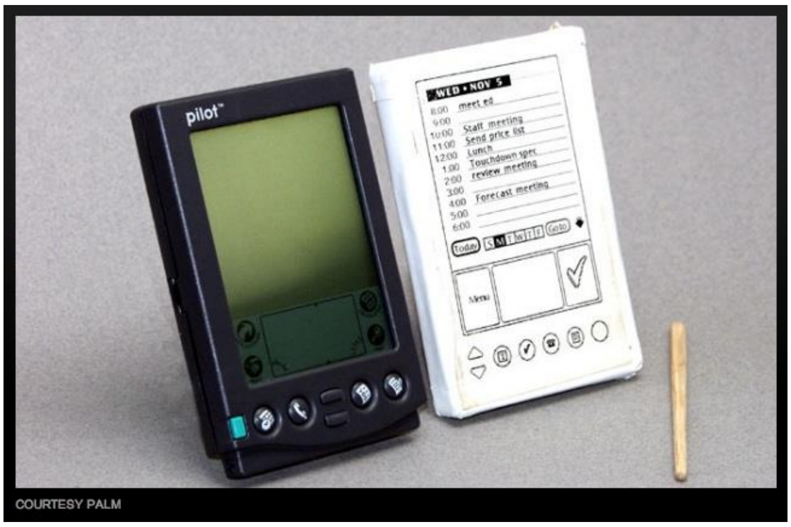


Figure 6: Mockup of the Palm Pilot made of wood and paper[17]

*"[…] he cut a block of wood to fit his shirt pocket. Then he carried it around for months, pretending it was a computer. Was he free for lunch on Wednesday? Hawkins would haul out the block and tap on it as if he were checking his schedule. If he needed a phone number, he would pretend to look it up on the wood. Occasionally he would try out different design faces with various button configurations, using paper printouts glued to the block."* [52]

Exploring interactions with a non-functional prototype can impose certain challenges. A way to solve those problems is "Wizard of Oz[18] prototyping" where complicated functionality is simulated by a human "wizard", as in the name-giving movie ("pay no attention to the man behind the curtain"). This way, the user is able to interact with the system, even though the prototype is non-functional. [53]

#### 2.2.1.5.5 Technology Probe

Technology probes are used to explore certain technological opportunities while omitting any look and feel considerations. [54] The head-mounted augmented reality device created at project "KIBITZER" [55] is a good example for such kind of prototype as it uses no integrated components. Instead, it was built using a combination of a standard bicycle helmet, a camera, a projector and a battery pack.

[17] http://2.bp.blogspot.com/_1IFw6Y1Zmec/TG6rq2a-grI/AAAAAAAACEY/PojezcTiG98/s1600/PalmPretotype.jpg [Accessed: 08-Aug-2014]

[18] http://www.imdb.com/title/tt0032138 [Accessed: 08-Aug-2014]

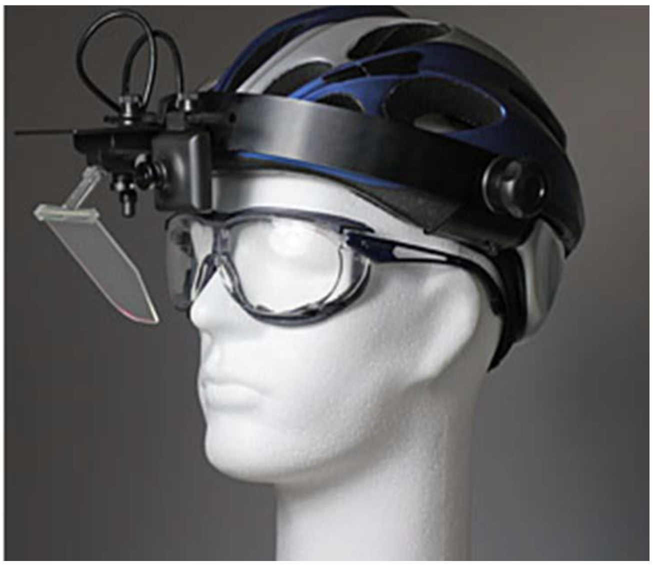


Figure 7: Prototype of project "KIBITZER" [55]

#### 2.2.1.5.6 Working Prototype

Working prototypes usually form the last stage of the prototyping process as they are both functional – at least in one respect – and demonstrate the look and feel of the final product, i.e. their level of maturity is rather high. They are utilized to test and verify the overall user experience. Sometimes, they are also used for advertising and marketing purposes. [50]

### 2.2.2 Conclusion

This section described a common approach to user-centered design and how proven methods are used for primary data gathering, exploration, user testing and communication between stakeholders. Interviews and observations build the basis for qualitative knowledge collection. Personas, design games and various prototyping techniques help refining and augmenting knowledge about users. User testing can already be employed in the early stages of the design process, e.g. using sketches to minimize the risk of resource wasting and annoying users. Furthermore, the interconnected roles of researchers and designers in the design process were analyzed and shown to be changing over time.

In the following chapter I will describe my concrete implementation of the user-centered design process for this thesis as well as the results from the methods used.

# 3 Design and Development

Unlike the previous chapter, which gave background information on spreadsheets and the method used in this thesis, this part of the work describes the whole process from the first idea to the final prototype.

In fact, my starting point was not a concrete idea but the following assumption: *spreadsheets as they are today are too bulky and complicated for many users; they could and should be more user-friendly*.

To confirm or refute my assumption, I conducted a comprehensive user study. As stated by Chambers and Scaffidi in their paper, there are two ways to approach user research: a) gather knowledge through interviews, observations and surveys or b) analyze artifacts created by real users. The former is good to collect opinions and beliefs of users and find out how they actually do their work. However, users' memories are limited and so they might not remember everything. Also, people automate routine behaviours and interactions to minimize cognitive effort. This is why talking to people about their everyday use of spreadsheets might not reveal all their problems and concerns. The analysis of artifacts on the other hand, enables the researcher to examine closely *what* people have done but not *how*. With advantages and disadvantages of both apporaches being clear, a combination of the two seems to be the right thing to do. [9]

Following their recommendation, I employed three different methods for primary data gathering – surveys, observations and interviews – in conjunction with an analysis of real-world spreadsheets created by the participants of my study.

The rest of this chapter is organized in an order that matches the sequence of steps in my user-centered design process. First, an online survey was conducted, which built upon findings of related work and served as a basis for subsequent qualitative research in the form of participant observations and semi-structured interviews with a group of real users. As a result, a web-based prototype was developed and continuously tested and evaluated in collaboration with users. This way, users were involved in all phases of the design and development process.

## 3.1 Survey

Related work, which I described in previous chapters of this text, provided a basis for my initial online survey. The aim of this survey was to confirm results from older studies (e.g. by Chambers and Scaffidi [9]), while clearly setting the focus on the private sphere, which, up until this point, had been completely left aside.

The target group for the survey was rather loosely defined, given the fact that spreadsheets are used – or *not* used – by all kinds of people of different ages, origin, education, profession etc. Furthermore, my goal was not the collection of statistically relevant data; I wanted to use this survey as a form of brainstorming to gather as many ideas as possible, giving a first indication as to whether my initial assumption could hold true or not. Results of this survey were planned to set the direction for the following qualitative user research.

With my questionnaire I could – at least partly – answer questions related to three of my research goals:

2. Identify what tasks people try to accomplish using spreadsheets;
3. Identify if and why they fail accomplishing their tasks;
4. Identify which spreadsheet software people use and why.

### 3.1.1 Design

The questionnaire was conducted in German, acknowledging the fact that a majority of expected participants has German as their first language. It was divided into three parts:

1. "Computers and I": three multiple-choice questions about computer knowledge and use;
2. "The gist of the matter: Spreadsheets": six questions about spreadsheet knowledge and use, including an assessment of Excel's complexity as well as two free text questions about usage scenarios and problems in the private domain;
3. "The usual": two questions asking about age and gender of the participant.

Two questions required participants to rate on a scale from 1 to 10; one to self-assess their Excel knowledge (from "not existent" to "expert") and another to assess Excel's complexity (from "underdesigned" to "overdesigned and complicated"). For such questions it is important to choose an adequate scale. On the one hand, too few options would increase the cognitive effort needed to answer the question because the differences between options would be rather big. Too many options on the other hand, can hinder analysis and reduce comparability. Also, it was shown that participants tended to choose options that lie in the middle of the scale, as choosing a presumably average value helps them keep the time and cognitive effort for answering the question to a minimum. Choosing an even number of options can temper that problem, although not solve it. [56]

### 3.1.2 Implementation

I created this survey in the form of an online questionnaire accessible to anyone with access to a link with the help of Google Forms[19]. Google Forms is a free, lightweight tool for easy form creation. Results can be viewed either online or downloaded in the form of a spreadsheet. Additionally, the software can also generate chart-enriched summaries. The number of question types is limited; however the repertoire was enough to serve my purposes. Unfortunately, answer-options to multiple-choice questions cannot be defined as mutually exclusive.

[19] https://docs.google.com/forms/create [Accessed: 08-Aug-2014]

In order to reach a diverse group of people, I distributed the questionnaire's link via multiple channels:

- My private Facebook wall, the link was reposted couple of times by me and other people;
- Emails to private and University contacts;
- Intranet and internal mailing-list of the Austrian Federal Ministry of Education and Women's Affairs[20].

Afterwards, results were analyzed and coded using built-in methods of Google Forms as well as Microsoft Excel.

### 3.1.3 Results

A total of 378 people filled out the questionnaire between November 13th and December 17th 2013. 58%, more than half of participants, were female, whereas 42% were male. The age of participants ranged from 16 to 64 years. Around two thirds were over 30 years old. However, probably due to the distribution via Facebook, my own age group was the largest in terms of absolute numbers (around 30 people born in 1988).

#### 3.1.3.1 Computer Knowledge and Use

The majority of participants' computer knowledge was self-taught. Courses organized by employers as well as primary and secondary educational facilities, friends and relatives are other important sources of computer know-how. This question was defined as multiple-choice with more than one answer allowed.

Table 5: Sources of computer know-how

| Source of know-how | Count |
|---|---:|
| Self-education | 293 |
| Work-related training | 176 |
| School | 167 |
| Friends and family | 91 |
| University | 55 |
| Other | 20 |
| Adult evening classes[21] | 3 |

[20] https://www.bmbf.gv.at/ [Accessed: 08-Aug-2014]
[21] "Volkshochschule" in Austria

Office software (including spreadsheets) is very popular in professional computer use. Nearly 97% of participants stated to use office software at work. Together with Email and browsing the Internet, this forms the by far biggest group of applications.

For private use the picture looks a bit different. Email and Web browsing are the most popular applications, followed by organizing photos and videos, consuming music and games. However, spreadsheets and other office programs are also used by around 75% of participants.

Table 6: Professional and private computer usage

| **Professional applications** | **Count** | **Private applications** | **Count** |
|---|---|---|---|
| Email | 369 | Internet | 364 |
| Office<br>(Word, Excel, Powerpoint etc.) | 368 | Email | 351 |
| Internet | 248 | Photos/Videos<br>(edit, organize, archive etc.) | 298 |
| Reporting, timekeeping etc. | 169 | Office<br>(Word, Excel, Powerpoint etc.) | 285 |
| Programming | 53 | Music<br>(listen to, organize, etc.) | 211 |
| Other | 47 | Games | 131 |
| Controlling machines etc. | 39 | Other | 32 |
| Creative | 37 | Creative | 31 |

#### 3.1.3.2 Spreadsheet Knowledge, Usage and Perception

When asked about whether or not people already heard of the market leader Microsoft Excel, they answered yes without exception. Around 98% of the participants already used Excel, too. Curious about the users' awareness of other spreadsheet products, I included a question requesting their knowledge about such products. Interestingly, only around 27% of people who took part in the survey and answered this question had used a spreadsheet program other than Microsoft Excel so far. This is further confirmation of Excel's market power.

Table 7: Awareness of alternative spreadsheet programs

| **Awareness of alternative spreadsheet programs** | **Count** | **Percentage** |
|---|---|---|
| “never seen before” | 149 | 39% |
| “already seen before” | 165 | 43% |
| “already used before” | 103 | 27% |

Only a fraction of participants identified themselves as spreadsheet experts or complete spreadsheet beginners. Most of them rated their corresponding knowledge somewhere between 3 and 7 on a 10-point scale. This could be explained by the fact that spreadsheet programs are so common nowadays that an average computer user is very likely to stumble upon and use it at some point. Basic office software skills are also taught in most schools. Another possible explanation of this result is the above mentioned tendency towards the value in the middle of the scale. Figure 8 shows the distribution of answers.

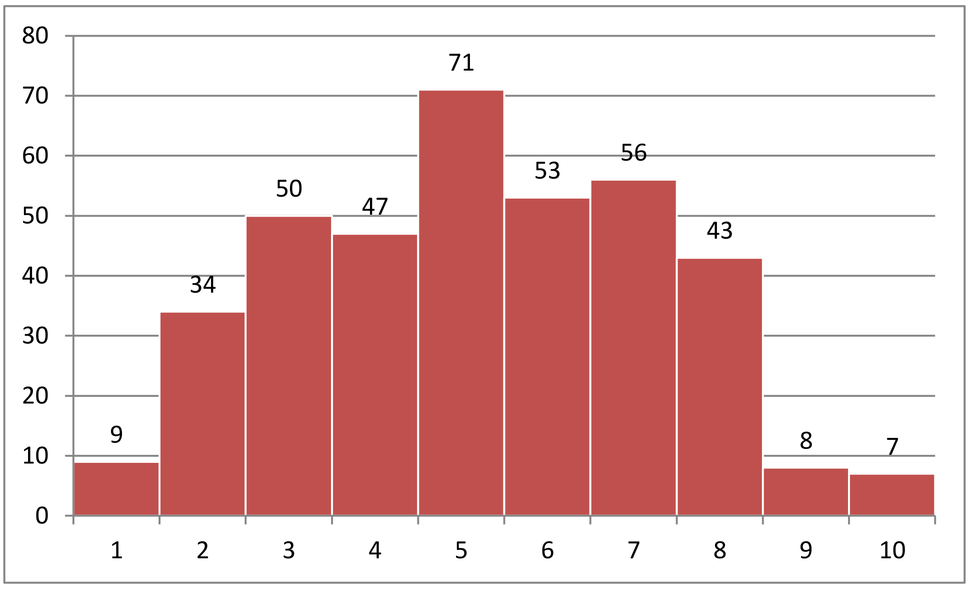


Figure 8: Assessment of own spreadsheet skills (1 = "no skills at all", 10 = "expert")

Question 8 asked participants how well they could realize the following feature categories with their favorite spreadsheet software. Those were found by Chambers and Scaffidi to be the main components of spreadsheets:

- Lists and tables;
- Simple calculations;
- Diagrams and graphs;
- Complex formulas and calculations;
- Macros;
- Forms. [9]

The results summarized in Figure 9 paint a clear picture: users are confident with basic tasks like lists and tables. Simple calculations also do not seem to impose major difficulties to most participants. As the complexity of features increases, people feel decreasingly confident in using them. While these results are no big surprise, they still serve as indicator for features and user interfaces in need of improvement.

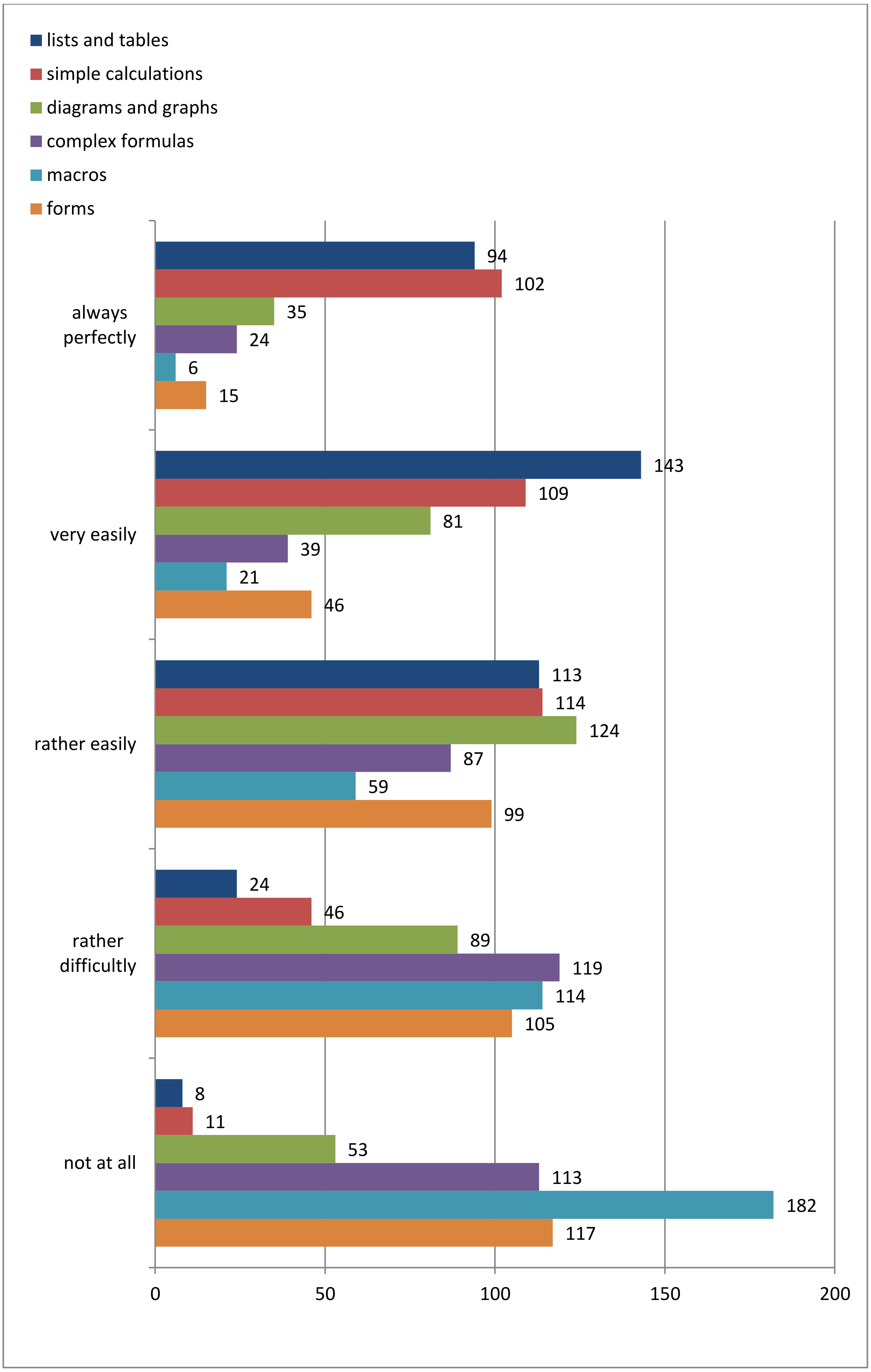


Figure 9: How well people can use major spreadsheet features

Surprisingly, around one third of participants rated Excel to be neither “underdesigned” nor “overdesigned and complicated”, but something in between. There is a clear trend towards the upper end of the scale, though. Nevertheless, I would have expected more people to give answers closer to 10. However, this could again be due to the tendency towards the value in the middle of the scale mentioned by Häder [56].

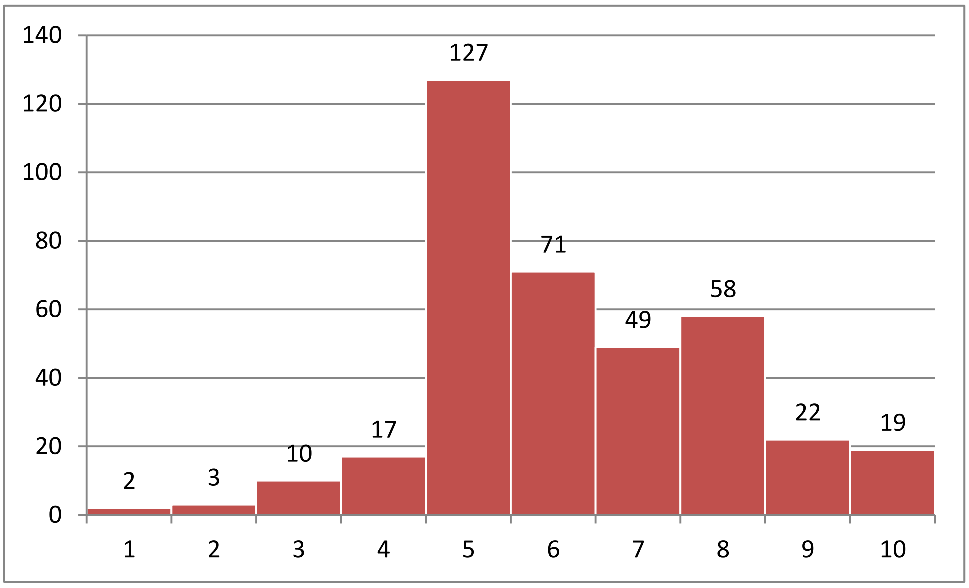


Figure 10: Assessment of Excel's complexity
(1 = "underdesigned", 10 = "overdesigned and complicated")

Furthermore, the questionnaire included two questions to be answered as free text of unlimited length without any preset answer options. They were designed to work as a form of collective brainstorming and to collect as much user input as possible. Question 11 was titled “I use Excel (or another spreadsheet program) privately for these purposes” while the heading of question 12 was “Excel (or another spreadsheet program) imposes these difficulties on me”.

Fortunately, most participants were confident speaking about their use-cases and difficulties they have with spreadsheets and gave a lot of input. I worked through all the answers and coded them with categories which I developed as I did the categorization. Some categories were inspired by what Chambers and Scaffidi found in their work [9], while others emerged during the process.

Table 8 shows the results of the coding process for question 11: use-case categories together with the numbers of occurrences.

Table 8: Use-cases for spreadsheets in the private area

| Category | Count | Category | Count |
|---|---|---|---|
| Lists | 86 | Training, workout | 8 |
| Tables | 52 | Database | 8 |
| Planning | 16 | Word-like tasks | 6 |
| Overview, comparison | 14 | Forms | 4 |
| Charts, graphs | 13 | Education | 4 |
| Time management | 13 | Data analysis | 2 |
| Inventory management | 11 | Formulas | 1 |
| Statistics | 9 | | |

Planning (e.g. vacations, household and cleaning), overview and comparison (e.g. products and prices), inventory management (e.g. personal CD or book collection) as well as training and workout (e.g. workout plans and time tables, tracking of weight and body fat) are rather particular use-cases that could be subsumed under “lists” or “tables”, resulting in even more dominant numbers for those categories.

Asked about their difficulties using spreadsheets, many participants pointed out their perceived lack of knowledge. They felt like they knew too little, which indicates that they see the problem on their side, rather than as a shortfall of the software they use. On the other side are answers falling under “complexity”, where participants blamed their favorite spreadsheet program for the difficulties they have. A small fraction of participants mentioned Word to be their alternative in case they have troubles using Excel.

Table 9: Difficulties with spreadsheets in the private area

| **Category** | **Count** | **Description** |
|---|---:|---|
| Knowledge | 19 | The user thinks she needs to learn more about the spreadsheet program. Not using the program for a certain timespan makes it hard to orient oneself afterwards. |
| Connectivity, import/export, multi-user support | 13 | Sharing spreadsheets can be tedious, collaborative editing is not well supported. Import and export of data poses various difficulties. |
| Specific functionality | 11 | Problems with specific features, e.g. “calculations with time-variables”, “copy-paste issues with hidden rows in newer versions of Excel”. |
| Complexity | 10 | The spreadsheet software is seen as too complex and therefore hard to use. |
| Formulas | 7 | Problems with creating, understanding and maintaining formulas. |
| Layout, formatting | 6 | Formatting of cells and cell-groups imposes difficulties. The fixed grid is a limiting factor. Problems with the presentation of large amounts of data. |
| Charts | 6 | Problems creating charts. |
| Help | 5 | Help is unclear and/or hard to find. Searching for content can be problematic. |
| Hardware, environment | 2 | Problems don’t originate in the spreadsheet software itself. |
| File-formats | 2 | Differences between old and current file-formats can be tedious, e.g. `.xls` vs. `.xlsx` format. |

### 3.1.4 Conclusion

The findings of this survey are clear: a majority uses spreadsheets to create (simple) lists and tables, as indicated by the 187 mentions of these two use-cases. This confirms what Chambers and Scaffidi found in their user study [9], where representatives of the two categories “data entry” and “database” accounted for 81% of all spreadsheets they analyzed. Therefore it seems valid to extend their statements to the private area.

Similar results were presented by Hendry et al. who spoke to ten spreadsheet users in 1994 and found out that “*spreadsheets are an effective communication tool and that computations performed are generally simple*”. [2]

An explanation for their popularity was given by Gyssens et al. who stated that "*tables are one of the most natural ways in which real-life data can be represented*". [57]

My evaluation of spreadsheet usage difficulties also showed some overlap with problem categories unveiled by Chambers and Scaffidi [9] (see Table 10).

Table 10: Categories of spreadsheet problems [9]

| **Category** | **Percentage** |
|---|---:|
| Problem setup | 21% |
| Feature finding | 19% |
| Formulas | 18% |
| Config | 16% |
| Data viz | 9% |
| Macro | 9% |
| Integration | 8% |

Findings from research in end-user programming may explain those numbers. End-user programmers often find it hard to create a mental model of a problem; this is what is labelled "problem setup" above. Problems in finding the right features can have two reasons. Firstly, not being sure about the problem makes it hard to choose the right tools to solve it. Secondly, overloaded user interfaces and a plethora of features packed into modern spreadsheet programs can also be a challenge.

## 3.2 Observations

In this second part of my user research, my goal was to investigate *how* people interact with spreadsheet software. *What* they do was at this point already pretty clear considering information gleaned from literature review and results from my own survey.

> *"Given the astounding success, one would like to know about spreadsheets in great detail, and one would like to know not just about the low-level details (e.g. speed of entering formulae) but about spreadsheet "ecology", the place of the spreadsheet in its users' lives, its relationship to competing tools and to collaborating tools."* [2]

Sharing Hendry's and Green's thirst for knowledge, I planned and executed observations with a handful of real spreadsheet users. The goal was to learn from them solving everyday tasks within their natural environment and real-world context.

### 3.2.1 Design

For this study I used a hybrid approach to participant observation, combining moderate participation and video observation (see 2.2.1.2 Observations). During the observations, subjects were at home using their own computer and equipment to solve a simple task. I took the role of the observer and watched from my own place via a live screencast. This way, I could leverage key advantages of both approaches. Firstly, participants could work in their usual environment without anybody physically looking over their shoulders. My assumption was that they would forget or block out that somebody is watching them after a short while, making the situation as realistic as possible for them. Secondly, using a live audio connection, I could ask questions and clarify ambiguities at any time. Records of screencast video and audio were valuable during analysis, as they allowed for repeated playback. Subjects were asked for permission for the recordings.

Additionally, I asked participants to comment on their thoughts, decisions and actions. This method is known as “think aloud” and it can help to make latent (expert) knowledge visible. On the downside, thinking aloud can disrupt the cognitive process, especially when subjects have never done it before. Also, “*there are substantial differences in the ease with which people verbalize their thoughts*”, as Van Someren et al. pointed out. [58] Nevertheless, think aloud is a practical way of enriching observable data.

At first, I planned to observe subjects working freely with spreadsheet software of their choice. After a pre-test with one subject, this approach turned out not to be suitable for answering my research questions. Therefore I created a small assignment which I gave to all subjects from then on. The latter approach worked substantially better and provided a lot more valuable data. Each observation lasted somewhere between 30 and 60 minutes.

As Hendy and Green pointed out, “*it is not enough to listen only to users or to HCI experts. […] their different backgrounds and experience with spreadsheets have made different aspects salient*” [2]. Selecting the right group of subjects is essential for gathering rich data. For my project, this meant choosing a fairly diverse group of people to work with:

- Subject 1: “Alice”[22], female, 26 years old, works at an online travel portal;
- Subject 2: “Bob”, male, 42 years old, railway officer, has 2 children;
- Subject 3: “Carol”; female, 20 years old, attends art school;
- Subject 4: “Dean”: male, 25 years old, technical physics student;
- Subject 5: “Eric”: male, 60 years old, mid-level official in an Austrian federal ministry.

All subjects were located either in Vienna or Upper Austria, which was not a problem given the fact that remote observations were used.

[22] Names are imaginary.

### 3.2.2 Implementation

As implied above, I implemented remote observations using live screencasts. TeamViewer[23] was the right software choice to comply with my requirements. It is free and can be used without installation on the customer-side (TeamViewer QuickSupport). I hosted the executable in a shared Dropbox[24] folder. Participants received the link to the file together with a brief explanation via Email or Skype[25]. TeamViewer comes with built-in functionality for recording screencasts, both audio and video. Additionally, I took notes on paper as I saw fit.

After setting up the software and establishing audio and video connection, each observation started with the handover of the assignment, which required the subjects to:

- Create a new spreadsheet;
- Understand the given financial information of a fictional record store;
- Create calculations of purchase costs and profit;
- Create a chart visualizing sales;
- Format the spreadsheet as desired, based on data and calculated values.

The full assignment, which I developed based on an existing lessen for Microsoft Excel[26] can be found in Appendix A.

All participants used their own computers and software to solve the tasks, which was important in order to be able to observe their habits. While briefing the subjects, I specifically stressed the voluntariness of their participation and that there would be absolutely no rating whatsoever; all results were treated strictly confidential. Additionally, I explained to them how the think aloud method works and asked them to apply it.

Following each observation, I asked the subjects to show and explain some of the spreadsheets they created for private use. All of the participants allowed me to inspect some of their files, including sheets for managing household accounts, trip-planning, calculations for shared travel spending and data analysis for a physics lab course.

For the analysis of the observations, I watched the recorded screencasts a total of three times each. The first time, I took notes of everything that seemed of interest for my research questions. Immediately afterwards, I watched the footage a second time and tried to focus on things I couldn’t detect before and enhance the notes taken during the first round. A couple of days later, I watched again, using a blank sheet for taking notes. After that I combined the notes of all rounds and categorized findings to identify common problem areas.

[23] http://www.teamviewer.com/ [Accessed: 08-Aug-2014]

[24] https://www.dropbox.com/ [Accessed: 08-Aug-2014]

[25] http://www.skype.com/ [Accessed: 08-Aug-2014]

[26] http://www.reviseict.co.uk/lessons/excel/task3.htm [Accessed: 08-Aug-2014]

### 3.2.3 Results

As expected, all subjects reacted differently to the assignment, due to individual experiences made prior to this experiment. Four out of five persons found the tasks difficult, three of them could not complete all steps and opted for aborting the assignment. Only one participant could complete all steps without major difficulties.

The following sections describe the most important observations I made for each subject. Interpretation of the results follows in Section 3.2.4.

#### 3.2.3.1 Alice

Alice had only minor difficulties solving the given tasks using a copy of Microsoft Excel 2007 on her notebook computer. She opened a new file which contained three sheets per default. However, she worked solely in one of the sheets. There she created three tables, separated with spacing (empty rows and columns) and custom borders: one for sales, another for purchases and even another one for profit. Each of these tables she furnished with a custom formatted heading above each table. Also, she put a large, colorful heading on top of the sheet. During these first steps she explained her usual routine where she puts data in first – already with a vision of how it should look like later on – and then turned to the formatting.

The subject complained about the “*annoying*” popups containing “*merely the same buttons that are in the top menu anyway*” that appeared with the context menu. Formulas she typed by hand using the formula bar relying on the autocomplete-feature for function names. Alice said that she does not like the way images – including charts – are displayed “*in a layer above*” the grid in Excel.

#### 3.2.3.2 Bob

Bob used the recent 2013 version of Microsoft Excel on a desktop computer. His way of approaching the assignment was rather uncommon: he started by enabling the page boundary indicators by opening the print preview. Then he spent roughly 30 minutes for the layout of the page by modifying column widths and row heights without even putting in any numbers. At first I just watched him, amazed by what I was seeing, but then I asked him what he was doing and he explained that this was his way of working with Excel. He said: “*I always print out the tables I create with Excel and so I need to make sure that they fit on the sheets of paper*”.

Later, after reading the task description multiple times, he simply said “*well, I don’t know how to use formulas*” and started calculating the required sums by hand on a sheet of paper. Afterwards, he typed the numbers into his spreadsheet. Presumably, he sees spreadsheets – and maybe also other digital documents – as a way of creating some kind of clean copy that can then be replicated through printing. At that point Bob and I together decided to stop the observation as I would probably not have provided me with any further valuable information.

#### 3.2.3.3 Carol

Carol behaved rather similar to Alice. Firstly, she also used Microsoft Excel 2007 to solve the task. Secondly, the way she organized her spreadsheet was also similar: two separate tables – one for sales, another for purchase and profit – were separated with borders and spacing. However, the latter was significantly larger than seen with Alice.

Like Alice, Carol also used the formula bar, but she did all cell editing there. She only typed directly in cells when they were still empty. Carol tried to place a large “record store” heading on top of the sheet, but had problems formatting it because it spanned over multiple cells.

When she needed to do the requested calculations, she had trouble remembering function names. “*I can’t remember how to do formulas. I’m not sure how to make my own formula. Is this even possible? I don’t like the abbreviations, they are not explained well*”, she said. After a lot of trial and error, she managed to do additions and subtractions. Giving her a hint to use the dollar sign notation for absolute cell references, she commented with: “W*hy dollar? Can’t I also do it with Euro?*” However, she did not mean that as a joke; she was rather confused by the two meanings a dollar sign can have in the context of a spreadsheet (currency symbol and part of formula syntax). Without being able to create the requested chart and setup conditional formatting, the subject ended this observation with the words “*this is awful*”.

#### 3.2.3.4 Dean

Normally, Dean uses LibreOffice Calc for creating spreadsheets. However, due to sudden problems with his LibreOffice installation occurring when we tried to set up the observation, he had to use Google Spreadsheets for the assignment. Being a rather experienced and interested user, this posed no major difficulties for him. Also, it was interesting to see in a real-world scenario, how the intentional use of familiar interface and usage concepts helps users learn new software. Dean stated that he had never used Google Spreadsheets before, but still he was the only subject who could complete the whole assignment without difficulties. During the observation he often said “*normally, I would do it like this…*” and then tried to do something like he would with LibreOffice Calc, for example absolute cell-references with a dollar-sign and function-names (e.g. “SUM”).

Just as Alice and Carol, Dean created spatially separate tables for purchase and sales data within his spreadsheet. He decorated them with headings and framed the connected cell areas with borders. “*I will draw some borders so this sets apart from the rest*”, he commented on his thought process. Interestingly, he did not use the cells in the top left area of the grid, but instead started his first table at cell B3.

Unlike all other subjects, Dean mainly used the keyboard for fast navigation within the grid. He also typed formulas by hand directly inside the cells and did not use the formula bar.

#### 3.2.3.5 Eric

The last subject, Eric, unfortunately missed out on activating his copy of Microsoft Excel and therefore had to work with Google Spreadsheets for this observation. I gave him some extra time to look through the software and orient himself. Already from the very beginning I noticed that Eric had resentments towards spreadsheet software. For example, after I showed him the key interface elements of Google Spreadsheets and told him that it works very similar to Excel, he replied: "*I cannot copy what I know from Excel because I don't really like that one either. I mean, when I get an Excel list and I have to change something in there, I don't use functions and other things. This is already where my relationship to Excel ends.*"

When he read that he should use formulas he cringed and complained. The subject said "*oh god, this is too complicated, I can't do it*" before he even tried. He said that he would rather do the calculations in his head or with a calculator and insert the results afterwards. Eric seemed so affected by his opinion about functions and formulas that he did not even think about using arithmetic signs to do calculations (e.g. `=A1+B2`). Being asked about that specific observation, he confirmed that calculating something in a spreadsheet automatically means for him "*I have to write a complex formula with functions I don't know*".

Throughout the observation, Eric used a trial-and-error strategy to work through the assignment and relied heavily on the undo function. Eventually, he got annoyed and stopped the observation during step 4.

### 3.2.4 Conclusion

After watching five subjects solving the assignment and allowing them to explain some of their real spreadsheets to me, there were three major conclusions I could draw from this part of my user study.

1. The grid seems unpractical.

The pure grid structure of modern spreadsheets which usually contain a sheer unlimited amount of rows and columns seems not to comply with common use cases, especially in the private area where the amount of data contained in spreadsheets is not as big as in many professional applications. Nearly all spreadsheets I analyzed contained one or more spatially separate areas which people formatted to look like tables and set them apart from the rest of the grid which remained unused. Most of the time, they had to use borders and color to achieve that effect. In cases where one sheet contained a larger number of table-like areas, people had problems finding a convenient way to align them, especially when different tables required different column widths. The most extreme cases I could find in Bob's spreadsheets, in which he always made all rows and columns as small as possible and then used combinations of connected cells to build the desired layout.

2. People like to annotate.

It showed that users often want to enhance pure data with annotations, such as headings and comments. Naturally, headings need to stand out, thus they need to be formatted with a larger font, for example. Having to put larger and smaller text or short numeric data along with long describing text together in the same grid can pose certain difficulties.

3. Some people are scared of formulas.

Even though most spreadsheets I could analyze only contained basic functions such as sum, maximum, or average, people still have problems creating formulas. This matches with what was found in a 1988 study of spreadsheets in Finland. Researchers revealed that only a small minority of available functions was used and only 5% of cells contained formulas at all. `SUM`, `IF` and `ROUND` were by far the most popular functions.

As I understood the issue, the problem lies in remembering function names and their syntax. Most people I observed struggled to find the right functions to use. Features like autocomplete can definitely help, but they require the user to start typing, i.e. the user at least needs to know what she is looking for. The plethora of functions available in modern spreadsheet applications of course make it hard for developers to find a convenient way of making them accessible.

With their user study, Hendry and Green showed similar findings. Their interview subjects complained about problems with writing, understanding and reverse-engineering formulas and bad visibility of relations between cells.

## 3.3 Interviews

As a complementary data gathering method to participant observations, I chose to conduct semi-structured interviews with the same people who participated in the observations. This section describes the setup and implementation of the interviews as well as results and newly gained insights.

### 3.3.1 Design

The interview guideline was based upon findings from the survey and designed to build a natural conversation in order to find out about people’s positive and negative experiences with spreadsheets. I also used the interviews to speak to participants about the prior observations, how they felt solving the tasks and interesting situations I found while watching them. Therefore, I extended the guidelines with keywords from each observation before the interviews to formulate a unique set of questions and topics for each conversation.

Most interviews were arranged in a face-to-face setting. However, one interview had to be conducted via Skype because the interviewee lived too far away and we could not arrange a personal meeting within a reasonable timeframe. This personal setup helped me not only learn from people through what they say, but also how they physically react to questions. Body language and facial expression are a valuable source of information. Interviews were conducted in German, given the fact that all participants have German as their first language.

### 3.3.2 Implementation

Whenever possible, I interviewed participants right after their observations. This way, it was fresh in our minds what we had previously seen and done, respectively. Needing to arrange only one appointment per person was another practical advantage.

As mentioned above, most interviews were arranged in a face-to-face setting in an environment comfortable for the interviewees. Some participants invited me to their homes while one person visited me at my place. For every conversation I made sure that both conversational partners had plenty of time to assure that we would not need to interrupt a lively conversation because of one person needing to leave for another appointment.

From each subject I had permission to audio-record the interview, which allowed me to focus on the interviewee as I did not need to take notes during the conversation. However, I usually wrote down thoughts, interpretations and striking observations such as people’s reactions to certain questions immediately after the interview. The audio-recordings were later transcribed and evaluated.

### 3.3.3 Results

The interviews definitely helped me understand the spreadsheet users better. I had a lively conversation which each one of the subjects, usually about one hour in length.

This section is similarly structured to the one last as it describes the most important insights I gained for each subject. Interpretation of the results follows in Section 0.

#### 3.3.3.1 Alice

“*I feel creative and look forward to creating something new*”, Alice replied as I showed her an empty spreadsheet and asked her about her first thought. However, she feels limited because of the rather rigid grid structure that Excel offers. In her mind, there is a strict divide between Excel and Word; she uses Excel for everything table-related tasks because she finds the table tools in Word rather complicated and cumbersome. She does like the feature to draw tables in Word, though.

Alice has hard times working with spreadsheets that were either created by another person or by herself a while ago. The mixture of data and formula cells render such spreadsheets unintelligible, making it hard for her to understand why certain things happen, e.g. how values are calculated. Writing formulas herself is also difficult for Alice.

The young woman prefers using Excel, but also knows Google Spreadsheets and does not notice substantial differences in the usability of the two. She prefers the latter for collaboratively working together and sharing files with others because she does not have to bother thinking about different file formats and program versions.

Creating lists and basic calculations poses no difficulties for Alice. However, Alice prefers specialized programs and apps to Excel for simple applications – such as to-do-lists – as they come with use-case specific features that simplify certain tasks. Excel is her first choice for documents that take a longer time to create and are reused over time, e.g. her book of household accounts.

Having problems with certain features, Alice likes to use the Web as a source of information because that way she gets faster results as with the built-in help in Excel.

#### 3.3.3.2 Bob

When Bob opens a new spreadsheet in Excel, he already has a vision of what the final result will look like. He uses Excel solely for creating printable documents, for example a multi-page newspaper for a club he manages. Interestingly, Excel is his first choice for that task, even though he has to spend a lot of time setting up rows and columns to even be able to design multiple pages with different layouts below each other. One reason for not considering Word, for example, is his lack of confidence in using table-features in that software. Bob said that he had not had any bad experience with Excel, and that he is pleased with what he can do with the program. He estimated that he uses about 0.5 to 1 percent of Excel’s features. When asked about whether he would consider using an alternative to Excel, he replied “*no, because I would be even less well versed in that then*”.

#### 3.3.3.3 Carol

For Carol, Excel is a synonym for numbers and tables and “*something that you first have to learn*”. The young woman feels rather discouraged when she has to work with spreadsheets, and she also felt that way when given the assignment for the observation.

Carol has problems remembering function names and when to use which feature. “*There are thousands of things in there*“, she explained. This is why Carol does not like to use Excel for her private tasks. The variety of features and formulas overtax her and she would prefer only a small pre-selection of functions to be visible to help her decide what is important for her and what is not. She also expressed the wish for better in-line help and explanations on how to link cells in a useful way.

From early on, she liked to experiment with Word because she sees the program as “*less straight*” than Excel and enjoys the freedom a blank white page gives her. Therefore, she uses Word also for creating documents containing tables, saying that “*this works super-fast and easy*”. She pointed out that software she wants to use on a daily basis has to provide her with a simple user interface enabling fast interaction without the need of “*having to consult the manual each time*”.

Besides communication, Carol uses her Smartphone for small specialized apps, e.g. to manage notes; she likes how synchronization between her different devices works. For managing her appointments she uses a pocket-size paper calendar because she particularly enjoys it’s flexibility and the fact that she can write down, cross out and annotate stuff as she likes.

#### 3.3.3.4 Dean

Dean is a rather advanced spreadsheet user and utilizes it both for private and university tasks, e.g. timetable, data analysis of test series from the lab, shared expenses, etc. He does not like to work on a spreadsheet together with other people because he has a hard time understanding, at first glance, which cells contain data and which contain formulas.

Because Dean uses different operation systems on his devices and a combination of Excel and LibreOffice Calc, he often runs into problems with incompatible file formats, as well as the import and export of data. A unified solution for all of his devices could solve this problem, he thinks. He empathized how important conventions are that spreadsheet program makers (silently) agree on, for example function names or formula syntax.

Speaking about how he solved the observation task, he confirmed his preference for separating data from row- and column-labels and borders to delimit tables from the rest of the grid.

#### 3.3.3.5 Eric

Right at the beginning of our conversation, Eric noted that he is "*definitely not an Excel-fan*". He used Excel, Smartsheet[27] and Google Spreadsheets before and described each as a "*disappointing experience*". "*Oh dear*" was the first thing he said when I showed him an empty spreadsheet and asked him about his first impression.

Eric has a hard time creating spreadsheets because he lacks a starting point; the large empty grid strains him, as he says. Also, he does not like editing spreadsheets created by other persons because the outcome of editing often surprises him when formulas are involved. He finds it "*not transparent enough for somebody who edits afterwards*" and sometimes he even thinks re-creating a spreadsheet from scratch would be easier than understanding and editing an existing one. Explanatory comments could help him but only if they were not in Excel terminology.

The man associates Word with a typewriter where "*you just have to start typing to begin*". Excel, on the other hand, he connects with numbers and calculations. The fact that menus and the general user-interface look very similar in these two applications often makes him believe that with Excel he could start right away just as with Word, which was not the case.

Since he owns a Smartphone, Eric prefers the clear structure of mobile apps over multi-purpose software like Excel. "*In any case*", he pointed out, "*computer programs only make sense when they take work off of you. Excel does not offer that for me.*" He has the feeling that Excel is often used for simple tasks and he would like to see a solution where simple tasks are also presented in a simple way.

[27] http://www.smartsheet.com/ [Accessed: 08-Aug-2014]

### 3.3.4 Conclusion

Interesting conversations with five diverse spreadsheet users helped me gain insight into common usage scenarios and their personal experiences and problems. Results are of course not representative for millions of users, but again confirm what other studies found for the professional domain.

1. People are overstrained with Excel's featuritis[28].

Over the years, spreadsheet programs grew as new features were added to every new version. The times when features that cannot be described with a few words on a two-page reference card would not make it into a release are over (see Section 2.1.1). Today, Excel and other spreadsheet products aim to please every potential customer, making it even harder for people with little know-how and training. Due to the lack of alternatives, many people accept the struggle and use Excel for their personal day-to-day tasks.

2. People enjoy the simplicity of modern mobile apps.

As mobile devices and applications gain popularity and market share, people begin to appreciate the simplicity of most apps. In fact, we see history repeating itself. A few decades ago, software had to be simple because computers lacked computing and rendering power. With the advent of mobile devices, developers were confronted with similar problems: slow processors, small screens, low screen resolutions and limited memory. Today, some of those issues have been resolved and mobile devices are more powerful than ever. Nevertheless, the usability of mobile apps is still a key factor in their success and we have seen a trend towards specialized applications simplifying certain tasks. One challenge for the future will be to transfer this new knowledge in order to enhance the user experience for classical computer applications.

3. People have problems creating and understanding formulas.

While the creation of lists and tables works quite well for most users, people seem to have major problems with formulas. I already explained this issue in the observations section; however, the topic came up again in all the interviews. This is so important because it probably is *the* limiting factor why people don't like using spreadsheets.

## 3.4 Prototype

Talking to users about their requirements and needs is a noble thing to do, but it is worthless if their concerns are not taken seriously and incorporated in the design of a new approach to spreadsheets. At this stage it makes sense to recap the last three goals of my thesis:

6. Define requirements for a web-based software framework prototype as a basis for small, easy-to-use spreadsheet applications;
7. Build a working implementation of said prototype;
8. Continuously evaluate and refine the prototype together with real users.

[28] http://en.wikipedia.org/wiki/Feature_creep [Accessed: 08-Aug-2014]

After doing a survey with around 380 participants, as well as observing and interviewing five spreadsheet users, I had enough material to define requirements for a modern spreadsheet program designed especially for private use.

### 3.4.1 Requirements

There are three main problem areas I discovered through my own research as well as literature review that I wanted to address with my prototype, namely:

- Feature finding;
- Formulas;
- Learnability.

It is evident that all three are somehow connected, yet have different implications for the design.

The problem of finding the right feature for a certain task makes software cumbersome to use. By and large this may be due to the sheer amount of features packed into current spreadsheet programs, making it hard for developers to provide good user interfaces for them. Colborne [59] described four strategies for simplicity, all of which I used in my design to improve the user interface:

- Remove: remove features not needed in a vast majority of use-cases, keep only the most important ones;
- Hide: intelligently show or hide menus, make interface elements visible when they can be used;
- Organize: group interface elements that belong together, give the user control over created elements;
- Displace: move interface elements into the focus of the user, make them easy to access, and provide important elements near to where the user currently has her attention.

They are illustrated below using a TV remote control – a popular device for presenting user interface flaws and improvement ideas as remote controls are often overloaded with buttons of which users usually only need a small number.

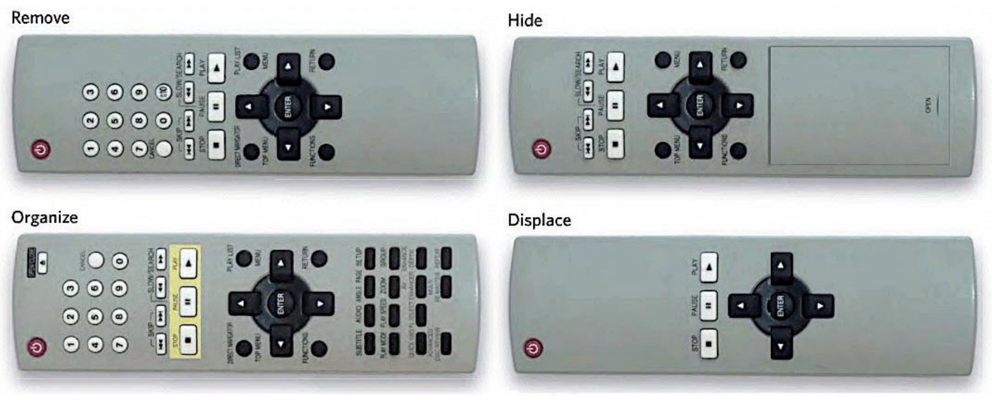


Figure 11: Four strategies for simplicity [59]

Formulas account for a big part of the usefulness of spreadsheets. However, as my research has shown, they are often hard to write and understand. Users struggle with finding function names as well as formula syntax. Therefore, one requirement for my prototype was to simplify the usage those functions that my research showed to be the most commonly used:

- Sum;
- Average;
- Minimum;
- Maximum;
- Count;
- If (Condition).

Other than professional software which is most often used on a day-to-day basis, programs employed in the private domain are sometimes less regularly and less frequently used. This can make it necessary to repeatedly learn both the software and the spreadsheet itself. Cleaned-up menus, a clear layout, labeled formulas as well as annotations in the form of comments should be included in my prototype to ease that situation and increase learnability while decreasing the need for repeated learning.

Seeing how often users have to draw borders to delimit their data and tables from the rest of the grid which is shown per default in every spreadsheet, the fact that most of the time only a fraction of the grid is used, and problems with layout and formatting due to different font sizes made me scrutinize the grid concept. After I found out that Apple also developed an alternative approach for their software Numbers, I ultimately decided to implement and evaluate a canvas-approach.

### 3.4.2 Technology

With the advent of high-speed Internet connection lines, powerful web technologies and performant web browsers during recent years, there is a trend towards web-based software. One of many examples is the recent version of Microsoft Office, which brought Excel & Co. into the web. Therefore, and because web technologies are ideal for rapid functional prototyping, it was a clear decision to build my prototype as a web-based application. It should be available anywhere, anytime and accessible by anyone using a modern web browser.

Usually, web applications consist of two main parts: the frontend which renders and controls user interfaces on the client-side, and the backend which is located on one or multiple servers providing storage facilities and processing background jobs. The communication between the two usually works over HTTP connections. Because building a fully functional spreadsheet application would have gone beyond the scope of this thesis, I decided to focus solely on the frontend part, including user interfaces and interactions. Therefore, all features of this prototype were implemented on the client-side using the standard web technologies JavaScript, HTML and CSS.

### 3.4.3 Iterations

As described in Section 0, prototyping allows for iterative refinement of an artifact with constant feedback of real users. In this specific case, I did seven iterations, each with a user-testing round at the end. Ten different test users were involved in the process, including four out of five participants of my user study. The lists of feedback, suggestions and defects collected in the user tests served as input for the respective next round of implementation.

#### 3.4.3.1 First sketch

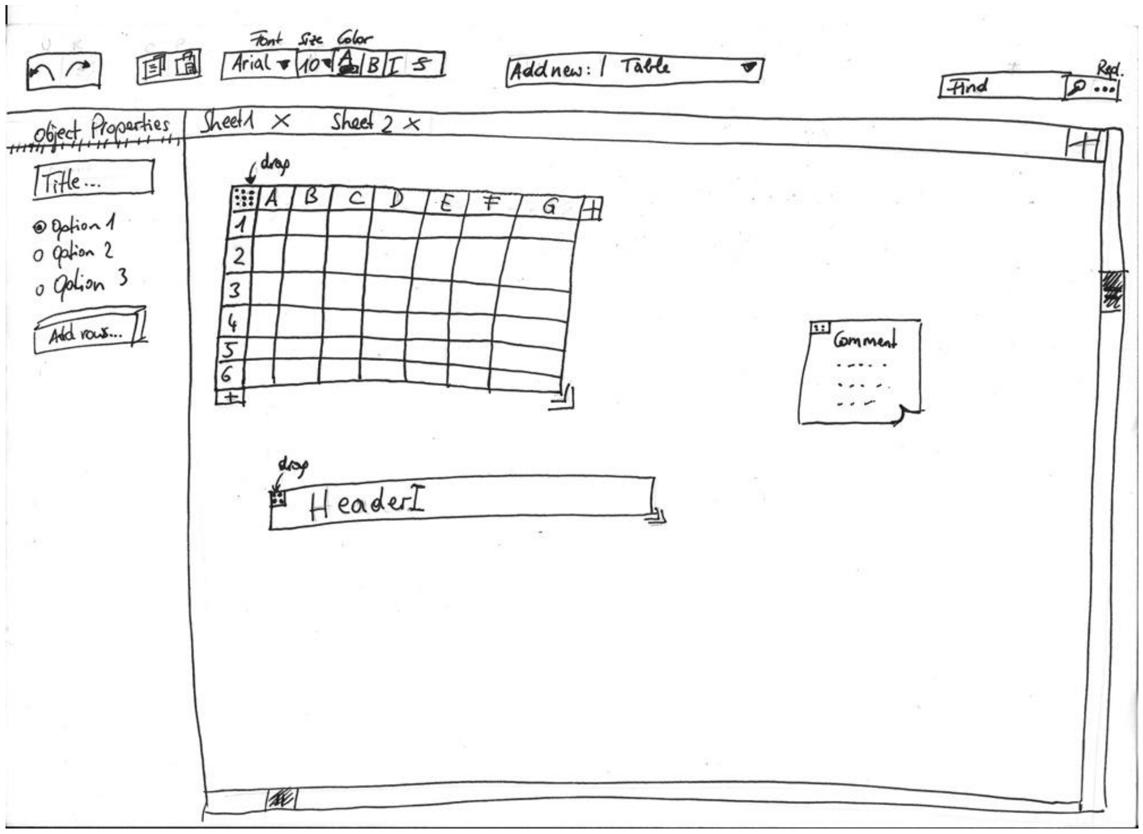


Figure 12: First sketch

Figure 12 shows the first hand-drawn sketch for the prototype interface. It includes a main toolbar on top, containing UI controls for basic features that can be applied to any element on the workspace, as well as a prominently placed search field on the top right. Below is the actual workspace which replaces the well-known grid and contains all kinds of elements, such as tables and headers. Another toolbar is located on the left which contains controls specific to the currently focused element, e.g. table or image tools.

### 3.4.3.2 Wireframes

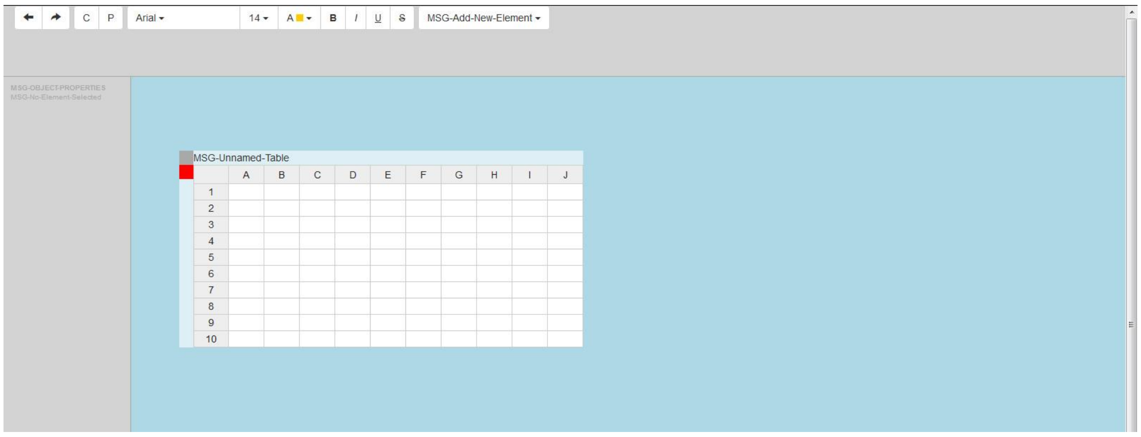


Figure 13: First wireframe

I decided not to use any special wireframing tools, but rather build the sketched interface in HTML and CSS right away. Employing the popular Bootstrap[29] frontend framework significantly reduced the work needed to create standard UI elements, such as buttons or dropdown menus.

The wireframes shown in Figure 13 and Figure 14 were completely non-functional, but nevertheless were not “dead” artifacts which I threw away after evaluating them. Instead, they were the basis for the following functional prototypes.

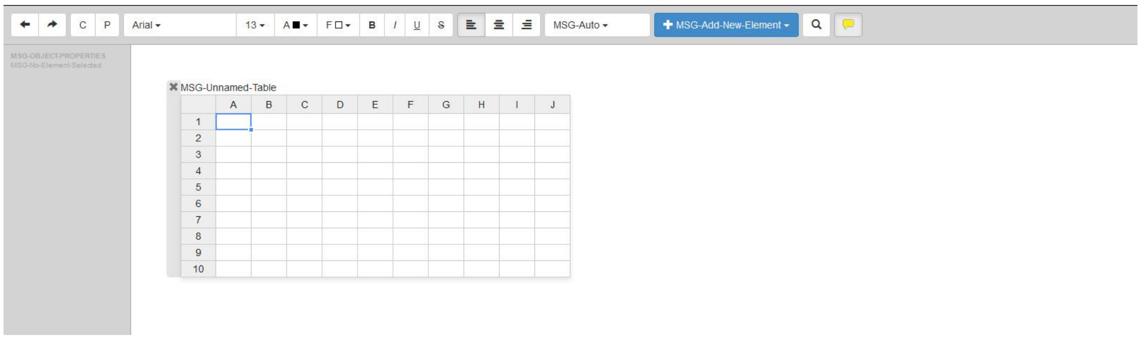


Figure 14: Second wireframe

[29] http://getbootstrap.com/ [Accessed: 08-Aug-2014]

### 3.4.3.3 First semi-functional prototype

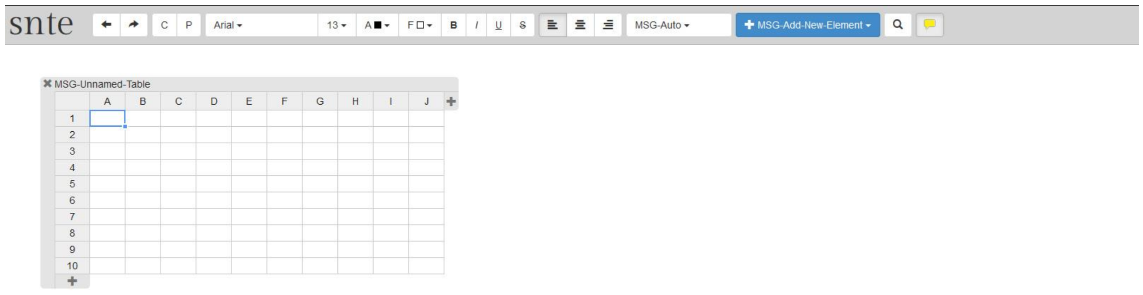


Figure 15: First semi-functional prototype

Figure 15 shows a screenshot of the first version of the prototype, which was user-tested. It included support for tables, textfields, comments and images, including creation, deletion and drag'n'drop. WYSIWYG editing with features such as font-size, font-color, background color, text align and text decoration was fully implemented for all applicable elements. During development it showed that the left toolbar is not needed and was therefore removed. This was also due to the decision to move interaction elements spatially close to where they affect something, e.g. the buttons to add table rows and columns. Wherever possible, I decided to make labels themselves editable, rather than having separate controls. For example, table headings are inline-editable.

### 3.4.3.4 Formula editing considerations

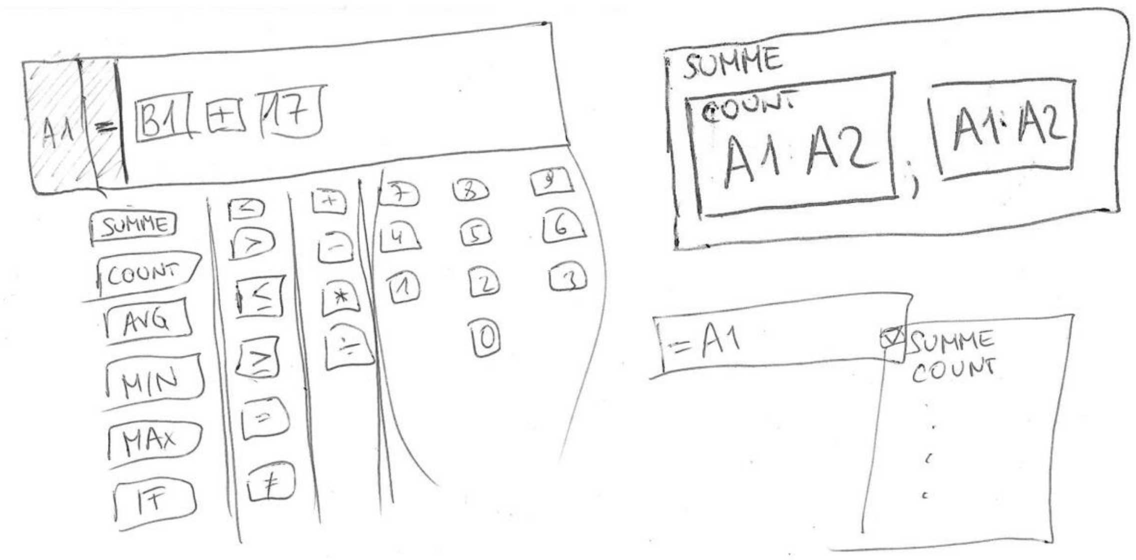


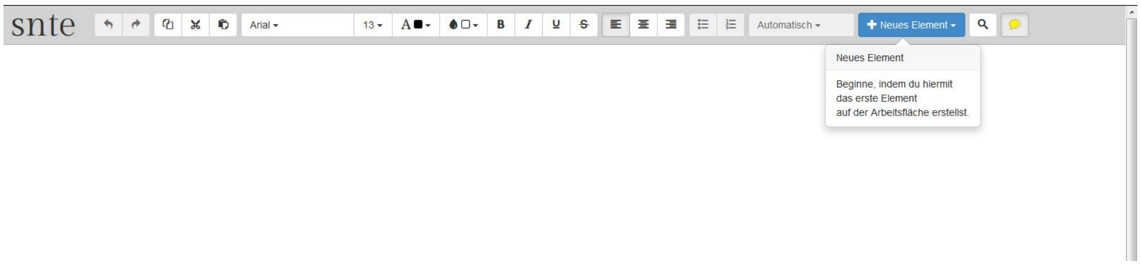
Figure 16: Sketched formula editing and visualization approaches

A lot of time and effort was invested in designing and testing a formula editor that constitutes a significant improvement over what is present in Excel and other spreadsheet programs. Brown and Gould made some suggestions on how to improve user interfaces for formula editing as early as 1987 [60]; some of them were also part of my considerations. Table 11 lists different ideas along with their advantages and disadvantages. I presented the concepts to some test users in the form of sketches (see Figure 16); some of them are shown below. In the end, the formula bar without number block turned out to be the best solution, as it increases the visibility of available functions while being rather compact and can be dynamically shown or hidden on demand. The final solution is shown in Figure 18.

Table 11: Different approaches to formula editing

| Approach | Pro | Contra |
|---|---|---|
| Simple text-based editing directly in the cells | Well-known concept | No improvement over current solutions |
| Text-based editing directly in cells with auto-complete | Well-known concept; can be used rapidly with the keyboard | Users have to know what they are looking for; at least one character needed to see suggestions |
| Excel-like formula bar located below top menu | Well-known concept; provides more space than direct editing; does not cover anything else | Is detached from the formula-containing cell |
| Calculator widget located within top menu | Makes available functions visible; easy to turn on/off automatically or on demand | Is detached from the formula-containing cell; requires a lot of vertical space |
| Calculator widget located on the workspace | Very flexible; can be freely positioned by the user | Can be invisible on large workspaces, therefore turning it on/off may be counter-intuitive; can cover other workspace elements |
| Calculator widget located in the context menu of table cells | Does not take up space per default; only visible on demand | Completely different to current concepts, additional right-click needed to display the widget; rather hard to find |
| Calculator widget toggled by an icon in cell editing mode | Does not take up space per default; only visible on demand | Cell editing has to be started to see the icon |
| Formula bar without number-block | Makes available functions visible; easy to turn on/off automatically or on demand; Takes up less space than calculator widget | Is detached from the formula-containing cell |

### 3.4.3.5 Feature-complete fourth prototype



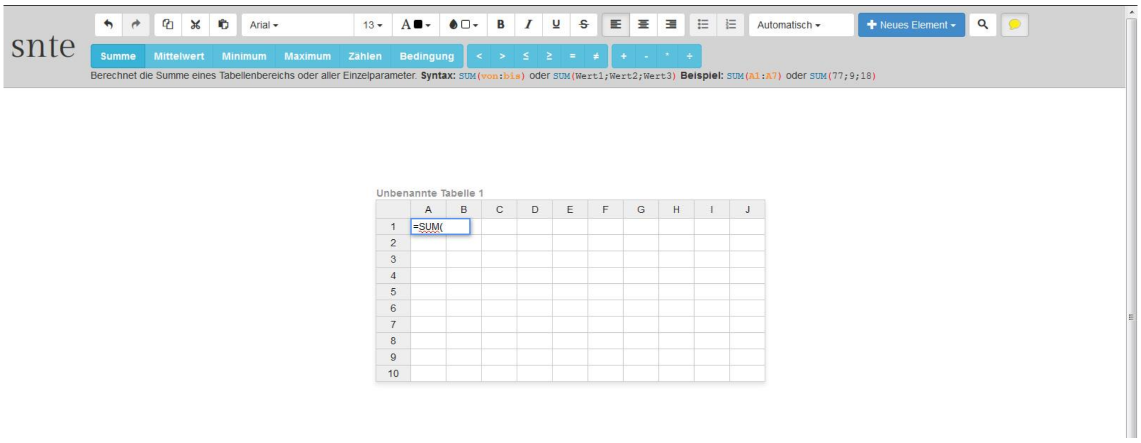
Neues Element

Beginne, indem du hiermit das erste Element auf der Arbeitsfläche erstellst.

Figure 17: Fourth prototype (feature-complete)

After four iterations, this prototype contained all the features that I decided to implement for this thesis based on prior requirement analysis. Support for basic charting was now included, as well as an internationalization component with configurable language files, which automatically chooses the appropriate language for a user based on the browser language. The program was fully translated into German and English. Other key features, such as the automatic labeling of formula cells and the formula toolbar were also introduced in this version.

### 3.4.3.6 Final seventh prototype



Berechnet die Summe eines Tabellenbereichs oder aller Einzelparameter. **Syntax:** `SUM(von:bis)` oder `SUM(Wert1;Wert2;Wert3)` **Beispiel:** `SUM(A1:A7)` oder `SUM(77;9;18)`

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | =SUM( | | | | | | | | | |
| 2 | | | | | | | | | | |
| 3 | | | | | | | | | | |
| 4 | | | | | | | | | | |
| 5 | | | | | | | | | | |
| 6 | | | | | | | | | | |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | | |
| 9 | | | | | | | | | | |
| 10 | | | | | | | | | | |

Figure 18: Final prototype after seven iterations

The sixth prototype which was used for the final user-testing round did not contain new features compared to version 4, but implemented various small tweaks for issues that became evident during three other testing rounds. All findings of the last round were then implemented in the seventh and final version of the prototype. A screenshot of the latest user interface is shown in Figure 18.

### 3.4.4 User-Testing

The whole implementation process took over three months. However, the iterations did not follow a strict time schedule; instead, I released a version and tested it together with users whenever I felt the need for user feedback. During the process the number of people involved in testing varied from one to four persons per round. The last evaluation round with version 6 was conducted with six people, four of them being from the group of observation/interview subjects.

Testing the prototype with users followed a similar scheme as the prior observations. As soon as all necessary features were implemented, I asked the subjects to do the same assignment that some of them had already done during the observation phase. However, I required all participants to use my own laptop computer to rule out possible difficulties resulting from unsupported browser versions and the like. Also, I stayed with the subjects during the whole time and took notes as we stumbled upon bugs or ambiguities. In the case of those who had already done the assignment before, I could compare the results of both rounds and they could give me feedback about which program worked better for them.

One perfect example of how important and effective user testing can be, is the way drag’n’drop was implemented for table elements in an early version of the prototype. People expect things that look similar to something they know to also behave in a similar way. In version 1, tables were only draggable using the grey handle on the left side but not the one on the top. This was due to my decision to put all control elements (drag, delete, etc.) in a control bar which appears as soon as the cursor is placed over a workspace element. However, as people are used to dragging program windows and other elements with their title bars, this flaw was immediately detected and caused astonishment.

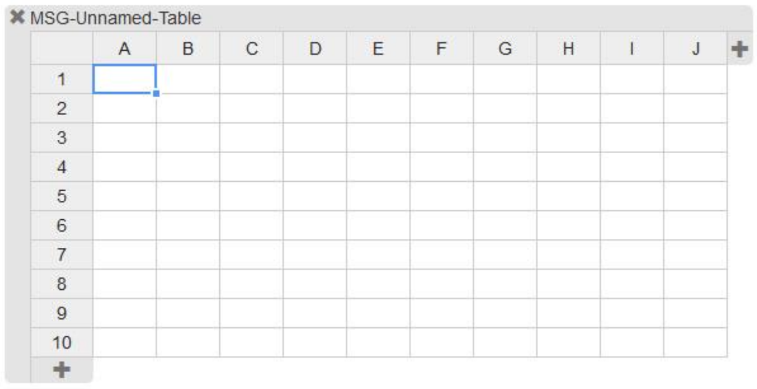


Figure 19: Table element with drag'n'drop handles on top and on the left

Another mismatch of expectations became evident during the prototyping process, this time related to comment and text elements. Originally I planned to make comments manually resizable, while textfields should be created with a certain size and then grow as needed depending on their content. However, they would never shrink below their initial size. The reason for the differing behavior of these two element types was that comments have visible boundaries (because of their background color), while textfields do not, i.e. the true size of a textfield does not matter. Figure 20 illustrates this fact.

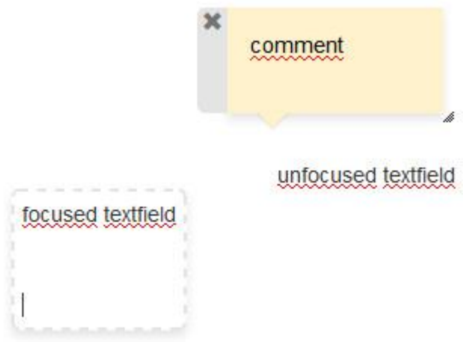


Figure 20: Textfields and comments

People then wondered why they could resize only comments but not textfields. After explaining my intention and showing them how a textfield automatically resizes after losing focus they understood, but still were not happy. And they were right: they could not know about the size of the field being irrelevant because at the time of editing the field *does have* a border, therefore it *is* relevant. A textfield's minimum size also created confusion because from a user perspective, sometimes it made a field shrink, sometimes it did not. In the end I decided for the following behavior: textfields are created with a certain rectangular size (about five lines of text high) to assure a good visibility of the fields right after creation. As soon as a textfield loses focus, it shrinks to fit the contained text. In fact, the final solution was still subject to discussion during the last testing round. However, I decided to not change it any further because it seems to be a good compromise between people's expectations and my original intention, which implies a reduction of work for the user as she does not need to resize textfields by hand.

Optimization of the formula bar, which can be seen in Figure 18, was also done with the help of test users. In its first version the bar appeared as soon as a cell was edited and a formula was started with the "="-sign. During the test this turned out not to be ideal. Indeed, it helped users to find the function name they needed, but it still required people to know how to start a formula. To fix this issue, I tweaking the bar to appear as soon as a table cell receives focus. In the latest version, the formula bar can be used in several scenarios because its buttons help the user to start a formula in case the cell is empty (by adding "`=SUM(`" to the cell, for example) or simply put a function name in other cases (by adding only "`SUM(`").

### 3.4.5 Evaluation

At the beginning of the prototyping process, my plans were to focus on the following problem areas:

- Feature finding;
- Formulas;
- Learnability.

Four strategies for simplicity – remove, hide, organize, displace – should help to reach that goal. After seven iterations some of my initial ideas remained in the product while others were developed further and even others were removed again.

#### 3.4.5.1 Remove

Removing a lot of features that are available in Excel and other similar spreadsheet programs, such as a number of formula functions and chart types, helped a lot in simplifying the user interface and building things such as the formula bar. The removal was justified by the results of my own user study and the work of other researchers ([9],). It could be shown that a majority of users only uses a small minority of the features currently available. By removing all those less frequently used features, the presented prototype is of course less versatile than Excel & Co., but on the other hand simplifies and speeds up the majority of tasks common in the private domain. Therefore, the advantages outnumber the disadvantages of the presented approach.

The remove strategy simplified feature finding because less control elements had to be hidden, thus increasing overall visibility. Building formulas also became easier as a result, as the new formula bar contains dedicated buttons for all available functions.

#### 3.4.5.2 Hide

Hiding control elements that have a low priority for a user – or even cannot be used – at a given time is a strategy to keep menus clear, thus facilitating feature finding. I employed this strategy when I designed the formula bar to be visible only when a table cell is focused. This way, buttons that are of no use while editing a comment element for example, don’t get in the way of the user.

#### 3.4.5.3 Organize

The organize strategy was employed in two different ways. Firstly, similar control elements in the main toolbar were grouped together to increase clarity. Secondly, the canvas approach with its unlimited workspace and freely movable elements facilitates the urge of users to organize and annotate their data. Both increase the learnability of the system, as well as spreadsheets created with it.

#### 3.4.5.4 Displace

Displacing certain control elements moves them into (or out of) the focus of the user as it provides important elements near to where the user currently has her attention. Microsoft, for example, follows that approach throughout their Office suite with small context-aware control panels that pop up after a right-click on certain elements. For the presented prototype this concept was used for adding table rows and columns, removing elements, editing images and element headings, for example. Consequently, the initial idea of having an element-specific toolbar on the right was dropped.

Overall, the user feedback was positive. All subjects particularly liked the canvas-approach, which replaces the traditional grid with a workspace of unlimited size where users can freely organize different elements. For example, being able to place multiple tables next to each other facilitates parallel working. One person found this especially useful, saying that "*this is genius*!"

Another pleasant comment came from Alice, who liked the way features from spreadsheet and word-processing software are combined in the prototype. She said: "*if it was already released, I would only use this instead of a thousand other programs*". Eric doubted that the prototype would change things for the better for him, but I could observe a substantial difference in how well he could work with formulas. The formula bar and the indicator icon labeling formula cells helped him a lot while the simple chart creation process even appeared "*optimal*" to him. During testing, Carol often muttered "*this is logical*" or "*this makes sense*" and in the end noted that "*this does not at all look and feel like a business program. I like it!*"

Of course the presented solution is far from perfect. Suggested features and improvements are described in Chapter 6.

# 4 Description of the Prototype

This chapter describes the final prototype, which is the result of seven rounds of development and constant user testing. The program is called SNTE, which stands for "say no to Excel", and incorporates all findings from my user study, literature review and prototyping process. All code was released as open source software under the MIT license[30] and published on GitHub[31]. A working demo is available online under http://snte.ims.tuwien.ac.at.

## 4.1 Key Features

In some ways SNTE works similarly to popular spreadsheet programs, while in others it is fundamentally different. In this section I will list distinguishing features and solutions and explain their value.

#### 4.1.1.1 Unlimited Canvas Workspace

Research has shown that the rather inflexible grid implemented in Excel & Co. often does not go well together with what people use personal spreadsheets for and how they use them. The canvas approach taken in this prototype aims to increase clarity of related data and other objects. Therefore, objects that are connected logically are also displayed on the workspace as optically and spatially separated, distinct elements.

In fact, similar solutions were already proposed in scientific papers years ago, for example by Dinmore, who wrote:

> "*In designing a literate spreadsheet, some basic ideas about the spreadsheet had to be rethought. First, the grid-centric structure did not lend itself well to the book-like arrangement suggested in literate programming. Breaking spreadsheets into chunks logically follows suggested best practices for organizing a spreadsheet, but requires the use of chunk labels in order to refer to sheets from within formulas (just as one might across worksheets in a workbook). This results in a structure […], which suggests a fusion of the spreadsheet and word processor.*" [25]

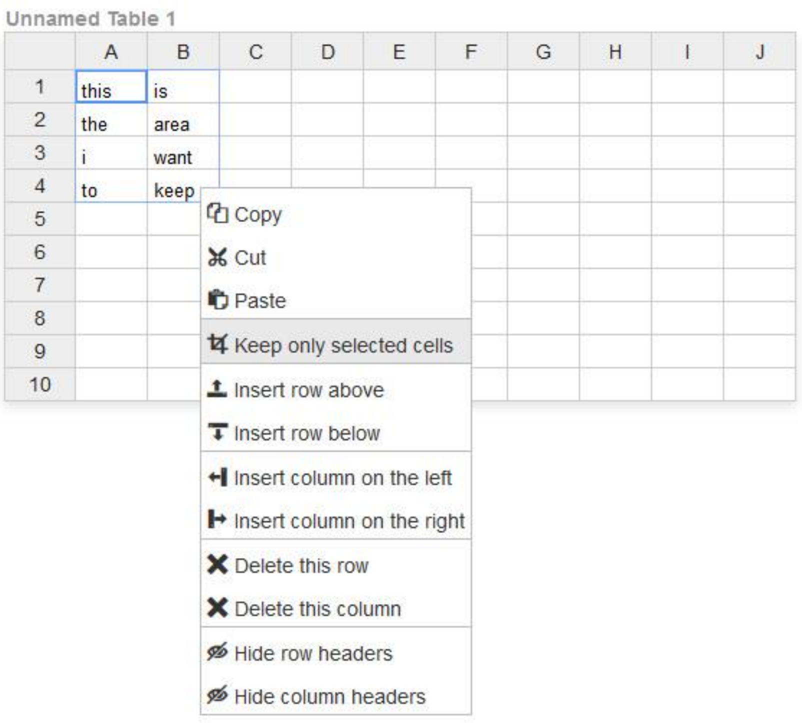
My research confirmed what Sajaniemi et al. described in "Goals and plans in spreadsheet calculation" [20], i.e. that in typical spreadsheets a variety of physical and logical data structures occur, which often show strong connections. Kohlhase called that phenomenon "Neighborhood of Information" [61]. With the canvas approach, I aimed to increase visibility of those "mental building blocks" by making them stand out more in comparison to the traditional grid. SNTE thereby implements what Sajaniemi suggested over a decade ago:

[30] http://opensource.org/licenses/MIT [Accessed: 08-Aug-2014]

[31] https://github.com/csizmazia/say-no-to-excel/ [Accessed: 08-Aug-2014]

> "*Users of spreadsheets mentally group cells into larger structures: 1-dimensional lists and 2-dimensional tables, possibly with headings. Such areas form the mental building blocks of spreadsheet applications, and users see them as separate areas, i.e., lists and tables are individual entities that may well be repositioned anywhere in the sheet.*" [14]

While images, charts and comments can be freely resized and textfields adjust their size automatically to fit their content, new tables are created per default with ten rows and ten columns. This size allows users to immediately start inserting data while still being small enough to take up only a fraction of the screen space. A clever feature, which was suggested by one of my test users during the prototyping process, allows users to "condense" tables by deleting all unwanted cells, thereby keeping tables as compact as possible. It works by selecting those cells a user wants to keep, opening the context menu with a right-click, and selecting "keep only selected cells" (see Figure 21). The program asks for confirmation in case some cells that are to be deleted contain data.


Unnamed Table 1

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | this | is | | | | | | | | |
| 2 | the | area | | | | | | | | |
| 3 | i | want | | | | | | | | |
| 4 | to | keep | | | | | | | | |
| 5 | | | | | | | | | | |
| 6 | | | | | | | | | | |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | | |
| 9 | | | | | | | | | | |
| 10 | | | | | | | | | | |

Figure 21: Ways to condense tables in SNTE

### 4.1.1.2 Integration of Spreadsheet and Word Processing Features

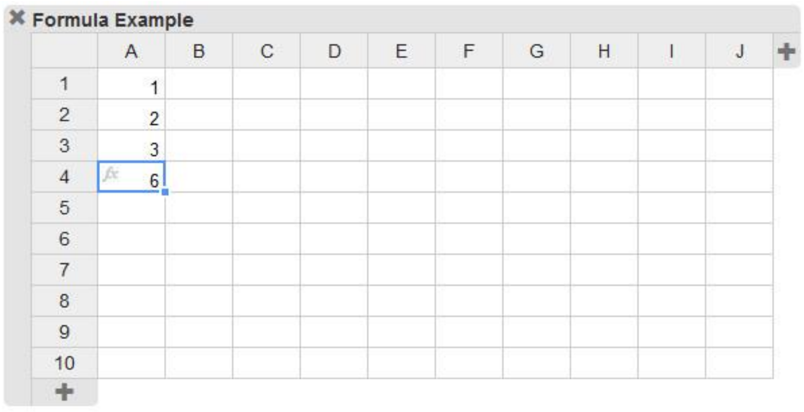
The decision to merge features from spreadsheet programs as well as word processors into a single piece of software was a logical implication of the canvas approach. It was also based on insight from my user study indicating that spreadsheets often contain a considerable amount of text.

Sajaniemi recognized the same issue when he wrote that reasonably rich annotation of spreadsheet data can only be achieved by copying the data into a text document. During this process, references and context might get lost. In such cases users also have to learn using two separate programs. The author even presented a prototypical implementation of the integrated approach, but did not conduct any kind of user testing. [14]

Since most UI controls available for formatting purposes are the same for tables and text-fields, this solution provides great value for the user while having a relatively low impact on development. Furthermore, the reuse of buttons and other UI elements made a compact menu structure possible.

### 4.1.1.3 Indicator for Formula Cells

Visibility is a key principle of functional user interface design, as suggested by Donald Norman in his famous book “The Design of Everyday Things” [62]. It is even more important for systems where a lot of “magic” happens in the background, e.g. formula calculation in spreadsheets. My research showed that people struggle to understand spreadsheets that contain formulas because it is hard for them to see which cells contain formulas and how they are connected. Editing such sheets often results in a trial and error process.



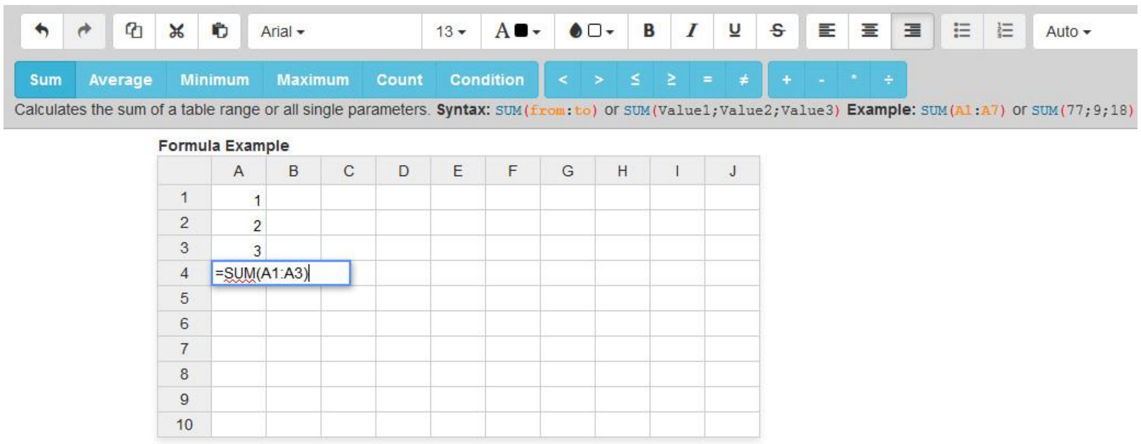
|  | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 |  |  |  |  |  |  |  |  |  |
| 2 | 2 |  |  |  |  |  |  |  |  |  |
| 3 | 3 |  |  |  |  |  |  |  |  |  |
| 4 | fx 6 |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |

Figure 22: Formula indicator in SNTE

A small icon labeling formula-containing cells was introduced in SNTE to increase visibility and make spreadsheets easier to understand. It helps users catch a glimpse of where formulas are located in a given sheet. This kind of annotation can also help finding and debugging formula errors as denoted by Sajaniemi [4].

### 4.1.1.4 The Formula Bar

Even though popular spreadsheet applications also contain “formula bars”, the concepts are fundamentally different. In Excel & Co. the formula bar is just another input field giving the user more space to edit a cell’s content. Finding functions and their syntax requires at least two more clicks and involves a sub-menu in a pop-up.



Calculates the sum of a table range or all single parameters. **Syntax:** `SUM(from:to)` or `SUM(Value1;Value2;Value3)` **Example:** `SUM(A1:A7)` or `SUM(77;9;18)`

**Formula Example**

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | | | | | | | | |
| 2 | 2 | | | | | | | | | |
| 3 | 3 | | | | | | | | | |
| 4 | =SUM(A1:A3) | | | | | | | | | |
| 5 | | | | | | | | | | |
| 6 | | | | | | | | | | |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | | |
| 9 | | | | | | | | | | |
| 10 | | | | | | | | | | |

Figure 23: Formula bar in SNTE with inline help

The reduction of functions as a result of requirement analysis made it possible to create a new kind of formula bar in SNTE. It shows all available functions and arithmetic operations at a glance, thus increasing visibility. Placing the cursor over a button also reveals a short explanation of how to use a specific function and its syntax. All this is placed within the top menu and does not require any sub menus or additional steps of interaction. The formula bar is context-aware in a sense that it is hidden per default, but automatically shown as soon as a table cell receives focus. Actual editing of formulas and static data is done directly within the cells, helping the user to concentrate on the element of interest, which was also suggested by Brown and Gould [60].

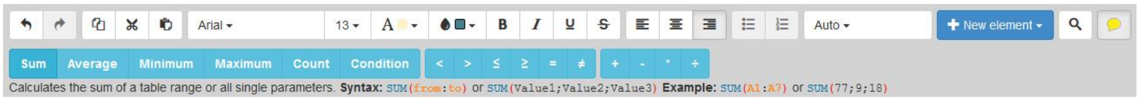
Dedicated buttons for functions and mathematical operations solve the problem accurately described by Hendry and Green:

> *„Spreadsheet users encounter both kinds of problem [sic]. Faced with an impasse, they often do not have a route to forming the appropriate plan; and when they have a plan, they often cannot find the appropriate function. When such a function is called GET.DOCUMENT(10), who can blame them?"* [2]

## 4.2 User Interface

As mentioned above, the SNTE user interface was built incorporating the four strategies to simplicity: remove, hide, organize and displace. This section describes important menus and interaction paths.

The main menu is located on the top of the screen, following the convention in most modern software applications. It consists of two rows of grouped elements: the upper row contains controls for undo/redo, copy/paste, formatting, text align, lists, cell types/formatting, creation of new workspace elements, search, as well as toggling visibility of comments. The second contains the formula bar and is visible only as long as a formula cell is in focus.



Calculates the sum of a table range or all single parameters. **Syntax:** `SUM(from:to)` or `SUM(Value1;Value2;Value3)` **Example:** `SUM(A1:A7)` or `SUM(77;9;18)`

Figure 24: SNTE main menu

Familiar icons were used for standard features for the sake of consistency with other applications. Additionally, every interactive element – buttons, dropdowns, etc. – in SNTE is equipped with a tooltip that briefly describes its function. To amplify the helpfulness of those tooltips, they become visible immediately after the cursor is placed over an element – as opposed to tooltips in many other programs that show after about one second.

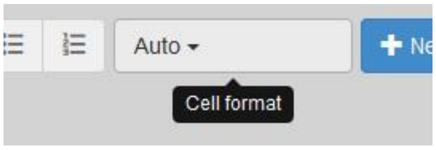


Figure 25: Tooltip in SNTE

The control for adding new elements to the workspace is highlighted using a different color to increase visibility. As long as the workspace is empty, it is further emphasized by an additional element helping the user to get started. This turned out to be necessary because people are used to seeing the traditional grid after creating a new spreadsheet. Some modern mobile applications also implement such guides to familiarize novice users with an interface.

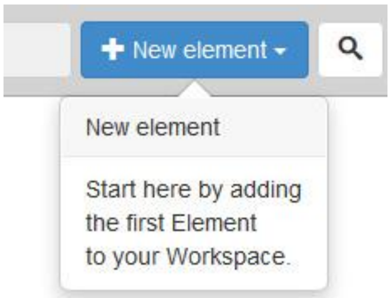
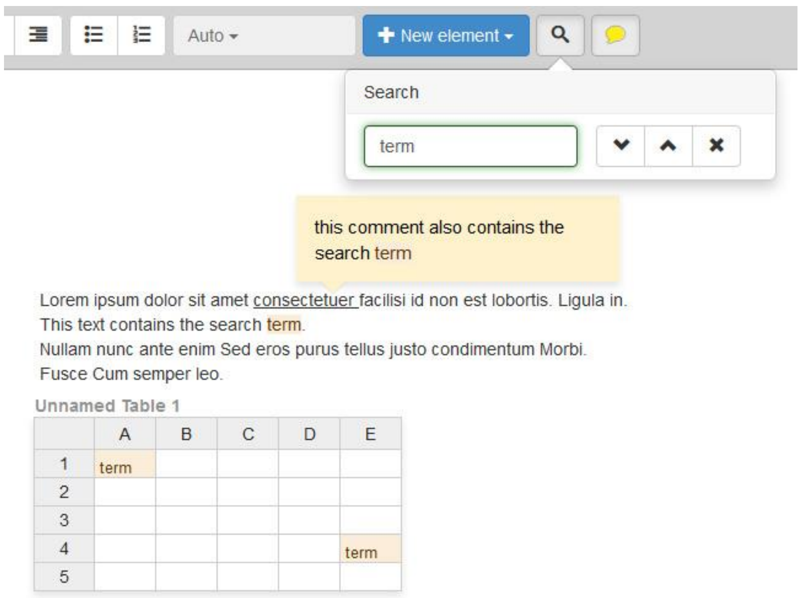


Figure 26: Guide helping novice users to get started with an empty workspace

As can be seen in Figure 24, the formula bar buttons also have a different color than the other controls of the main menu. This is due to two reasons: Firstly, it helps make them appear as one group of connected elements. Secondly, it increases visibility and attracts the user's attention.

The search widget works analogously to many others, for example the one in Google Spreadsheets. Search results are highlighted by color and can be traversed using the two buttons or the enter key.



this comment also contains the search term

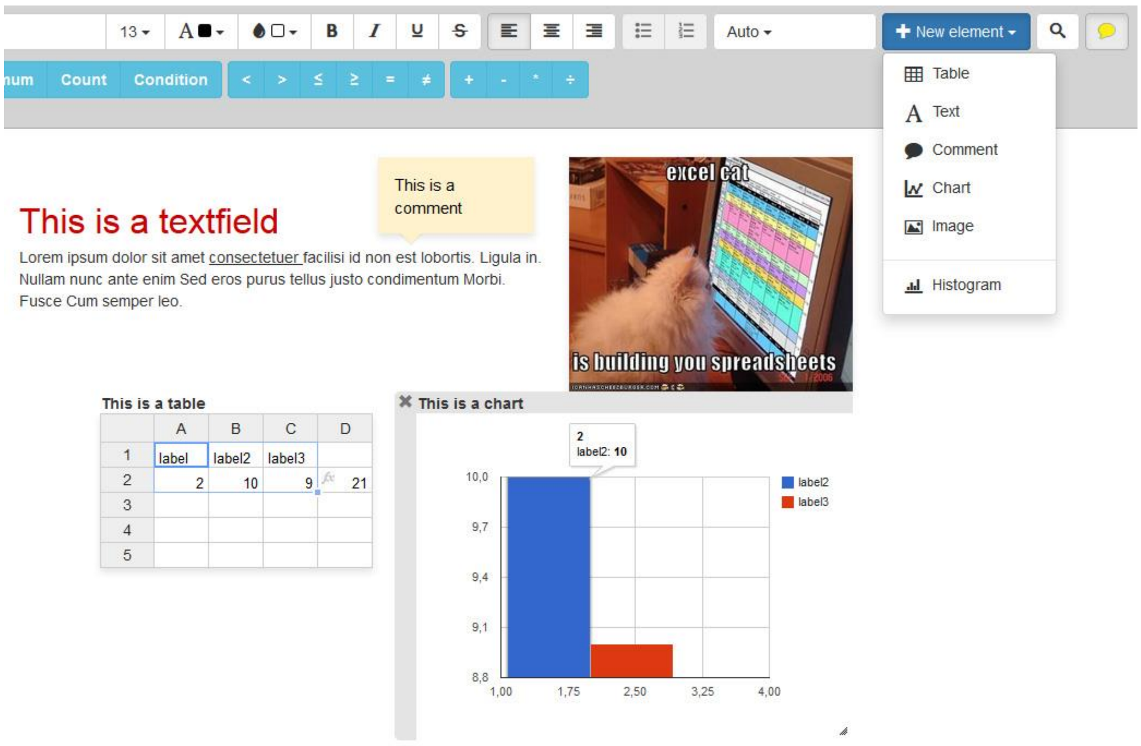
Lorem ipsum dolor sit amet consectetuer facilisi id non est lobortis. Ligula in.
This text contains the search term.
Nullam nunc ante enim Sed eros purus tellus justo condimentum Morbi.
Fusce Cum semper leo.

Unnamed Table 1

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | term | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | term |
| 5 | | | | | |

Figure 27: SNTE search widget

Figure 28 shows the five different basic elements currently implemented in SNTE: tables, textfields, comments, charts and images. All elements can be freely positioned on the work-space using the grey handles which become visible when the cursor is placed above an element; a “magnetic snap” feature helps aligning them nicely.



This is a textfield

Lorem ipsum dolor sit amet consectetuer facilisi id non est lobortis. Ligula in.
Nullam nunc ante enim Sed eros purus tellus justo condimentum Morbi.
Fusce Cum semper leo.

This is a comment

This is a table

| | A | B | C | D |
|---|---|---|---|---|
| 1 | label | label2 | label3 | |
| 2 | 2 | 10 | 9 | 21 |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |

This is a chart

Figure 28: Different elements available in SNTE

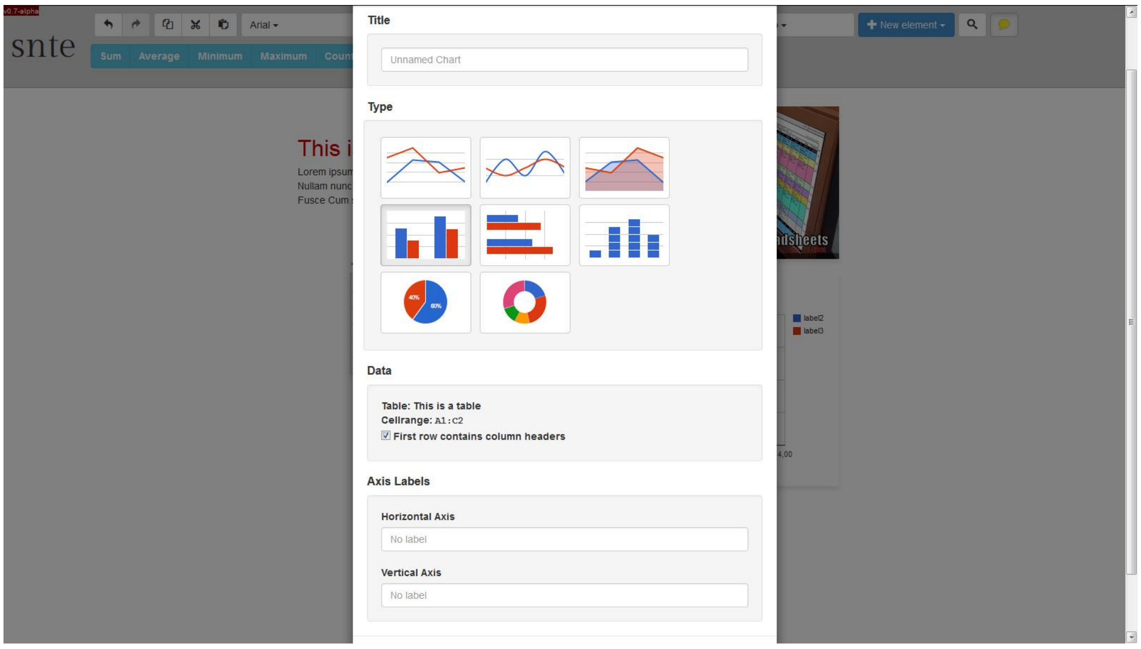
Tables and charts both have built-in fields for headings, incorporating the existing habit of some users to annotate their data and encouraging others to do so as well. This follows the results of my user study as well as what Kohlhase claimed when she wrote:

> "*As a consequence authors should be compelled to create context, e.g. respective Headers or Legends if the spreadsheet is meant to be distributed[…]*" [61]

Images can be added to the workspace either from the local file system or by providing a web link. They are resizable using a handle on their bottom right corner and can be brought back to their original size any time double clicking said handle.

Comments are designed to look similar to post-it notes, even though the user can adjust the color. To be able to connect comments to specific points on the workspace, they have small triangles at the bottom, analogous to speech-bubbles.

Charts are created with the simple dialog shown in Figure 29. Users can define the chart title as well as axis labels, if applicable. Currently available chart types include line-, area-, column-, bar-, pie- and donut-charts, as well as histograms. Those were found to be the most commonly used by Chambers and Scaffidi [9] and confirmed by my user study.


Title

Unnamed Chart

Type

Data

Table: This is a table
Cellrange: A1:C2
☑ First row contains column headers

Axis Labels

Horizontal Axis

No label

Vertical Axis

No label

Figure 29: SNTE chart dialog

## 4.3 Technical Aspects

SNTE was built using standard web technologies to work with any modern web browser. The focus lay on the use with desktop operating systems because building software that also works well on mobile devices – especially in terms of user interaction principles – would have gone beyond the scope of this thesis.

HTML5[32] serves as a basis for the SNTE user interface. When a user directs her web browser to a website serving SNTE (e.g. http://snte.steviec.at), an HTML5 page is loaded together with CSS3[33] style information and rendered by the browser. The browser then loads different JavaScript[34] files containing program code of the following components:

- Bootstrap[35]: a popular frontend framework used to build standard UI elements such as buttons, dropdowns, tooltips, etc.;
- Google Charts[36]: a comprehensive API to create various kinds of data visualizations used to render charts in SNTE;
- Handsontable[37]: a powerful jQuery component, which, together with the handsontable-excel[38] plugin, served as a basis for the table elements in SNTE;
- i18next[39]: a JavaScript internationalization framework used to translate SNTE into different languages;
- jQuery[40]: one of the most popular JavaScript libraries used for easy DOM (Document Object Model) access and manipulation;
- jQuery UI[41]: a curated set of user interface interactions, effects, widgets, and themes built on top of jQuery;
- Numeral[42]: a JavaScript library for formatting and manipulating numbers used in the parsing of formulas and printing of their results;
- phpJS[43]: a library of PHP[44] functions ported to JavaScript used for parsing and manipulating dates in SNTE;
- SNTE: the program itself containing all application logic and putting together frameworks and libraries.

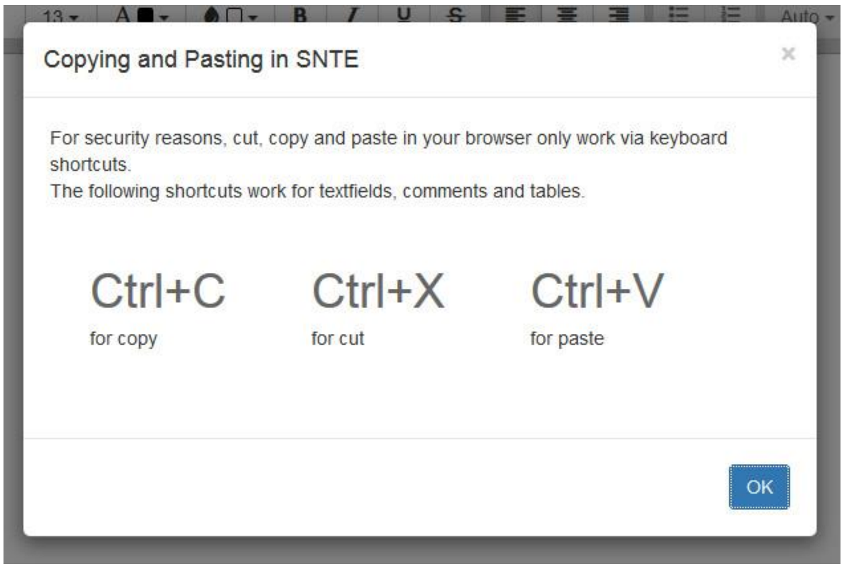
---

[32] http://www.w3.org/TR/html5/ [Accessed: 08-Aug-2014]
[33] http://www.w3.org/TR/CSS/ [Accessed: 08-Aug-2014]
[34] An interpreted programming language widely used on the Web.
[35] http://getbootstrap.com/ [Accessed: 08-Aug-2014]
[36] https://developers.google.com/chart/ [Accessed: 08-Aug-2014]
[37] http://handsontable.com/ [Accessed: 08-Aug-2014]
[38] https://github.com/gurubenka/handsontable-excel [Accessed: 08-Aug-2014]
[39] http://i18next.com/ [Accessed: 08-Aug-2014]
[40] http://jquery.com/ [Accessed: 08-Aug-2014]
[41] http://jqueryui.com/ [Accessed: 08-Aug-2014]
[42] http://numeraljs.com/ [Accessed: 08-Aug-2014]
[43] http://phpjs.org/ [Accessed: 08-Aug-2014]
[44] https://php.net/ [Accessed: 08-Aug-2014]

As mentioned above, the latest prototype of SNTE does not contain any backend functionality. All features are implemented in client-side JavaScript code which is executed in the web browser. This setup guarantees great responsiveness, but of course limits the abilities to store and share spreadsheet files. However, this was intentionally excluded from the list of planned features for this prototype, as it would not have contributed to reaching the specified research goals.

A special solution had to be found for copying and pasting text in SNTE because programmatic access to the user's clipboard is disabled in all major web browsers due to security reasons. However, after observing a lot of people to use them in other programs, I deliberately decided to place dedicated buttons for cut, copy and paste in the main menu as well as in the context menu of table elements. Textfields and comments use the browser's standard context menu for HTML5 "contenteditable" elements, which contains these controls by default.

Pressing said buttons opens a modal window explaining why and how to use keyboard shortcuts. This solution was inspired by Google Spreadsheets, which apparently has to cope with the same problems.


Copying and Pasting in SNTE

For security reasons, cut, copy and paste in your browser only work via keyboard shortcuts.
The following shortcuts work for textfields, comments and tables.

Ctrl+C for copy

Ctrl+X for cut

Ctrl+V for paste

Figure 30: Cut, copy and paste in SNTE

# 5 Evaluation of the Prototype

"*A usable system reduces to a minimum the cost of the user in reaching the appropriate enabling states for their goal tasks*." [2]

Is SNTE a usable system? This chapter describes the measures taken to answer this question.

In his article "Number Crunching without Programming: The Evolution of Spreadsheet Usability" [5], Martin Campbell-Kelly defined three major parameters that were decisive for the success or failure of spreadsheet programs released until this point:

- Responsiveness: being especially important in the early days of spreadsheets where automated parallel computation of multiple formulas was *the* killer feature, this is still a quality attribute of any interactive system;
- Familiar UI elements: facilitate the change from one software to another and reduce learning effort because they make a system "guessable" based on the user's previous interaction with other systems;
- Compatible file formats: being able to re-use existing spreadsheets and not losing all data is even more important for changing to new software.

Except for the latter, which is out of scope of the presented prototype, we believe that SNTE fulfills these criteria for success. The user interface contains familiar icons and adopts concepts from current spreadsheet programs wherever it is in line with the users' requirements. Furthermore, due to an entirely client-side implementation, it is responsive and fast; STNE was designed so that every user action generates immediate feedback.

## 5.1 Concluding Online Survey

To get even more people to try out and assess SNTE, I conducted another online survey after the prototyping process ended, i.e. after findings from the last user testing round with version 6 were incorporated into a final version 7 of the software. This survey was similar to the one described in Section 3.1, as it also had no strictly defined target group; its goal was to get as many people as possible to try out the prototype. The survey was again done using Google Forms and distributed in the same channels as the previous survey.

The questionnaire consisted of five sections focusing on different aspects:

1. "The prototype": people were asked to try out SNTE and specify the amount of time invested, their feelings during or after the test as well as at least one aspect (positive or negative) that struck them;
2. "The competitors: Excel & Co.": six questions about spreadsheet knowledge and use, including an assessment of Excel's complexity as well as two free text questions about usage scenarios and problems in the private domain. This section was exactly the same as in the previous questionnaire;

3. “SNTE compared to competitors”: similar questions as in section 2, but relating to SNTE;
4. “Histograms in SNTE”: two questions to assess what people like more: a dedicated histogram component/app or standard elements;
5. “The usual”: two questions asking about age and gender of the participant.

A primer explained which web browsers were safe to use with SNTE and pointed out that there was no support for storing created spreadsheets. The latter was shown by a pre-test to be especially important to avoid annoying the participants.

Unfortunately, this survey only reached far fewer people than the first, one reason probably being the substantially increased amount of time and effort needed to complete it. Within one month, 14 people took part and filled out the questionnaire; 13 of them were male, 1 person was female. The age of participants ranged between 19 and 49 years.

Participants of the survey worked with SNTE an average of 24 minutes (minimum 3 minutes, maximum 120 minutes, median 15 minutes).Table 12 lists the feelings participants had after their first contact with the prototype. When asked about what stood out for them, subjects gave the following valuable feedback.

Positive:

- Intuitive GUI, clear arrangement of the main menu;
- Main menu automatically adapting to the active object;
- Similarity to known designs is helpful for fast orientation and learning;
- Easy way to work in parallel with multiple tables;
- Possible future scenario: hidden data tables with visible visualization to hide complexity while still being able to “look behind the curtains” if necessary;
- Use of different building blocks (formulas, charts, etc.) is easy to understand;
- Simplified processes (e.g. insertion of textfields and images);
- Possibility to annotate spreadsheets with comments;
- Surprisingly fluid interactions;
- SNTE contains the most important features of Excel;
- Practical workspace with different movable elements.

Negative:

- Fiddly drag’n’drop of table, text and comment elements. A bigger handle could help (e.g. the whole element as it already works for charts and images);
- Missing features (pivot tables, printing, import/export, formula references between tables);
- Missing support for keyboard shortcuts.

Table 12: People's feelings after trying out SNTE for the first time

| Feeling | Count |
|---|---:|
| Euphoric | 0 |
| Satisfied | 3 |
| Pleasantly surprised | 7 |
| Curious | 8 |
| Sceptic | 5 |
| Bored | 0 |
| Overstrained | 2 |
| Disappointed | 1 |

A majority of participants assessed themselves as advanced spreadsheet program users – 11 of 14 picked a value of 7 or higher on the scale from 1 to 10 described in Section 3.1.1. The same number of people has also practical experience with at least one program other than Microsoft Excel.

To be able to compare results, I repeated the question on how well people can use major features – this time both for existing spreadsheet programs and for SNTE. For the latter, macros and forms were not included as they were not implemented in the prototype. The results are shown in Figure 31 (current spreadsheet programs) and Figure 32 (SNTE) and indicate two main facts. Firstly, SNTE supports the applications it was designed for well. Secondly, there is room for improvement for the charting component. However, charting was not the main focus of the development work.

With all answers being in the upper half of the ten-point scale, participants by tendency rated current spreadsheet programs as rather complex and over-engineered, thus confirming results from the initial survey. SNTE seems to implement a good mixture of functionality and simplicity: 13 subjects rated the prototype between 3 and 7 on the same scale. One person found the software to be under-engineered.

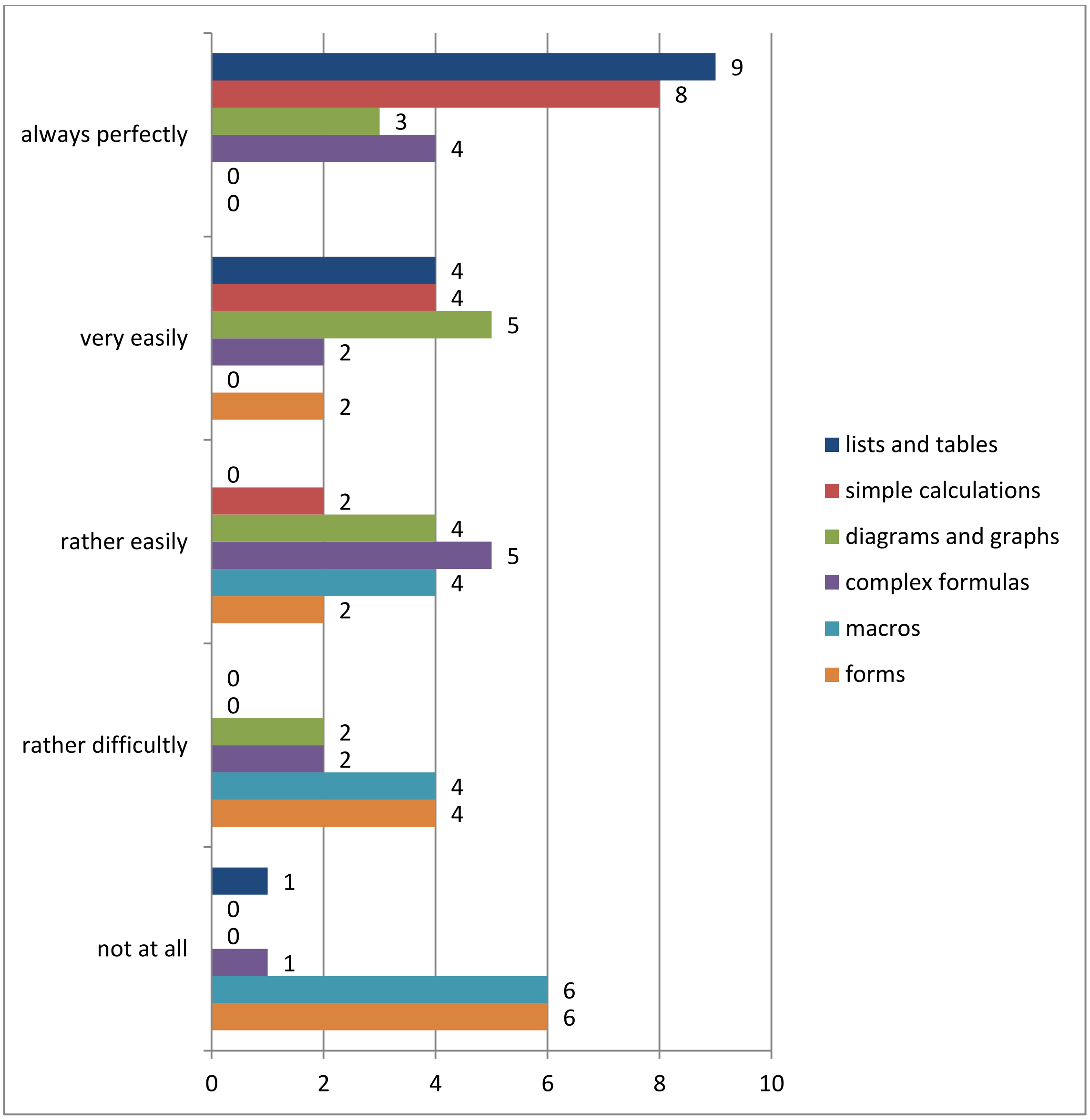


Figure 31: How well people can use major features in current spreadsheet programs

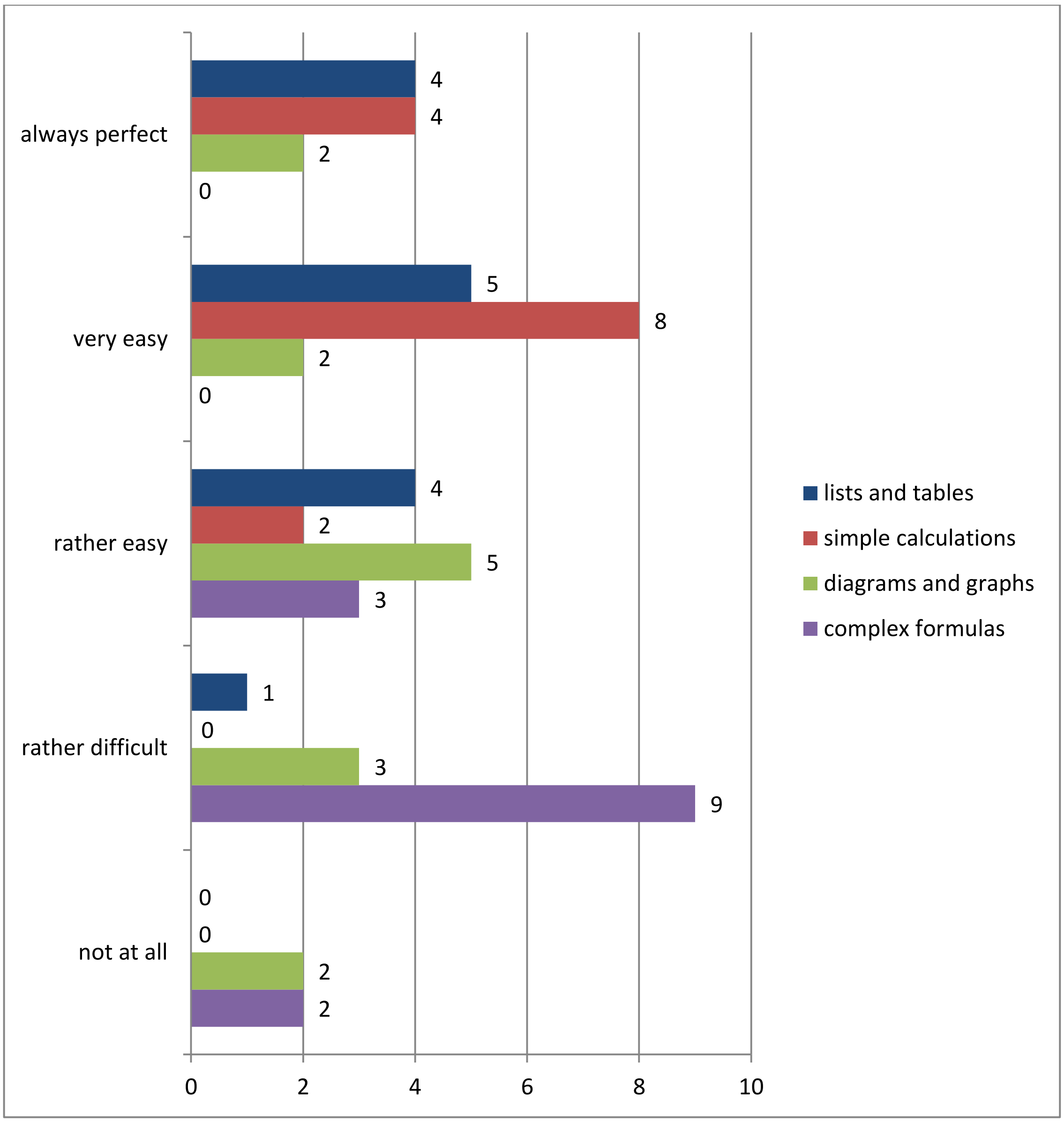


Figure 32: How well people can use major features in SNTE

Another question aimed to assess the acceptance of the innovative canvas approach with freely moveable elements. All participants found it to be “very” or “rather” lucid and practical. For 57% the concept felt “very” or “rather” unfamiliar, while the rest seemed to know the approach from other applications. The workspace facilitates parallel working for 13 subjects and over 70% find the canvas approach “better than the grid known from Excel & Co.”, while only one person prefers the traditional interface.

SNTE was built to contain a majority of features necessary for private applications. However, subjects of this survey came up with the following list of missing functionalities that would be useful for their everyday tasks:

- Printing;
- Import and export (CSV for data, PDF for viewing, etc.);
- Saving files;
- Support for collaborative work;

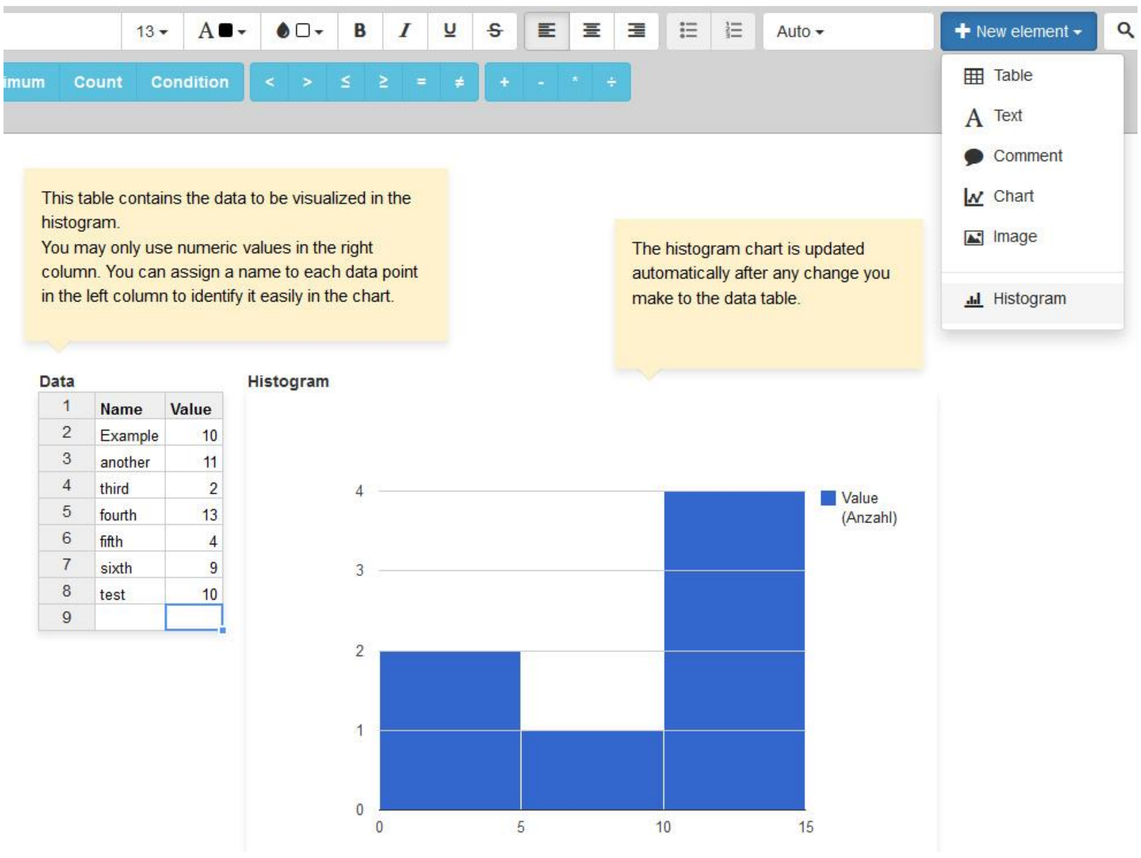
- Merging of cells;
- Pivot tables;
- Support for calculations with time-units;
- A simple way to add a number of rows to a table at once;
- References between tables;
- Support for exponential numbers;
- Improved formula editing (automatic closing of brackets);
- Logical operators.

Even though SNTE is currently in the status of an early prototype, more than half the number of participants said they would consider using the program as an alternative to what they currently use. However, some essential features are missing, for example saving files and collaborative editing. Nearly all of the subjects complimented the simplicity of SNTE and the workspace solution. Some see the software being web-based, open source and not developed by a large IT enterprise as additional benefits.

## 5.2 Multi-purpose Tool vs. Small Specialized Apps

The way workspace elements in SNTE are built, allows for reuse and combination. Therefore, SNTE can serve as a framework for small, easy-to-use spreadsheet applications. The presented prototype is drafted as such an application built with SNTE components, yet it can be used for various different tasks. Therefore, I built an even more specialized app on top of SNTE and evaluated it with the above-mentioned online survey to answer the question of whether people prefer small, specialized apps over more complex multi-purpose tools. For testing purposes it was integrated into the presented SNTE user interface as an additional component. The app specializes in creating histograms, a graphical representation of the distribution of data, commonly used in statistics, image processing and other areas.

Figure 33 shows a screenshot of what this “Histogram” app, consisting of an SNTE table, a chart and comments, looks like. The data table has the pre-configured headers “Name” and “Value” and consists of only two columns with pre-set formatting. Adding more columns is disabled, as they would not be of use for the creation of the histogram. To make the insertion of data as uncomplicated as possible, the button for manually adding rows is also hidden. Instead, the table grows automatically in a way that there is always a spare row at the end. The actual histogram chart is tied to the data table and at any time visualizes all rows that contain data. Two pre-defined comment elements give assistance on how the histogram component works.



This table contains the data to be visualized in the histogram.
You may only use numeric values in the right column. You can assign a name to each data point in the left column to identify it easily in the chart.

The histogram chart is updated automatically after any change you make to the data table.

Data

| | | |
|---|---|---|
| 1 | Name | Value |
| 2 | Example | 10 |
| 3 | another | 11 |
| 4 | third | 2 |
| 5 | fourth | 13 |
| 6 | fifth | 4 |
| 7 | sixth | 9 |
| 8 | test | 10 |
| 9 | | |

Figure 33: Histogram component built with SNTE workspace elements

As mentioned above, the adoption of this approach – as opposed to building a histogram manually with SNTE – was evaluated in one section of the final survey. The results are not as clear: 64% (9 persons) prefer the dedicated histogram element, 36% (5 persons) would rather build histograms manually. Some were unsure about the histogram chart and “all elements being of the same size”. One issue I could see is that there is currently no explanation for how SNTE calculates buckets automatically. This is probably too oversimplified and may have confused users, especially given the fact that people who do histograms usually know what histograms are and how they function. Not being able to define buckets might hinder them more than actually helping them.

## 5.3 Comparison with Competitors

Most spreadsheet programs currently available have a very similar range of functions and user interfaces alike. In fact, this is nothing new in the history of spreadsheets, where successful products were often copied to a certain extent (see Section 2.1.1). Today, the biggest difference between Microsoft Excel and LibreOffice Calc is that the latter is free and open source software. Both try to consider as many use cases as possible, resulting in complex software packages. The presented prototype takes a completely different approach, as it contains only a reduced set of features concentrating on private use-cases. For the sake of interoperability and learnability, SNTE adopted conventions and interaction paradigms known from Excel and Calc, for example function names or cell types. Probably the most distinctive part is the replacement of the traditional grid with a more flexible workspace and freely movable elements. Google Spreadsheets have in common with SNTE that they are purely web-based, although otherwise they function in most ways very similarly to Excel and Calc.

Apple Numbers follows a similar approach to SNTE with elements to be positioned on a white canvas, yet the concept focuses on printable documents, therefore limiting the standard size of the canvas. It also comes with a number of ready-made templates for specific use-cases. Such templates can help users getting started with certain tasks, which, in this sense, make them similar to specialized apps. However, templates like these do not affect the program's user interface in a way that it gets more specific to the respective task; menus and toolbars still offer all available functionality no matter if it could be helpful for a task or not. Therefore, such templates are kind of “half-way” solutions to easy-to-use spreadsheet apps. In fact, Numbers incorporates a lot of the findings and insights of this thesis, which, from my point of view, makes it the best solution currently available on the market. Its severest disadvantage, however, is its exclusive availability for the Mac operating systems.

# 6 Summary and Future Work

Before I started working on this thesis, all I had was the assumption that a majority of spreadsheet software users only use a minority of the features available and do not know, do not understand and/or do not need the rest of them. My goal was to find solutions on how to make spreadsheets easier to use and enable users to accomplish their everyday spreadsheet-tasks faster and with less effort.

Now, about nine months later, I am able to present a working prototype which incorporates findings of extended user research and a successful user-centered design process. During the course of this process, I learned a lot about how people use spreadsheets as a tool for their personal tasks, which difficulties they encounter and what their demands are.

After reading every paper and scientific publication I could find on the topic of spreadsheets, I conducted an online survey in which about 380 people took part. This survey revealed what tasks people try to accomplish using spreadsheets, where they fail and which software they use. Participant observations and semi-structured interviews conducted with five diverse spreadsheet users generated even more detailed knowledge about problem areas and user behavior.

All results from three different yet complementary primary data gathering methods defined the requirements for a new, easy-to-use spreadsheet application and formed the basis of an iterative prototyping process with seven rounds of development and intensive user testing. The goal of this process was to minimize the difference between "*the tasks that an end-user wants to do with a design and the task that the design forces them to do*" [2].

In the end, a web-based application called SNTE is the result of a long process in which I tried to understand as much as possible about spreadsheet users and worked closely together with some of them to create a new kind of user experience. SNTE received positive feedback by test users and is more than a purely academic artifact. One pleasant comment was written by a subject of the final survey, unintentionally applauding the successful design work: "*Even though I immediately recognized the reduction of control elements in the menu, I missed only one feature after using the software for 15 minutes.*"

"*Finally, lest there be any doubts: there is still much to learn about how people use spreadsheets, and spreadsheet ecology.*" [2]

Even after some months in which I intensively worked on this topic, I *still* believe this statement to be true. As far as I am aware, this project was the first to solely concentrate on the private domain. This field has so far received very little if any attention in scientific research, perhaps also because there was too little economic interest. In fact, it is hard to encourage substantial change in an area where the market leading software has a superior market share and can be seen as a quasi-standard. However, I believe strategies could change in the near future, as a new mindset develops amongst end users. Inspired by how simple most mobile apps work and how low the effort required to try out and switch between apps has become, users could be encouraged to scrutinize whatever they have been unhappily using for years.

Due to the limited amount of time available for the work on this project, I had to narrow down the list of features to be implemented in the final version of the prototype, even though there were enough ideas to do many further iterations. In addition to the list of features suggested by survey participants described in Section 5.1, I propose the following interesting points to be implemented in future versions:

- Import and export of common spreadsheet file formats (mainly Excel);
- Easy resizing of tables analogous to image and chart elements;
- Basic features that are well known from other spreadsheet programs: drag'n'drop of cell contents, conditional formatting, copy and paste of formats;
- Features for increased workspace usability: select and modify multiple elements at once (move, delete, etc.), manually controllable stacking of elements, navigator widget known from some image processing programs for better overview over large workspaces;
- Templates with pre-defined structures for common use-cases.

Possible business cases could involve charged support as well as a market place for small apps and templates based upon SNTE elements. Furthermore, I think that developing with web-based technologies was the right choice. Not only because of the resulting rapid prototyping, but also because of a clear trend towards rich online services and web applications that became visible during recent years.

I conclude that, even though it required a lot of time, the results justified the increased effort of the user centered design process compared to traditional software development processes. I enjoyed working closely together with real users and am sure that their constant input was key to the high level of usability SNTE reached at the end. However, it would have been beneficial to pre-test the questionnaires more intensively, to be able to refine questioning and obtain even better results. The way I conducted my observations with live screencasts would also need to be evaluated. A comparative study would be required to find out whether this setup worked better than traditional participant observation with the observer being together in one room with the subject or not.

Due to their utility for many tasks, spreadsheets are widespread and Microsoft Excel is amongst the most popular computer programs of all time. However, since currently available multi-purpose spreadsheet programs are far from perfect and small user-friendly apps dedicated to specific tasks are on the rise, it will be interesting to see how this area of end-user computing further evolves.

# References

[1] M. R. Zynda, “The first killer app: a history of spreadsheets,” *interactions*, vol. 20, no. 5, pp. 68–72, Sep. 2013.

[2] D. G. Hendry and T. R. G. Green, “Creating, comprehending and explaining spreadsheets: a cognitive interpretation of what discretionary users think of the spreadsheet model,” *Int. J. Hum. Comput. Stud.*, vol. 40, no. 6, pp. 1033–1065, 1994.

[3] Wikipedia, “Spreadsheet.” [Online]. Available: http://en.wikipedia.org/wiki/Spreadsheet. [Accessed: 26-Jul-2014].

[4] J. Sajaniemi, “Modeling Spreadsheet Audit: A Rigorous Approach to Automatic Visualization,” *J. Vis. Lang. Comput.*, vol. 11, no. 1, pp. 49–82, 2000.

[5] M. Campbell-Kelly, “Number Crunching without Programming: The Evolution of Spreadsheet Usability,” *IEEE Ann. Hist. Comput.*, vol. 29, no. 3, pp. 6–19, Jul. 2007.

[6] G. Gable, C. S. Yap, and M. N. Eng, “Spreadsheet investment, criticality, and control,” in *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, 1991, vol. iii, pp. 153–162.

[7] J. Hihn, S. Lewicki, and B. Wilkinson, “How Spreadsheets Get Us to Mars and Beyond,” in *2009 42nd Hawaii International Conference on System Sciences*, 2009, pp. 1–9.

[8] A. Ginige, L. Paolino, M. Sebillo, R. Shrodkar, and G. Vitiello, “User requirements for a web based spreadsheet-mediated collaboration,” in *Proceedings of the International Conference on Advanced Visual Interfaces - AVI ’10*, 2010, p. 133.

[9] C. Chambers and C. Scaffidi, “Struggling to Excel: A Field Study of Challenges Faced by Spreadsheet Users,” in *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, 2010, pp. 187–194.

[10] J. J. J. Laconich, F. Casati, and M. Marchese, “Social Spreadsheet,” in *13th International Conference on Web Engineering*, 2013, vol. 7977, pp. 156–170.

[11] W. Kongdenfha, B. Benatallah, J. Vayssière, R. Saint-Paul, and F. Casati, “Rapid development of spreadsheet-based web mashups,” in *Proceedings of the 18th international conference on World wide web - WWW ’09*, 2009, p. 851.

[12] M. Fisher and G. Rothermel, “The EUSES spreadsheet corpus,” in *Proceedings of the first workshop on End-user software engineering - WEUSE I*, 2005, vol. 30, no. 4, pp. 1–5.

[13] T. Chintakovid, “Cultural Differences and End-User Computing,” in *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC’05)*, 2005, pp. 325–326.

[14] J. Sajaniemi, “A new interface to spreadsheet programming: a truly seamless fusion of spreadsheet and word processing paradigms,” in *Proceedings IEEE 2002 Symposia on Human Centric Computing Languages and Environments*, 2002, pp. 40–42.

[15] A. J. Ko, B. Myers, M. B. Rosson, G. Rothermel, M. Shaw, S. Wiedenbeck, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, and H. Lieberman, “The state of the art in end-user software engineering,” *ACM Comput. Surv.*, vol. 43, no. 3, pp. 1–44, Apr. 2011.

[16] C. Scaffidi, M. Shaw, and B. Myers, “Estimating the Numbers of End Users and End User Programmers,” in *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC’05)*, 2005, pp. 207–214.

[17] J. Rieman, S. Davies, and J. Roberts, “A visit to a very small database,” in *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI ’92*, 1992, pp. 471–478.

[18] K. J. Rothermel, C. R. Cook, M. M. Burnett, J. Schonfeld, T. R. G. Green, and G. Rothermel, “WYSIWYT testing in the spreadsheet paradigm,” in *Proceedings of the 22nd international conference on Software engineering - ICSE ’00*, 2000, pp. 230–239.

[19] M. B. Rosson, “The end of users,” in *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications - OOPSLA ’05*, 2005, pp. 3–3.

[20] J. Sajaniemi, M. Tukiainen, and J. Väisänen, “Goals and plans in spreadsheet calculation,” 1999.

[21] R. R. Panko, “What We Know About Spreadsheet Errors,” *J. Organ. End User Comput.*, vol. 10, no. 2, pp. 15–21, 1998.

[22] S. P. Jones, A. Blackwell, and M. Burnett, “A user-centred approach to functions in excel,” *ACM SIGPLAN Not.*, vol. 38, no. 9, pp. 165–176, Sep. 2003.

[23] S. Gulwani, W. R. Harris, and R. Singh, “Spreadsheet data manipulation using examples,” *Commun. ACM*, vol. 55, no. 8, p. 97, Aug. 2012.

[24] W. R. Harris and S. Gulwani, “Spreadsheet table transformations from examples,” *ACM SIGPLAN Not.*, vol. 46, no. 6, p. 317, Jun. 2011.

[25] M. Dinmore, “Design and evaluation of a literate spreadsheet,” in *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2012, pp. 15–18.

[26] R. R. Panko, “The Cognitive Science of Spreadsheet Errors: Why Thinking is Bad,” in *2013 46th Hawaii International Conference on System Sciences*, 2013, pp. 4013–4022.

[27] R. R. Panko and R. H. Sprague, “Hitting the wall: errors in developing and code inspecting a `simple’ spreadsheet model1An earlier version of this paper was presented at the Hawaii International Conference on System Sciences, January 1996.1,” *Decis. Support Syst.*, vol. 22, no. 4, pp. 337–353, 1998.

[28] J. Carver, M. Fisher, and G. Rothermel, “An empirical evaluation of a testing and debugging methodology for Excel,” in *Proceedings of the 2006 ACM/IEEE*

*international symposium on International symposium on empirical software engineering - ISESE ’06*, 2006, p. 278.

[29] M. Burnett, C. Cook, and G. Rothermel, “End-user software engineering,” *Commun. ACM*, vol. 47, no. 9, p. 53, Sep. 2004.

[30] A. Wilson, M. Burnett, L. Beckwith, O. Granatir, L. Casburn, C. Cook, M. Durham, and G. Rothermel, “Harnessing curiosity to increase correctness in end-user programming,” in *Proceedings of the conference on Human factors in computing systems - CHI ’03*, 2003, p. 305.

[31] L. Beckwith, S. Sorte, M. Burnett, S. Wiedenbeck, T. Chintakovid, and C. Cook, “Designing Features for Both Genders in End-User Programming Environments,” in *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC’05)*, pp. 153–160.

[32] M. Erwig, R. Abraham, I. Cooperstein, and S. Kollmansberger, “Automatic generation and maintenance of correct spreadsheets,” in *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pp. 136–145.

[33] D. Flood, R. Harrison, and C. Iacob, *Lessons Learned from Evaluating the Usability of Mobile Spreadsheet Applications*, vol. 7623. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 315–322.

[34] C. Abras, D. Maloney-Krichmar, and J. Preece, “User-Centered Design,” *Bainbridge, W. Encycl. Human-Computer Interact.*, vol. 37, no. 4, pp. 445–456, 2004.

[35] E. B.-N. Sanders, “From User-Centered to Participatory Design Approaches,” in *Design and the Social Sciences*, J. Frascara, Ed. Taylor & Francis, 2002.

[36] C. Helfferich, *Die Qualität qualitativer Daten: Manual für die Durchführung qualitativer Interviews*, 2nd ed. VS Verlag für Sozialwissenschaften, 2005.

[37] H. S. Becker and B. Geer, “Participant Observation and Interviewing: A Comparison,” *Hum. Organ.*, vol. 16, no. 3, pp. 28–32, 1957.

[38] K. M. DeWalt, B. R. DeWalt, and C. B. Wayland, “Participant Observation,” in *Handbook of methods in cultural anthropology*, H. R. Bernard, Ed. Walnut Creek, CA: AltaMira Press, 1998, pp. 259–299.

[39] J. P. Spradley, *Participant Observation*, 1st ed. Holt, Rinehart and Winston, 1980.

[40] M. S. Schwartz and C. G. Schwartz, “Problems in Participant Observation,” *Am. J. Sociol.*, vol. 60, no. 4, pp. 343–353, 1955.

[41] P. Atkinson and M. Hammersley, “Ethnography and Participant Observation,” *Handb. Qual. Res.*, vol. 23, no. 1, pp. 248–261, 1994.

[42] L. K. Hong and R. W. Duff, “Modulated Participant-Observation: Managing the Dilemma of Distance in Field Research,” *Field methods*, vol. 14, no. 2, pp. 190–196, 2002.

[43] K. M. DeWalt and B. R. DeWalt, *Participant Observation: A Guide for Fieldworkers*, 2nd ed. AltaMira Press, 2010.

[44] J. Pruitt and T. Adlin, *The Persona Lifecycle: Keeping People in Mind Throughout Product Design*, 1st ed. Morgan Kaufmann Publishers Inc., 2006.

[45] J. Pruitt and J. Grudin, “Personas: Practice and Theory,” in *Proceedings of the 2003 conference on Designing for user experiences - DUX ’03*, 2003, pp. 1–15.

[46] S. Bødker, “Scenarios in user-centred design—setting the stage for reflection and action,” *Interact. Comput.*, vol. 13, no. 1, pp. 61–75, Sep. 2000.

[47] K. van der Heijden, *Scenarios: The Art of Strategic Conversation*, 2nd ed. John Wiley & Sons, 2005.

[48] J. M. Carroll, *Making Use: Scenario-Based Design of Human-Computer Interactions*, 1st ed. MIT Press, 2000.

[49] E. Brandt, “Designing exploratory design games: a framework for participation in Participatory Design?,” in *Proceedings of the ninth conference on Participatory design Expanding boundaries in design - PDC ’06*, 2006, vol. 1, pp. 57–66.

[50] Y. Rogers, H. Sharp, and J. Preece, *Interaction Design: Beyond Human - Computer Interaction*, 3rd ed. John Wiley & Sons, 2011.

[51] B. Buxton, *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann Publishers Inc., 2010.

[52] D. S. Jackson, “Palm-To-Palm Combat,” *TIME Magazine*, 1998.

[53] S. Dow, B. MacIntyre, J. Lee, C. Oezbek, J. D. Bolter, and M. Gandy, “Wizard of Oz Support throughout an Iterative Design Process,” *IEEE Pervasive Comput.*, vol. 4, no. 4, pp. 18–26, Oct. 2005.

[54] H. Hutchinson, H. Hansen, N. Roussel, B. Eiderbäck, W. Mackay, B. Westerlund, B. B. Bederson, A. Druin, C. Plaisant, M. Beaudouin-Lafon, S. Conversy, and H. Evans, “Technology Probes: Inspiring Design for and with Families,” in *Proceedings of the conference on Human factors in computing systems - CHI ’03*, 2003.

[55] M. Baldauf, P. Fröhlich, and S. Hutter, “KIBITZER: a wearable system for eye-gaze-based mobile urban exploration,” in *Proceedings of the 1st Augmented Human International Conference - AH ’10*, 2010.

[56] M. Häder, *Empirische Sozialforschung - Eine Einführung*, 2nd ed. VS Verlag für Sozialwissenschaften, 2010.

[57] M. Gyssens, L. V. S. Lakshmanan, and I. N. Subramanian, “Tables as a paradigm for querying and restructuring (extended abstract),” in *Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems - PODS ’96*, 1996, pp. 93–103.

[58] M. W. Van Someren, Y. F. Barnard, and J. A. C. Sandberg, *The Think Aloud Method - A practical guide to modelling cognitive processes*, 2nd ed. London: Academic Press, 1994.

[59] G. Colborne, *Simple and Usable Web, Mobile, and Interaction Design*. New Riders, 2010.

[60] P. S. Brown and J. D. Gould, “An experimental study of people creating spreadsheets,” *ACM Trans. Inf. Syst.*, vol. 5, no. 3, pp. 258–272, Jul. 1987.

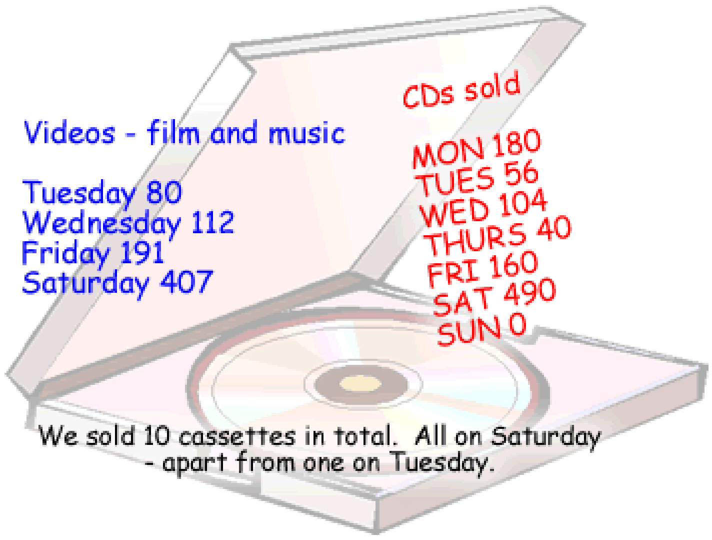
[61] A. Kohlhase, “Human-Spreadsheet Interaction,” *INTERACT 2013*, vol. 8120, pp. 571–578, 2013.

[62] D. A. Norman, *The Design of Everyday Things*. Basic Books, 2002.

# Appendix A – Assignment for Participant Observations

First of all and most important: This is not a test and you will not be rated in any way! This task helps me observe how you interact with spreadsheet software. You are going to create a “computer model” for a record store's accounts. Please read the instructions carefully, but feel free to ask whenever you’re stuck or need assistance.

1. Open a new file with your favorite spreadsheet software.
2. A record store has the following financial information for one week. Design and create your own spreadsheet to show this.
3. For this week, the record store bought 1000 CDs, 1000 videos and 100 cassettes. Add this information, together with the “purchase cost” information, to your spreadsheet. You should show “purchase cost per unit” and “total purchase cost” for each item. Each video costs the shop 3 €, but is sold for 12 €. Each CD is sold for 12 €, but costs the shop 5 €. Each cassette is sold for 8 €, at a profit of 4 €.
4. Develop your spreadsheet further. Add in additional columns to calculate “Total sales”, “Daily profits” and “Weekly profit”. Remember to use formulas wherever you can as this will allow you to add in different numbers and automatically calculate results.
5. Experiment with the “Chart wizard” button. Highlight the CD sales column and press the button. You can work through and create charts to show your data. Try to create one that shows the sales of CDs, videos and cassettes all together.
6. Go back to your main spreadsheet for the final work. Format your results as desired. Additionally, set up conditional formatting for the cell that shows the “Weekly profit” to show a loss in bold red and a profit in bold black.



(http://www.reviseict.co.uk/lessons/images/excel_records.gif)