A Usage Study of Desktop Users

Technical Realisation and Coding Software

Günter Modes

A Usage Study of Desktop Users

Technical Realisation and Coding Software

Master's Thesis

at

Graz University of Technology

submitted by

Günter Modes

Institute for Engineering- and Business Informatics,
Graz University of Technology
8010 Graz, Austria, Kopernikusgasse 24/III

22th September 2005

Advisor:	Univ.-Prof. Dipl.-Ing. Dr.techn. Siegfried Vössner
Co-Advisor:	Dipl.-Ing. Dr.techn. Stefan Grünwald

TUG Graz University of Technology
Erzherzog-Johann-University

Eine Usage Studie von Desktop Benutzern

Technische Realisierung und Kodierungssoftware

Diplomarbeit

an der

Technischen Universität Graz

vorgelegt von

Günter Modes

Institut für Maschinenbau- und Betriebsinformatik,
Technische Universität Graz
8010 Graz, Austria, Kopernikusgasse 24/III

22. September 2005

Diese Arbeit ist in englischer Sprache verfasst.

Begutachter:	Univ.-Prof. Dipl.-Ing. Dr.techn. Siegfried Vössner
Betreuer:	Dipl.-Ing. Dr.techn. Stefan Grünwald

TUG Technische Universität Graz
Erzherzog-Johann-Universität

Abstract

Nowadays many software applications are already designed with usability in mind. Therefore software is tested in designated usability laboratories with selected test users. These users are observed and analysed trying to manage predefined tasks in artificial test environments. Based on these findings the software is improved.

However the knowledge about how users really work in their normal environments with the available software is limited. The goal from the usage study introduced in this thesis is to gather more information about how software is used in business for different tasks.

A framework for being able to observe and analyse the actual usage of software products in real life business environments including the user's task context is designed. A portable setup for recording, coding and analysing is introduced.

This thesis covers mainly the technical issues of this study. The technique for recording the screen contents and environment of the user is presented and the Anvil software application for coding these recorded video files is described. As a part of this work the FUSSY software is created for being able to quickly generate results.

Although this thesis conducts a feasibility study with administrative assistants in an educational university environment, the study design always keeps the possibility of an easy extension for other business areas in mind.

Kurzfassung

Mittlerweile werden bereits viele Softwareprogramme mit dem Ziel einer einfachen Bedienbarkeit (Usability) entwickelt. Dafür wird Software in speziellen Usability Labors mit selektierten Probebenutzern getestet. Diese Benutzer werden, während sie versuchen vordefinierte Aufgaben in einem künstlichen Umfeld zu erfüllen, beobachtet and analysiert. Die Software wird aufbauend auf diesen Erkenntnissen verbessert.

Das Wissen, wie Benutzer in ihrem normalen Umfeld tatsächlich mit Softwareprodukten arbeiten, ist allerdings eingeschränkt. Das Ziel, der in dieser Diplomarbeit vorgestellten Usage Studie, ist das Sammeln von Daten über die Benutzung von Softwareprodukten für verschiedene Aufgaben im Beruf.

Es wird ein System entworfen um eine Beobachtung und Analyse der tatsächlichen Benutzung von Softwareprodukten in einer realistischen beruflichen Umgebung, unter Berücksichtigung der aktuellen Aufgabe, zu ermöglichen. Ein portables System für die Aufnahme, Kodierung und Analyse wird vorgestellt.

Diese Diplomarbeit befasst sich vor allem mit den technischen Aspekten dieser Studie. Die Technik für die Aufnahme des Bildschirminhaltes und der Umgebung des Benutzers wird präsentiert und die Anvil Softwareanwendung für die Kodierung dieser aufgenommenen Videodateien wird beschrieben. Als Teil dieser Arbeit wird auch die FUSSY Softwareanwendung erstellt um eine schnelle Generierung von Ergebnissen zu ermöglichen.

Obwohl diese Diplomarbeit eine Pilotstudie mit administrativem Personal in einem universitären Umfeld durchführt, wird mit dem Studiendesign auf eine leichte Erweiterbarkeit auf andere Berufssparten Rücksicht genommen.

I hereby certify that the work presented in this thesis is my own and that work performed by others is appropriately cited.

All trademarks are used without the warranty of free availability and possibly are registered trademarks. All trademarks are the properties of their respective owners.

Ich versichere hiermit, diese Arbeit selbständig verfaßt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient zu haben.

Alle Marken werden ohne Gewährleistung der freien Verfügbarkeit benutzt und sind möglicherweise eingetragene Marken. Alle Marken sind Eigentum der jeweiligen Rechteinhaber.

Acknowledgements

I especially thank my advisors Siegfried Vössner, Stefan Grünwald and Keith Andrews for their valuable help including hardware support and feedback during the course of my thesis. I would also like to thank the other team members Sandra Brückler, Martin Grabner and Hannes Kollmann for their important work on this study.

Special mention also goes to all test participants of this study.

Last but not least, without the help and understanding of my girlfriend Petra and my parents this thesis would not have been possible.

Günter Modes
Graz, Austria, August 2005

Credits

- Chapter 4 and Chapter 6 are designed and written together with Sandra Brückler and also included in her thesis [Brückler, 2005].
- A very important part of this work is done using the advanced Anvil software coded and provided by Michael Kipp. This software is used with permission.
- This thesis was written using Keith Andrews' skeleton thesis [Andrews, 2004] hopefully considering all of the valuable tips concerning the English language.

Contents

1 Introduction 1

2 Study Motivation and Design 2

2.1 Motivation . . . 2

2.2 Usage versus Usability . . . 3

2.3 The FUSS-B Goal . . . 3

2.4 Data Collection Techniques . . . 4

2.5 Coding and Analysis . . . 6

3 User Recording and Analysis Technology 7

3.1 Recording Requirements . . . 7

3.2 Hardware-Based User Recording . . . 8

3.2.1 Scan Converter Evaluation . . . 9

3.2.2 VGA Splitter with VGA Capture Card . . . 10

3.2.3 Tapeless Recording . . . 11

3.2.4 Biobserve Spectator . . . 12

3.3 Software-Based User Recording . . . 12

3.3.1 Remote Controlling and Viewing Software . . . 12

3.3.2 Remote Usability Studies . . . 15

3.3.3 Commercial Software-based Recording Systems . . . 16

3.3.3.1 Techsmith Camtasia Studio . . . 16

3.3.3.2 Techsmith Morae . . . 17

3.3.3.3 VisualMark . . . 17

3.3.3.4 Noldus Observer Video-Pro . . . 18

3.3.4 Codecs . . . 19

3.3.4.1 Test Recordings . . . 21

3.3.4.2 DivX and XviD Codecs . . . 24

3.4 Coding and Analysis Software . . . 25

3.4.1 Mongold Interact . . . 27

3.4.2 MacSHAPA . . . 27

3.4.3 Techsmith Morae Manager . . . 28

3.4.4 SignStream . . . 28

3.4.5 Transana 30
3.4.6 Anvil 31
3.5 Additional Used Software 35
3.5.1 Eclipse 35
3.5.2 Revision Control Systems 36
3.5.3 Wiki 37
3.5.4 VirtualDub and VirtualDubMod 38
3.6 Evaluation Conclusion 40
3.6.1 Ideal Recording Solution 40
3.6.2 Chosen Software for Study 40
3.6.3 Chosen Recording Solution 41
3.6.4 Coding 43
3.6.5 Analysis 45

4 The FUSS-B Study 47
4.1 Recording 47
4.1.1 Hardware Equipment 48
4.1.2 Practical Recording Tips 49
4.1.3 Checklist for User PC 50
4.1.4 Checklist for Recorder PC 52
4.1.5 Recorded Files 54
4.1.6 Non-Technical Issues 54
4.1.7 Limitations 55
4.2 Taxonomy 55
4.3 Anvil Coding Software 55
4.4 Analysis 57
4.5 Summary 58

5 FUSSY Analysis Software 59
5.1 Requirements for the Analysis Software 59
5.2 Installation 60
5.2.1 Windows Installation 61
5.2.2 Linux Installation 63
5.3 Manual 67
5.4 FUSSY Technologies and Standards 70
5.4.1 The Java Language 70
5.4.2 XML Standard 74
5.5 Building FUSSY 75
5.5.1 Ant Installation 75
5.5.2 Fussy Source Code Extraction 77
5.5.3 Using Ant 77
5.5.4 Development 80

5.6 Internal Structure 80
5.6.1 Startup 81
5.6.2 Taxonomy 81
5.6.3 Waiting Time Handling 82
5.6.4 XML Structure 83
5.6.5 Chart Creation 85
5.6.6 Chart Viewing and Storage 86
5.6.7 Performance 86
5.7 Summary 86

6 Results of the FUSS-B Study 87
6.1 Goals 87
6.2 Activities 88
6.3 Tools and Applications 91
6.4 Waiting Time Analysis 95
6.4.1 Definitions 96
6.4.2 Application And Waiting Time 97
6.5 Feature Groups 99
6.6 Discussion 103

7 Outlook 104
7.1 Possible Recording Improvements 104
7.1.1 Installation Avoidance 104
7.1.2 Connection Alternatives 105
7.1.3 Video and Audio Streaming 105
7.1.4 Recording Automation 105
7.1.5 Visionary Recording Hardware 105
7.2 Coding Tool Ideas 106
7.3 Proposed FUSSY Improvements 106
7.4 FUSS-B Study Future 107

8 Concluding Remarks 108

A Anvil Coding Specification 109

B FUSSY API Documentation 117

Bibliography 137

List of Figures

2.1 Coding and Analysis Overview 5

3.1 Hardware-Based Recording Methodology 9
3.2 Software-Based Recording Methodology 13
3.3 Camtasia Recorder Screenshot 16
3.4 The Noldus Observer Screenshot 19
3.5 Software-Based Screen and Video Recording 20
3.6 Test Recording Environment 22
3.7 SignStream 2 Screenshot 29
3.8 Transana Screenshot 30
3.9 Anvil Screenshot 32
3.10 Eclipse Screenshot 36
3.11 Twiki Screenshot 37
3.12 Twiki Editor Screenshot 38
3.13 VirtualDub Audio Codec Dialog 39
3.14 VirtualDub Video Compression Dialog 39
3.15 VirtualDub Progress Dialog 40
3.16 Transana Coding Setup 44
3.17 Taxonomy in Transana 45

4.1 FUSS-B Coding and Analysis Overview 48
4.2 FUSS-B Recording Methodology 49
4.3 Recording Hardware Equipment 50
4.4 FUSS-B Recording Session 51
4.5 FUSS-B Software 52
4.6 Anvil Coding Setup 56

5.1 System Properties Dialog 62
5.2 Environment Variables Dialog 63
5.3 FUSSY Charts on the Screen 69
5.4 Java Logo 71
5.5 Productive Waiting 83

6.1 Computer Goals 88
6.2 Computer Goals Including Non-Computer 89
6.3 Computer Activities 90
6.4 Activities Including Non-Computer Work 91
6.5 Tools 92
6.6 Applications 93
6.7 Email Applications 94
6.8 Online Applications 94
6.9 Application Waiting 95
6.10 Waiting Time Definitions 96
6.11 Waiting with Productive Waiting 98
6.12 Applications With Included Waiting and Productive Waiting Time. 99
6.13 Application Relative Waiting 100
6.14 Feature Groups Including Non-Computer And Away 101
6.15 Computer Related Feature Groups 102

List of Tables

3.1 Screen Recordings without Embedded Video 23
3.2 Recording with Embedded Video and Different Codecs 23
3.3 Recording Including XviD Codec 24

4.1 FUSS-B Recorded Files 54

5.1 PDF/PNG File Size Comparison 68

6.1 Countable Feature Group Occurrences 103

Chapter 1

Introduction

This thesis covers the design and implementation of the *Framework for Usage Studies of Software in Business Environments* (FUSS-B) usage study. While the technical aspects of this study are primarily discussed in this work, other aspects like designing the analysis scheme (taxonomy) are described in a second parallel thesis by Brückler [2005].

The next chapter (Chapter 2) introduces the motivation and goals for this study. The relations between usage and usability are discussed. Different data collection techniques and an overview about the coding and analysis process is shown.

The technical aspects of this study are covered in Chapter 3. The evaluation of the different hardware and software products in respect to the study requirements are discussed. The emphasis is placed on recording and analysis issues. The chosen coding software Anvil is introduced.

Chapter 4 looks at the practical implementation of the recording sessions with the users. This is a shared chapter with Brückler [2005]. The methodology of the recording, coding and analysis process is described in detail. Practical challenges and their chosen solutions are given.

Issues concerning the implementation of the analysis tool FUSSY for the evaluation of the results from the different coding sessions are described in the Chapter 5. The design of the FUSSY application and the internal object oriented structure are explained. The FUSSY tool is coded in the Java programming language, the input files originate from the Anvil coding software tool.

The generated output diagrams from the FUSSY application are presented in Chapter 6. This is the second shared chapter also contained in Brückler [2005]. The results are shown and analysed. Unexpected and surprising findings are explained and discussed.

Finally, Chapter 7 outlines some concepts and ideas for future work in this research area. Next steps for continuation and extension of this study are proposed.

The Anvil coding specification file and the FUSSY API documentation can be found in the Appendixes A and B respectively.

Chapter 2

Study Motivation and Design

2.1 Motivation

Some years ago most notably employees in technical, administrative or design related occupations had to use the computer for most of their time. Nowadays the usage of software applications is increasing in almost any working environment. More and more activities which have formerly been done by hand are now computer supported. It has become absolutely necessary for staff to know this new tools and to efficiently use them. Most of the time the employees use the applications which are installed on the computer. They do not want or they have no rights to install additional software. It depends on several factors which applications are installed. For normal users it is often not allowed to install new applications because of security concerns. Of course there are exceptions from this scenario. Users which evaluate new software applications, software developers and system administrators have the rights to test and install new software. who defines the applications for installing on someones computer? What reasons and opinions will be taken into account if an installation of a new software is decided?

A research for evaluating the actual requirements and needs of the users is done very seldom. It is the same problem with usability issues for software applications. Some large software companies like Microsoft or Oracle implement more and more usability tests. Many studies imply an immediate return of invest through extensive usability tests [Black, 2002; Marcus, 2002] however only a small amount of smaller software companies are regularly conducting usability tests during the software development process.

A quite common scenario in smaller companies is the installation of the same conventional software applications on almost any computers regardless what tasks the users of these computers have to accomplish. Normally some kind of office suite is installed. The installed operating system is in most of the cases a Windows operating system. If there is some special definitely necessary software application, this application will be also installed. After this initial setup in many cases nothing is changed until the computer has a serious software or hardware problem. Sometimes many applications are bought and installed but never really used. In other cases big software suites are installed, which support many not needed features and are maybe difficult to use. Even if some appropriate software application for a useful and important task is already existing on the market this software will be unlikely ever found and used. This is a very inflexible way of configuring computer systems. The demands of the users are more or less ignored. The users often work in a sub-optimal environment for achieving their goals. [Grabner et al., 2005b]

2.2 Usage versus Usability

Many studies concentrate on the usability of software. Results from those studies are normally some kind of usability guidelines for avoiding common problems. Most of these studies cover web usability issues. There are also some studies focusing on usage. Unfortunately the terms usage and usability are mixed up.

> "Usability is the measure of the quality of the user experience when interacting with something – whether a Web site, a traditional software application, or any other device the user can operate in some way or another" [Jakob Nielsen]

A more formal definition from the *International Organisation for Standardisation* (ISO) for usability:

> "Usability: The extend to which a product can be used by specific users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use." [ISO 9241:11 1998]

Usability deals with the interaction between the user and a given product. A lot of aspects are covered through usability including psychology and physiology of the user [Wikipedia, 2005a]. Another usability definition covering learnability, efficiency, memorability, few and noncatastrophic errors and subjective satisfaction can be found in Nielsen [1994].

In contrast here is a definition from Diamond Bullet Design [2005] for usage studies:

> "Usability Glossary: usage study as opposed to usability testing, which typically brings people into a lab to examine specific usability questions against often predetermined benchmarks, a usage study examines how a system is actually being used in its actual work setting, which may entail discovering that it is being applied to unexpected problems and in unexpected circumstances and also provides a sense of how the task context affects usability issues."

Usage in the context of software application usage is interested in which applications are used and how long, how often and under circumstances in what context they are used. FUSS-B in its actual form primarily deals with usage, however the framework offers the possibility to extend the field of research to usability aspects too. A deeper insight into the differences and connections between usage and usability in relation to the FUSS-B study can be found in Brückler [2005].

2.3 The FUSS-B Goal

An important goal of the FUSS-B study is to reveal how users actually work with the computer in different work task contexts. It should be analysed what goals users have to achieve in their daily work. Therefore different kind of data has to be retrieved. Statistical data for the used software applications should be available such as the used features of these applications. Furthermore the time of waiting for applications is collected too. Even the occurrence of telephone calls and other non-computer tasks should be considered.

The users are observed in their actual working environment. A desirable goal is to have the least possible impact on the PC of the user (User PC). Installation of software and performance problems should be avoided on the User PC. Besides the observation should be as unintrusive as possible.

With the study requirements and the above definitions the FUSS-B study is mainly a usage study. This thesis covers particularly the technical aspects of the FUSS-B usage study. A classification for events and user actions occurring during the observation has to be defined. This classification is labelled taxonomy from now on. A definition of a taxonomy:

> "Taxonomy (from Greek taxis meaning arrangement or division and nomos meaning law) is the science of classification according to a pre-determined system, with the resulting catalog used to provide a conceptual framework for discussion, analysis, or information retrieval." [whatis.techtarget.com]

The task is to find out all possible actions and events in the work of administrative assistants to define a valid and comprehensive taxonomy. Prior to the study interviews with the participating users and observations were done. So an initial taxonomy was specified. This taxonomy is adapted and extended during the course of the study. So the definition of the taxonomy becomes an iterative process. The design of the coding taxonomy for this study can be found in Brückler [2005].

Particularly in academic environments, in small and medium business environments and in public administration it may be an advantage to migrate some or all operating systems and software applications from closed proprietary commercial systems to Open Source software. One of the first main institutions which executed a migration on a larger scale is the municipal authority of Munich [München, 2005], which completely operates on Linux, both on servers and on desktop computers. In Vienna a soft migration process is just now taking place [Wien (MA 53), 2005]. In many other cities and countries such as Brasil and China migrations are taking place.

Such a migration may have advantages concerning licensing costs, interoperability and security issues. Also a locked-in effect can be avoided [Grünwald, 2003]. Although the initial idea for this study was to check out the possibility of an Open Source migration the proposed goal is now to better understand and help the user. If there are only a few features used from some applications it will also be an advantage for the user to install another smaller and simpler commercial application for this task. Another example is one application handling a task where several applications where used in conjunction before. This single application will replace all of these other applications if these are not used in other contexts. The main goal is to assist the users in archiving their goals in a simple and efficient way.

It is explicitly not a goal of this study to observe the efficiency or productivity of the administrative assistants. The goal is to gather information about how software is actually used. Non-computer tasks are evaluated too, but it is not the intention of the FUSS-B study to rate or benchmark the work of individuals. No analysis in the direction of productivity is done or necessary. The interents of this study are concentrated only on the usage of computer software. Complying with these interests a consent form [Brückler, 2005, Appendix B] has been designed and is signed by all participants and analysts of the study. Because of data protection issues, all involved persons must explicitly grant permission to use the data and the analysts agree by contract that the data is not misused. Results are always published anonymised.

2.4 Data Collection Techniques

Usage studies concerning observation of computer users are often based on automatically collected data. These logfile are evaluated statistically. There are several possibilities to acquire this data. It is possible to use clientside logfiles like for example history files from a browser [Cockburn and McKenzie, 2000; Sellen et al., 2002]. A special event logging software can be installed on the computer of the user for gathering raw data of low level user events like the active application, mouse movements,

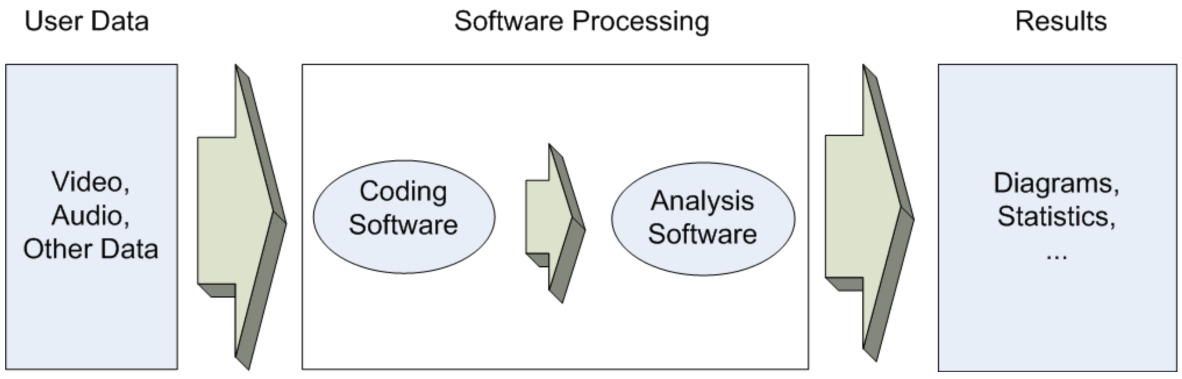


Figure 2.1: Overview of the Coding and Analysis Process of Usage Studies.

key typing and so on. Of course the used software applications can also be adapted to log in a desired manner [Catledge and Pitkow, 1995]. Another possibility, if applicable, is to analyse serverside log files. For example web usage studies can use proxy log files. All of these techniques have their own strengths and weaknesses [Grabner et al., 2005a; Cockburn and McKenzie, 2000, Section 6.1]. The main advantage of an automated evaluation of logfiles is the smaller time consumption for gaining and analysing the data in contrast to a manual evaluation method. As a consequence, a larger sample of users can be evaluated with a small amount of resources.

The FUSS-B study is interested in general usage of programs and features, not in details what event is exactly executed when. So low level events like mouse clicks are not needed. Therefore the installation of the event logging software is not used as data retrieving technique here. Neither are serverside logfiles or adapted software possible solutions in this case, because data of every software (and even non-computer) task should be collected. Obviously it is not possible to change all of the used proprietary software, simply because the source code is not available. All of these mentioned data gathering solutions have the weakness not having available the task context. It is impossible to detect the goal the user wants to achieve.

A different approach is chosen. A manual evaluation of video material is chosen. The screen of the User PC, a audio signal and a video from the environment of the user is recorded by means of a camera in the first step. Because the environment and audio is recorded it is guaranteed do have all necessary data for the desired analysis available.

A similar technique was used in a web usage study. The user was tapped including audio with a video camera and analysed afterwards. No log files were used. The advantage having the task context every time available is emphasised there. [Byrne et al., 1999; Byrne, 2001]

In contrast to automated methods, the results of manually evaluated studies contain more qualitative information. For example data like intentions of the users and events in the user's environment can be dealt with. For the FUSS-B study telephone calls and other non-computer related tasks are also easily observable with this setup. Goals can be guessed by means of previous or later user actions. If it is not possible to guess the goal sometimes, the user will be questioned while showing the problematical part of the videotaping.

2.5 Coding and Analysis

Most of the usage studies in literature have a similar coding and analysis process (Figure 2.1). Usage and usability studies often have some observational data in form of video, audio and other formats. After retrieving the raw material from the users, the material is postprocessed. First it is coded with a given taxonomy.

For the analysis of time based observational data the term *Exploratory sequential data analysis* (ESDA) was introduced lately [Fisher and Sanderson, 1993, 1996]. The term was created according to the term *Exploratory data analysis* (EDA) introduced by Tukey [1977]. There are many commercial and non-commercial applications available for ESDA. All evaluated applications for the coding and analysis process of this study are shown in detail in Chapter 3.

After using the defined taxonomy for the coding process the results have to be created. In the case of the FUSS-B study the results will be diagrams. For example Morae by TechSmith Corporation [2005b] has also features for statistical analysis and can be directly used for creating charts. If the coding software supports no statistical features the analysis will be done with another separate software application. In this case there are either intermediate files generated by the coding software available such as with Anvil by Kipp [2004a]) or the intermediate coding data is stored in a database like with Transana by Woods and Fassnacht [2005]. Anyway the intermediate data is gathered, statistically processed and charts are generated by the analysis software. This step could also be done manually by importing the data into a spreadsheet application for chart creation.

The FUSS-B taxonomy is a hierarchical classification. Actions and events have defined start and end points in the timeline. These time segments are the basic part of the coding. These segments are called feature groups. Feature groups describe a set of functionalities of certain software tools. *Browse Content* or *Text creation and formatting* are feature group examples. Feature groups belong to different tools such as wordprocessing, spreadsheet or emails tools.

Each tool has one ore more specific implementation instances, the applications. Examples are the MS Outlook, The Bat or Mozilla Thunderbird applications. The next hierarchy in the taxonomy are activities which are a sequence of used applications. Activities are contained in the highest level of the FUSS-B taxonomy, the goal level. Such goals are Teaching Support, Accounting or Staff Support.

An important feature group is *Non-Computer Work*, because this feature group allows to see the fraction of computer work of the overall working time. The taxonomy will also consider waiting times in front of the computer if an application or the network connection is slow. This is an example where an typical usability issue is covered. More information about the used taxonomy for this study can be found in Brückler [2005] and Grabner et al. [2005b].

Chapter 3

User Recording and Analysis Technology

The screen and the environment of the user has to be recorded. The gathered data has to be analysed afterwards. Several persons are working together on the FUSS-B study. So hardware and software applications for recording the data, collaboration in a team and finally analysing the gathered data has to be chosen. Several evaluations are done for these purposes.

The first step is the recording of the user's activities. A view of the computer screen in front of the user has to be recorded. So the open and active applications and the mouse pointer movements are available for later coding and analysis. The term screencasting for the process of recording movies of a computer desktop was introduced by Udell [2005] in an article about the benefits of this technique.

> "A screencast is a recording of computer screen output, usually containing audio narration typically published as a video file. ... Screencasts are typically created to produce software and web application demonstrations. This was mostly done within corporations as a way to facilitate employee training. However, further interest has been sparked through the increased blogging trend, which has eased content publishing."[Wikipedia, 2005d]

The term (web-)casting is often used synonymously with the term streaming meaning continuous, often live, delivery of multimedia data over the internet. Therefore the name screencast is a bit misleading because screencasting has nothing to do with live streaming of content, only a video file is created for later usage [DonationCoder.com, 2005]. Screencasts can be used for several purposes. It can be used as training tool for computer users. Another domain is the creation of demonstration videos for educational or commercial purposes. Features and complicated operations can often be shown more easily in a video than describing them in detail with written descriptions. With screencasts bugreports can be generated for programmers for exactly showing how the bug can be reproduced. And last but not least screencasts can be used as tool for user interface designers and usability testing.

3.1 Recording Requirements

Besides the screencasting of the user screen an additional recording of the user's video is also necessary for being able to see if the person is working on the computer, is doing non-computer related work or if the user is away from the workplace. Also the reactions on possible disturbing events, such as a telephone call or problems with the computer, is important to detect for the FUSS-B study. Facial expressions from the user may also be important for deeper analysis. To make it easier to determine the goal and intention of the user an audio recording is needed too. There are many technical realisation possibilities to fullfill these requirements. All of these are either hardware based, software based or both and all of them have their advantages and disadvantages. The initial requirements, although not all of them are met in the final design, of such a screencast solution include amongst others:

- Small fonts on the recorded user screen should be readable in an easy way. Therefore a resolution from the User *Personal Computer* (PC) screen of at least 1280x1024 should be recorded in sufficient quality.
- Mouse movements should be defined precisely. A framerate of at least seven frames per second is enough for most cases. This is tested by empirical experiments.
- A second parallel video stream showing the user in his work environment should be recorded too.
- Long time recordings should be possible. A normal working day with eight hours ought to be taped without interruption.
- For later analysis these two recorded streams should be already synchronised during recording or at least afterwards in some way.
- It should be possible to start and stop the recording of the video and nevertheless always have both of the streams synchronised.
- Non-intrusive observation for having the least possible impact on the workplace environment of the user.
- No installation of software on the PC of the user.
- A connection to the *local area network* (LAN) should be avoided.
- The necessary recording hardware should be as small as possible. These include microphones, cameras, computers and other necessary hardware devices.

Changes in the user's environment and on the User PC should be avoided as far as possible. The least impact on the User PC would be achieved without installation of any software. The videocamera and the microphone for observation of the surrounding area should be small and unobtrusive. The other necessary recording equipment should not interfere with the normal workflow and environment of the user.

3.2 Hardware-Based User Recording

The methodology of a hardware based recording is to split the monitor signal and to record the screen of the User PC simultaneously. The recording could be done by means of a second computer via hard disk recording or with some other kind of hardware capable of recording such signals (see Figure 3.1). For all hardware based solutions at least the following hardware parts are required:

- A hardware for splitting the monitor output signal from the User PC to be observed for parallel recording.

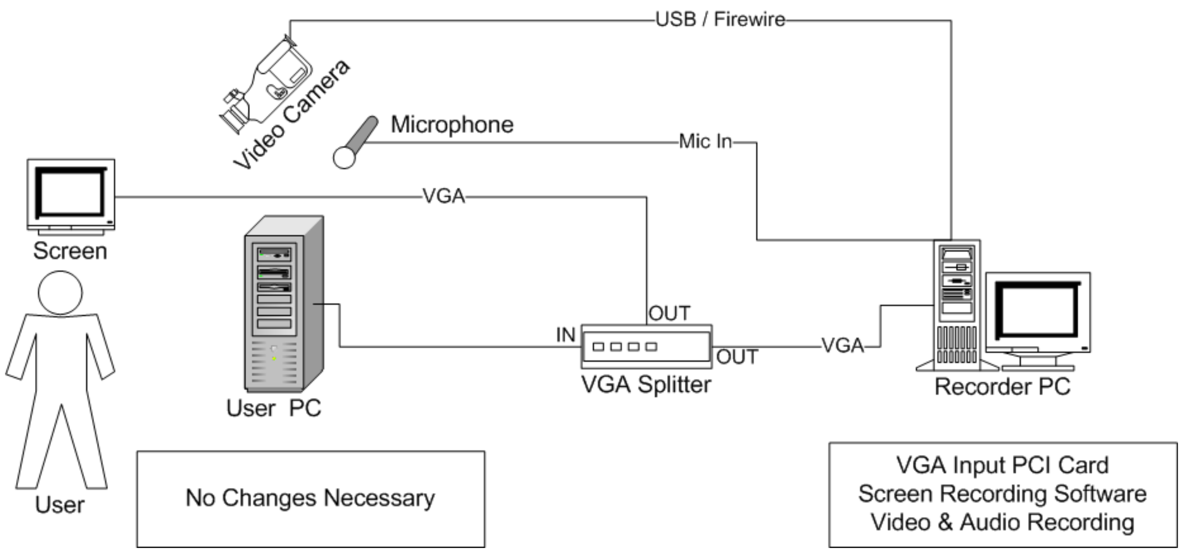


Figure 3.1: Recording with a Hardware-Based Setup. A Signal Splitter is used. No Software Installation on the User PC is necessary.

- A camera for capturing the user and the environment.
- One or two recorder capturing both signals.

There is different kind of hardware available for splitting or converting computer video signals. Such a hardware is called a splitter. Splitter output signals are identical to the input signal. If the splitter input signal originates from a computer monitor output signal then this signal is a *Video Graphics Array* (VGA)/*Red Green Blue* (RGB) signal. A splitter is used if the output signals should be again VGA/RGB signals with the same framerate. Unlike a splitter, a scan converter is used for converting from one format into another format.

> "A scan converter accepts data (e.g., video) from one format and converts it to another format. In videoconferencing technology, a scan converter typically converts a digital signal from a computer into an analog signal that can be displayed on a regular television monitor. Its particular role is to convert computer video scan rates (how many times per second the image is refreshed, or repainted, on the screen) to scan rates for TV video."
> [The Utah Education Network, 2005]

3.2.1 Scan Converter Evaluation

A first evaluation is the test of a configuration with a scan converter and a *Super Video Home System* (S-VHS) recorder. For further analysis and archiving the video is copied to a *Digital Versatile Disc* (DVD) recorder. The recording is not done directly on the DVD recorder because of not being able to acquire a portable DVD recorder. The following hardware is used:

- Extron high resolution scan converter: Emotia Xtreme
- S-VHS JVC video recorder HR-S6600EU

- Panasonic DVD-RAM/DVD-R recorder DMR-E20

A detailed description of the Emotia Xtreme by Extron Electronics [2005]:

> "The Emotia Xtreme is a 1600 x 1280 up to 92kHz resolution workstation (PC, Mac, SUN, SGI) to composite video (NTSC/PAL), S-video, component video and RGBS scan converter. ... With digital signal processing the Emotia Xtreme also offers a full spectrum of 16 million colours without the need for any software drivers."

The quality of the resulting recording with this setup is quite unacceptable although state of the art hardware is used. The resolution is very bad, the fonts are not readable for resolutions above 800 x 600. Maybe a second experiment with direct recording on a DVD recorder (without intermediate S-VHS recording) could achieve slightly better results. However the resolution of television standards is too low for computer generated high resolution VGA signals. All presently used standards like *Phase Alternation Line* (PAL) or *National Television Systems Committee* (NTSC) and signals like FBAS-Composite (FBAS is used in German language area: *Farbe Bild Austast Synchron*) and others are not suitable for high resolution signals. The situation will change if the *High-definition television* (HDTV) technology is successfully established in the near future.

Thus the S-VHS and also the DVD recorder solution idea is cancelled due to the apparent disadvantages following the technical specifications of television compatible video signals. To summarise the main findings of the scan converter evaluation are:

- The resolution of a DVD is at most 720x576. So only screen resolutions up to 800x600 are recorded in adequate quality.
- The possible duration of a DVD recording is two hours with SP (Standard Play) setting, four hours with *Long Play* (LP) and six hours with *Extended Play* (EP).
- Separate recording of the second (camera) video signal has to be done on a second recorder.
- No Synchronisation between these two video streams during the recording process.
- It is difficult to use the recorded video directly from DVD while analysing. So either the DVD player has to be controlled from the computer or the video has to be copied (this process is also called ripping) from the DVD directly on the hard disk. Having the video stored on the computer eases the coding process and enables direct jumps to specific time positions in the video.

3.2.2 VGA Splitter with VGA Capture Card

Another possible setup, quite similar to the preceding scan converter setup, is to have a dedicated computer with a special hardware for recording the user screen from the other User PC. This is still possible without changing the hardware or software on the User PC. The second signal from the video camera is also taped on this computer (see Figure 3.1).

The hardware requirements of such a setup include among others:

- A VGA signal splitter, splitting one VGA signal from the User PC into two identical signals, one for the monitor from the user PC and the other one for recording on the Recorder PC.
- The second PC for capturing and recording the signals.

- One (S-)VGA capture card built into the Recorder PC.
- A camera for capturing the environment.

Getting a proper VGA capture card is the major problem. The capture card should be able to capture a resolution of at least 1280x1024 in good quality. There is no large market for this kind of hardware. It is difficult to receive such a card for evaluation purposes and these cards are quite expensive.

One example of such a piece of hardware is the VisionRGB-Pro *Peripheral Component Interconnect* (PCI) card. This card provides the capability to handle two RGB input channels, 24 bit colour depth, a resolution of 1280x1024 in realtime and a resolution of 1600x1200 pixels with two frames per second. It is also supports scaling the image to smaller resolutions. Drivers and a software application for Windows systems is also included. Here is a feature overview from Datapath Limited [2005]:

> "A standalone PCI plug in card, the VisionRGB-PRO captures the display output (e.g. the analogue RGB data) from one computer screen and then displays it as an independent application, in real time, on your Windows desktop. The VisionRGB-PRO has two VGA compatible inputs for simultaneously feeding two independent data sources directly into your main PC. The data is then converted on the VisionRGB-PRO and displayed as a standard Windows application on your desktop."

Buying the card is not possible due to budget constraints and unfortunately no test hardware could be obtained hence an evaluation of the VisionRGB-Pro hardware is not possible. A recording solution based on this card would most likely be superior to the other proposed and finally used solutions.

The VisionRGB-PRO card is a PCI card thus a laptop can not be used. Laptops does not have PCI bus slots available. This is a disadvantage of this solution. Of course a mini barebone system could be used for more portability of the system.

3.2.3 Tapeless Recording

> "Non-linear editing for film and television postproduction is a popular modern editing method. Non-linear editing means being able to put any frame next to any other frame. This method was inherent in the cut and glue world of film editing from the beginning. Unfortunately, in the analogue world of film, to change it means to destroy what existed before. Video editing, when it first became possible, was a rerecording method, which eliminated the need to destroy previous versions, but was by definition linear. Non linear, non - destructive methods began to appear with the first digital images." [Wikipedia, 2005b])

The proposed hardware setup in Section 3.2.2 is a kind of a tapeless recording solution. Tapeless recording technology is a synonym for non-linear editing.

Some years ago hard disk recording systems had special and very expensive hard- and software. Nowadays normal Personal Computers provide hard disk capacities with several 100 Gigabytes and the *Central Processing Units* (CPUs) are fast enough to encode and store video and audio in realtime in many different formats.

There are suppliers of hardware-based recording solutions. Most of them also include software for controlling the hardware. Some suppliers have a all-in-one solution, where the software also covers the coding and analysis process. Also groupware tools, for a team of observers working together, are sometimes included.

3.2.4 Biobserve Spectator

This commercial example is a solution by Biobserve GmbH [2005]. It consists of a hardware based screen recorder and the Spectator software application designed for Windows XP. The recorder, named SR 5, has the size of a VCR and contains a hardware for VGA signal splitting and capturing the monitor signal and a internal hard disk for recording the data in *Moving Pictures Experts Group* (MPEG) 4 format. The user video and audio is stored separately. These signals are directly encoded to a MPEG file on the computer. *Digital Video* (DV) or firewire cameras are supported directly. The signal of the monitor is recorded full screen on the hard disk of the Biobserve SR 5 Recorder. Both files are synchronised during the recording. The screen recorder is controlled by the software running on a separate PC.

The video is stored in full resolution (up to 1600x1200 with 10-15 frames per second). The recording is tapeless (see Section 3.2.3) and no software installation is required. Further information is provided at `http://www.usability.biobserve.com/`.

Features include a project database for managing and scheduling different sessions and the definition of a taxonomy (tasks, behaviours and modifiers) for coding. The taxonomy is not complete freely configurable, it consists of tasks and behaviours. The coding can take place already during the session and also afterwards with definable keyboard shortcuts. Statistical tools and a report generator for summarisation and analysis of results in *Portable Document Format* (PDF) files are supported. Generating highlight videos with an included video editing tool is possible. The free Spectator Player provides the capability to play this videos.

3.3 Software-Based User Recording

The recording part of this study may also be implemented with no special kind of recording hardware only with a combination of software applications with over the counter computer hardware. Since this is not possible without an intervention in the User PC. At least some software application has to be installed or at least started on the User PC. So the goal in this proposed software-only setup is to have the least possible impact concerning usage of resources such as harddisc space, memory and CPU time.

A remote viewing software is installed on the User PC and the recording of all the necessary video streams (the screen and the user video) is done on another PC, the Recorder PC. So most of the used software is located in the Recorder PC. The videos are stored on this Recorder PC and also memory and CPU usage primarily takes places there.

The only hardware used in this proposed setup is the already available and observed PC from the user, an additional Recorder PC, a hub or switch for the connection of the Recorder PC to the User PC (not necessary if a patched ethernet connection is available), a webcam for recording the user and the user's environment, an additional microphone and of course the necessary cables for connecting the different devices. No expensive or special hardware is needed.

3.3.1 Remote Controlling and Viewing Software

Several available applications could be used for such a remote viewing process. It is for example possible to use remote controlling software. A remote control software provides the capability to control a computer from some remote location via the internet. Remote controlling is extensively used from help desk personal supporting other computer users from remote. Most of these applications

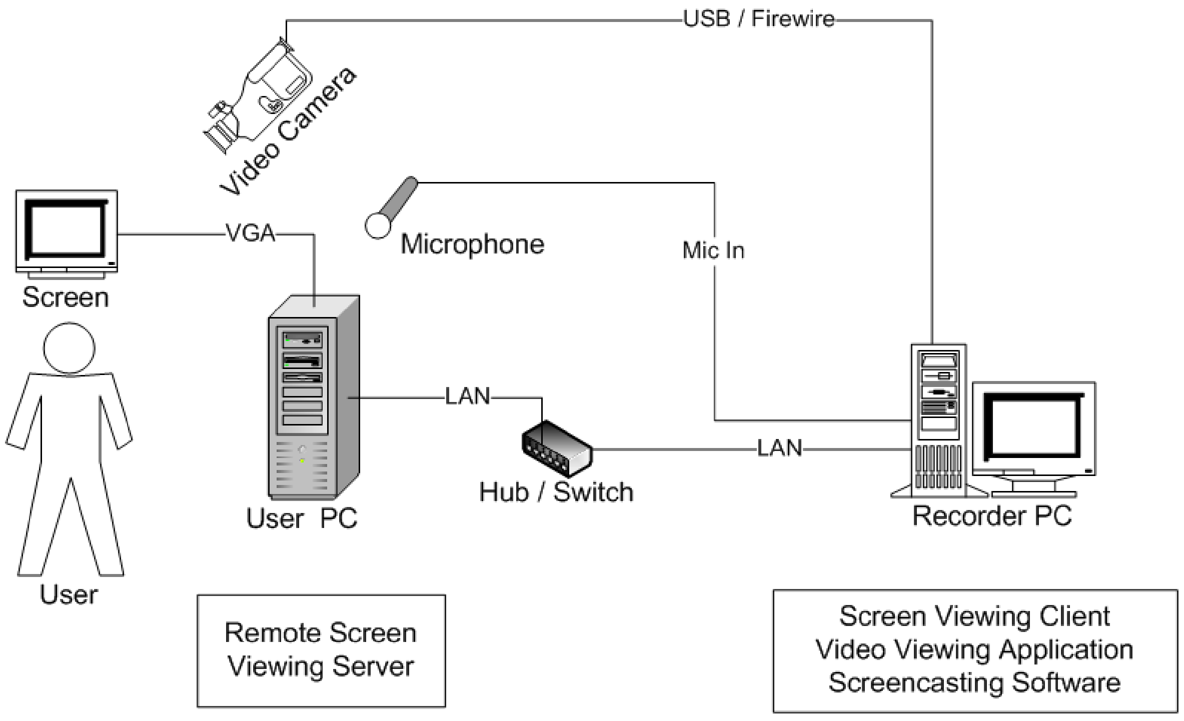


Figure 3.2: Recording with a Software-Based Setup. A Remote Viewing Software has to be installed on the User PC.

support some kind of a view only mode for monitoring a remote screen without interacting. Only this is the software requirement for this study.

All remote controlling applications have to deal with several aspects of security issues. For example it is necessary in most environments to implement some kind of permission control. Only defined persons should be able to gain control over the computer. It is also often necessary to only allow some special defined actions. An other important security aspect in remote controlling sessions over the internet (and also in larger company local area networks) is the possibility of encrypting the transmitted data. And of course there is also an issue with privacy of the user, most of the mentioned applications provide features for observing a computer user. With special prepared and configured software this can be done transparently and unnoticeable from the observed user in the background.

Also technical problems like controlling other computers behind a firewall with *Network Address Translation* (NAT) eventually arise when using remote controlling software in real world environments. Performance problems are common too. Protocols for remote controlling software have to consider limited bandwith or bad response times caused by the network connection. Also the performance impact on the server- and clientside can be troublesome. And last but not least in the modern heterogeneous business environments cross-platform compatibility could be necessary.

Many proprietary commercial applications are available for remote controlling computers like PC Anywhere by Symantec Corporation. [2005], Timbuktu by Netopia, Inc. [2005], Radmin by Famatech, Inc. [2005] and others. Open source remote controlling solutions are also available. Most of them are based on the *Virtual Network Computing* (VNC) protocol. This study uses VNC based software applications for remote viewing purposes.

VNC based Software

The VNC protocol was originally developed at AT&T. The original VNC source code is Open Source under the GNU General Public License. The VNC protocol transmits keyboard and mouse events (clicks and movements) from one computer to another for controlling purposes. The resulting screen changes are sent back to the remote computer over a network.

The original VNC protocol had no encryption for the username, password or transmitted data. If security is important there will be always the possibility to tunnel all of the data over an VPN or SSH connection. VNC derivatives which already include a built-in encryption module (for example UltraVNC and TridiaVNC) are also available.

The VNC protocol is originally designed as a cross-platform application, there are servers and clients for almost any computer platform, including Java, available. It is even possible to connect from several VNC clients to one VNC server simultaneously. There are many variants based on the original VNC codebase. Well-established variants are for example:

Original Source based Cross-Platform

- RealVNC is the official VNC version maintained by the original team from AT&T Laboratories. Other variants are normally based on this source and compatibility is tested with this version. It is now hosted at RealVNC (`http://www.realvnc.com/`).

- TightVNC is known for its more efficient compression algorithms which allow better responsiveness over slow connections like the internet.

- TridiaVNC has also many added features to the original version. TridiaVNC supports encryption, a stateless viewer (it is possible to reconnect from another location and continue th work), a very small footprint, secure file transfer, a built-in telnet client and more.

- MetaVNC is a windows aware VNC which has the possibility of a seemless integration of the local desktop and the remote desktop when running on a windows computer. There exist also MetaVNC versions for Linux and even a Java version is not missing.

Windows only

- UltraVNC is a windows only VNC client and server with many added features such as file transfers, an optional installable high performance video driver, encryption plugins, a text chat communication and more.

- TeamViewer is based on UltraVNC therefore it only runs on windows based computers. There are no administrative rights for the installation of the TeamViewer software required. This is an advantage. An separate, not available as Open Source however available with a cost free license, application named dyngate can be used if the computers are behind firewalls with private IP address ranges to tunnel the data.

- Other windows only VNC software variants are eSVNC, ZVNC, LapLinkVNC, xVNC, Cayote VNC, EchoVNC, ...

Other VNC Variants

- Some common Macintosh variants are: OSXvnc, OS9vnc, ChromiVNC, VNCThing, CoTVNC and VNCDimension.
- DirectVNC is a Linux variant which uses the linux framebuffer device, therefore no X-Window System is necessary.
- xf4vnc is a Linux variant which provides two implementations of VNC in an Xserver.
- SSHVNC is a now part of SSHTerm Pro, a Java based SSH client providing the ability to connect to systems using SSH, SFTP, Telnet and Secure VNC.
- *Java 2 Micro Edition* (J2ME) VNC is a client for embedded devices such as mobile phones.
- VNC viewer are also available for PocketPCs and Palm devices.

3.3.2 Remote Usability Studies

Although motivations and goals for remote usability studies vary in relation to this FUSS-B usage study, some findings from these remote usability studies can nevertheless be useful. Especially the study design, the used software for sharing the User PC screen with the Recorder PC and the used recording technology have similarities. [Ames, 2003; Aufrecht, 2005]

Aufrecht [2005] describes a solution with a observed User PC (Subject PC) and a dedicated Recorder PC (Monitor PC).

> "In this setup, the Subject PC is monitored from another PC, which records the screen and audio to hard drive. The Monitor PC serves these recordings on demaind [sic]. It also provides a live feed during tests. Compared to VCR-based labs, this requires less equipment, provides a higher-quality image, and is much faster to edit. Most importantly, it is much easier to share results."

This proposed solution requires installation from a remote viewing server application on the User PC for screaming the screen contents on the network. On the Recorder PC a remote viewing client application has to be installed. In this special setup a WinVNC server from RealVNC Ltd [2005] is used as remote viewing server. On the other side as remote viewing application a VNC Viewer is installed on the Recorder PC for capturing the screen contents. The Camtasia Recorder application, available as part of the TechSmith Corporation [2005a] Camtasia Studio Suite, is used for recording the screen and audio stream on the hard disk. The Windows Media Encoder tool streams the audio signal live on the ethernet. Further information on Camtasia can be found in Section 3.3.3.1.

With this solution no video from the user is recorded, however a live audio stream is available for every computer on the network via the Recorder PC audio stream. The live screen video is also directly available from the User PC VNC server. The Recorder PC captures the User PC screen with the VNC Software and encodes the screen and audio signal with Camtasia to hard disk. Joel Aufrecht states only a small performance impact from the monitoring software on the User PC.

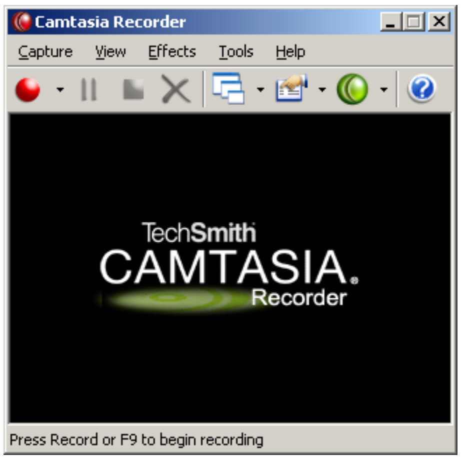

Camtasia Recorder
Capture View Effects Tools Help
TechSmith
CAMTASIA.
Recorder
Press Record or F9 to begin recording

Figure 3.3: The Techsmith Camtasia Recorder Application which is a Part of the Camtasia Studio Suite.

3.3.3 Commercial Software-based Recording Systems

3.3.3.1 Techsmith Camtasia Studio

TechSmith Corporation [2005a] Camtasia Studio is a solution for creating videos of a PC desktop screen activity (screencasting). Its main purpose is the generation of presentations. The recording part of this software supports recording screen contents in real time full motion and full resolution videos. The resulting video is not assembled via a series of screenshots instead one codec is used for the realtime encoding of the resulting video.

It is possible to record simultaneously an audio track from a source such as a microphone. This additional audio track is stored within the video file. Another feature, although not used for this study, is the possibility to provide visible and audible cues for mouse clicks and key strokes. Camtasia has many options for controlling output quality and other details. Flash, AVI, Windows Media Format, Apple Quicktime and RealMedia formats are supported. On DonationCoder.com [2005] a review from Camtasia and comparisons with other screencasting applications is shown.

For the FUSS-B study only the Camtasia Recorder part of the Camtasia Studio Suite is evaluated and used. Until around the third quarter 2004 it was possible to obtain a license only for the Camtasia Recorder. Now it will be necessary to acquire a license for the complete Studio Suite, even if only the Recorder Part is required. Results from undertaken test recordings will be found in Section 3.3.4.1. The performance impact and various different codecs are evaluated.

The whole Studio Suite supports many features like video import, video editing and audio mixing and dubbing tools. Also titles, graphics and interactive parts, especially useful for a presentation, are supported. A multi-track movie editing interface is provided. Audio and video segments can be mixed and edited. The interface is designed very intuitively and easy-to-use. Many different transitions and fade effects are provided. Placing of so named callout objects (text boxes, balloons and other objects can be placed on screen to emphasise a special event or region) for using in presentations is supported. Callout objects and effects such as clipping and zooming are shown on a separate timeline where they can be created and edited.

Many additional extras are also included. A tool for easy integration of Camtasia recordings into

Powerpoint presentations is provided. For web conferences a device driver supporting streaming parts or whole of the screen contents is also included in the Camtasia Suite.

3.3.3.2 Techsmith Morae

Morae has the ability to examine and mark user actions and reactions. It is a software based solution supporting recording and synchronising user and system data. The software is designed for Windows platforms. The main goal is to gather data for usability or user experience analysis purposes. Morae consists of three components: the Morae Recorder is for collecting the test data, the Morae Remote Viewer is for observing and annotation of the test data and, Morae Manager provides analysis capabilities of the captured data and can be used for the assembling of relevant pieces into a presentation. The recorder software has to be installed on the PC to be tested. The test computer should be equipped with audio and video capture capabilities. No special hardware is required for using the Morae application.

It is possible to record video, audio and data tracks of system activity and user interaction actions and events. Screen actions by the tester are recorded such as a video and audio of the tester. The recorder software will automatically synchronise these streams. With The video from the user can originate from a digital videocamera or webcam and the audio track source may be an external microphone. Tools for detecting web page changes (only for Internet Explorer), mouse clicks, keyboard activity and other user initiated events are included. It can be defined what events should be captured. These events are saved in a synchronised data track together with the other recordings.

Morae supports collaborative working of the research team by means of the Morae Remote Viewer tool. The Morae Remote Viewer software can be installed on other networked systems. This tool shows the test participant's computer screen, a picture-in-picture windows of the participants face and also the audio track can be transferred in real-time over a network to the Remote Viewer. Several team members in remote locations are able to watch and annotate the ongoing recording. These annotations are markers or text notes and are included in another synchronised data stream in the recording file. Each Remote Viewer can set markers and text notes hence this enables for example each observer to focus on specific monitoring and annotation to log. Developers or consultants can observe real -world live testing and see how the software is actually used. For a description of analysis and presentation features of Morae also see Section 3.4.3.

Export from the data to spreadsheet or statistic software packages is provided. Morae provides an utility for producing an executable file including a highlight video, the codec and a media player in one single file. The correlation of multiple datastreams, automatic event tagging and monitoring is good which is already shown by independent reviews however the statistical analysis and reporting capabilities could be improved. [Network World, 2005]

3.3.3.3 VisualMark

The VisualMark software by Users First, LLC [2004] is a software based observation laboratory. The analysis software is running on Macintosh computers. The software is VNC based hence Windows, Linux and Macintosh are supported as observed operating system on the User PC. It is not necessary to install any software on the User PC, because the observation software with the VNC server is directly started from CD-ROM on the User PC. Brignull [2004] made an interesting comparison between Morae and VisualMark covering features, license models and more.

3.3.3.4 Noldus Observer Video-Pro

The Noldus Observer Video-Pro is an integrated commercial software system for collection, analysis and presentation of observational video data [Noldus Information Technology bv., 2004]. It is a tool for recording behavioural activities and processes from animals and humans. Time linked events are coded during repeated passes through a video. User can record and annotate observational data with this tool. Quantitative analysis is immediately possible through a built-in analysis module. There are also other variants of the Observer available: The Observer Basic is for data collection with a stand-alone computer and the Observer Mobile is designed for data collection with a handheld computer. The Observer Video-Pro extends the Observer software to an integrated system for collection, management and analysis of time-structured data from live observations. These data can come from analogue or digital video tapes or from digital media video files.

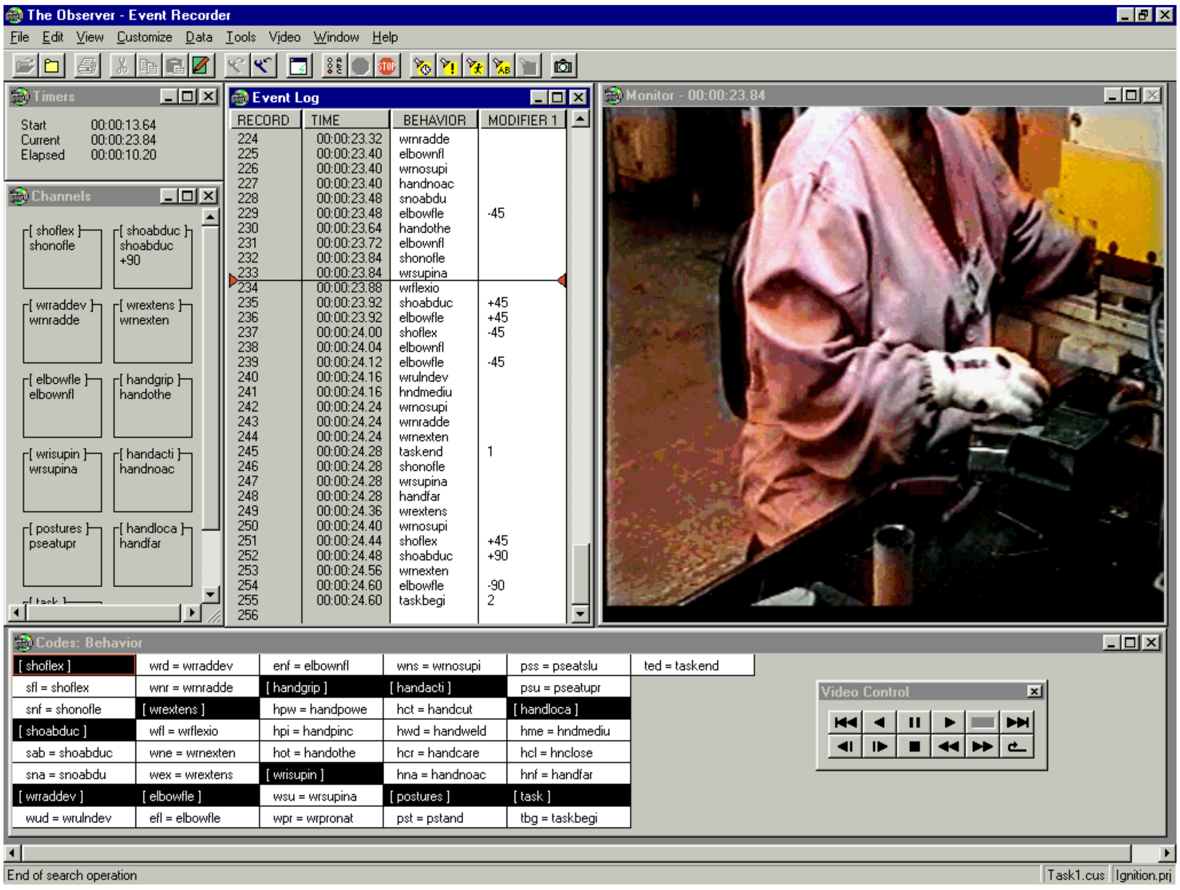
The Observer has a modular structure. The Project Manager is the centre of all modules and controls the creation and management of projects, data import and export, and also the activation of other program modules. These modules include amongst others the Configuration Designer, the Event Recorder and the Analysis Procedures. Other add-on modules like a video highlight module, a report generator or a software screen capture module are available too.

With the Project Manager the properties of an annotation project are defined. An annotation project consists of the configuration data, the video data and the annotation data files. The configuration data includes variables which are used during the event recording and for the later analysis of the annotation data files. These variables include subjects (actors), behavioural elements, modifiers (associated with behavioural elements) and channels (combination of a subject and a class of behavioural elements). These channels are equivalent to layers in other coding software.

In the Event Recorder module the actual observations are performed. A screenshot of the Event Recorder is shown in Figure 3.4. The user assigns a value to the variables already defined with the Project Manager before. Actors, behaviours and modifiers are coded by typing predefined key codes or clicking codes on the screen during the annotation process. During observation it is possible to add notes and comments, which are stored together with the annotation data. The data is saved in observational data files. Multiple channels (layers) of annotation can be used. Simultaneous events can be viewed, edited and analysed aligned to the same timeline. The user can control the playback of the video. The Observer Video-Pro has built-in drivers for a wide range of digital and analogue video recorders. Many videotape hardware devices can be controlled directly from the application. Digital camcorders with firewire (IEEE 1394) can also be connected. Playing with different speeds and jumps to any position with a slider is supported. Depending on the video source or media file format not all options are always available. An advanced search for a specific time stamp, text and other parameters is also included. [Dybkjær et al., 2001]

In the Data Analysis module the collected data can be analysed. Advanced filter can be used for a preselection of the data and complex queries for selection of data can be defined. Several analysis options are available. The user can generate different plots, graphs and reports. Elementary statistic functions such as frequencies and durations are available. Also features like simultaneous occurring events and advanced analysis like lag sequential analysis (how often are certain events preceded or followed by other events) and reliability analysis (measuring the level of agreement between pairs of different variables) are provided. Export functions for spreadsheet applications, databases and statistics packages are available.

The video capture add-on module provides support for 20 frames per second. The high resolution screen video from the User PC and the video from the user is captured on the Recorder PC. Digital media files including sound are created. The files have a maximum length of two *gigabytes* (GB) which is approximately equivalent to a two hour video recording. Logging and coding is possible



Start	00:00:13.64
Current	00:00:23.84
Elapsed	00:00:10.20

RECORD	TIME	BEHAVIOR	MODIFIER 1
224	00:00:23.32	wrnradde	
225	00:00:23.40	elbownfl	
226	00:00:23.40	wrnosupi	
227	00:00:23.40	handnoac	
228	00:00:23.48	snoabdu	
229	00:00:23.48	elbowfle	-45
230	00:00:23.64	handothe	
231	00:00:23.72	elbownfl	
232	00:00:23.84	shonofle	
233	00:00:23.84	wrsupina	
234	00:00:23.88	wrflexio	
235	00:00:23.92	shoabduc	+45
236	00:00:23.92	elbowfle	+45
237	00:00:24.00	shoflex	-45
238	00:00:24.04	elbownfl	
239	00:00:24.12	elbowfle	-45
240	00:00:24.16	wrulndev	
241	00:00:24.16	hndmediu	
242	00:00:24.24	wrnosupi	
243	00:00:24.24	wrnradde	
244	00:00:24.24	wrnexten	
245	00:00:24.28	taskend	1
246	00:00:24.28	shonofle	
247	00:00:24.28	wrsupina	
248	00:00:24.28	handfar	
249	00:00:24.36	wrextens	
250	00:00:24.40	wrnosupi	
251	00:00:24.44	shoflex	+45
252	00:00:24.48	shoabduc	+90
253	00:00:24.56	wrnexten	
254	00:00:24.60	elbowfle	-90
255	00:00:24.60	taskbegi	2
256			

[shoflex] shonofle	[shoabduc] shoabduc +90
[wrraddev] wrnradde	[wrextens] wrnexten
[elbowfle] elbownfl	[handgrip] handothe
[wrisupin] wrsupina	[handacti] handnoac
[postures] pseatupr	[handloca] handfar
[task]	

[shoflex]	wrd = wrraddev	enf = elbownfl	wns = wrnosupi	pss = pseatslu	ted = taskend
sfl = shoflex	wnr = wrnradde	[handgrip]	[handacti]	psu = pseatupr	
snf = shonofle	[wrextens]	hpw = handpowe	hct = handcut	[handloca]	
[shoabduc]	wfl = wrflexio	hpi = handpinc	hwd = handweld	hme = hndmediu	
sab = shoabduc	wne = wrnexten	hot = handothe	hcr = handcare	hcl = hnclose	
sna = snoabdu	wex = wrextens	[wrisupin]	hna = handnoac	hnf = handfar	
[wrraddev]	[elbowfle]	wsu = wrsupina	[postures]	[task]	
wud = wrulndev	efl = elbowfle	wpr = wrpronat	pst = pstand	tbg = taskbegi	

End of search operation

Figure 3.4: The Noldus Observer Event Recording Module. [Trienes et al., 1998]

already during the recording and of course afterwards. All streams are synchronised and controlled from The Observer software. The captured file is automatically synchronised with the observation data log too. A dual screen set-up can be used.

Noldus also provides portable solutions for field observations. The software is designed for windows operating systems and hand-held computers for use in the field. The Observer is a very powerful and efficient tool for recording, annotation and analysis of observational data [Noldus Information Technology bv., 2004].

3.3.4 Codecs

Codec is the abbreviation for compressor and decompressor. Codecs are applied to audio, video and image files to compress their size while storing on digital media or streaming over a network. If an adequate codec is used smaller files will be gained. To open, view or listen to a compressed file, the same codec must be used to decompress the file. Lossy is a compression technique decreasing image, audio, or video quality in order to reduce the file size. These codecs can not reconstruct the video at its original quality.

On the other site with lossless data compression algorithms the original data can be reconstructed exactly from the compressed data without losing any information. Lossless data compression is often

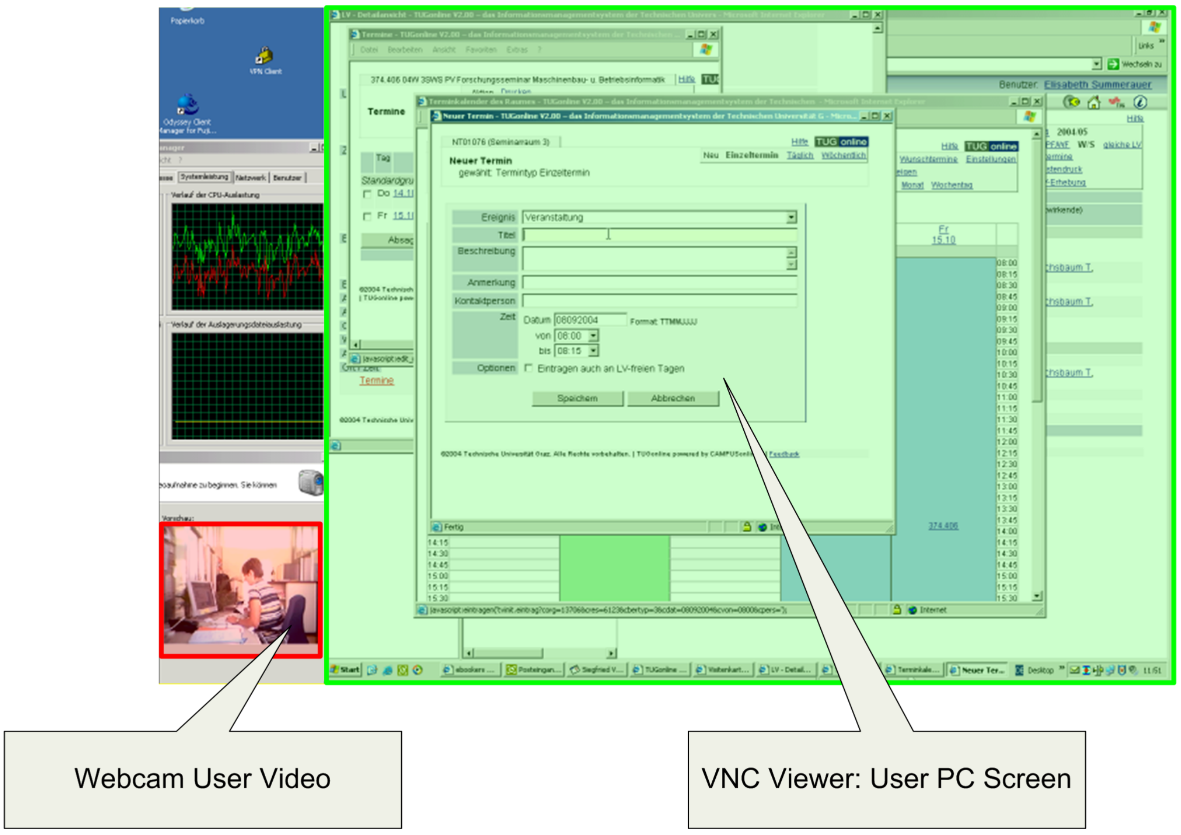


Figure 3.5: Recording of the User Screen and Video on the Recorder PC.

used in software compression tools such as the common ZIP file format. Lossless compression would be used if the original and the decompressed data had to be identical. Typical examples are executable programs or text files. Image compression use both, lossless and lossy algorithms. Lossless examples are *Graphics Interchange Format* (GIF) and *Portable Network Graphics* (PNG). *Joint Photographic Experts Group* (JPEG) is a lossy compression algorithm for images, MPEG is a lossy compression format for movie files derived from the JPEG format for still images.

In some cases a lossy method can produce a much smaller compressed file than any lossless algorithm, while still meeting the given requirements. Lossy methods are most often used for compressing sound, images or movies. In these cases, the retrieved file can be quite different to the original at the bit level while being almost indistinguishable to the human ear or eye for many practical purposes. Noticeable quality problems to the human eye or ear caused by lossy compression are known as compression artifacts.

The Figure 3.5 shows the two different videos which have to be recorded. The user video from the webcam and the full resolution User PC screen. The video from the User PC may already have a resolution of 1280 by 1024 if a state of the art flat 17 inches *Thin-Film Transistor* (TFT) display is used. Also refer to the Section 3.1 for an exact definition of the recording requirements. Both of these videos have very different requirements for a video codec. The webcam video is low resolution however has in general more motion and faster changes like in a TV movie. Many lossy compression algorithms exists for such signals like for example the already mentioned MPEG algorithm. The high resolution screen video has very seldom changes affecting the whole screen, only every time an application is started or changed. Most of the time just the mouse pointer is moved or some text is typed and so just a very small part of the screen is changed. The fonts are often very small with the used high resolution and so it is essential to avoid compression artifacts caused by a lossy compression algorithm. Due to the used colour palette is fixed in a digital generated screen as in opposite to a video originating from a webcam it is also possible to achieve a very high compression with lossless compression for screen recording.

The best technical solution would be to record the user webcam video and the forwarded screen content video separately. Different optimised codecs could be used. The synchronisation of these videos is a major challenge, since the synchronisation should also be kept even if the recording is several times stopped and started again. Also while analysing the videos the synchronisation should always be preserved. It was not possible to find a simple solution for this described synchronisation of the videos. This is the reason the idea emerged to record just one video containing both, the webcam video and the forwarded screen from the user (Figure 3.5). With this solution a synchronised video is already available after the recording process. An evaluation is conducted for such a combined screencasting.

Therefore the resulting geometric resolution for screencasting is around 1440 by 868 pixel. To keep the files smaller, the first step is to reduce the frame rate. Empirically a frame rate of seven frames per second was found as the minimal rate to obtain enough image quality for the coding process. With lower framerates it is not possible anymore to follow quick mouse movements.

3.3.4.1 Test Recordings

Some experiments take place for the evaluation of different codecs. No screen from another PC is forwarded in the first experiment, just the actual screen and for some of these test recordings also the video of a video camera which is previewed on this screen is screencasted. The following hardware and software is used:

- A video camera Canon DM-MV600E with a firewire connection and a tripod.
- The first test recordings are realised with a 1,4 GHz AMD Athlon computer with one GB RAM.
- A Windows XP Professional Edition operating system.
- The Microsoft Windows Moviemaker, which is a part of Windows XP, for showing the video from the camera.
- The Camtasia Recorder part from the proprietary Techsmith Camtasia Studio Software Suite (see Section 3.3.3.1).

Figure 3.6 presents such a recording session. It should be possible to record User PC screen resolutions up to 1280 by 1024 pixels. This can be achieved by switching the resolution of the Recorder PC to a resolution higher than the viewable screen resolution. The additional space is available via scrolling (this is similar to the virtual resolution feature available in the Unix/Linux X-Window System graphical user interface). The screen recording software can not recognise this virtual resolution and is able to record any chosen screen area inside or outside of the actual visible screen.

The standard codec used by the Camtasia Recorder is the *Techsmith Screen Capture Codec* (TSCC). The TSCC decoding part is freely available and freely redistributable. For normal desktop application screen recordings the TSCC is a very good lossless codec and achieves very good quality combined with very small file sizes while recording normal screen contents.

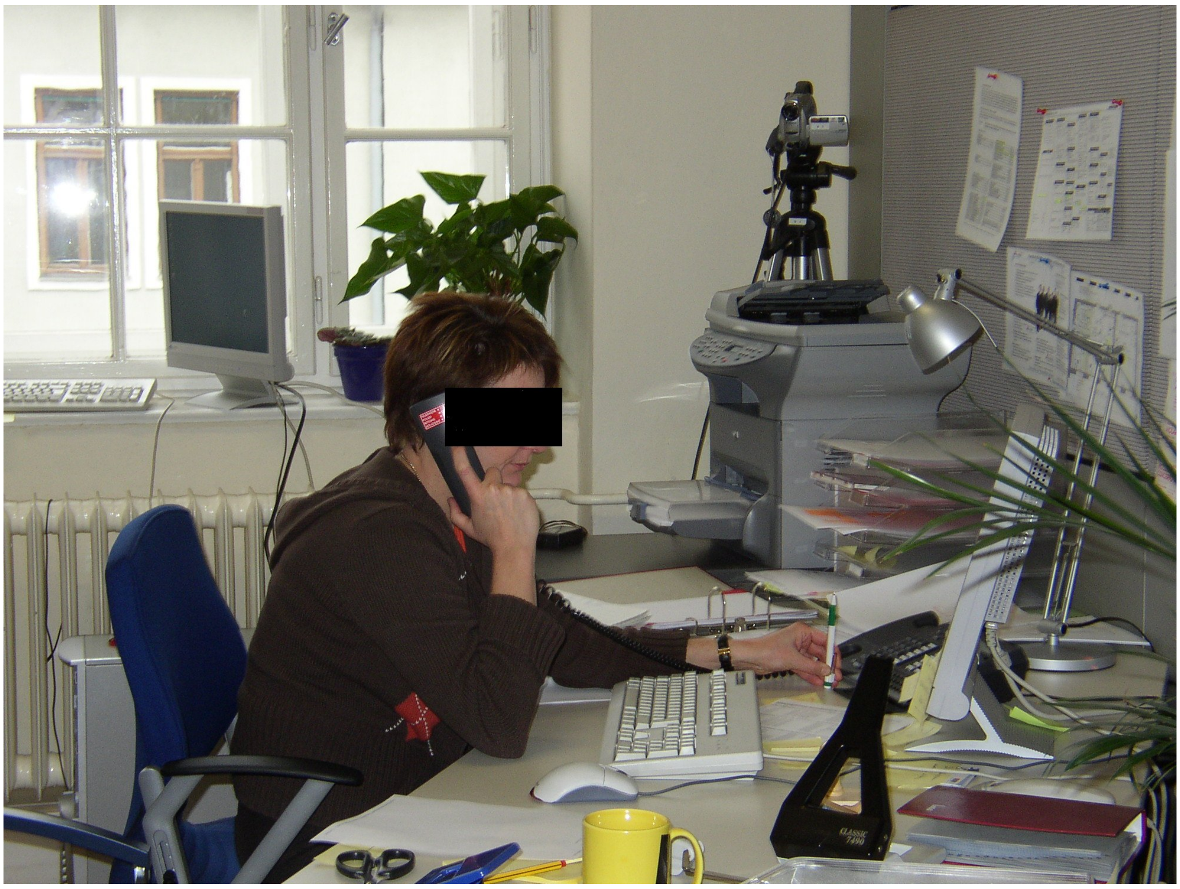

[redacted]

Figure 3.6: A First Experimental Recording with a Test Participant of the Study. Due to the Used Tower PC this First Test Recording Equipment is Not Very Portable. The Used Video Camera with Tripod is also Rather Intrusive.

Screen Test Recording without embedded Video The recorded screen size is 1280 by 1024. The capture performance will be better if hardware acceleration from the graphics card driver is disabled. Therefore the hardware acceleration is disabled for all recordings. The TSCC compression is switched to best compression. The system load values are approximate average values for an, besides the proceeding recording, idle system. This load is acquired with the Windows Task Manager application. If the mouse is moved very quickly or other user events are initiated, the load will be of course increase. These first results are shown in Table 3.1.

Advantages of the TSCC are nevertheless the very good quality (same as original) and the efficiency. It is no problem to record with TSCC compression on thy fly in realtime with the given hardware. Obviously the framerate and the bitdepth have a direct impact on the file size and system load. One possibility to reduce the file size is to reduce the recorded video bit depth. Width the TSCC 16 bit recordings are only half the file size of 32 bit ones.

Screen Test Recording with embedded Video For the next recording evaluation the screen with the previewed video from the camera is recorded. The stored video from Camtasia is therefore a combined single video file containing the screen and the video from the user. The camera video is

Framerate	Video Bitdepth	Audio Settings	System Load	File size / h
10 Frames / sec	32 Bit	PCM 16 Bit, Mono	70 %	273 MB / h
10 Frames / sec	32 Bit	No Audio	51 %	227.5 MB / h
5 Frames / sec	32 Bit	No Audio	26 %	69 MB / h
10 Frames / sec	16 Bit	No Audio	26 %	61 MB / h
5 Frames / sec	16 Bit	No Audio	14 %	46 MB / h

Table 3.1: Techsmith Camtasia Screen Recordings (TSCC) without Embedded Video.

Codec	TSCC	MS Video (CRAM)	Techsmith Ensharpen	DivX
Video Settings	32 Bit	16 Bit	24 Bit, Default Quality	24 Bit
Screen Quality	Very Good	Good	Good	Medium
Video Quality	Very Good	Bad	Good	Good
File size	875 MB	191 MB	457 MB	101 MB

Table 3.2: Short Test Video with Screen and Embedded Video Recording (18 Minutes, 8 Seconds, 7 Frames / Second) Compressed with Different Codecs.

positioned on the left hand side of the screen. No audio is recorded for this comparison. A laptop is now used for this experiment instead of a tower PC such as shown in Figure 3.6. The laptop is a Fujitsu Siemens Amilo M 1420 Centrino 1.7 GHz, 60 GB hard disk, 512 MB RAM with a resolution of 1280x800. As operating system a Windows XP Home Edition with Service Pack 2 is used. The length of the testvideo is 18 minutes and 18 seconds, the framerate is 7 frames per second, the size of the video picture is 242 by 184, this is the default size of the Moviemaker application preview screen, the total recorded resolution is 1548 by 1028 pixel. Audio is disabled for this experiment. The Camtasia Recorder application records an *Audio Video Interleave* AVI file. The AVI file format is a container format. It provides the possibility to embed different codecs in it. For the results with the different codecs see Table 3.2.

TSCC is not optimised for camera video compression, this is a problem with this described setup. If there is some moving picture, like a TV screen or video from a camera, part of the recorded screen area, TSCC will produce really large files. This is the case in this combined recording with the camera movie preview included in the screen recording area. Although this camera preview video is only 242 by 184 pixel in size the resulting video file is 875 MB. Without the camera video the resulting file would only have some MB (compare with Table 3.1).

The MS Video codec has an unacceptable quality for this recording setup. The files encoded with the Techsmith Ensharpen codec are too large, so these two codecs drop out for further evaluations. The XviD codec is included in the next experimental recording setup. An one hour test video is used. The framerate is again 7 framer per second, the size of the video picture is 242 by 184 and the total recorded resolution is 1288 by 1212. The camera video is again combined recorded as in the setup before. One difference to the prior recording is the reduced video bit depth from 32 to 16 bit for the TSCC. Also audio is recorded with 16 Bit *Pulse-Code Modulation* (PCM) and mono for this test. The results of this test can be found on Table 3.3.

Codec	**TSCC**	**DivX**	**XviD**
Video Settings	16 Bit	24 Bit	24 Bit, Cartoon & Turbo Mode
Audio Settings	16 Bit PCM Mono	mp3 30 KBit/s	mp3 30 KBit/s
Screen Quality	Very Good	Medium	Very Good
Video Quality	Very Good	Good	Very Good
File Size	1,446 MB	349 MB	228 MB
Encoding Time	1:00 (Realtime)	1:01	1:06
Comment	Large Files, Lossless	Small Files	Very Small Files, Very Good Quality

Table 3.3: Long Test Video (1 Hour, 7 Frames / Second, with Audio) Compressed with Different Codecs.

With the TSCC a file with one hour is still far beyond one gigabyte large. DivX and also XviD will only compress videos to a resulting video with 24 or 32 bit depth even if the resulting video is 16 bit depth. An attempt of direct encoding with XvID (with enabled cartoon and turbo mode) failed due to dropped frames. The CPU from the testsystem was too slow for this direct encoding process, although the visible load from the taskmanager did not actually show such a large load. The computer was not responsible to mouse events during this recording. That is why the video can only be recorded in realtime with the TSCC in the first step. Consequently for later processes like archiving and further analysis and coding processes the file size has to be reduced by using other codecs.

3.3.4.2 DivX and XviD Codecs

XviD is an Open Source MPEG-4 video codec. It is based on OpenDivX. After the OpenDivX source was closed a group of programmers started the XviD project. OpenDivX was an Open Source MPEG-4 video codec based on a version of the reference MPEG-4 encoder and started by DivXNetworks, however the code was placed under a restrictive license and only some people had write access to the CVS repository. After some struggles with open source developers about the license and the behaviour of DivXNetworks the OpenDivX project was ended and a closed source version was released. A fork of OpenDivX was created and released under the Open Source *GNU General Public License* (GNU GPL or simply GPL) and named XviD. [Wikipedia, 2005f; Sem, 2005]

XviD's main competitor is the proprietary DivX codec (DivX, Inc. [2005]). DivX is available as freeware in a binary format. A commercial version named DivX Pro is available too. DivX Pro has significantly better compression and speed. XviD performs better regarding the encoding speed an quality than the freeware DivX codec. With XviD 1.0.2 finally file sizes between 200 and 300 MB per hour depending on codec settings are possible as shown in Table 3.3.

3.4 Coding and Analysis Software

Usage and also usability studies based on digital media file data are a quite new field of research. Some years ago it was impossible to store as large data directly on digital systems. Analogue tapes were the common storage medium for long lasting recordings with high resolution. No more than some years ago it was impossible to handle such large amounts of digital data. The evolution in the computer hardware market, with the powerful CPUs, large hard disks and last but not least, new software developments made this possible. The operating systems are now able to handle very large files and hard disk partitions and new codecs emerged such as MPEG or DivX (see Section 3.3.4). The famous and common mp3 audio compression format is actually a subset from MPEG, exactly its name derives from MPEG-1 Audio Layer 3.

There is no standard product for coding and analysis of screencasted data from people performing real-world tasks. As already mentioned in Section 2.5 for the process of analysing time based observational data the term *Exploratory sequential data analysis* (ESDA) is commonly used in literature. ESDA software applications from different research areas are evaluated for being able to analyse the screencasted data. Several annotation software tools for linguistic research and gesture analysis are available. More and more research is done in the field of annotation of digital multimedia (for example video, gesture and audio) data files, this is also called multimodal data annotation.

Tools can be differentiated by the tasks handled. Data retrieval, annotation or coding of data and statistical analysis of coded data are different aspects. Not every software tool is supports all of these features. This section provides an overview of the most popular available software packages which may be adapted and used for the analysis of digital media files for the purpose of usage studies. The process of coding video data in the linguistic research area of spoken languages often also contains the transcription of the spoken language. Thus the overall process of coding is sometimes also called the transcription of a video. From now on the terms transcription and annotation are also used meaning the overall coding process of a video. An appropriate software application for the analysis stage in this study has to meet several requirements:

- A freely definable hierarchical coding scheme should be supported.
- Iterative changes from the taxonomy should be possible without loosing already coded transcription material. This flexibility is an essential factor, due to the fact not being possible to completely predefine the FUSS-B taxonomy.
- The handling of simultaneous events which overlap in time must be possible.
- If it is not possible to synchronise the two video signals the application must be able to handle large files. Additionally an appropriate codec must be supported. The software has to run efficiently and stable with these large digital media files. Section 3.3.4 covers the background regarding codecs and file sizes. If a combined recording was used the supported video file formats and codecs would have to include the AVI video file format with embedded DivX or XviD codecs. With one of these codecs acceptable file sizes are possible.
- If the two video streams are not already synchronised during the recording, the analysis software will have to handle two parallel video streams. These streams should always kept synchronised during the ongoing coding of the the different events.
- If there are different applications used for the coding and analysis step then the coding software should be able to export the data in an exchangeable format. The analysis software must be able to import this format.

- There are several separate video files from different users and recording sessions. In the coding or at latest in analysis stage it must be possible to combine the coding annotation data together to calculate summarised results finally.
- Direct jumps to video scenes where special coding events happened must be possible. So questions can be asked to the user afterwards and adaptions of the coding taxonomy can be easily applied. Interesting parts of the recording session can be found quickly.
- Some kind of search functionality in the transcript should be available.

Other essential requirements cover the usability and other features of the proposed coding software. To keep the time-consuming annotation process as efficient as possible, the interface should be generally easy and quick to use. The performance is important. It should be possible quickly jumping around even in large video files with the corresponding annotation data updated in realtime. An intuitive optical representation of the data is required. The annotation data has to be shown clearly arranged. Configuration of the layout and other attributes like colours and the arrangement of the annotation data should be provided. Due to the resolution of the video working on two monitors (dual head) should be possible. Frequently used operations should need only few mouse clicks and additionally shortcuts should be available.

Another important point covers the coding taxonomy. Leaving beside the already mentioned requirements regarding the hierarchical taxonomy and the flexibility in iteratively changing the taxonomy during the coding process another important fact is how the coding definition is handled by the software. Is it necessary to change some external configuration file or can the taxonomy be changed during the coding process? Is it possible to add some coding guidelines to the taxonomy definition? If parts of the taxonomy were changed, how would these changes be applied to the already coded data?

Is it necessary to code parts again? Can the start and end points of events easily be set with the user interface of the application during the coding stage? Is it possible to just count certain events instead of assigning a period of time? For some events only the occurrence instead of the duration is relevant (for example pressing the save button of an application). The defined taxonomy should be easily accessible in the interface. It should not be necessary to type in whole words. Choosing a proper coding item from a drop down listbox or something similar during the coding process with a state-of-the-art *Graphical User Interface* (GUI) should be the standard. It should be possible to add comments to the annotation too.

The flexibility and extensibility is also evaluated. Are there interfaces to other software applications available? Is an import or export available via common file formats or standards such as *Open Database Connectivity* (ODBC) for *Database Management Systems* (DBMS)? These covers the the storage of the coding taxonomy as well as the stored transcription data. Text files, structured text files like XML or *Structured Query Language* (SQL) DBMS are used for this purpose. XML files can immediately be accessed, transformed and analysed. Many software applications and standards are available for XML based data such as *Extensible Stylesheet Language Transformations* (XSLT) for transforming and *XML Path Language* (XPath) for querying XML based data. Another preferable export format is *Comma-Separated Values* CSV for the import in spreadsheet applications for further processing. Does the software has an published *Application Programming Interface* (API) for having the possibility to control and use it from other software applications? If a relational database is used as backend storage, would it be possible to gain access to this data with SQL queries or would the database be used somehow hidden and embedded by the coding software?

Due to the fact that several people are working together in the coding and analysis stage groupware features from the software can also be helpful. Is some kind of version management system included? Does the software provide tools for managing the collection of digital media files from the recording sessions? If the software was also used for analysis, some common statistical operations would become necessary. At least total durations of a given kind of events, calculation of mean values and counting for the different possible events should be supported.

3.4.1 Mongold Interact

The software by Mangold Software & Consulting GmbH [2005] is an analysis solution for studies on behaviour. It is a very flexible solution, has a modular design and is extensible. The coding taxonomy is freely definable no predefined behavioural categories have to be used.

Several streams can be coordinated, like the user video and the screen video. Also other data can be synchronised like key strokes or biophysical data for medical research purposes with the dataview add-on module. The software has multiple user capabilities. A first iteration of coding is already possible during the observation. Interact has no problems in handling simultaneous events which overlap in time. It is also possible to control many different analog consumer and broadcast video equipment.

Many different statistical analysis processes are possible with the Interact software. Calculation of durations and frequencies in percentage or absolute or graphical representations are possible. There are also built-in export capabilities in customisable formats. Exporting data for spreadsheets or other statistical software is no problem. A powerful highlight movie creator is available as add-on. So it is possible to combine special coded scenes from several input videos in different formats to one output video.

An additional IXL (Interact eXtension Language) module is available for integrating the software into existing environments. Additional functionality can be obtained by writing own plug-ins for specific purposes. The language for writing these plugins is Delphi which is a object oriented successor from Pascal.

To summarise Mangold Interact is a quite mature (over ten years development and enhancements) coding and analysis software application. It has many useful features and is very extensible and integrates in many different environments. Academic licensing models are available.

3.4.2 MacSHAPA

MacSHAPA by Sanderson [1997] is a Macintosh software developed by Penelope Sanderson at the University of Illinois. It is a tool for analysing sequential data. Transcriptions with and without user defined annotation coding schemes are provided. Codes can be defined for describing different human or system activities. Annotation, manipulation and visualisation of data is possible. Unstructured comments, annotations and events can be coded. All of them are associated with a particular point in time. A spreadsheet like view is used for entering and viewing the data. Individual data is contained in single spreadsheet cells and a sequential data streams are arranged in columns.

Import and export of data and results from and to other applications is supported with various textformats. MacSHAPA can be used for controlling a video recorder at different speeds. Timecodes can be captured from video and included into the data. A videotape position associated with a given timecode can be found. The Video can be displayed at an external monitor or with appropriate hardware videos can also be displayed on the computer screen. The professional video recorders Panasonic AG-7750 or AG-DS850 are directly supported and other video recorders are supported via the additional Abbate VTK third-party software.

MacSHAPA offers many different statistical functions for quantitative and qualitative analysis of the transcoded data. Content analysis, duration analysis, lag sequential analysis, reliability analysis and other statistical methods are provided. A built-in query language can be used to query the data also in other ways than already supported ba the statistical methods.

The major shortcoming with MacSHAPA is the necessary hardware for complete functionality including controlling video recorders because digital media files cannot be used directly. Although it is still possible to gain the software and a very detailed manual (800 pages) presently MacSHAPA does not seem to be maintained or improved any more (the last update of the webpage was 1997 and the last version 1.0.3 release dates back to 1995). There is no Linux or Windows version of MacSHAPA available.

3.4.3 Techsmith Morae Manager

The Morae Manager supports editing, analysing and presentation preparation. The user and screen videos are synchronised automatically and all events generate timestamps and indexes in the video supplemented by markers and annotations of human viewers. An assembling of a highlight video is easily possible.

The Manager also includes several built-in analysis possibilities like time on specific tasks, number of clicks or page views, time spent on a web page, delay times and more. Visualisation of these results can be viewed in several with the recorded video synchronised formats (for example time-stamped lists or interactive graphs). An export to comma-delimited files is also supported . [TechSmith Corporation, 2005b]

3.4.4 SignStream

SignStream is a tool assisting in the transcription and analysis of video-based language data. SignStream is being developed by researchers at Boston University, Dartmouth College, and Gallaudet University. SignStream is part of the American Sign Language Linguistic Research Project and is supported by the National Science Foundation. [Neidle et al., 2004].

The main focus of the tool is the sign language research. Digital video files are the basis for the transcription process. Despite of its original orientation towards representation of signed language data, SignStream can be used for different kind of linguistic research relying on video data, such as research in phonetics or gestural components. SignStream is distributed on a non-profit basis to students, educators, and researchers.

The tool is designed to facilitate research on signed languages and gesture. A video is divided into a collection of so called utterances. One utterance is a video segment associated with a transcription. Each utterance can be associated with one or more synchronised media clips at once. Since version 2.0 up to four synchronised video files are allowed. In addition, it is possible to associate a synchronised audio file and access a visual display of the wave form in another window. Overlapping segments of a video can be coded as distinct utterances. Every utterance has a number of glosses (tracks). Within these utterances and glosses the single items are coded. The used coding scheme is relatively extensible. SignStream supports multiple synchronised movie and sound files. Direct access to video and audio segments corresponding to utterances or to specific items, coded within utterances, is provided. Sophisticated and complex search capabilities are included. Script facilities are also provided, making it possible to save and play subsets of utterances in a given order. [MacLaughlin et al., 2000; Neidle, 2000]

The video and transcription are displayed in one ore more video windows and a gloss window. In Figure 3.7 three video windows and the gloss window situated below are visible. This gloss window the main input environment, where the annotation of video frames is entered. For each entered data

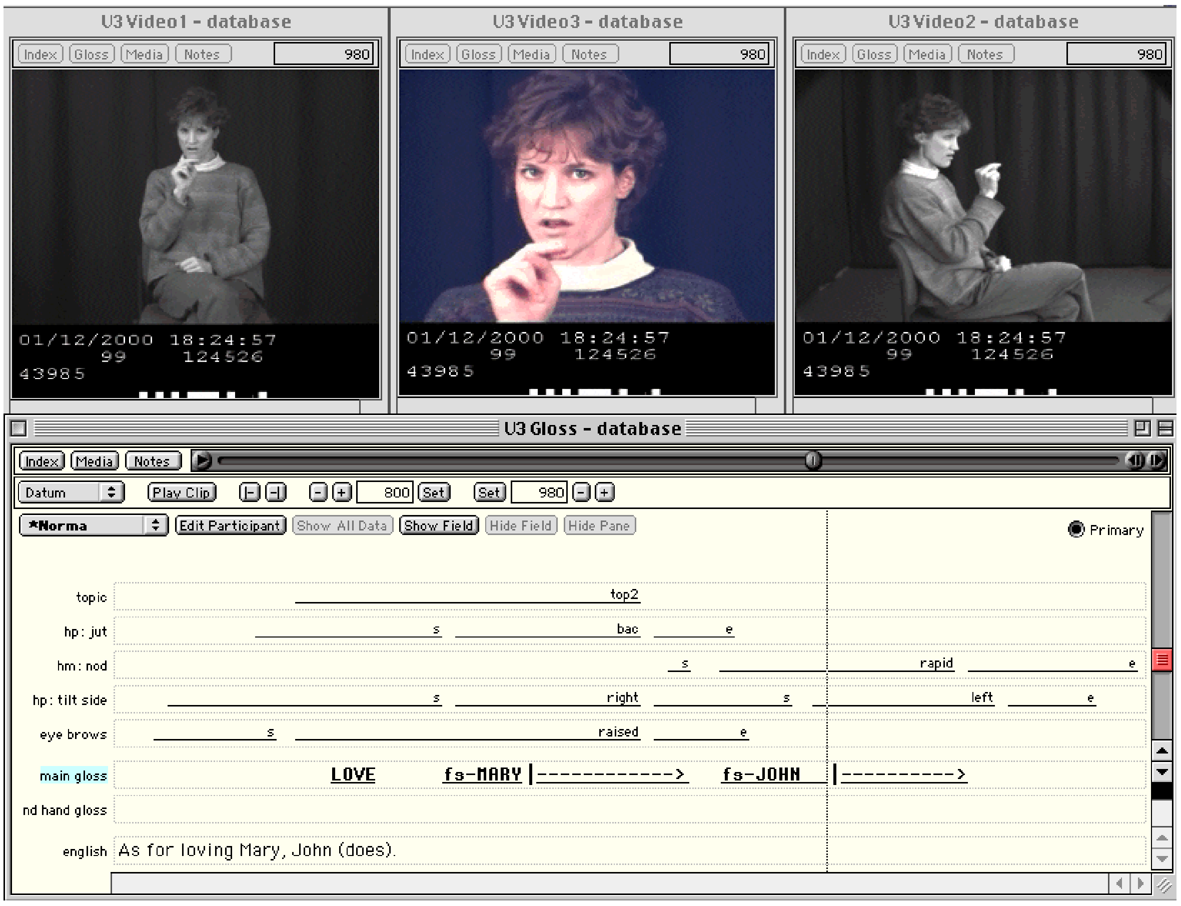
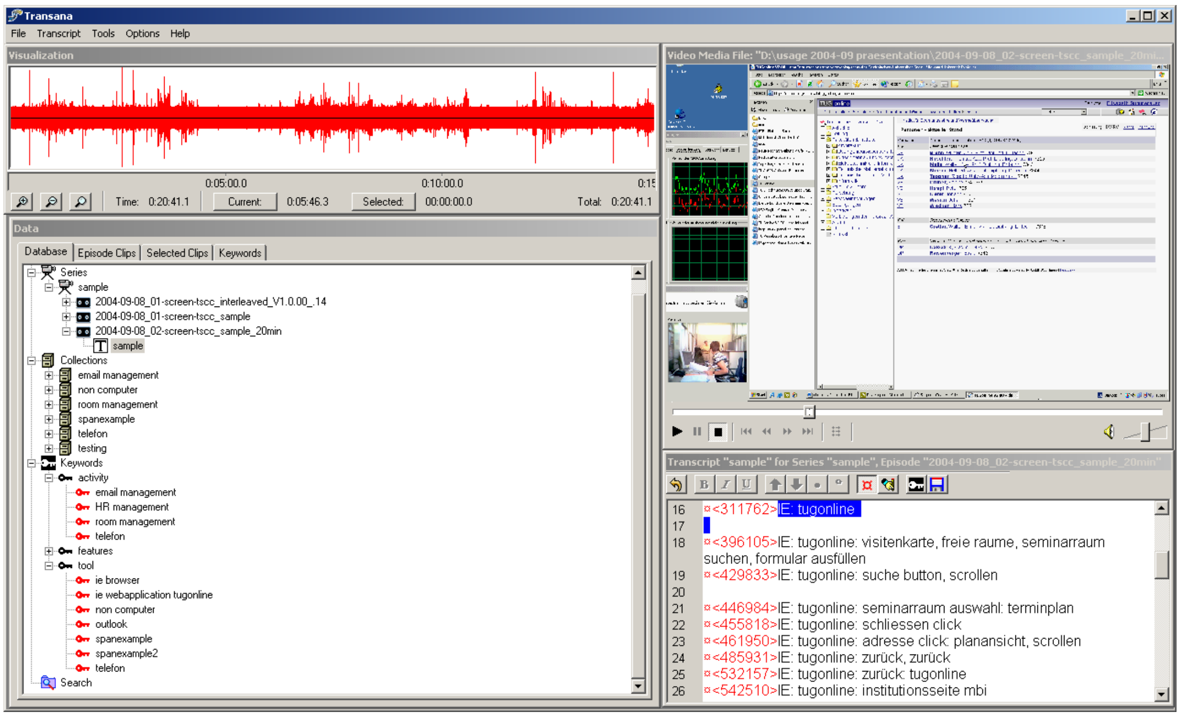


Figure 3.7: SignStream allows several synchronised Video Files. The draggable media alignment indicator in the gloss window shows the current frame position. [Neidle, 2005]

item the user defines the start and end frames. These annotated data items can be edited or removed in the ongoing annotation process.

These are the two main environments of the user interface. The video can be viewed with a controls including frame-by-frame advance and scroll bar for positioning the video at an arbitrary point in time. SignStream also provides a variable speed controller, which allows the user to change the speed at which the video is played. The SignStream tool provides a set of predefined fields specifically designed for representing sign language data. However, the application also allows the user to define new fields and new field values. The layout can be changed by the user in a number of ways. The user can select which fields should be displayed, fields can be rearranged, positions and colours can also be customised. The coded items are represented with their start and end points. The tool allows the user to carry out advanced search queries in the database.

The actual SignStream V2.2 application runs on MacOS systems. Videos must be QuickTime format. The supported audio format is *Audio Interchange File Format* AIFF format are supported. Information from a SignStream database can be exported into a system-independent text format. SignStream is not currently available for Windows or UNIX platforms. Currently a new implementation of SignStream (version 3) is in progress. This new version is being ported to Java and will be initially released for windows and macintosh operating systems. SignStream 3 is also designed to use the *Java*



```
¤<311762>IE: tugonline

¤<396105>IE: tugonline: visitenkarte, freie raume, seminarraum
suchen, formular ausfüllen
¤<429833>IE: tugonline: suche button, scrollen

¤<446984>IE: tugonline: seminarraum auswahl: terminplan
¤<455818>IE: tugonline: schliessen click
¤<461950>IE: tugonline: adresse click: planansicht, scrollen
¤<485931>IE: tugonline: zurück, zurück
¤<532157>IE: tugonline: zurück: tugonline
¤<542510>IE: tugonline: institutionsseite mbi
```

Figure 3.8: A Screenshot of the Transana Application.

Media Framework (JMF) for media playback, so easy porting to Linux and other Java-enabled UNIX platforms is provided. JMF enables sound and video facilities in the Java programming language. SignStream 3 will also include XML (Extensible Markup Language) import and export capabilities and many other improvements. SignStream version 3 will be released under an Open Source license. [Zhuravlova et al., 2004, 2005]

To summarise in short, the main functionalities of SignStream include browsing of video data consisting of up to four synchronised files, the annotation of video data with simultaneous layers of description. Simultaneous occurring events in different layers are aligned vertically from SignStream on the screen.

SignStream would meet the FUSS-B study requirements for the coding process however the actual stable version SignStream 2.2 is only available for Macintosh computers. The actual development status of the Java based SignStream 3, which would also support windows operating systems, is still pre-alpha. There is no Macintosh operating system available for the FUSS-B study hence SignStream is not used for this study.

3.4.5 Transana

Transana is a software application for researchers for analysing large collections of digital video or audio data. With Transana these digital media data can be transcribed, analysed and managed in different ways. This transcription process generates links between the positions in the written transcript and to the corresponding time positions in the video. Transana can be used by researchers in many diverse disciplines. Possible areas of research include educational practices, the analysis of languages, the analysis of animal behaviour and so on.

Transana was originally created by Chris Fassnacht. It is now developed and maintained by David K. Woods at the Wisconsin Center for Education Research, University of Wisconsin-Madison. Transana is since version 2.0 released as Open Source under the GPL. It is used in the education research community, where the analysis of video data is very important. Transana runs on Windows in both single-user and multi-user versions. A Macintosh version is actually in development and is available as an alpha version. [Woods and Fassnacht, 2005]

Many methods are provided for organising the video based data into different categories. Related videos can be grouped together into a series. On the other side every video file can also be splitted into categorised clips. Keywords can be assigned to both clips and videos. Related clips can be grouped together into so called collections. The keywords are searchable and so it is possible to identify and access special segments of video data, even is spread across many different files. A large number of media files can be managed with the built-in database from Transana. A database window, arranged in a tree structure, is provided to help manipulate and organise these groups. In Figure 3.8 a screenshot of transana is shown. The relatively prominent waveform screen can be used for quickly navigating in the video. If a position in this waveform is selected with a mouse click it will be immediately jumped to this position in the video.

A multiuser version is available for collaboratively working together via the internet over distances. Working in teams on a common base of video data files and analytic annotation data is possible.

Transana stores the annotation transcripts into a database. Video and audio files are not stored in this database. With the Export Database tool a Transana XML file containing all of the data in the database can be created for export and analysis purposes or to move the database from one computer to another. Transana 1.xx used a Paradox database, while Transana 2.xx uses an embedded MySQL database engine. The transcript can be exported into the common *Rich Text Format* RTF format for cross-platform document interchange. Most word processors are able to read and write RTF documents. At this time Transana works with MPEG-1, MPEG-2, and most AVI video formats. MP3 and WAV are supported as audio formats.

3.4.6 Anvil

Anvil is a generic research tool for the analysis of digital data. Anvil stands for *Annotation of Video and Language Data*. It is a powerful software application for areas like linguistics, gesture research, human computer interaction and others. It is developed by Michael Kipp [2004a], University of the Saarland, Germany and is freely available for research use. It comes with a highly intuitive graphical user interface. Anvil provides support for multiple parallel tracks for video annotation.

Anvil is Java based and is actually available in version 4.5. The *Java Media Framework* (JMF) is needed for processing the media files. It is therefore almost platform-independent and runs on any operating systems where Java and JMF is available. It is tested on MacOS X, Windows, and Solaris yet it should also run on other Unix systems including Linux. Quicktime and AVI media file formats are supported, both are container formats. These container formats provide the possibility to embed different codecs. Anvil can handle any codec supported by JMF 2.1.1, so many video and audio codecs are supported (`http://java.sun.com/products/java-media/jmf/2.1.1/formats.html`). Like the classification scheme the internal configuration and the annotation results from the coding process are also stored in XML files.

Anvil is designed and coded object-oriented. The top object is an annotation. This annotation object is the container for all the encoded information from one video file. The annotation depends on a coding scheme which is a formal description of the annotation scheme. This file is called specification file in Anvil terminology and is actually an XML file. This specification must be defined in terms

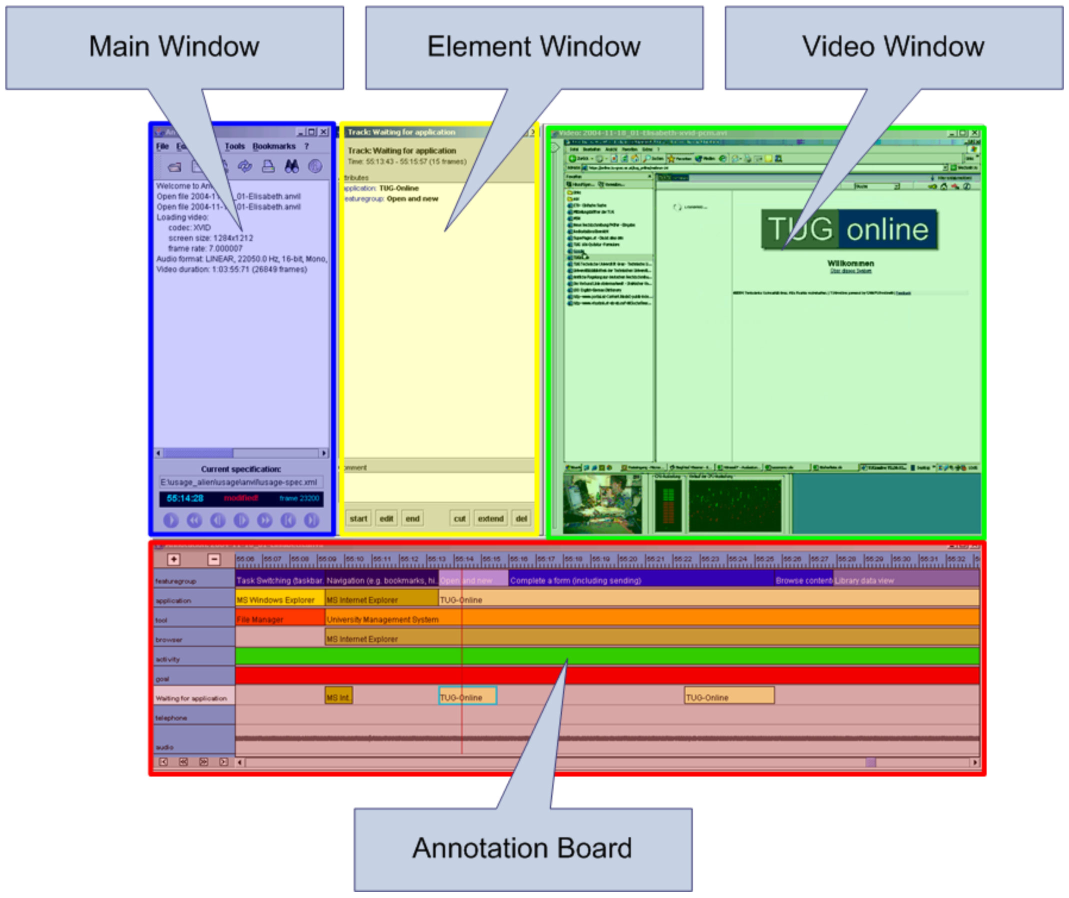


Figure 3.9: Screenshot of Anvil Using a Single Screen Setup with Description of the Windows.

of hierarchical tracks. Each track represents a special kind of elements. The tracks for themselves contain several track elements also called the annotation elements. This elements are configurable by means of the XML specification file too. The track elements can hold a number of attributes. This provides an almost freely definable taxonomy for annotation.

Three different types of tracks are available. There are primary tracks on the one side. Primary tracks consist of primary elements with defined start and an end points. On the other side there are two types of secondary or reference tracks. Secondary tracks contain secondary (or reference) elements. The first type of these secondary tracks is the singleton track. It has elements which exactly point to one corresponding element in the primary track. The other type of secondary track is the so called span track. The span track has elements with defined start and end primary elements references. So this newly defined element from the reference track spans over the time period covered by the two given primary track elements. Tracks can be grouped together with the structural group node. Groups can have attributes and value sets which can be inherited by containing tracks or other groups.

The user interface consists of a Main Window, the Video Window, the Element Window and the Annotation Board. For a screenshot of Anvil see Figure 3.9. The Main window includes the toolbar with icons for the most important functions and it contains a window with a list of user actions and other information like the format of the video file. Also the specification of the currently loaded annotation is shown in the Main Window. Video controls are also a part of the Main Window.

The Annotation Board is the most important window of the Anvil application. The actual coding takes place in this window. A time aligned view with all tracks containing their elements is given. Whole groups of tracks can be hidden by deselecting the corresponding group in the view menu of the Main Window. Each element is displayed as a coloured rectangle in the Annotation Board on a kind of timeline. This timeline can be zoomed. With the highest zoom rate single frames are visible as ticks on the timeline. Simultaneousness is clearly visible. An element contains at least following data: start time, end time and one or more attributes. The attributes can be of different types such as string, boolean or value set and are defined in the annotation XML scheme file. The value set type is the most interesting type for this study because it contains a list of user-defined strings. Each value of an element can be assigned a specific colour. This colour is shown in the Annotation Window. One of the defined values of a value set can be selected from a selection box during the coding process. A value set track corresponds to a defined coding taxonomy where for example only a set of predefined applications or feature groups exist. It is not possible to add a new element without changing the specification.

The Element Windows gives some information about the actual selected element of the currently active track. Also new track elements can be added or already existing elements created with this window. In the Video Window the video can be played or the currently position is shown with a still picture.

A project tool is included for browsing and searching across several annotation files. The project tool also supports exporting data to external software for further statistical analysis purposes. This can be imported into common spreadsheet applications or other powerful commercial software tools such as SPSS or Statistica.

Anvil is tested with AVI video files. TSCC, DivX and XviD codecs are evaluated. Plenty of *Random Access Memory* (RAM) is definitely needed when using large high resolution files like it is necessary for this study. When jumping from one position in the video to another, which is possible by clicking on a new position in the annotation board, it lasts some time until the playback can be started from this position. Maybe even more RAM would enable a better performance as it is often the case with Java programs. XviD compressed files are handled better regarding these waiting times than TSCC compressed files. This is an interesting finding. Probably this is introduced due to the large TSCC coded files. The embedded video causes this large file sizes. The video can be compressed much better with the other codecs (compare with Table 3.3).

A drawback with Anvil and XviD compressed files appear in connection with the mp3 audio codec. These files cannot be used due to the fact the audio playback is not possible. The videos can be played normally, the codec is detected (this is visible in the Main Window) however no sound is audible. That is why the TSCC files from the recording are converted into XviD files with a PCM encoded audio layer. These files are actually a bit larger than files with mp3 encoded layer though having the advantage of using less processor power while decoding.

If the taxonomy is iteratively changed during the annotation, the already coded parts will not correspond to this new taxonomy. All coded annotations will have to use the same taxonomy for later analysis. The necessary changes can be done either manually in Anvil or manually in the XML file. Anvil offers a search function within tracks, however no replace feature is supported. Nevertheless it is very easy for example to rename an element because the Anvil annotation XML files are text based. All of the action can be accomplished with a single string search and replace in all existing annotation files. If a new element is needed this will cause no problem in the already existing coded annotation. Some bugs were encountered while existing annotations were changed with the Anvil GUI because of changed coding guidelines. If the cut, extend or delete feature is used very often, sometimes wrong values will be written in the annotation file. It was not reproducible however sometimes elements from referenced tracks showed up with the end element referencing to an smaller element number than the corresponding start element. Such elements are invisible in the Anvil annotation area. These problem were fixed directly in the XML file.

The configuration, the annotation scheme and the results are all stored in the open XML standard. These is an advantage of Anvil. Many tools can be used for creating, editing and analysing such files. Furthermore many libraries for different programming languages exist for the work with XML files.

It is possible to be reasonably efficient while coding with Anvil. However if jumping around in the annotation, Anvil will sometimes be a bit slow. This is particularly caused by the large high resolution files. Anvil is not initially designed for these kind of files. Kipp states:

> "Screen Size: I recommend 384x288 (European TV half size) or 320x240 (American TV half size) - double values (768x576 or 640x480) if you really need the detail but be aware that you need a very powerful computer to cope with the increased data processing, and also, a large video pane leaves little room for Anvil's Annotation Board." [Kipp, 2003, Section 2.2]

The problem with the large video pane can be workarounded with using a dual-screen (dual head) setup. On the other side the Java programming language in principle is not very fast and needs much memory when using multimedia features due to the internal virtual machine structure. A further upgrade of main memory could possibly improve the performance although the PC for conducting the analysis already has one GB of main memory. In summary although Anvil was particularly created as tool for linguistic and gesture research, it is a very suitable tool for human computer interaction studies.

Anvil Feature Overview

- Anvil runs on MacOS X, Windows, Solaris and Unix systems including Linux.
- Anvil supports both AVI and Quicktime video format. Also the XviD codec is supported.
- The classification scheme or taxonomy is stored and defined in an external XML file.
- The resulting annotation file after coding are also stored in a file with the XML format.
- Although written in the Java language, version 4.5 achieves a fluently playback of the videos, however it requires at least 1024 MB RAM to work efficiently. Combined with a fast CPU (AMD64 3500+) it was possible to analyse videos in XviD with a duration over 80 minutes with an acceptable performance. Interestingly this contrasts to the opinion of the Anvil author:

  > "What is the maximum length of a video for usage in Anvil?
  > Currently, due to the way graphics in used in Anvil, you can only use video files up to a certain length (about 10 minutes) depending on your computer's working memory (so try bigger files first). This restriction will certainly vanish at some point in the future. In the meantime, researchers can improvise by cutting their files to 10 minute chunks. Note that you can still work quite comfortably with these chunks using Anvil's Project Tool (included in the distribution)." [Kipp, 2004b, Anvil FAQ]

- Statistical analysis of the data is not supported instead analysis can be performed by directly working on the XML output files or using the included export features for statistical applications like SPSS or spreadsheat applications.

- By means of a bookmark function it is possible to mark specific points in the annotation. Unfortunately in the tested Anvil 4.5 this feature is broken. If the bookmark feature was used the annotation result XML file would be destroyed. In this case an error would be shown if the file was opened thus the file has to be repaired manually.
- Freely available for research purposes.
- Intuitive user interface enables an efficient coding process. Shortcuts are provided for the most important functions. The main coding functions can be done using all three mouse buttons.
- Anvil can generate a coding manual if documentation elements are included in the specification file. This manual is generated in HTML and can be useful for ensure consistency between different coders.

3.5 Additional Used Software

Several people are working together for this FUSS-B study. Thus the recording, coding and analysis software also other software products could be useful. In this section other software products, particularly software for supporting team collaboration is dealt with.

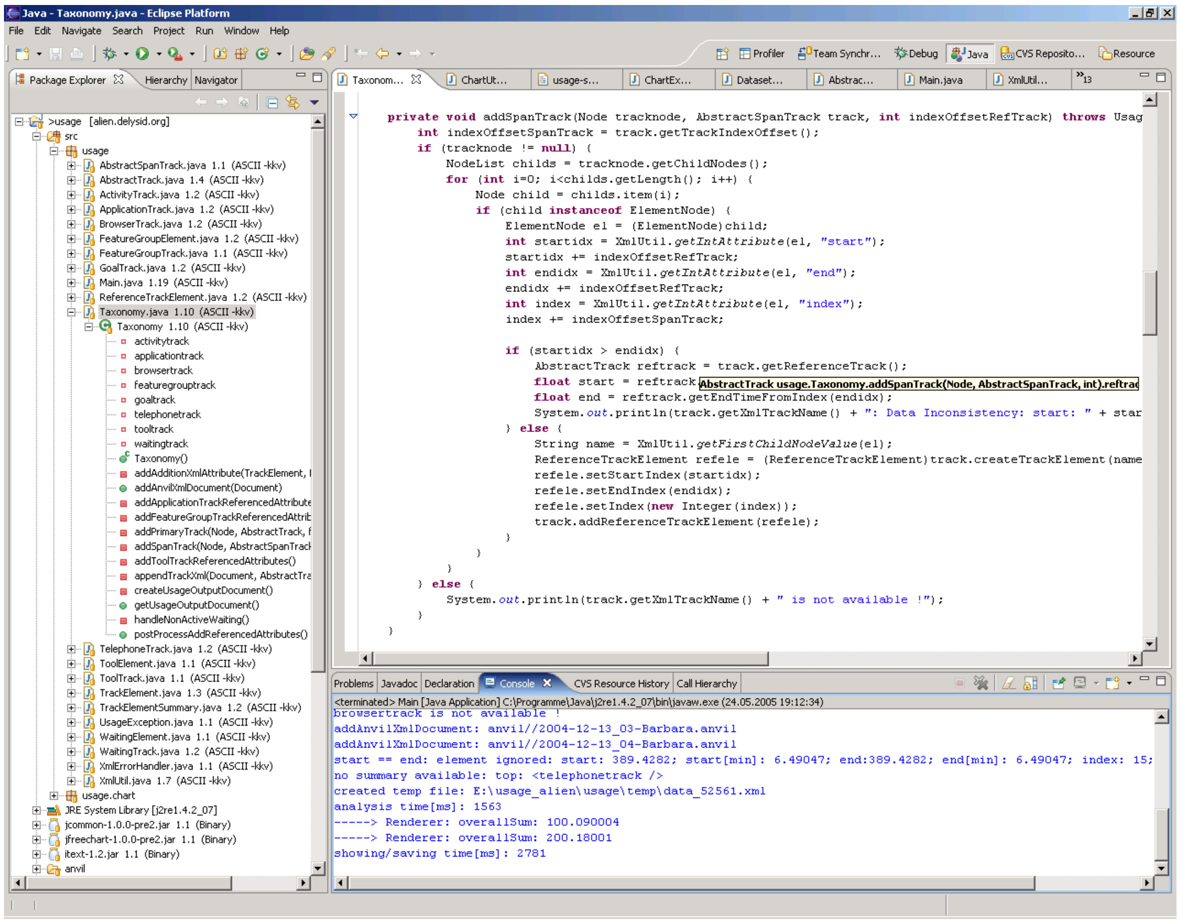
3.5.1 Eclipse

Eclipse [2005] can be used as an *Integrated software Development Environment* (IDE). Eclipse was initially developed by IBM and is now Open Source. Eclipse was at first designed for the Java programming language. Meanwhile many other programming languages are supported by the Eclipse IDE including C, C++ and PHP.

> "Eclipse is an open platform for tool integration built by an open community of tool providers. Operating under an Open Source paradigm, with a common public license that provides royalty free source code and world wide redistribution rights, the eclipse platform provides tool developers with ultimate flexibility and control over their software technology. ...The Eclipse Platform is written in the Java language and comes with extensive plug-in construction toolkits and examples."

Eclipse is very usable as part of a software engineering development system. Some very special features are available. Many refactoring possibilities are implemented. Refactoring means restructuring of the code for gaining better extensibility without implementing new features. For example it is possible to extract some parts of the code to a method (function) using autodetection of parameters or debugging running remote application. Many features support the programmer such as showing the possible methods of an object, code templates (for example a template for a loop including local variables can be automatically generated) and many more.

For Java an incremental compiler is working in the background, so every code change immediately shows possible problems. The debugger is even capable of inserting new code parts into running remote machines. Eclipse can be customised in many ways. Many plugins are available for different purposes: profilers, graphical template designer for report generators, JDBC (Java database connectivity) database frontends to name just a view. Most of these plugins are available as some kind of Open Source license however there are also commercial plug-ins on the market.



```
>usage  [alien.delysid.org]
  src
    usage
      AbstractSpanTrack.java  1.1  (ASCII -kkv)
      AbstractTrack.java  1.4  (ASCII -kkv)
      ActivityTrack.java  1.2  (ASCII -kkv)
      ApplicationTrack.java  1.2  (ASCII -kkv)
      BrowserTrack.java  1.2  (ASCII -kkv)
      FeatureGroupElement.java  1.2  (ASCII -kkv)
      FeatureGroupTrack.java  1.1  (ASCII -kkv)
      GoalTrack.java  1.2  (ASCII -kkv)
      Main.java  1.19  (ASCII -kkv)
      ReferenceTrackElement.java  1.2  (ASCII -kkv)
      Taxonomy.java  1.10  (ASCII -kkv)
        Taxonomy  1.10  (ASCII -kkv)
          activitytrack
          applicationtrack
          browsertrack
          featuregrouptrack
          goaltrack
          telephonetrack
          tooltrack
          waitingtrack
          Taxonomy()
          addAdditionXmlAttribute(TrackElement, I
          addAnvilXmlDocument(Document)
          addApplicationTrackReferencedAttribute
          addFeatureGroupTrackReferencedAttrib
          addPrimaryTrack(Node, AbstractTrack, f
          addSpanTrack(Node, AbstractSpanTrack
          addToolTrackReferencedAttributes()
          appendTrackXml(Document, AbstractTra
          createUsageOutputDocument()
          getUsageOutputDocument()
          handleNonActiveWaiting()
          postProcessAddReferencedAttributes()
      TelephoneTrack.java  1.2  (ASCII -kkv)
      ToolElement.java  1.1  (ASCII -kkv)
      ToolTrack.java  1.1  (ASCII -kkv)
      TrackElement.java  1.3  (ASCII -kkv)
      TrackElementSummary.java  1.2  (ASCII -kkv)
      UsageException.java  1.1  (ASCII -kkv)
      WaitingElement.java  1.1  (ASCII -kkv)
      WaitingTrack.java  1.2  (ASCII -kkv)
      XmlErrorHandler.java  1.1  (ASCII -kkv)
      XmlUtil.java  1.7  (ASCII -kkv)
    usage.chart
  JRE System Library [j2re1.4.2_07]
  jcommon-1.0.0-pre2.jar  1.1  (Binary)
  jfreechart-1.0.0-pre2.jar  1.1  (Binary)
  itext-1.2.jar  1.1  (Binary)
  anvil
```

```java
private void addSpanTrack(Node tracknode, AbstractSpanTrack track, int indexOffsetRefTrack) throws Usag
    int indexOffsetSpanTrack = track.getTrackIndexOffset();
    if (tracknode != null) {
        NodeList childs = tracknode.getChildNodes();
        for (int i=0; i<childs.getLength(); i++) {
            Node child = childs.item(i);
            if (child instanceof ElementNode) {
                ElementNode el = (ElementNode)child;
                int startidx = XmlUtil.getIntAttribute(el, "start");
                startidx += indexOffsetRefTrack;
                int endidx = XmlUtil.getIntAttribute(el, "end");
                endidx += indexOffsetRefTrack;
                int index = XmlUtil.getIntAttribute(el, "index");
                index += indexOffsetSpanTrack;

                if (startidx > endidx) {
                    AbstractTrack reftrack = track.getReferenceTrack();
                    float start = reftrack.AbstractTrack usage.Taxonomy.addSpanTrack(Node, AbstractSpanTrack, int).reftrac
                    float end = reftrack.getEndTimeFromIndex(endidx);
                    System.out.println(track.getXmlTrackName() + ": Data Inconsistency: start: " + star
                } else {
                    String name = XmlUtil.getFirstChildNodeValue(el);
                    ReferenceTrackElement refele = (ReferenceTrackElement)track.createTrackElement(name
                    refele.setStartIndex(startidx);
                    refele.setEndIndex(endidx);
                    refele.setIndex(new Integer(index));
                    track.addReferenceTrackElement(refele);
                }
            }
        }
    } else {
        System.out.println(track.getXmlTrackName() + " is not available !");
    }
}
```

```
<terminated> Main [Java Application] C:\Programme\Java\j2re1.4.2_07\bin\javaw.exe (24.05.2005 19:12:34)
browsertrack is not available !
addAnvilXmlDocument: anvil//2004-12-13_03-Barbara.anvil
addAnvilXmlDocument: anvil//2004-12-13_04-Barbara.anvil
start == end: element ignored: start: 389.4282; start[min]: 6.49047; end:389.4282; end[min]: 6.49047; index: 15;
no summary available: top: <telephonetrack />
created temp file: E:\usage_alien\usage\temp\data_52561.xml
analysis time[ms]: 1563
-----> Renderer: overallSum: 100.090004
-----> Renderer: overallSum: 200.18001
showing/saving time[ms]: 2781
```

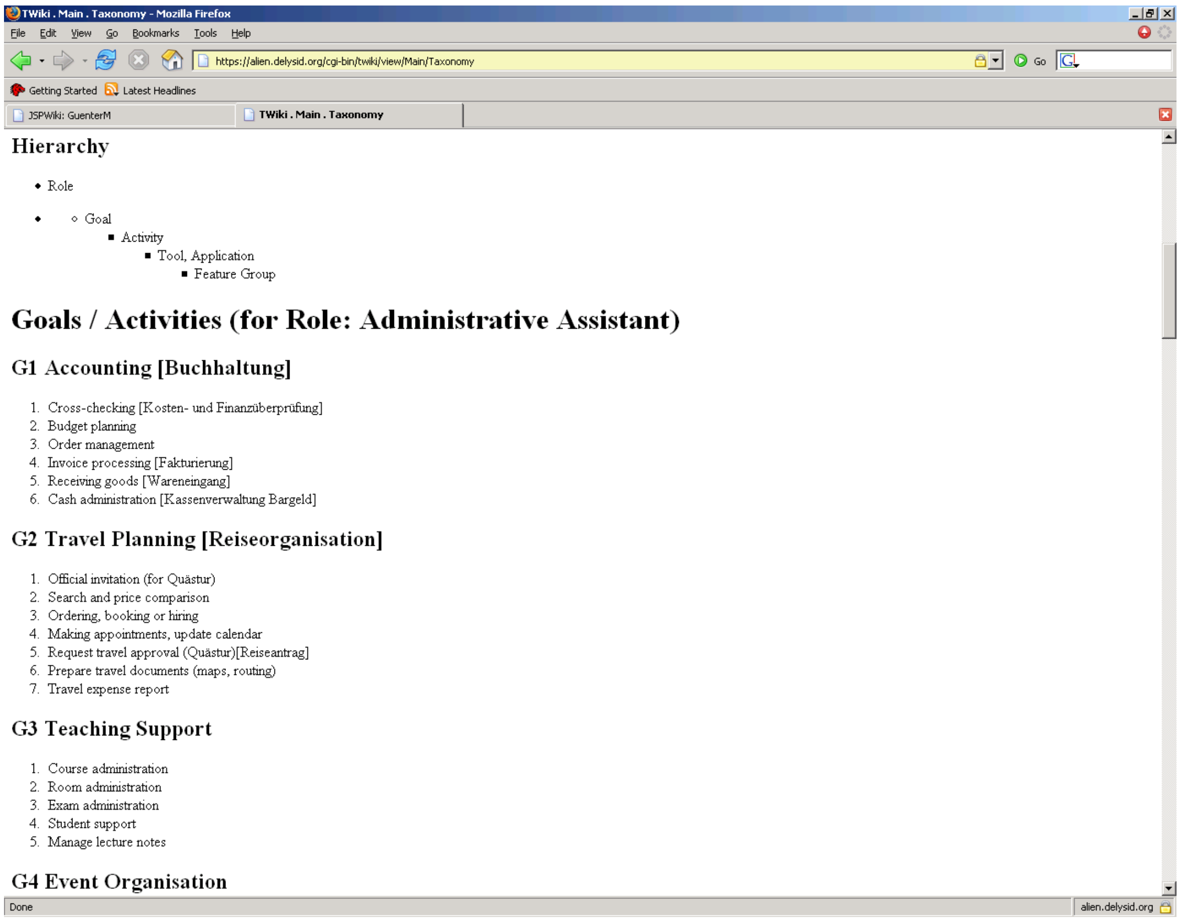
Figure 3.10: Screenshot of Eclipse while Coding.

3.5.2 Revision Control Systems

By means of revision control systems different versions of files can be handled. Revision Control Systems are very common in software development for being able to manage iterative processes and evolution in the development process. Changes to these files are identified by incrementing an associated version number.

Normally revision control systems are server-client systems. A revision control server has a common storage area shared between all clients. This storage area is called repository. It is possible to revert to old versions of a file and the history of changes on files in the repository can be overseen. In the software area particularly documentation, configuration and source code files are stored in a revision control system. It is possible to store different type of files, text or binary, in such systems. Examples of binary files are images or sound files, examples for text based files are source code or XML files.

Concurrent accesses to the same files from several people are handled different from the available versioning applications. Locking the actual files is one possibility. Another possibility are automated or user assisted merges if one ore more files are changed at the same time from different users. In any case no changes should be overwritten and lost.


TWiki . Main . Taxonomy - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
https://alien.delysid.org/cgi-bin/twiki/view/Main/Taxonomy
Go
Getting Started
Latest Headlines
JSPWiki: GuenterM
TWiki . Main . Taxonomy
Done
alien.delysid.org

Hierarchy

- Role
-
  - Goal
    - Activity
      - Tool, Application
        - Feature Group

Goals / Activities (for Role: Administrative Assistant)

G1 Accounting [Buchhaltung]

1. Cross-checking [Kosten- und Finanzüberprüfung]
2. Budget planning
3. Order management
4. Invoice processing [Fakturierung]
5. Receiving goods [Wareneingang]
6. Cash administration [Kassenverwaltung Bargeld]

G2 Travel Planning [Reiseorganisation]

1. Official invitation (for Quästur)
2. Search and price comparison
3. Ordering, booking or hiring
4. Making appointments, update calendar
5. Request travel approval (Quästur)[Reiseantrag]
6. Prepare travel documents (maps, routing)
7. Travel expense report

G3 Teaching Support

1. Course administration
2. Room administration
3. Exam administration
4. Student support
5. Manage lecture notes

G4 Event Organisation

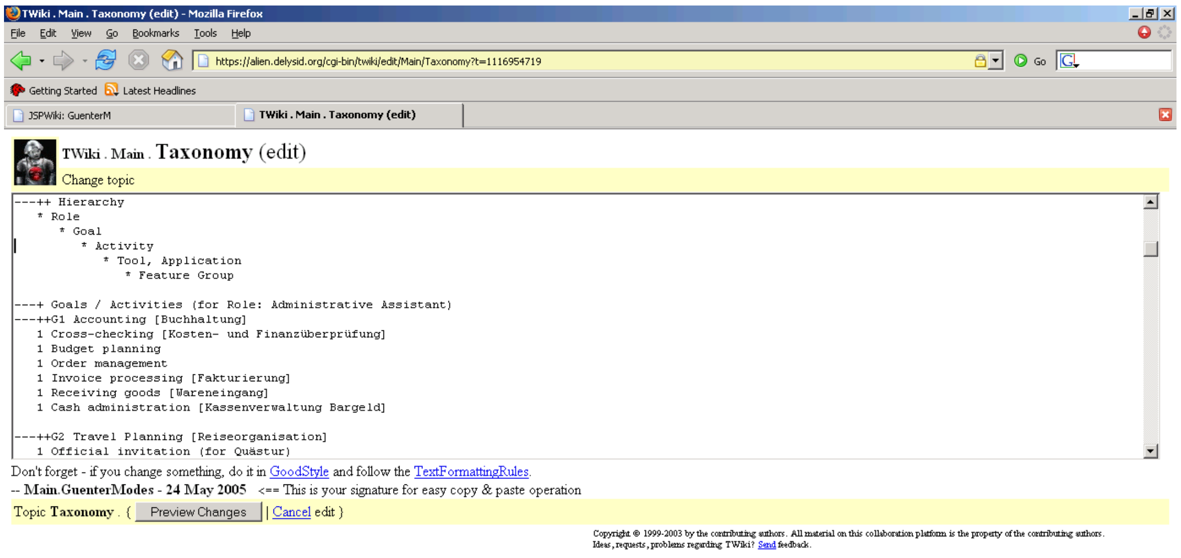
Figure 3.11: Screenshot of a Twiki Wiki Rendered Page.

Version management is a very useful tool for collaborating in a team. It can also be useful for a single person working on several different workplaces and computers for always having a common repository of files with their respective changes in history available. Examples for Open Source revision control systems are CVS, subversion and arch. Examples of common commercial revision control systems include IBM/Rational Clearcase and Microsoft Sourcesave.

Some IDE have plug-ins available for revision control systems. Eclipse has plug-ins for CVS and subversion. In the FUSS-B study the CVS revision control via the Eclipse plug-in is used.

3.5.3 Wiki

A wiki is an interactive website allowing users to add and edit content. Also the serverside software running such a website is sometimes called wiki [Wikipedia, 2005e]. With wikis it is easy to change and add content without using a *HyperText Markup Language* HTML editor or manually writing the HTML code. New pages can be created easily. Web available documents can be written collectively in a simple markup language using a web browser. The syntax of wiki markup languages varies among the different wiki implementations. The wiki server application generates out of this simple wiki markup language the HTML code which is shown in a web browser when viewing the wiki page.


TWiki . Main . Taxonomy (edit) - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
https://alien.delysid.org/cgi-bin/twiki/edit/Main/Taxonomy?t=1116954719
Getting Started Latest Headlines
JSPWiki: GuenterM
TWiki . Main . Taxonomy (edit)
TWiki . Main . Taxonomy (edit)
Change topic

```
---++ Hierarchy
   * Role
      * Goal
         * Activity
            * Tool, Application
               * Feature Group

---+ Goals / Activities (for Role: Administrative Assistant)
---++G1 Accounting [Buchhaltung]
   1 Cross-checking [Kosten- und Finanzüberprüfung]
   1 Budget planning
   1 Order management
   1 Invoice processing [Fakturierung]
   1 Receiving goods [Wareneingang]
   1 Cash administration [Kassenverwaltung Bargeld]

---++G2 Travel Planning [Reiseorganisation]
   1 Official invitation (for Quästur)
```

Don't forget - if you change something, do it in GoodStyle and follow the TextFormattingRules.

-- **Main.GuenterModes - 24 May 2005** <== This is your signature for easy copy & paste operation

Copyright © 1999-2003 by the contributing authors. All material on this collaboration platform is the property of the contributing authors.
Ideas, requests, problems regarding TWiki? Send feedback.

Figure 3.12: Screenshot of the Twiki Wiki Editor.

The first wiki ever was The Portland Pattern Repository. It was created 1995 by Cunningham [2005]. It is sometimes called WikiWikiWeb or just Wiki written with a capital 'W'. Most wikis are open to the general public without the need to register any user account. Nevertheless there are only rare cases of wiki vandalismprobably since it is no fun for hackers to break into an unsecured system.

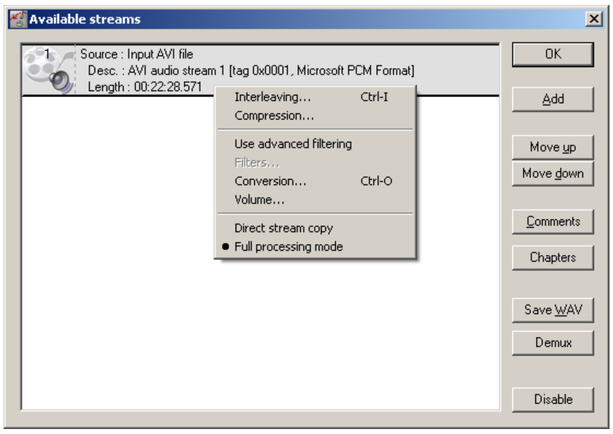
Most wikis have built-in search capabilities. Most of the wikis allow the attachment of files to every page. This is mostly used for images to display though normally any type of file can be attached. Many wikis also offer some kind of revision control for having access to older versions of the page. Often also file attachments are also included in the revision control. Most wikis offer some kind of a *Recent Changes* page for having available the last changes at once.

Wikis are used for many different purposes. Documentation of ongoing (software) projects and common knowledge bases are often wiki based. A very famous example is the free Wikipedia encyclopedia (`http://www.wikipedia.org`). In the FUSS-B study the twiki wiki is used. Coding guidelines, documents and comments are developed together and can be viewed from everywhere with an internet access.

3.5.4 VirtualDub and VirtualDubMod

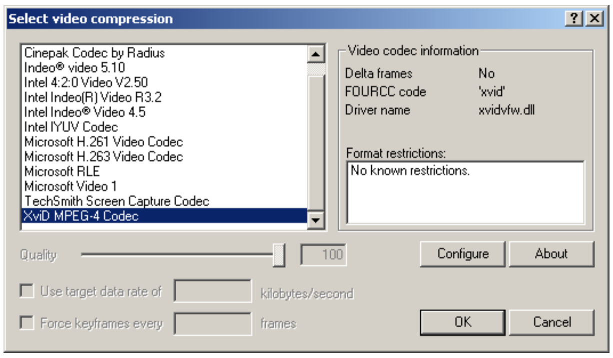
VirtualDubMod is based on the video editing software VirtualDub by Avery Lee [Abreu et al., 2005]. VirtualDub is a video capture and processing utility for Windows platforms and is Open Source licensed under the GPL. Many features are provided. For example it is possible to remove, replace or reencode only the audio track without touching the video track of a multimedia video file. Of course the same is true for changing only the video track. Segments of the video can be removed without recompressing the video. Results can be previewed including audio. Many video filters are available such as sharpen, blur, rotate, brightness and many, many more.

Other provided functionalities include an integrated volume meter and histogram for input level monitoring, a verbose monitoring including compression levels and CPU usage. It is even possible



1 Source : Input AVI file
Desc. : AVI audio stream 1 [tag 0x0001, Microsoft PCM Format]
Length : 00:22:28.571

Figure 3.13: Screenshot of the VirtualDubMod Audio Codec Dialog.



Cinepak Codec by Radius
Indeo® video 5.10
Intel 4:2:0 Video V2.50
Intel Indeo(R) Video R3.2
Intel Indeo® Video 4.5
Intel IYUV Codec
Microsoft H.261 Video Codec
Microsoft H.263 Video Codec
Microsoft RLE
Microsoft Video 1
TechSmith Screen Capture Codec
XviD MPEG-4 Codec

Video codec information

Delta frames	No
FOURCC code	'xvid'
Driver name	xvidvfw.dll

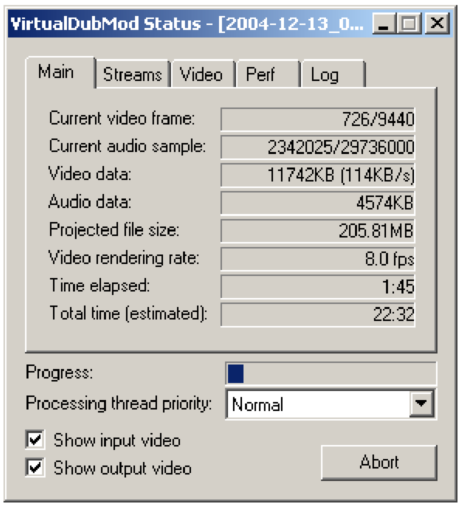
Format restrictions:
No known restrictions.

Quality 100

Figure 3.14: Screenshot of the VirtualDubMod Video Compression Dialog.

to use fractional framerates (such as 15.23 frames per second). VirtualDub has batch-processing capabilities for processing large numbers of files. VirtualDub's primary focus is the processing of AVI files, although it can read although not write MPEG-1 files too. The application is designed for speed. The user interface is intuitive and powerful and the processing pipeline is fast. The Homepage of VirtualDub is available at `http://www.virtualdub.org/`.

VirtualDubMod is an unification of several existing modifications (VirtualDubMPG2, VirtualDubOGM and VirtualDubAVS) of VirtualDub. VirtualDubMod adds additional features to VirtualDub such as MPEG-2 processing and support for additional audio formats such as AC3, OGG vorbis and VBR (Variable Bit Rate) mp3. Multiple audio stream support for all output formats is also added. Figures 3.13, 3.14 and 3.15 respectively show some of the powerful configuration and monitor dialogs from the VirtualDubMod application. Further information is available from `http://virtualdubmod.sourceforge.net/`.



Current video frame: 726/9440
Current audio sample: 2342025/29736000
Video data: 11742KB (114KB/s)
Audio data: 4574KB
Projected file size: 205.81MB
Video rendering rate: 8.0 fps
Time elapsed: 1:45
Total time (estimated): 22:32
Progress:
Processing thread priority: Normal
☑ Show input video
☑ Show output video

Figure 3.15: Screenshot of the VirtualDubMod Progress Dialog.

3.6 Evaluation Conclusion

3.6.1 Ideal Recording Solution

The ideal recording solution for this FUSS-B study would have been some kind of hardware based recording solution. Either with a self designed system consisting of a VGA splitter and a second Recorder PC with a VGA capture card like the Vision-RGB Pro or with an all-in-one commercial system like the Biobserve Spectator. The advantages of one of these hardware based solutions would be:

- The least impact on the user environment.
- No software installation is required on the User PC.
- Hence also no administrative rights are required on the User PC.
- No access to the ethernet or any other connection from the Reorder PC to the User PC besides the splitted monitor signal is necessary.
- A hardware based recording solution automatically supports any operating system on the User PC.

The only, however for this study unfortunately decisive drawback, is the higher price of such a solution. The FUSS-B study has a constraint budget so a setup with a commercial product or some kind of very special hardware like the powerful Vision-RGB Pro is not possible for now. Nevertheless a visionary hardware design for further studies is considered and can be found in Section 7.1.5. Currently a setup with standard computer hardware and only software based recording is chosen.

3.6.2 Chosen Software for Study

The study recording requirements are defined in Section 3.1 and the requirements for the coding and analysis process are found at Section 3.4. Morae, VisualMark and the Noldus Observer are examples of commercial all-in-one software based recording, coding and analysis tools. All of these above mentioned commercial applications would provide the necessary features (and even more) regarding recording, coding and analysis. So all of this different phases of the study could be done with the same tool. Although cheaper than hardware based solutions shown in Section 3.2, also these products unfortunately exceed the budget from the FUSS-B study. Thus these software based solutions are also not chosen and used for this study.

Instead it was decided to use a software based remote viewing application in combination with a screencasting tool as described in Section 3.3 for the recording stage. For the coding the Anvil software application is elected, thereby an additional requirement arise for the recording software to already combine the two videos during the recording process as already mentioned in Section 3.3.4.

3.6.3 Chosen Recording Solution

To have the least possible impact on the User PC the remote viewing application software is the only software installed on the User PC. A second Recorder PC is connected via ethernet to the User PC and receives the signal from the User PC via a remote viewing client application. A software for recording the screen contents (including the User PC remote screen and the camera signal) has to be installed on the Recorder PC too (see Section 3.3 and Figure 3.2).

UltraVNC

A VNC based software is picked as remote viewing software. There are many different variants hence some criteria were applied for getting the most suitable for this study. The installation should be easy and most important the hardware requirements, particular on the User PC and also on the Recorder PC, should be as minimal as possible.

At first the TightVNC was tested as of some good reviews found in the internet. On the first User PC an enormous performance impact was caused by the TightVNC Server application. The mouse pointer was flickering and the overall performance of the computer was very bad. This problem was not reproducible on other computers, therefore it was tried to upgrade the memory of the User PC and even to change the network card. The problem was not solved with these measures so another VNC server software was tried, the UltraVNC server.

One advantage of UltraVNC is the optional available high performance video driver, named hook driver. UltraVNC is available for Windows platforms only. Although the different VNC server and clients are more or less compatible often better results are achieved when using the same software on the server- and clientsite. This is the reason why on the Recorder PC also the UltraVNC client software is used. With the combination of UltraVNC server and client applications an usable performance was gained on the first User's PC. On the computers of the other participants of the study such disturbing performance impacts were not observed.

The webcam video from the user is displayed on the screen with the Windows Movie Maker application. Despite the fact that this preview video is already being recorded with the screen recording software it is cautionary recorded redundantly with a higher resolution in a separate file. The performance of the Recorder PC is high enough to store both video streams without dropped frames. Actually it was *not* necessary to use this parallel recording anytime during the analysis process for further investigation of events.

Camtasia Recorder and Codecs

On the Recorder PC the screen from the User PC and the webcam video preview is shown. To be able to record the screen a screen recording software has to be installed. The Camtasia Recorder from Techsmith is used for this purpose. The Camtasia Suite is a very powerful and innovative collection of applications and is described in Section 3.3.3.1. Only the Camtasia Recorder part is used for this study.

Test recordings suggest (Section 3.3.4.1) creating AVI files using the Techsmith default TSCC during the coding process and converting the video files to other formats for archiving and coding afterwards. A framerate of seven frames per second , a video bit depth of 16 bit is used and audio is recorded in PCM 16 bit mono.

For the conversion from the large AVI files with TSCC into smaller files using the XviD codec the VirtualDubMod (see Section 3.5.4) software is used. The file sizes can be reduced significantly (see Table 4.1). VirtualDubMod is a robust Open Source software application even when handling very large files with sizes above one GB.

Recording Solution Discussion

With these described setup most of the initial recording requirements as defined in Section 3.1 are fulfilled. The full resolution from the User PC screen is recorded hence also small fonts are easily readable. The screen video is already synchronised with the webcam video during the recording. Mouse movements can be followed sufficiently when using the chosen framerate.

Not fully satisfied initial requirements include the long time recording. The user can not be observed non-stop over several hours. With the described setup it is only possible to record video continuously up to approximately 1 hour and 30 minutes otherwise the resulting AVI files would create files with sizes above two GB and Camtasia Recorder is not being able AVI files above this limit. Another small issue with the Camtasia Recorder will be a required additional step when a recording with audio track is stopped. The application interleaves the audio track into the video track. This is done with help of temporary files and after this interleaving stage is complete the file is copied to the defined end location. This procedure will last for about six minutes if one hour is recorded with the described setup.

If the resulting video was more than two GB in size the Camtasia Recorder software will crash during the interleaving process and will destroy the file. To be on the secure side (depending on actual screen, video and audio data the actual file size can vary) it is decided to record videos up to approximately 1 hour and 15 minutes duration. So an uninterrupted recording is not possible, because at least every one hour and 15 minutes around seven minutes is interrupted because of the mentioned interleaving process and a few more minutes are required to setup and start the new recording again (choosing file location for screencast, select the region to record and so on). The users will be pleased if they are not observed continuously without interruption for very long periods. This fact is observed in the recording sessions with several test participants. Thus in practice this forced break is rather seen positive by the users.

Another not supported initial requirement is the easy start and stop possibility. With the used recording setup, before the recording takes place, it is necessary to manually start the used software applications and to arrange the windows sizes and positions. If the screen recording with Camtasia is started, the region for recording will have to be defined too. It has not been evaluated how these procedures can be automated. Thus in the beginning of each recording a person has to initialise all applications (UltraVNC server and client, Camtasia Recorder, Windows Movie Maker and optionally the Windows Task Manger for viewing the actual load on the Recorder PC for later analyses regarding the Recorder PC performance) and start the tapping. The location for saving the recorded file must be chosen too.

This is a software based setup hence the initial requirements regarding the least possible impact on the User's environment, meaning no installation of software on the User PC and no required ethernet connection, are not met with this setup. It is at least necessary to install the VNC server application on the User PC. And an ethernet access is also required for getting the forwarded screen to the Recorder PC VNC client application. The VNC server application is quite small and the actual recording takes place on the Recorder PC hence the performance and resource impact on the User PC is manageable.

With this setup the resulting recorded files are quite large. These files are converted into smaller files after the recording process. It is also not possible (although not necessary for this study) with this setup to apply an initial coding already during the recording process. Some commercial software applications support this feature. For the Recorder PC a laptop is chosen, so the required hardware for being able to conduct a recording consists of a microphone, a webcam, a laptop and some cables and is very portable.

The microphone and webcam is connected with cables to the Recorder PC hence this PC has to be near the user probably in the same room. Administrative rights are required for the installation of the VNC server and also for the installation of the optional high performance video driver therefore this will yield to problems if used in the field. Also the required ethernet connection with a necessary valid LAN IP address is problematic considering several security issues. Security experts will mention the possible threat from viruses and other malicious code which may be spread from the Recorder PC into the LAN, possible attacks on other computers of the LAN, the possibility of sniffing passwords, etcetera. For the conducted test recordings of this study these issues could be discussed and were finally accepted by the respective system administrators.

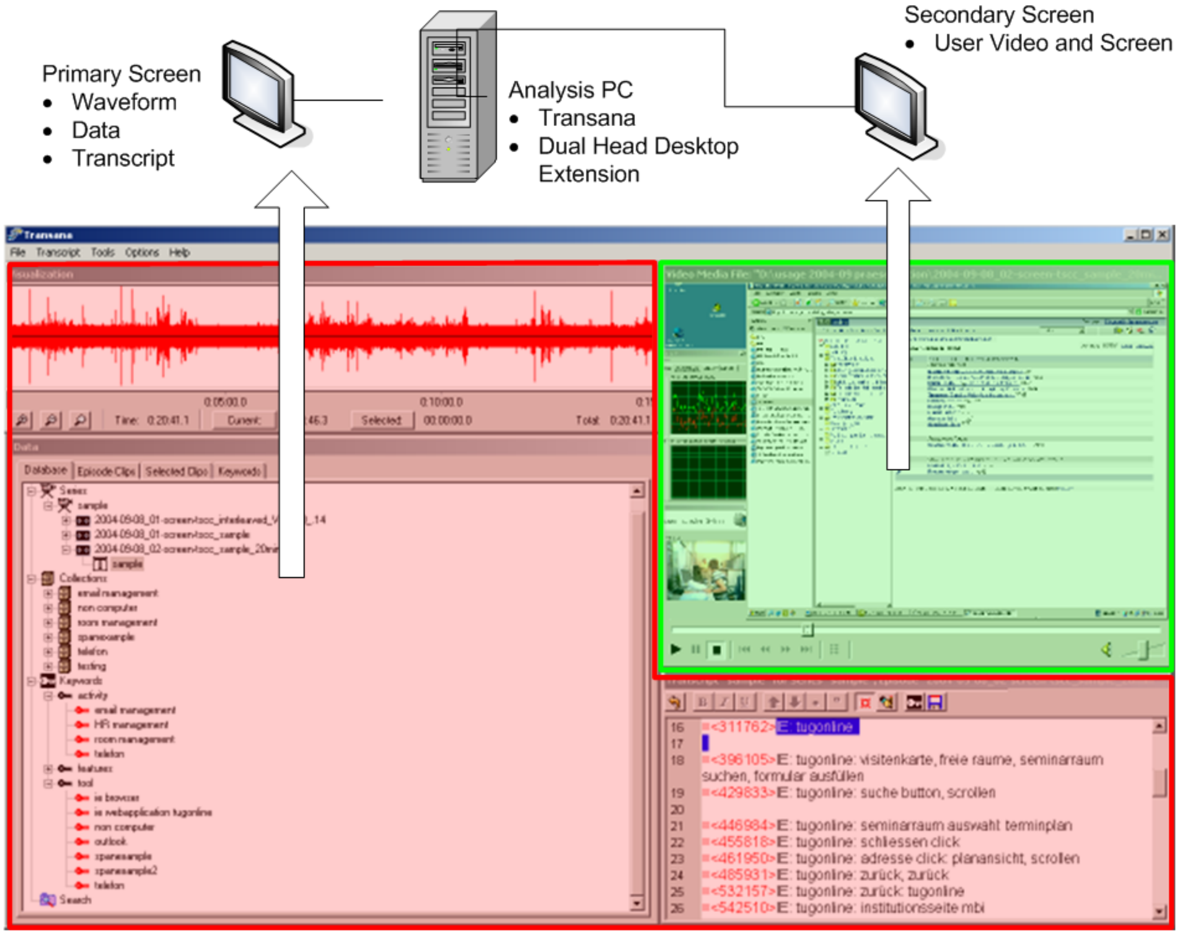
Despite these mentioned drawbacks the described setup proved to be quite usable. It is portable and was accepted by the test users. Furthermore the acquisition of the necessary hard- and software is not expensive compared to other solutions. Some ideas for improvements regarding the software based recording solution setup are outlined in Section 7.1.

3.6.4 Coding

SignStream would meet the requirements however the actual stable version is only available for Macintosh computers. The actual development status of the Java based SignStream 3 is still pre-alpha. After the release of SignStream 3 a new evaluation of this application should be considered. VisualMark on the other side would also be a very interesting solution for the FUSS-B study because no software is installed on the User PC. Nevertheless VisualMark is not picked because the analysis software only runs on Macintosh computers and no Macintosh computer is available for this study. Both applications in the actual stable versions only run on Macintosh computers so they drop out from the group of evaluated coding software. MacSHAPA has the drawback not being able to handle digital video files directly from a computer and MacSHAPA can only directly control analog video recorder hardware devices. Like SignStream and VisualMark also MacSHAPA only runs on Macintosh computers too.

Anvil versus Transana

Anvil is platform-independent and has a good usability. All used files are XML files and can be easily processed from other software applications. A coding scheme can be almost freely defined and changed with little effort during the coding process. The handling of simultaneous events overlapping in time is no problem with the multiple layer architecture from Anvil.



```
16 <311762>E: tugonline
17
18 <396105>E: tugonline: visitenkarte, freie raume, seminarraum
suchen, formular ausfüllen
19 <429833>E: tugonline: suche button, scrollen
20
21 <446984>E: tugonline: seminarraum auswahl terminplan
22 <455818>E: tugonline: schliessen click
23 <461950>E: tugonline: adresse click: planansicht, scrollen
24 <485931>E: tugonline: zurück, zurück
25 <532157>E: tugonline: zurück: tugonline
26 <542510>E: tugonline: institutionsseite mbi
```

Figure 3.16: Proposed Dual Head Setup for Coding with Transana on the Analysis PC.

Anvil can not synchronise different video files. This problem can be bypassed if the recording is done with a combined video and screen file as described in Section 3.3.4. One drawback with this solutions are the resulting large files thus the hardware requirements for using Anvil effectively without much waiting and response time for the coding process are relatively high.

Anvil does not support direct statistical analysis however with the open architecture from Anvil using XML files for the annotation of data an export and further processing is easily possible. Either the annotation XML files are directly used and combined for analysis and generating overall results from all coding sessions or the export feature from the Anvil Project tool is used instead. In this case the combination of the files has not to be done from the analysis software.

At first almost Transana was elected as annotation tool for the FUSS-B study. The possibility of changing the coding taxonomy right in the application was rated as an advantage over Anvil, where this definition of the taxonomy had to be done before and cannot be changed directly within the application. The necessary events used for coding can be defined in Transana with the help of keywords. A possible mapping from Transana keywords to the FUSS-B taxonomy is represented in Figure 3.17. The necessary hierarchy can be implemented in Transana through summarising the different keywords into groups. Some promising test encodings already had taken place with Transana.

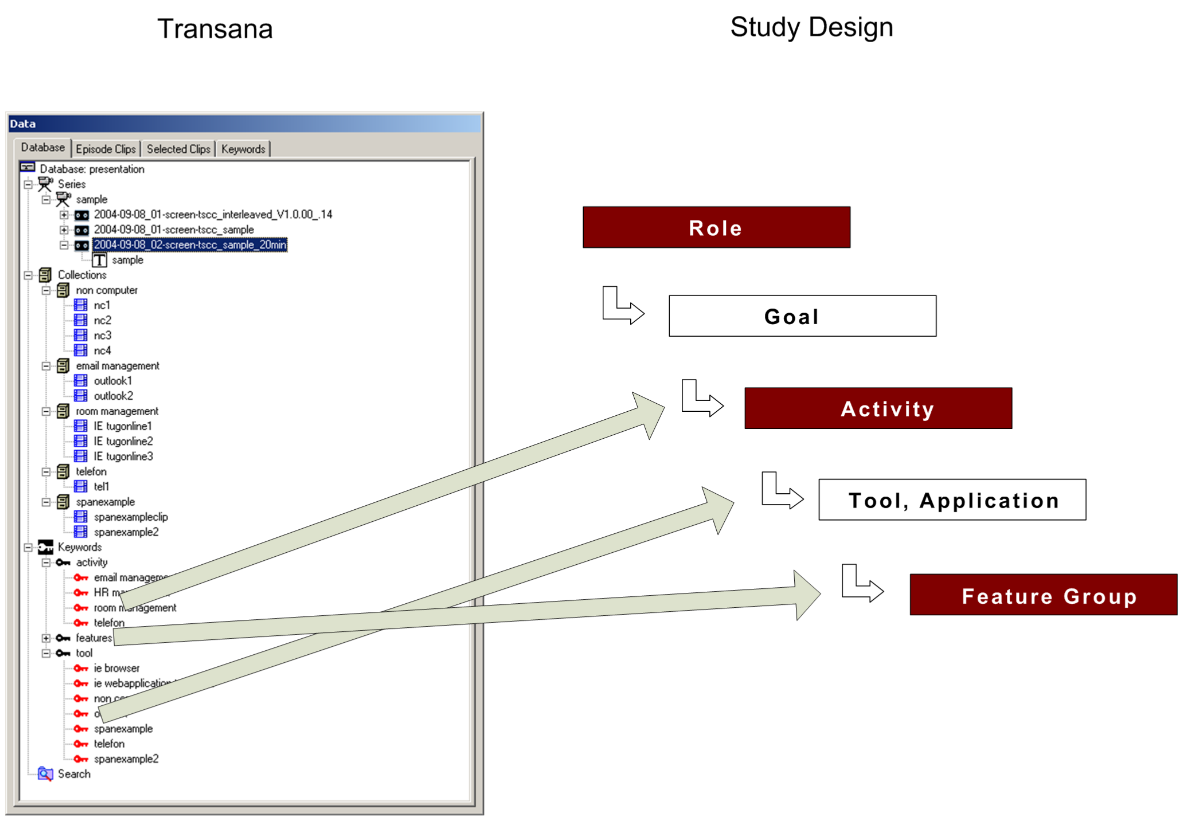


Figure 3.17: Proposed Coding and Taxonomy with Keywords in Transana.

One drawback of Transana is the not intuitive user interface regarding the requirements of this study. Transana is designed primarily as a linguistic transcription tool of spoken speech and although Anvil can import data from several speech transcription tools it is particularly designed as an annotation tool of videos. In Transana many mouse clicks and drag-and-drop events are necessary during the coding process and there is no good overview of the annotation available. Anvil has a very intuitive interface during the coding process by means of the zoomable timeline view. This timeline view also enables a good overview from simultaneous events via the multiple parallel tracks. So the idea of using Transana as coding tool in this study was finally abandoned in favour of using Anvil. Regardless of the used coding software application a dual head setup on the analysis PC is proposed for being able to deal with the high resolution recorded user screen. Such an example setup is shown in Figure 3.16 for transana. A similar setup is finally used in this study for the coding process with the Anvil application. The window of the user is viewed in full resolution on a second monitor for being able to read the actual screen contents during the coding process.

3.6.5 Analysis

For the realisation of the analysis process it was decided to implement a new software tool from scratch. This tool is called FUSSY and is implemented as part of this thesis. A verbose description covering FUSSY can be found in Chapter 5. The Anvil generated XML annotation data files are imported, processed and analysed. The results are charts regarding the defined taxonomy. On overview about the used software applications and features should be given. Of course it would also be possible to use the Anvil project tool to summarise the different annotation files and afterwards using software tools like SPSS or spreadsheets for generating charts. Nevertheless it was decided to design the FUSSY application since it is more flexible and it can be applied automatically on given result files. There is no need to us the Anvil software to export the data and to use other software tools afterwards to generate the results. The results can be generated very quickly without having the necessity to use *Graphical User Interfaces* (GUIs). If the study continues with more recorded users it will be immediately possible to analyse new annotations.

Chapter 4

The FUSS-B Study

This chapter presents the chosen design for the implementation of the FUSS-B study. An important goal of the FUSS-B study is to reveal how users actually work with the computer in different work task contexts. It should be analysed what goals users have to achieve in their daily work. It is explicitly not a goal of this study to observe the efficiency or productivity of the administrative assistants. The goal is to gather information about how software is actually used. This is a shared chapter and written together by Sandra Brueckler and Günter Modes.

The screen of the User *Personal Computer* (PC), an audio signal and a video from the environment of the user is recorded by means of a camera in the first step. The necessary recording equipment is described and configuration settings of the different software applications are explained. Tips are given for being able to avoid possible sources of problems. Two checklists are presented covering the necessary steps for starting a recording session with a test participant.

The recordings are evaluated manually. For this coding process the Anvil tool is used. The recorded videos from the test participants are coded with the help of this software. The results will be created, using the defined taxonomy, after the coding process ends. In the case of the FUSS-B study the results are diagrams. For the analysis stage the FUSSY tool, described in Chapter 5, was developed. This tool imports all coded Anvil files and creates charts with the results. These result charts can be found in the Chapter 6. Figure 4.1 outlines the FUSS-B coding and analysis process with the used software applications including file formats of their interfaces. The necessary software applications for all the different phases including the recording phase are covered and outlined in Figure 4.5.

4.1 Recording

The recording uses a setup as shown in Figure 4.2. On the User PC an installation of the Ultra *Virtual Network Computing* (VNC) Server application is obligatory. Also the high performance video hook driver could be installed for better performance. The study is designed for minimal possible impact on the User PC hence no other software is required on the User PC.

The screen and the video with the environment of the user is recorded on the Recorder PC. On the Recorder PC an installation of the Ultra VNC Client is necessary. Additionally the Techsmith Camtasia Recorder with the included *Techsmith Screen Capture Codec* (TSCC) is used on the Recorder PC. Codecs are described in the Section 3.3.4. The Windows Moviemaker is used for video previews and is normally already installed as part of the operating system.

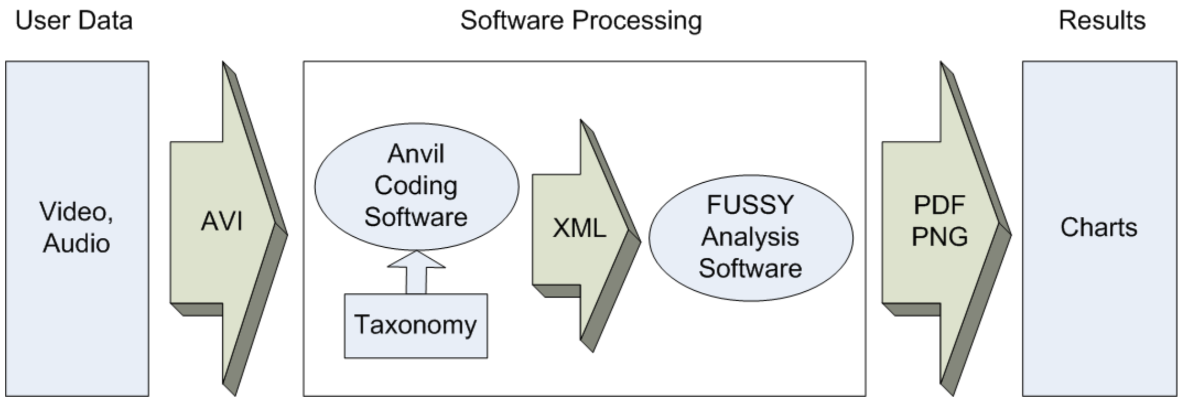


Figure 4.1: Overview of the Coding and Analysis Process of the FUSS-B Usage Study.

The screen is forwarded with the VNC server and client application and the video of the user is previewed in the Windows Moviemaker application. Both videos are recorded together with the Camtasia Recorder application. At first the TSCC is used as codec for the recording of the screen contents because with this codec this is possible in realtime. The file size of the files originating from this recording process are large thus a conversion to another format is done afterwards. As already stated in Chapter 3 video segments of at most one hour and 30 minutes can be recorded. If the two GB limit is exceeded the Camtasia Recorder software will crash.

4.1.1 Hardware Equipment

The Recorder PC is chosen being a laptop for having the necessary setup highly portable. However the hard disc space of the laptop is not sufficient hence an external hard disc is used for the actual recording. This hard disc can be connected with the *Universal Serial Bus* (USB) interface. The initially used video camera with tripod was to intrusive and thus disturbing the recording process. It is not necessary to have a high resolution recording of the user's environment thus it is decided to use a small webcam for recording the user's environment. This webcam uses the USB interface.

Another very important data is audio data. Therefore a microphone is connected into the microphone input jack of the laptop. Although audio is not recorded in high quality this recorded audio track is essential during the ongoing coding process. If only a screen recording was made and for example a telephone call was occurring it would be very difficult to get an idea of the underlying activity and goal. With the used setup of recorded screen, webcam and microphone information it was almost always possible to guess the actually activity and goal.

The Recorder PC is a Fujitsu Siemens Amilo M 1420 Centrino laptop with Pentium-M 1.7 GHz CPU, 60 GB hard disk, 512 MB RAM with a resolution of 1280 by 800. As operating system a Windows XP Home Edition with Service Pack 2 is used. The webcam is a Creative Video Blaster Webcam III USB with a resolution of 640 by 480 pixel. The microphone is a Creative Telex microphone. This microphone comes free with most Creative Labs sound cards. The external hard disk is connected via USB and has 160 GB capacity. The whole necessary hardware equipment is quite unintrusive as shown in Figure 4.3 and Figure 4.4.

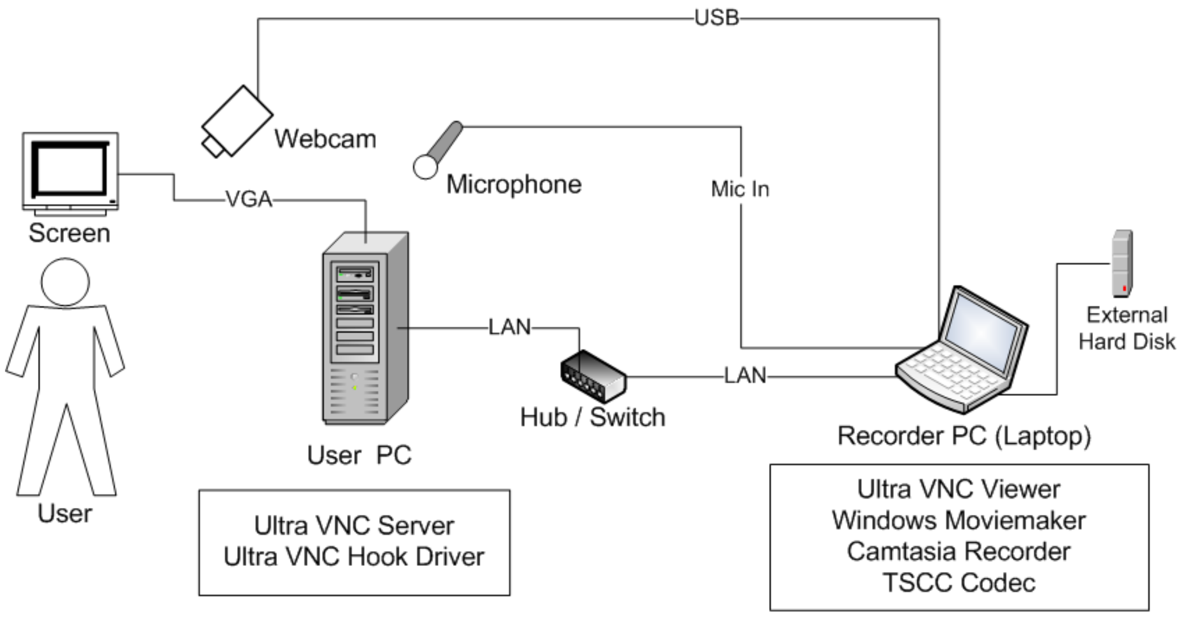


Figure 4.2: FUSS-B Software-Based Recording Setup. A Small Web Camera and a Laptop is Used as Recorder PC.

4.1.2 Practical Recording Tips

“ *Everything that can possibly go wrong will go wrong.* ”

[Major Edward A. Murphy, Jr.]

Practical experiences during the first test and the study recording sessions showed many possible sources of errors. The realisation of a recording session requires planning efforts concerning the technical setup. A Recorder PC has to be available, a webcam and a microphone. Several cables are used. At least line cords for the laptop and the external hard disk, an USB cable for the webcam, another USB cable for the external hard disk, a network cable for the connection of the Recorder PC to the LAN and a cable for the microphone are used. All cables must have sufficient length! Additionaly it is maybe handy to have a power distributor available.

For the connection to the LAN an unique IP address in the address range of the LAN is necessary. A network connection must be possible. *Registered Jack 45* RJ-45 connectors for *Unshielded Twisted-Pair* (UTP) cables are nowadays the most used technology in LANs therefore either an already patched and free jack has to be available or the User PC is connected with the Recorder PC by means of an ethernet switch like shown in the Figure 4.2. To be on the save side a switch should always be available.

One test participant of this study surprisingly still uses the old ethernet technology with coaxial cables and *Bayonet Neill-Concelman* (BNC) connectors. Fortunately a PCMCIA card for this technology was available and after successful installation of the necessary drivers the connection to the network was possible. This old technology only supports 10 Mbit/s instead of the 100 Mbit/s or even 1000 Mbit/s supported by up-to-date LAN technology. However now problems were observed regarding stability or performance during the recording session.

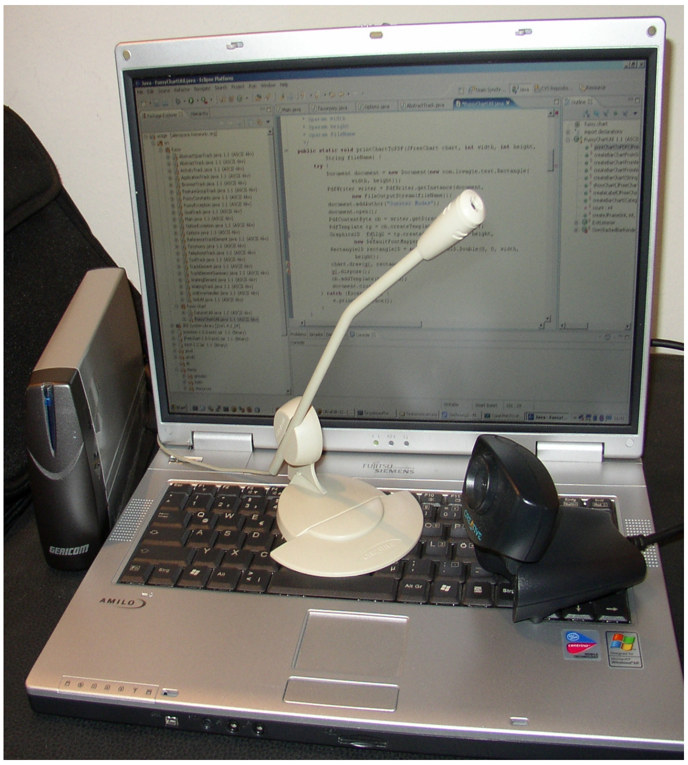


Figure 4.3: The Recording Hardware Equipment is Highly Portable.

Once the microphone cable was not connected to the microphone input but to the line-out jack. With the used laptop fortunately audio is recorded anyhow in this situation by means of the built-in microphone. However the quality is very bad because a permanent noise is caused by the fan. The fan is always running because the computer is under heavy load caused by the VNC Viewer and particularly by the Camtasia Recorder saving around one GB data every hour on the external hard disc.

Another incident caused the loss of some video material. A test participant wanted to leave the work place for some minutes hence automatically activated the screen lock. This interrupts the VNC connection from the server to the client thus killing the VNC client application on the Recorder PC. If the user deactivates the screen lock afterwards no forwarded screen will be recorded anymore. That is why one recorded file shown in the Table 4.1 only has a duration of around 22 minutes. The initially recorded length was over an hour.

Two checklists are developed to minimise sources of potential errors and problems. Some of the mentioned problems really occurred during the recording sessions thus recorded material was lost or degraded. An example of degraded material is a wrong connected microphone cable.

4.1.3 Checklist for User PC

For the initial installation of the Ultra VNC Server software and optionally the video hook driver administrator privileges on the User PC are necessary. If the software is already installed normal user rights will be sufficient for carrying out recording sessions.

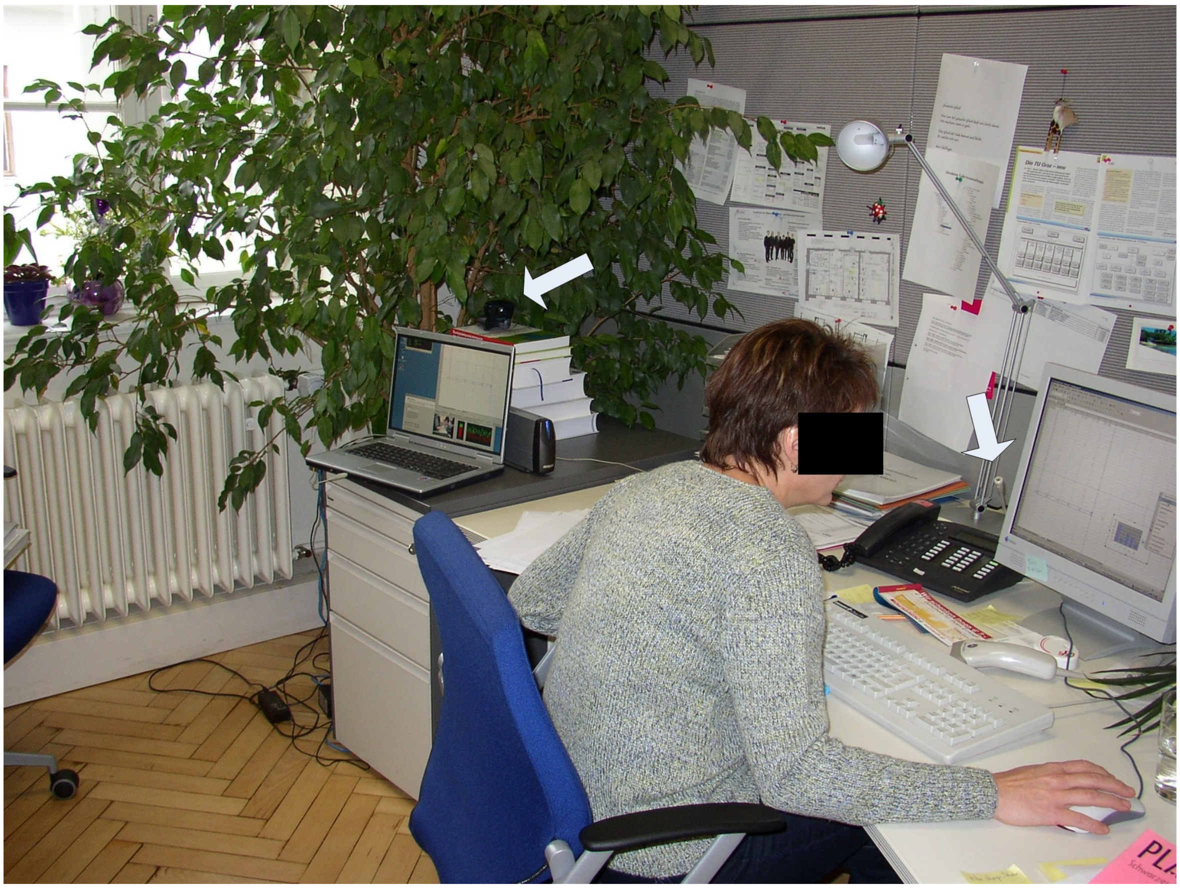

[redacted]

Figure 4.4: The Recording of a Test Participant for the FUSS-B Usage Study. The Used Recording Equipment is Quite Unintrusive. Arrows are Pointing to the Webcam and the Microphone for an Easier Recognition.

- The Ultra VNC Server application has to be installed.
- The VNC Server is started. A password is chosen and the following configuration settings are used:
  - Activated Options: Poll Full Screen, Poll Foreground, Poll Window under Cursor
  - Deactivated Options: Poll Console, Poll on Event
- If a firewall is active the TCP port 5900 has to be opened. This is the default port of the VNC Server application. Administrative privileges could be required for being able to do this.
- The standby mode has to be deactivated otherwise the VNC connection to the Recorder PC is lost whenever the computer activates the standby mode.
- The colour depth of the display settings is set to 16 bits. Thus the VNC server uses less system resources.
- Due to performance reasons an active virus scanner may be deactivated during the recording process.

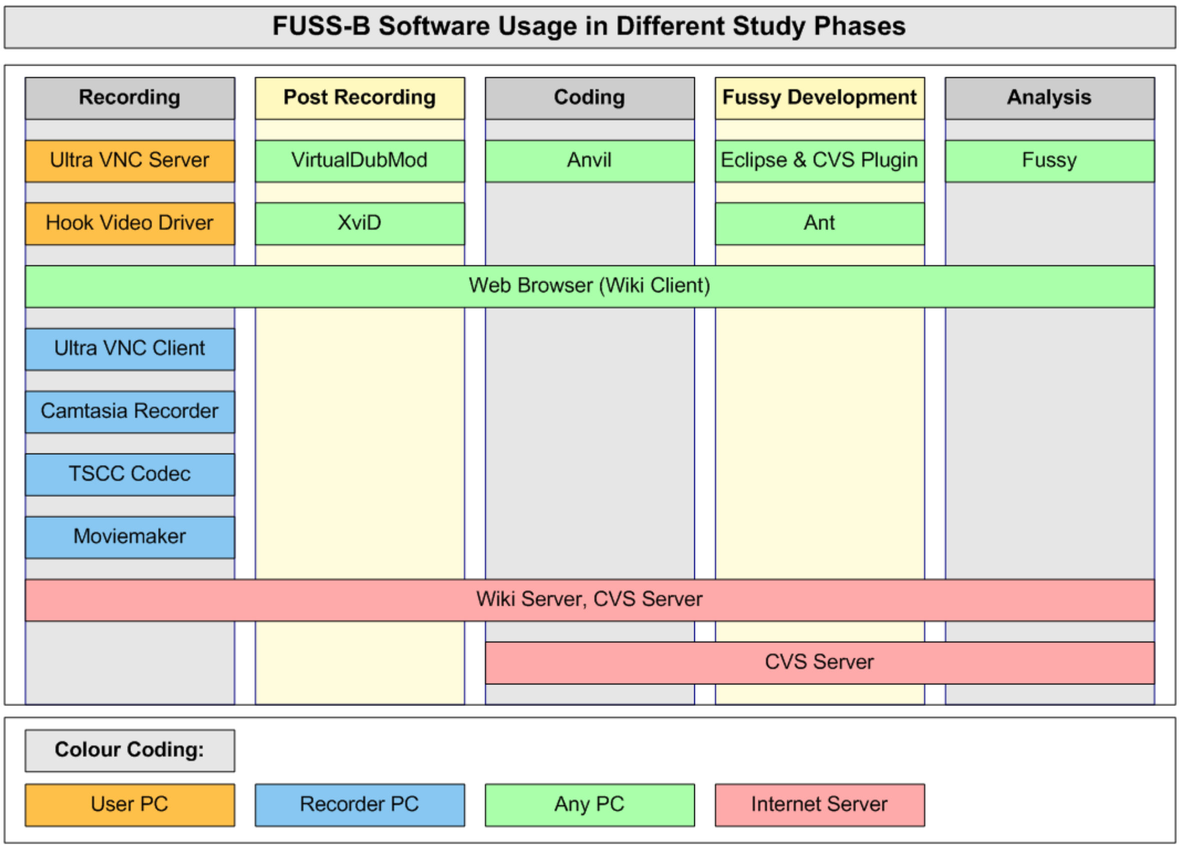


Figure 4.5: An Overview of the Necessary Software in the Different Phases of this Study.

- The hardware acceleration setting of the display driver should be deactivated. Software for capturing screen contents such as screen recording software and VNC server software will use interestingly less resources if the hardware acceleration is deactivated.
- A possible background image should be removed during the recording session. VNC will use less computer resources and network bandwidth if no background image is used.
- The user has to be alerted not using the lock screen feature of the operating system. If the screen is locked the VNC connection will be disconnected.
- The microphone has to be connected to the microphone input (mic-in) jack of the laptop. This should be double-checked.

4.1.4 Checklist for Recorder PC

Camtasia stores initially the recorded file in a temporary location. After the recording is stopped the audio interleaving will take place and finally the file will be copied to the chosen location. Due to the very large file sizes it is suggested to use the same hard disk partition for temporary and for the final location. Thus the final action is a simple file rename process instead of copying GBs of data allowing the saving of time.

- The standby mode has to be deactivated otherwise the VNC connection to the User PC is lost whenever the computer activates the standby mode.
- The screensaver has to be deactivated. This is definitely necessary otherwise only the screen saver is recorded on the Recorder PC.
- The hardware acceleration setting of the display driver should be deactivated here on the Recorder PC too.
- It is switched to a higher virtual screen resolution. If the User PC has 1280 by 1024 pixel a virtual resolution of 1792 by 1344 is recommended.
- The colour depth of the display settings is set to 16 bits. Thus the recorded files from Camtasia are half the size as with 32 bits setting.
- Similar to the User PC due to performance reasons an active virus scanner can be deactivated during the recording process.
- In the VNC Client configuration the following settings are activated: View Only (!), No Toolbar, Let Remote Server deal with cursor
- Camtasia is configured with the following settings:
  - 7 Frames / Second, Activate Audio (mono), Codec: TSCC
  - The directory for saving the recorded file is chosen. Enough space must be available.
- The Moviemaker application is started enabling the preview of the user video.
- The different application windows has to be layouted prior the recording is started. These applications are the VNC Viewer, the Moviemaker and optionally the Task Manager can be started to have the system load during the overall recording process available.
- The Camtasia Recorder application is started with a chosen recording region. The region should cover all of the VNC Client Window and the Moviemaker video preview. Due to the used virtual resolution is is necessary to scroll the screen area.
- During the recording the mouse must not be moved into the recorded region.
- It is not recommended to start any application on the Recorder PC during the ongoing session because the opened application window will most probably be located inside the recording region.
- Also the used temporary directories must have enough space available.
- The recording has to be stopped before the recorded Camtasia file reaches 2 GBs file size this corresponds approximately to 1 hour and 30 minutes recording time.

It is demanded to deactivate the hardware acceleration although the default setting of Camtasia does this automatically every time a recording starts. However the Moviemaker application will crash if the Camtasia Recording is stopped in this situation because of the reactivated hardware acceleration. Therefore it is better to deactivate the hardware acceleration manually.

Filename	**Time** [h:min:sec]	**Resolution** Width x Height	**TSCC Size** [KB]	**XviD Size** [KB]	**XviD / TSCC** [%]
2004-11-18_01	1:03:56	1284 x 1212	1,583,226	287,302	18.1
2004-11-18_02	1:00:12	1280 x 1216	1,454,046	251,420	17.3
2004-11-22_01	1:08:40	1552 x 1028	1,528,135	303,558	13.3
2004-11-22_02	1:02:37	1528 x 1028	1,404,647	267,390	19.0
2004-12-07_01	1:17:42	1284 x 768	1,630,209	349,028	21.4
2004-12-07_02	0:53:13	1280 x 772	1,118,842	257,160	23.0
2004-12-13_01	1:00:00	1404 x 872	1,454,245	308,854	21.2
2004-12-13_02	0:22:28	1404 x 868	547,728	134,096	24.5
2004-12-13_03	0:30:00	1400 x 868	750,023	158,524	21.1
2004-12-13_04	0:44:20	1392 x 864	1,088,953	213,606	19.6
2004-12-15_01	1:10:13	1268 x 772	1,470,803	292,902	19.1
2004-12-15_02	1:07:34	1260 x 768	1,424,860	325,258	22.8
Sum / Average	**11:20:55**		**15,455,717**	**3,149,098**	**20,4**

Table 4.1: Recorded Files, Durations, Resolutions and Filesizes Used for the FUSS-B Study.

4.1.5 Recorded Files

The Table 4.1 shows the file sizes, resolutions and durations of the recorded files. The overall duration of all recordings is almost 11 hours and 21 minutes. The recorded videos are converted with the VirtualDubMod tool from the lossless TSCC codec to the lossy XviD codec thus gaining much smaller file sizes [Abreu et al., 2005].

These differences are also outlined in Table 4.1. The file sizes with the TSCC added together equals over 15 GB in contrary the sum of the same files converted using the XvID codec is around 3 GB. Although The XviD encoded files still have a very good quality they only have the fifth of the size. A description of the VirtualDubMod software and an evaluation of different codecs is described in detail in Chapter 3. An overview of the different software tools necessary for the FUSS-B study is outlined in Figure 4.5.

4.1.6 Non-Technical Issues

The technical side is one aspect, on the other side consultations with the test participant and the participant's management are also very important. Security considerations of system administrators and management have to be taken into account. The consent form [Brückler, 2005, Appendix B] has to be discussed and signed by every test user and even more important by every FUSS-B team member including the persons which will perform the later coding and analysis work.

Also psychological effects should be taken into consideration. Most probably the test user is behaving in another way during being observed. Persons will alter their behaviour if they know that they are observed. This effect was discovered by Mayo [1933] and is called *"Hawthorne Effect"* since it was discovered in a research project of the Hawthorne Plant in Cicero, Illinois.

To keep this effect as small as possible we refrained from inviting the user to think aloud. During the first recordings it was tried to get more information by asking the user to fill out a spreadsheet showing the actual goal. However it was decided that the distraction of the user was too strong.

4.1.7 Limitations

This proposed recording setup has some limitations. An extensive discussion can be found in Chapter 3. Continuous recording is only possible for around 1 hour and 30 minutes. After the recording ends the interleaving of the audio track in Camtasia will last for around 6 minutes if one hour was recorded. Another minor drawback is that the background image should be removed thus perhaps annoying the test participant for changing the familiar environment.

Security has also some flaws which could make problems in larger LANs. A secure password for the VNC server should be chosen. Other persons should not be able to connect to the User PC and watch the user screen contents. Another problem may arise with the deactivated virus scanners on both computers. This could be a thread especially for the User PC. It is any time possible to get a virus or other malicious programs by email or other ways over the network. If the performance was enough it would be considered to not deactivate the virus scanners.

4.2 Taxonomy

Usage and usability studies often have some observational data in form of video, audio and other formats. After the raw material is retrieved from the users, the material will be postprocessed. First it is coded with a given taxonomy. The coding process is the application of the taxonomy on the recorded material.

The FUSS-B taxonomy is a hierarchical classification. Actions and events have defined start and end points in the timeline. These time segments are the basic parts of the coding. These segments are called feature groups. Feature groups describe a set of functionalities of certain software tools. *Browse Content* or *Text creation and formatting* are feature group examples. Feature groups belong to different tools such as wordprocessing, spreadsheet or emails tools.

Each tool has one ore more specific implementation instances, the applications. Examples are the MS Outlook, The Bat or Mozilla Thunderbird applications. The next hierarchy in the taxonomy are activities which are a sequence of used applications. Activities are contained in the highest level of the FUSS-B taxonomy, the goal level. Such goals are Teaching Support, Accounting or Staff Support. An important feature group is *Non-Computer Work*, because this feature group allows to see the fraction of computer work of the overall working time. The taxonomy will also consider waiting times in front of the computer if an application or the network connection is slow. This is an example where an typical usability issue is covered. More information about the used taxonomy for this study can be found in Brückler [2005].

A coding manual containing specific guidelines considering the FUSS-B taxonomy is available [Brückler, 2005, Appendix A]. Coding manuals are very important for the coders to keep a consistent view on their work for themselves and on the other hand to achieve consistency between different coders. The recordings for this study are coded manually. The coding is a very time-consuming process. In general it needs up to ten ten hours to work through one hour recorded material. The coders also worked together in teams of two persons to prevent differences in the application of the taxonomy.

4.3 Anvil Coding Software

For the analysis of time based observational data the term *Exploratory sequential data analysis* (ESDA) was introduced [Fisher and Sanderson, 1993, 1996]. There are many commercial and non-commercial applications available for ESDA. More and more research is done in the field of an-

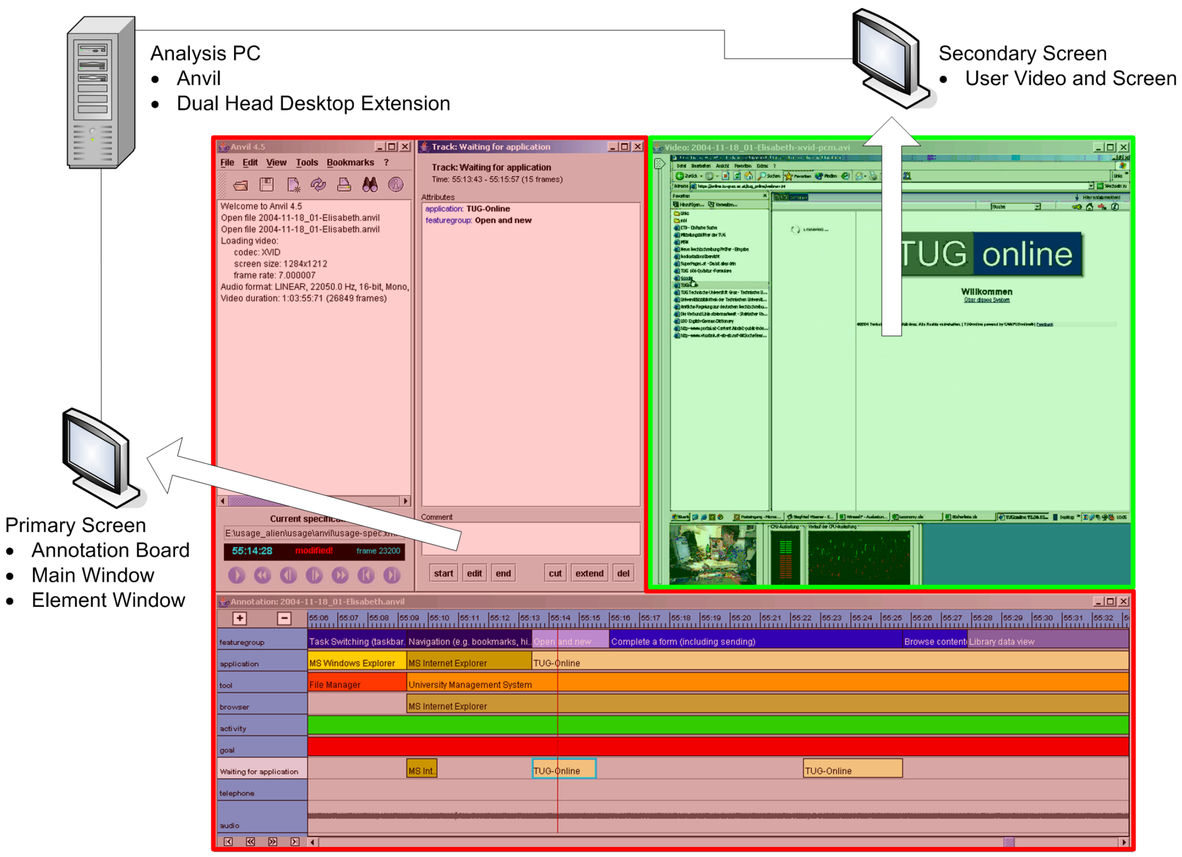


Figure 4.6: Dual Head Setup for Coding with Anvil on the Analysis PC.

notation of digital multimedia (for example video, gesture and audio) data files, this is also called multimodal data annotation.

Several annotation software tools for linguistic research and gesture analysis are available. Software used in this area has similar requirements as this study. The process of coding video data in the linguistic research area of spoken languages often also contains the transcription of the spoken language. Thus the coding process sometimes is also called transcription or annotation.

Several requirements have to be met by the coding software. A freely definable hierarchical coding scheme with possible iterative changes should be supported. The handling of simultaneous events which overlap in time must be possible. The coding software should be able to export the data in an exchangeable format for analysis. Direct jumps to video scenes where special coding events happened must be possible. Interesting parts of the recording session should be found quickly. The user interface should be generally easy and quick to use. In the evaluation of different software applications in Chapter 3 finally the Anvil software was chosen.

Anvil is a generic research tool for the analysis of digital data. Anvil stands for *Annotation of Video and Language Data*. It is a powerful software application for areas like linguistics, gesture research, human computer interaction and others and freely available for research purposes. It is developed by Kipp [2004a], University of the Saarland, Germany and is freely available for research use. Anvil provides support for multiple parallel tracks for video annotation.

Anvil is Java based and is actually available in version 4.5. Anvil is designed and coded object-oriented. The top object is an annotation. The annotation depends on a coding scheme which is defined in the so called Anvil specification file. This specification must be defined in terms of hierarchical tracks. The tracks for themselves contain several track elements also called the annotation elements. The track elements can hold a number of attributes. This provides an almost freely definable taxonomy for annotation. It is also possible to include documentation and colour specifications in the specification file.

There are primary and secondary or reference tracks. Primary tracks consist of primary elements with defined start and an end points. Secondary tracks contain secondary (or reference) elements pointing to specific primary track elements. The for this study used secondary track type is the so called span track. Every element of this span track spans over the time period covered by the two defined primary start and end elements.

The user interface consists of a Main Window, the Video Window, the Element Window and the Annotation Board. To be able to use the high resolution screen recording files without scaling down the video window a dual-screen (dual head) setup is strongly recommended. A screenshot with a proposed setup using two monitors is shown in Figure 4.6. The Main window includes the toolbar with icons and contains other information like the video file format. Video controls are also a part of the Main Window.

The Annotation Board is the most important window of the Anvil application. The actual coding takes place in this window. A time aligned view with all tracks containing their elements is given. Each element is displayed as a coloured rectangle in the Annotation Board on a kind of timeline. Simultaneousness is clearly visible.

The Element Windows gives some information about the actual selected element of the currently active track. Also new track elements can be added or already existing elements created with this window. In the Video Window the video can be played or the currently position is shown with a still picture.

If the taxonomy is iteratively changed during the annotation, the already coded parts will not correspond to this new taxonomy. The necessary changes can be done either manually in Anvil or manually in the XML file. Anvil offers a search but no replace function within tracks. Nevertheless it is very easy for example to rename an element because the Anvil annotation XML files are text based thus a simple text based search and replace can be done.

In summary although Anvil was particularly created as tool for linguistic and gesture research, it is a very suitable tool for human computer interaction studies. Anvil supports amongst others AVI video files with XviD codecs. Statistical analysis of the data is not supported instead analysis can be performed by directly working on the Anvil annotation files or using the included export features. The highly intuitive user interface enables an efficient coding process. However Anvil requires at least 1024 MB RAM to work efficiently with the large files of this study. The used hardware for the coding stage consists of two different computers. The first is a Pentium IV, 3.2 GHz, 1 GB RAM, the second PC is a AMD64 3500+, 2.2 GHz, 1 GB RAM respectively. Both computers have a Windows XP Professional Service Pack 2 operating system.

4.4 Analysis

For the realisation of the analysis process it was decided to implement a new software tool from scratch. This tool is called FUSSY and is implemented as part of the thesis. A verbose description covering FUSSY is found in Chapter 5. The FUSSY application is command line application without *Graphical User Interface* (GUI).

Anvil coded data in respect to the used software applications, the used features of these applications and other data such as waiting times should be analysed. Non-computer tasks should be considered too. Although waiting times and non-computer time is analysed it is not the intention of the FUSS-B study to rate or benchmark the efficiency of individuals.

FUSSY imports the Anvil generated annotation data files. All data is put together and result charts regarding the defined taxonomy are created automatically. Finally FUSSY creates result charts. These charts can be viewed on the screen or saved into files in different formats.

4.5 Summary

The FUSS-B study design is outlined in this chapter. The mainly software based recording setup including the necessary software applications are described. A highly portable laptop based solution is presented. Practical tips for easing the realisation of recording sessions with test participants are given. Possible pitfalls including their avoidance and checklists are shown.

The Anvil software for coding the recorded files considering the FUSS-B taxonomy is introduced. The features and the user interface are discussed. An overview about all necessary software applications in the different phases of the study is shown. Finally the most important features of the FUSSY software for creation of results are overlooked.

Chapter 5

FUSSY Analysis Software

The Anvil software is used for coding the recorded videos. The results of the coding process are stored in Anvil annotation data files. The main goal of this study is to analyse how users actually work with the computer in their everyday work. Coded data in respect to the used software applications, the used features of these applications and other data such as waiting times have to be analysed. Telephone calls and other non-computer tasks should be considered too.

It was decided to implement a software tool for being able to implement a fully automated analysis. The software is called FUSSY and is coded from scratch. The implementation of this FUSSY software is part of this theses. As coding language the Java language is chosen. It is decided to code this application instead of using already available statistical software packages or spreadsheet applications for having a greater flexibility during this analysis stage.

FUSSY imports a number of Anvil annotation files. These files originate from different recording sessions with test participants of the FUSS-B study. All of these files are coded with the same taxonomy thus every annotation uses the same Anvil specification file (see Appendix A). These files are analysed in respect to the defined taxonomy. Finally FUSSY creates result charts. These charts can be viewed on the screen or saved into files.

This chapter covers the initial requirements of the FUSSY software and describes the chosen Java language. The Installation of the FUSSY software is shown in detail for Windows and Linux. A manual is also included in this chapter. The building and deployment process such as the structure of the FUSSY application is presented and explained in detail.

5.1 Requirements for the Analysis Software

The Anvil generated annotation data files are imported, processed and analysed. The results are charts considering the defined taxonomy. An overview about the distribution of time of used software applications should be given.

FUSSY cerates automatically the result charts using a number of imported Anvil files. There is no need to use the Anvil software or any other software tools for the chart generation. These results can be generated very quickly without having the necessity to use *Graphical User Interfaces* (GUIs) at all. If the study continues with more recorded users thus having more Anvil coding files available, it will be immediately possible to analyse these new annotations. A short summary of the necessary features is given now.

- It should be possible to import several different Anvil annotation files at once.

- The data of these imported files should be analysed in respect to the defined FUSS-B coding taxonomy.
- The FUSSY application should be usable from the command line and important functions should be controllable with different options.
- The generated results should be charts showing the different aspects of the coded recordings.
- The units of the result values should either be seconds or percent.
- Different file formats should be available for storing these generated charts.
- A screen view of the generated results should be possible.
- Usage duration of applications and used features should be shown.
- Waiting periods from different applications should be analysed.

5.2 Installation

The FUSSY application has few hardware and software requirements. The needed system resources are primarily caused by the necessary Java environment. Although the software is actually only tested with Linux and Windows operating systems, the software should run on almost any operating system. The installation of the FUSSY software requires at least the following hard- and software.

- An operating system where Java is available. Almost any common operating system nowadays supports Java.
- Because FUSSY only has a command line interface available the operating system must support command line parameter. This may be a problem for example with Apple Macintosh operating systems.
- An installed JDK or JRE is required. For the development of FUSSY the JDK version 1.4 is used hence this version is at least required.
- For the 1.4 JRE a minimum of 32 MB RAM are required. Recommended are at least 48 MB of RAM.
- The installation of the 1.4 JRE uses around 75 MB disk space.
- The FUSSY software uses around 2.6 MB disk space.
- A software for the required extraction of the delivered ZIP file is necessary.

At first it has to be checked if already a JRE or JDK is installed. Subsequently the actual installation procedure for the FUSSY software consists of three steps. The FUSSY software is delivered in a single ZIP file. At first this file is extracted to an arbitrary location. Afterwards the start file has to be adapted to the chosen installation path. Finally the FUSSY application has to be found in the search path of the operating system. The installation is described for Windows and Linux computers respectively and should also be possible in a similar way in almost any other Java supported platform.

No setup application is actually delivered with the FUSSY software the installation has to be done manually. There are no additional Java libraries necessary. All required libraries are already delivered with the fussy software installation package (for technical details building the installation package see Section 5.5). The software was primarily developed on a Debian Linux computer using the Eclipse IDE and the CVS revision control system.

5.2.1 Windows Installation

Java Runtime Environment

As already mentioned an installed JRE is required. If the JRE is already installed the following command will execute successfully. For executing the command the Windows command processor (the *Command Prompt* application from the Windows *Start Menu*) is used:

```
C:\Dokumente und Einstellungen\gmodes>java -version
java version "1.4.2_09"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_09-b05)
Java HotSpot(TM) Client VM (build 1.4.2_09-b05, mixed mode)
```

When a similar outpout is given a Java VM is already installed thus Java applications can be used. Java can be downloaded from `http://java.sun.com`. The Installation of Java under Windows is quite simple with the InstallShield Wizard. The installation file is double clicked and the Java environment is quickly installed with several clicks of the *Next* button of the Wizard. Only the JRE is required for running the software. If development of FUSSY should take place it will be necessary to install the JDK. The JDK installation package also includes the JRE.

The default installation location from the JRE is the *"C:\Program Files\Java\j2re1.4.xxx"* directory using the 1.4 version. If the JDK is installed too the default location from the JDK is in *"C:\j2sdk1.4.xxx"*. The *xxx* in the filename is substituted with the actual version number. Of course also the latest stable 1.5 (5.0) version can be used. The *java* command should be found automatically after installation thus the above mentioned *java -version* command should execute successfully.

FUSSY Extraction

Any location in the filesystem can be chosen for the extraction of the FUSSY application. Examples are "C:\Program Files\fussy"or "C:\Programme\fussy"on a German Windows version. The Winzip shareware can be used for the decompression of this ZIP file. Open Source software for decompressing ZIP files is also available for Windows operating systems. One example distributed under the GNU *Lesser General Public License* (LGPL) is the *7-ZIP* (`http://www.7-zip.org/`) application.

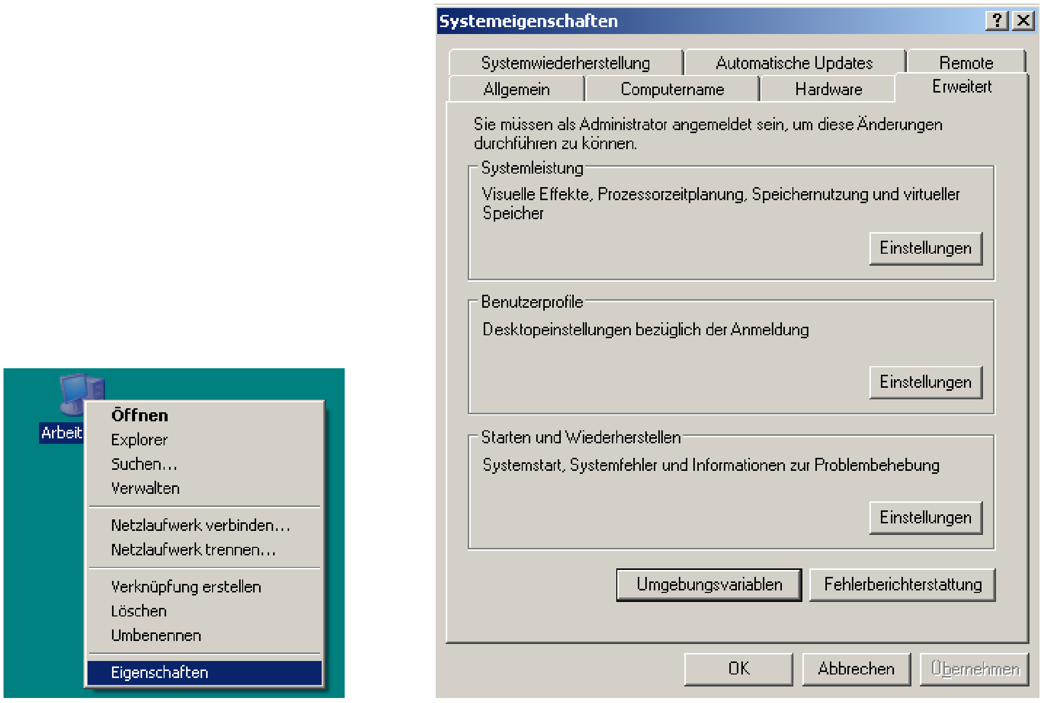
Configuring the Startup File

Similar to many other Java applications the FUSSY application is started with a batch file under Windows. The second installation step is the adaption of the contents in this file. The chosen FUSSY installation location has to be set. The short FUSSY batch file has the following contents initially:

```
@echo off
set FUSSY_HOME="."
java -jar %FUSSY_HOME%\dist\lib\fussy.jar %*
```

For being able to pass all given parameters on the Java application the special batch file variable *%** is used. With a common text editor the FUSSY_HOME variable in the second line has to be changed in respect to the chosen FUSSY installation path.

```
@echo off
set FUSSY_HOME="C:\Program Files\fussy"
java -jar %FUSSY_HOME%\dist\lib\fussy.jar %*
```



Sie müssen als Administrator angemeldet sein, um diese Änderungen durchführen zu können.

Systemleistung
Visuelle Effekte, Prozessorzeitplanung, Speichernutzung und virtueller Speicher

Benutzerprofile
Desktopeinstellungen bezüglich der Anmeldung

Starten und Wiederherstellen
Systemstart, Systemfehler und Informationen zur Problembehebung

Figure 5.1: At first a right click on My Computer is executed and Properties is chosen. From the System Properties Dialog the Advanced Tab is selected. Here the Environment Variables Button can be pressed.

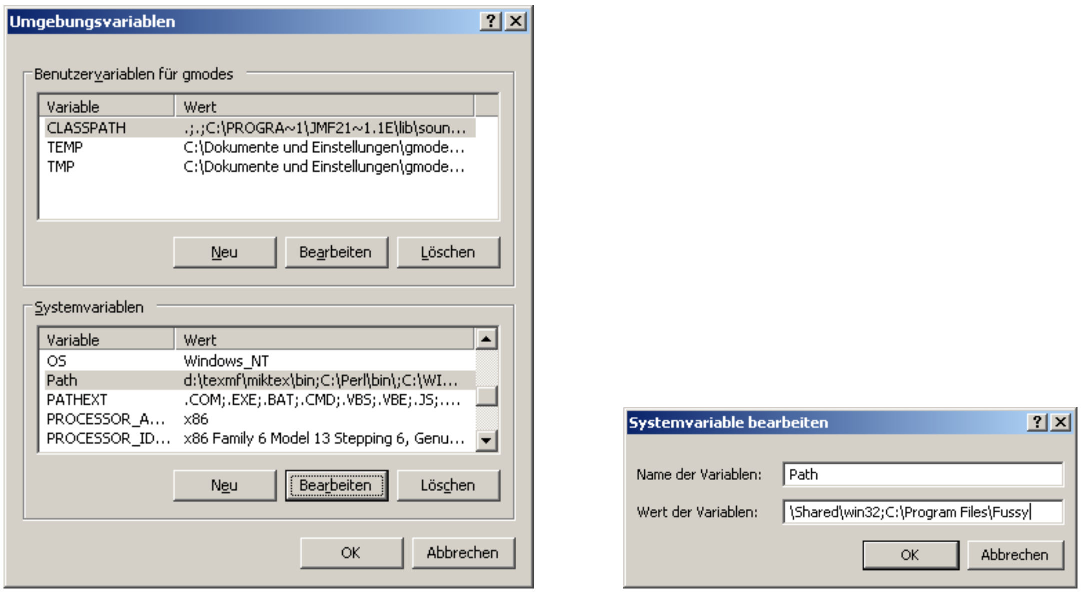
Setting Environment Variables

The third and last step for finishing the FUSSY installation process is to set the PATH variable from the environment. This is necessary for the operating system for being able to find the FUSSY startup script or batch file regardless of the actual location in the filesystem.

The definition of environment variables in Windows operating systems depends on the used version. In old Windows 95 systems environment variables have to be defined in the AUTOEXEC.BAT batch file. This file is automatically executed when the system is started. It is necessary to manually edit the file AUTOEXEC.BAT with a text editor. In Windows 98 this can be done the same way although it is also possible to use the *Msconfig* tool for this task. This tool also modifies the AUTOEXEC.BAT file. It is necessary to restart the computer before the new settings will take effect in these old Windows systems.

In Windows NT/2000 or XP systems environment variables can be changed using the GUI by means of the *System Properties* dialog. The fastest way to activate this dialog is using a right click on the *My Computer* icon of the desktop and choose *Properties* afterwards. These procedure is shown in Figure 5.1. It is also possible to navigate to this dialog with the *Control Panel*. In Windows NT the *Environment* tab has to be used, in Windows 2000 the *Advanced* tab must be chosen.

When clicking the *Environment Variables* button a new dialog opens and environment variables can be created or changed. Figure 5.2 outlines this process. This can be done for the whole system (administrator privileges required) or for the actual user. The screenshots are taken from a Windows XP computer with a German Windows version. It is not necessary to restart the computer although only new opened command line processor boxes have the changed environment available.



Benutzervariablen für gmodes

Variable	Wert
CLASSPATH	.;.;C:\PROGRA~1\JMF21~1.1E\lib\soun...
TEMP	C:\Dokumente und Einstellungen\gmode...
TMP	C:\Dokumente und Einstellungen\gmode...

Systemvariablen

Variable	Wert
OS	Windows_NT
Path	d:\texmf\miktex\bin;C:\Perl\bin\;C:\WI...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	x86
PROCESSOR_ID...	x86 Family 6 Model 13 Stepping 6, Genu...

Name der Variablen: Path
Wert der Variablen: \Shared\win32;C:\Program Files\Fussy

Figure 5.2: With the Environment Variables Dialog existing variables can be edited and new variables can be created. Here the FUSSY path is appended to the Path variable.

With the *echo* command the content of environment variables can be viewed. Many path definitions are contained in the PATH variable hence the output is shortened. The successful setting of the path variable can be tested with an execution of the FUSSY application.

```
C:\Program Files\Fussy>echo %PATH%
C:\WINDOWS\system32;...;C:\Program Files\Fussy

C:\Dokumente und Einstellungen\gmodes>fussy -v
Fussy 1.0 (c) 2005 Guenter Modes (gmodes@gmail.com)
Usage: fussy [options] [anvildir] [-h, --help] [-l, --license, --license])
```

The FUSSY application is found and thus the installation is successful! If the following error output was printed the FUSSY batch file would be found however the FUSSY_HOME variable in the startup script would not be set correctly. The Java VM can not find the necessary *fuss.jar* file in this case.

```
C:\Dokumente und Einstellungen\gmodes>fussy
Unable to access jarfile .\\emph{fuss.jar}
```

5.2.2 Linux Installation

Java Runtime Environment

When the JRE is already installed the following command will execute successfully. For executing the following commands a Linux shell is used. The default shell is the *Bash* shell. It can be started for example with the common KDE environment with the program menu choosing *System* and *Konsole (Terminal Program)*. Other terminal emulators can be chosen too. In any case a window with a command line input prompt is opened.

```
gmodes@amilo:~$ java -version
java version "1.4.2_05"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_05-b04)
Java HotSpot(TM) Client VM (build 1.4.2_05-b04, mixed mode)
```

Under Linux the installation procedure and default location depends on the used distribution. Due to Java is not free in the context of Open Source software under Debian Linux Java is not included in the standard distribution. Here the Linux self-extracting file (ending with *.bin*) distribution from Sun has to be downloaded and changed into a Debian packet with the help of the Debian ”java-package”and the included *make-jpkg* command.

Other distributions already distribute Java in their standard distribution and there are also *Red-Hat Package Manager* (RPM) packages for RPM based distributions such as Novell SuSE, Red-Hat and others available. With the following commands the location of an installed Java can be investigated. Here two symbolic links are used. The installation location can be identified as being *”/usr/lib/j2sdk1.4-sun/”*.

```
gmodes@amilo:~/fussy$ which Java
/usr/bin/java
gmodes@amilo:~/fussy$ ls -l /usr/bin/java
lrwxr-xr-x  1 root root 22 2004-09-22 14:13 /usr/bin/java
                                                      -> /etc/alternatives/java
gmodes@amilo:~/fussy$ ls -l /etc/alternatives/java
lrwxr-xr-x  1 root root 30 2004-11-30 16:15 /etc/alternatives/java
                                              -> /usr/lib/j2sdk1.4-sun/bin/java
```

FUSSY Extraction

The installation directory of *∼/fussy* is assumed for extraction on Linux. Normally in Linux the *unzip* (not the *gunzip*) command is used for extracting ZIP files thus resulting in the following output:

```
gmodes@amilo:~/fussy$ unzip fussy.zip
Archive:  fussy.zip
   creating: anvilexample/
   creating: lib/
  inflating: INSTALL
  inflating: LICENSE
  ...
```

Configuring the Startup File

Similar to many other Java applications the FUSSY application is started with a shell script under Linux. The chosen FUSSY installation location has to be set in this file. The short FUSSY shell start script has the following contents initially:

```
#!/bin/sh
FUSSY_HOME=.
java -jar $FUSSY_HOME/fussy.jar $@
```

In this initial default script the variable FUSSY_HOME is set to ”.”meaning the actual directory. With this default configuration it is already possible to start FUSSY in the installation directory. At first this script is started with the *sh* command because the file is not yet executable after extraction.

```
gmodes@amilo:~/fussy$ sh fussy
Fussy 1.0 (c) 2005 Guenter Modes (gmodes@gmail.com)
Usage: fussy [options] [anvildir] [-h, --help] [-l, --license, --license])

Could not find Anvil files in directory: /home/gmodes/fussy/.
gmodes@amilo:~/fussy$ ./fussy
bash: ./fussy: Keine Berechtigung
gmodes@amilo:~/fussy$ ls -l fussy
-rw-r--r--  1 gmodes gmodes 58 2005-09-13 14:29 fussy
```

Unfortunately it is not possible to define Unix like file permissions into ZIP files. This is no problem under Windows, because executable files are detected with their extension and not with file attributes. In Linux the file has to be set executable with the *chmod* command manually.

```
gmodes@amilo:~/fussy$ chmod 754 fussy
gmodes@amilo:~/fussy$ ls -l fussy
-rwxr-xr--  1 gmodes gmodes 58 2005-09-13 14:29 fussy
gmodes@amilo:~/fussy$ ./fussy  -v
Fussy 1.0 (c) 2005 Guenter Modes (gmodes@gmail.com)
Usage: fussy [options] [anvildir] [-h, --help] [-l, --license, --license])
```

Now the script is executable however the script cannot be started from other locations because the *Jar* file is not found in these cases. An according error message is printed out. Java cannot find the necessary *fussy.jar* file. The JAVA_HOME variable is set to the actual directory. Here the actual directory is the directory where the script is called from this the *fussy.jar* file is not found.

```
gmodes@amilo:~$ fussy/fussy
Unable to access jarfile ./fussy.jar
```

For being able to run FUSSY from every directory location on the system the FUSSY_HOME variable in the script has to be set accordding to the chosen installation location. When the assumed */home/gmodes/fussy* is chosen as installation location the variable has to be initialised in the script with this location.

It is possible to execute the script successfully from any other locations when the full path of the script is given. However it is not possible to execute FUSSY without full path because the operating system cannot find the command in the actual search path yet. Thus the final step of the installation process is to make this possible.

```
gmodes@amilo:~/fussy$ cat fussy
#!/bin/sh
FUSSY_HOME=/home/gmodes/fussy
java -jar $FUSSY_HOME/fussy.jar $@
gmodes@amilo:~/fussy$ cd ..
gmodes@amilo:~$ fussy/fussy -v
Fussy 1.0 (c) 2005 Guenter Modes (gmodes@gmail.com)
Usage: fussy [options] [anvildir] [-h, --help] [-l, --license, --license])

gmodes@amilo:~$ fussy
bash: fussy: command not found
```

Setting Environment Variables

Under Linux there are several locations where environment variables can be defined. It is possible to set environment variables for any user on the system or just for one user. The exact locations depend on the installed Linux system and the used shell. The default shell in Linux systems is normally the *Bash* shell. In Debian Linux the system-wide location of environment variable settings are defined in the */etc/profile* file. The PATH variable is now extended with the FUSSY installation path. The location for adding environment variables for a single user, when using the *Bash* shell, is normally the hidden *∼/.bashrc* file. The last two lines were added. In Linux an additional PATH location is appended with colons, in Windows with semicolons respectively.

```
gmodes@amilo:˜$ cat .bashrc
# /etc/bash.bashrc
[ -f /etc/profile ] && . /etc/profile

PATH=$PATH:/home/gmodes/fussy
export PATH
```

With the *which* command the location where the operating system executes a command is shown. The *which* command searches through all PATH locations in the same way it is done when executing the command. In this case the position of the FUSSY script is printed out.

```
gmodes@amilo:˜$ which fussy
/home/gmodes/fussy
```

Linux Alternative with Symbolic Link

Another elegant way for achieving the same goal namely finding the FUSSY application from any location in the filesystem is shown now. No files need to be changed. A symbolic link can be created instead. The link has to be placed in an already in the PATH environment variable contained directory. The already contained locations in the PATH variable can be found with the following commands:

```
root@amilo:˜# set |grep PATH
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Only the beginning of the PATH contents is shown. When the */usr/local/bin* location is selected the following commands create a symbolic link to the FUSSY shell script in this location:

```
root@amilo:˜# ln -s /home/gmodes/fussy/fussy /usr/local/bin/fussy
```

It is necessary to be the Linux super user (root) for having the rights for creating symbolic links in these system-wide important locations. The successful creation of the symbolic link can be verified with the *ls -l* command.

```
gmodes@amilo:˜$ ls -l /usr/local/bin/fussy
lrwxr-xr-x  1 root staff 24 2005-09-07 15:56 /usr/local/bin/fussy
                                        -> /home/gmodes/fussy/fussy
```

With the *which* command can be tested where the application would be started if being executed. In the case of this symbolic link solution in opposite to the changed PATH variant, the output is the location of the created symbolic link. Finally the successful installation can be tested with a very first execution of the FUSSY application from an arbitrary location:

```
gmodes@amilo:~$ which fussy
/usr/local/bin/fussy

gmodes@amilo:~$ fussy -v
Fussy 1.0 (c) 2005 Guenter Modes (gmodes@gmail.com)
Usage: fussy [options] [anvildir] [-h, --help] [-l, --license])
```

5.3 Manual

The necessary input files for FUSSY are Anvil annotation files in XML format. The most important generated results from the FUSSY applications are charts. These charts can be created in different formats. It is possible to directly view the charts on the screen. Therefore a window for every generated chart is opened. Figure 5.3 shows this functionality. The FUSSY application will exit if the last window is closed. Another possible output format are images stored in the *Portable Network Graphics* PNG format.

Although the PNG format is still not as commonly used as the *Graphics Interchange Format* (GIF) format for images, PNG is a powerful lossless bitmap image format. PNG is expected to replace the GIF format. GIF uses for the data compression a software patent protected algorithm thus GIF required a patent license to use. In the meantime the patent has expired though the GIF format has also other drawbacks such as the 256 colours limit. The PNG format even achieves a better compression for most images than the GIF format. PNG has more available transparency options including an alpha channel for defining the transparency for each pixel. PNG has a greater available bit depth with 48 bits for true colour images. There are also other advanced features included in the PNG format such as colour correction. Small animations can be stored in the GIF format, this is not possible with the PNG format. For animations the related *Multiple image Network Graphics* (MNG) format can be chosen.

A third supported output format is PDF. PDF is the native file format of the Adobe Acrobat family of products. Adobe products are available for all common operating systems. Electronic documents can be shared independent of the environment in which they were created. PDF documents can contain text, lists, tables and images. It is a very good format when it is necessary to scale the image because it is a vector-oriented format. The stored charts are considerably smaller than with PNG. The PDF files have about 10-15% the size of PNG files depending on the resolution and actual chart. Table 5.1 outlines the different file sizes for some charts in different resolutions. PDF files are used for the included result charts in the Chapter 6. The resolution has a direct impact on the PNG image format which really has the defined resolution. A vector format like PDF does not have a defined resolution. For the creation of the PDF file the resolution information is only used for the layout process.

The actual version of FUSSY is being used via command line. A short help is given if FUSSY is called with the *- -help* option. In the actual version a GUI interface of FUSSY is not available.

```
gmodes@amilo:~/fussy$ fussy --help
Fussy 1.0 (c) 2005 Guenter Modes (gmodes@gmail.com)
Fussy 1.0 (c) 2005 Guenter Modes (gmodes@gmail.com)
Usage: fussy [options] [anvildir] [-h, --help] [-l, --license, --license])

Anvil files must have the extension .anvil (case insensitive).

The possible options are:
```

Chart Name	**Used Resolution**	**File Size PDF** [Bytes]	**File Size PNG** [Bytes]	**PDF/PNG File Size** [%]
Applications	800x600	2964	21493	13.8
Tools	800x600	2946	20554	14.3
Applications	1024x768	3373	26701	12.6
Tools	1024x768	2952	25102	11.8
Applications	1280x1024	3396	30809	11.0
Tools	1280x1024	3330	30850	10.8

Table 5.1: A Comparison showing PDF and PNG File Sizes.

```
-v,    --version            show version information and usage
-s,    --show-screen        show charts on the screen (default)
-i,    --store-images       store charts in PNG images
-p,    --store-pdf          store charts in PDF files
-rWxH, --resolutionWxH      use this resolution, W: width, H:height
-t,    --time               use time in seconds instead of percent
-d,    --debug              write debug messages and store internal XML
-w,    --suppress-warnings  suppress warning messages

The minimum value for width is 640, for height 480 pixel.
If no directory is passed, the actual directory is used.
If -i or -p are not used, --show-screen is assumed.
If -i or -p options are passed, -s is not assumed anymore.
The default resolution is 1024x768

Examples:
---------
fussy (without parameter)
Use all Anvil files in the actual directory, show the charts
on the screen with a resolution of 1024x768.

fussy -p -i -r900x700 anvilcoding
Use all Anvil files from ./anvilcoding/ directory, storing results
in PDF and PNG files using a resolution of 900x700.
```

As stated in this help output, the *fussy* command can be called without any parameters. In this case the actual directory is used and the results are shown on screen (the *-s* or *- -show-screen* option is implicitly assumed). In this case also the default resolution from 1024 by 768 pixel is used. All values in these charts are further shown in percent.

All options have a short form started with a single hyphen (-), and a long form beginning with two hyphen (- -). Thus output files are stored in the PDF format when using the *-p* or *- -store-pdf* option. When an option is given for storing the images as files the charts are not shown on the screen anymore. This is also the case when the created charts are stored as PNG images with the options *-i* or *- -store-images*. If wrong options are passed to the FUSSY application an error message will be shown and the program exits.

The resolution of the generated charts can be changed with the *-r* or *- -resolution* followed by a width and height tuple. The tuple is separated with a single and lowercase *x* character. Thus a possible

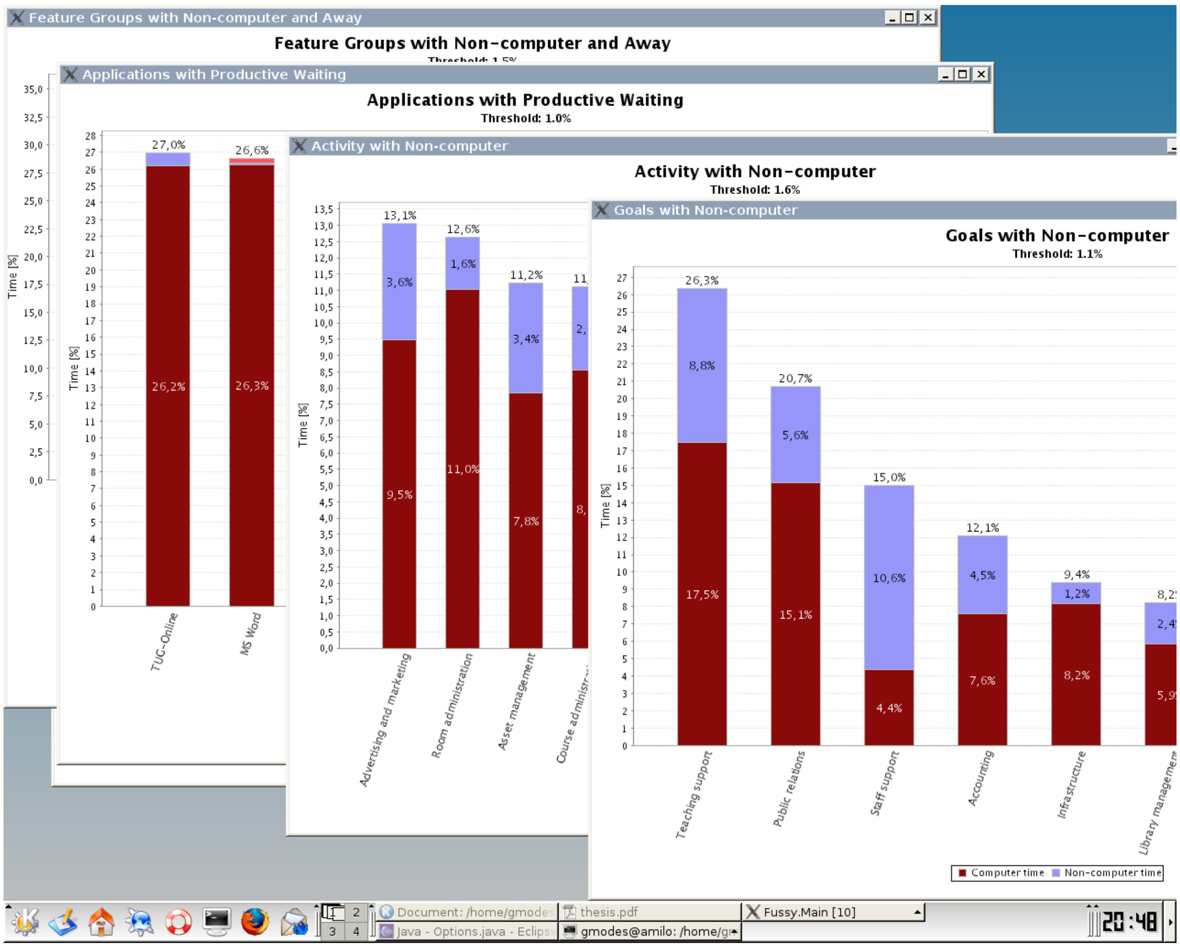


Figure 5.3: Generated Charts from FUSSY directly viewed on the Screen.

option could be *-r1280x1024*. It is possible to put a space between the -r and the width and height tuple. The minimum values for the width is 640, the minimum possible value for the height is 480 respectively.

Width the *-t* or *- -time* the values in the charts are shown in seconds instead of percent. This is particularly useful while actively developing the application such as implementing a new result chart. If the charts are created with percent values errors are more difficult to detect.

Width the *-d* or *- -debug* additional debug values can be shown in the console. And with the *-w* or *- -suppress-warnings* options, printed warnings while parsing the Anvil files can be avoided. Warnings are given if some parts of the annotation coding break rules of the defined taxonomy for the FUSS-B study. A warning is for example given if a feature group will have no corresponding goal defined. Other potential warnings include consistency problems such as elements where start equals end time or even the start time is greater than the end time. These wrong elements sometimes appear due to Anvil bugs and are simply ignored. To give an overview of the printed data when using the debug option, portions of a sample output is shown:

```
gmodes@amilo:~/fussy$ fussy --debug anvilexample
Fussy 1.0 (c) 2005 Guenter Modes (gmodes@gmail.com)
```

```
Usage: fussy [options] [anvildir] [-h, --help] [-l, --license, --license])

Using Anvil file directory: /home/gmodes/fussy/anvilexample
Showing charts on screen !
Using percent values !
Using a resolution of 1024x768

addAnvilXmlDocument: /home/gmodes/fussy/anvil/2004-11-18_01.anvil
addAnvilXmlDocument: /home/gmodes/fussy/anvil/2004-11-18_02.anvil
addAnvilXmlDocument: /home/gmodes/fussy/anvil/2004-11-22_01.anvil
...
featuregrouptrack: count: 413; Non computer work ...; time: 14156.843
featuregrouptrack: count: 300; Browse content(search, ...); time: 3632.71
featuregrouptrack: count: 287; Text creation and formatting; time: 7659.2783
...
featuregrouptrack: totalcount: 2066.0; ... totaltime [m:s]: 680:58
---------------------------------------------------------------------------------
...
created temp file: /tmp/data_42391.xml
analysis time[ms]: 1957
showing/saving time[ms]: 5928
```

With the activated debug mode at first the used parameters are shown. This includes the used directory, the chosen output formats, the resolution and whether percent or seconds are used as value units in the charts. Also the imported documents are shown with their corresponding full path. Afterwards the used feature groups including the count of their occurrences are printed. The same happens for all elements in the other tracks. These are the application, tool, activity, goal, browser, waiting and telephone tracks. All occurring elements with their respective counts are shown. In this debug mode also a temporary file is stored containing the internally used XML. The location of this stored debug file is shown and finally the needed durations for doing the analysis and showing or saving the created charts is displayed in millisecond units.

5.4 FUSSY Technologies and Standards

5.4.1 The Java Language

The FUSSY and the Anvil applications are both written in the widespread Java programming language. Java is object-oriented and originally developed by Sun Microsystems [2005b]. The development began 1990 as an internal project. Initially the language was called Oak. 1992 first demonstrations took place. The name Oak was already used as trademark thus Oak was renamed 1994 to Java. Although rumours exist Java is standing for *Just Another Vague Acronym* most likely the name originates from coffee products consumed by Java developers. Also the actual Java logo shown in Figure 5.4 supports this assumption. Java is often used in connection with the Javascript language, developed by Netscape Communications Corporation. Besides a similar syntax and the name the Javascript language has nothing in common with the Java language.

The upcoming world wide web influenced the further development. A prototype Java aware browser named Hotjava was created. The first public release of Java and the Hotjava web browser was 1995. The Java applications running in a web browser are called Java applets. The usage of Java applets was very popular for a few years though nowadays almost all interactive animations and websites are using Macromedia Flash for this purpose. Although Java applications suffer of a large overhead regarding memory consumption and used CPU power, recently Java applications gain a bit

Figure 5.4: The Official Logo of the Java Programming Language [Sun Microsystems, 2005b].

more popularity also on desktop clients. The Eclipse IDE is an example for a wide-spread popular Java desktop application.

Java always was and is still very successful on the server-side. Many different technologies exist for server-side Java applications. For websites *Java Server Pages* (JSP), Java servlets and many frameworks using this technologies as basis are widespread. On the server-side the overhead of Java applications is in many cases negligible, the advantage of being platform independent is often much more important. Other reasons for using Java on servers are many available libraries and third party applications. Many standards are developed or supported by Java such as JDBC. With JDBC it is possible to connect to almost any common third-party database with the same interface.

Software Development Kit

The API for being able to code and test Java applications is now called *Software Development Kit* (SDK), also known as *Java Development Kit* (JDK). In a *Java Runtime Environment* (JRE) Java programmes can be executed. Many improvements were made in the JDK API and Java Virtual Machines since the initial JDK 1.0 release 1996. Several new introduced features like *Just In Time* (JIT) compilation and better garbage collector algorithms improved the performance. Improvements also took place for the development of GUIs. JDK 1.4 introduced regular expressions and an XML parser is directly included in the JDK. Also non-blocking I/O is supported since version 1.4. With this new concept a number of so called *channels* can be handled with a single thread. The last stable release, originally numbered 1.5, is now numbered 5.0 and has the codename tiger. Tiger introduces many changes and improvements in the language syntax.

Improvements in the 5.0 version include the introduction of generics eliminating the need for most typecasts, autoboxing for automatic conversions between primitive and wrapper types and metadata can be added to several language constructs. The new *enum* keyword for defining an ordered list of values is introduced and an extended syntax for loops is available. Through these features a simpler, shorter and more robust code can be written. Very sophisticated libraries are already included in the JDK for the easy use of computer networks. Code from remote sources can be executed securely. Three defined Java platforms exist nowadays: *Java 2 Platform Micro Edition* (J2ME) for embedded environment, the *Java 2 Platform Standard Edition* (J2SE) for workstation client environments and the *Java 2 Platform Enterprise Edition* (J2EE) for large distributed enterprise server-side environments. All of these platforms come in a SDK or JRE flavour and are available for almost any platform including most Unix-like systems, Linux, Macintosh and Windows. Also many embedded platforms are supported with the J2ME platform.

Object-Oriented Paradigm

As already mentioned Java uses the object-oriented programming paradigm. The intent with the object-oriented language design is to make large software projects easier to manage. Also a better maintainability should be achieved. Another goal is to be able to create reusable code for usage in different projects. A good introduction and overview of the object-oriented paradigm can be found at Wikipedia [2005c]. A very important concept of object-oriented languages is the class concept. A class consists of inner variables and methods for working on these variables. A method is a piece of code similar to functions or procedures in procedural programming languages. Methods contain statements and optional a set of input parameters and also an optional return value. A class normally will have constructor methods for being able to create instances if this class to be able to create objects. Thus a class specifies the structure of data which each object has as well as the methods which manipulate the data of the object.

In reality even with object-oriented languages the software reusability have often been not as successfull as expected. Even if a language supports writting reuseable code with generic objects it will be often more difficult than it seems. Not all projects require an enterprise-level complexity. Java already has a large overhead regarding necessary lines of code and files (look at a simple Java hello world example). If the initial design of a large Java project is bad this will be even worse than writing bad designed code in other programming languages. Due to the bad maintainability of such a project it is very difficult adding new features and the complexity increases even more.

Inheritance One of the most important features of object-orientation is the concept of inheritance. An class can inherit behaviour such as attributes and methods from other super classes, and further change or extend their behaviour. While in other object-oriented languages it is possible to inherit from several classes at once this is not possible in Java. Java was designed without support for this multiple inheritance. Multiple inheritance can cause confusing situations so there is a debate whether the benefits of multiple inheritance are worth the problems. One such problem will arise if a given function is implemented in more than one super class. Which one is inherited and taken for execution? The Java designers made a compromise. Java is single inheritance language instead the concept of interfaces is provided. Interfaces provide some of the benefits of multiple-inheritance, because interfaces can inherit from more than one parent. The actual implementation of interface methods can only take place once hence the described problem of multiple inheritance languages is prevented.

Platform Independence

The same program can be executed on the different computer platforms without recompiling the code. This is often called *write once run anywhere*. The platform independence of Java is possible through Java compilers does not compile native bytecode instead an intermediate bytecode is created. These files have the extension *.class*. Where native bytecode is directly understood and can be executed by a given CPU such as Intel x86 or the Motorola 68xxx series this generated Java bytecode can be interpreted by a *Java Virtual Machine* (JVM). The JVM is compiled in a native code on the given host hardware and translates the generic Java bytecode to native instructions. The first JVMs interpreted every instruction thus having a very bad performance. Modern JVM uses the JIT technology for on-the-fly conversion from the Java bytecode to native code at runtime. The Java bytecode is compiled to native code the first time it is executed.

Real platform independence is difficult to achieve. Although it is possible to write programs hat behave consistently across many host platforms, some available platforms have errors or inconsistencies. Hence even in Java every target platform must be tested. For supporting a platform for Java it is unfortunately not only necessary having available a native VM for this platform, also an adapted JDK (API) is necessary. This is obvious when GUI based applications are taken into consideration. Most of the GUI functionality is handled in the JDK API and not in the VM. Hence for being able to access native GUI components another JDK API is necessary for a Linux based X-Window System as for a Windows platform.

Garbage Collector

With the Automatic garbage collection technology of Java it is not necessary for programmers to explicitly perform memory management as it is the case in many other languages including C++. In Java objects can be created, the necessary memory is allocated and the object is referenced in the program code. It is not necessary to deallocate memory areas, the automatic garbage collector detects unused objects, deletes them and frees the memory. Thus the infamous memory leak bugs, where memory areas are not released, can be often avoided.

With this advanced concept of garbage collecting, memory leak bugs are very seldom. References to objects are automatically freed if the actual scope is left. Nevertheless the automatic garbage mechanism makes management of objects in Java much easier and safer than in C++. One drawback with this solution is the usage of system resources from the garbage collector.

Proprietary Nature of Java

An ongoing issue with the Open Source community is the proprietary nature of Java. The source code of the JDK is available free of charge and can be viewed though the license does not permit implementing new features or bugfixes within the JDK. In Microsoft's .NET framework it is not even possible to view the implementation of the .NET classes framework. The same is true with the JVM implementation from Sun. The JVM source code is not available. Because the virtual machine specification is freely available, some other virtual machine implementations including Open Source implementations currently exist.

Nowadays specifications of the Java language, the Java Virtual Machine (JVM) and the Java API are maintained through the formalised *Java Community Process* (JCP). The JCP is controlled by Sun and allows interested parties to be involved in the definition of new features for future versions. Also this JCP is a subject of controversy. Nowadays IBM strongly supports the Open Source and also the Java community. IBM invests much money into Open Source implementations such as Eclipse and others. A powerful native widget library used in Eclipse for developing more responsive GUIs in Java is also given to the community. IBM and others have already frequently demanded Sun Microsystems for an Open Source implementation of Java. There are always rumours around though until now nothing has happened in this direction. Some features like operator overloading are unfortunately not supported in Java. In versions before the introduced generic types in JDK 1.5 many typecasting operations were necessary.

Java Summary

Many interfaces are available in Java for connecting to other applications. Interfaces for directly calling code written in other languages or over a network are available. The JDK offers many libraries for data structures, distributed development, connection to databases, processing multimedia content and applet development for running in browsers. Java also includes advanced security features, including encryption and signature. A compiler and powerful debugger is included in the JDK. IDEs like Eclipse and Netbeans and a vast amount of information and third party products, many of them are available as Open Source, can be found on the internet.

Java programs often use more memory and CPU than applications in other languages although with modern hardware and JVMs the performance of Java applications is getting better and better. On the technical side Java offers some very advanced and innovative features. Java is a robust and mature language and although some issues exist it will probably play an even more important role in different areas server- and clientside for the next years.

5.4.2 XML Standard

Since the XML file format is very important for the FUSSY application in several aspects here a short summary of the properties and features of the XML language is given. XML is a general-purpose markup language based on international standards. Many different kinds of data can be described with XML. One important purpose is the sharing of data across different systems. XML is also used as a format for document storage. The Open Office Suite uses XML as native file format and also the actual Microsoft Office 2003 is able to edit XML files with user defined schemas. Different possibilities of verification of XML exist.

On the one side it is possible to check if an existing XML is well-formed. The XML document has to fulfil several criteria for being well formed such as having only one root element, every non-empty element must have a start and end element and attributes have to be quoted.

On the other side several possibilities of definitions how the XML document has to be structured exist. Such definitions describe constraints on the structure and content of XML documents. *Document Type Definitions* (DTDs) are the oldest available syntax for this purpose. DTDs have some drawbacks such as no type definitions are possible. Hence it can not be enforced for certain XML attributes being a number instead of a string. A second shortcoming of DTDs is not using an XML syntax for themselves therefore XML parsers can not be used to parse DTDs. XML schema definitions solves some of the shortcomings of DTDs such as XML schema definitions are written in XML. XML schema definitions introduce instead other problems like a worse readability for humans.

To summarise DTDs and XML schema definitions, if an XML document complies with a particular schema it will be said to be valid. Many different tools exist to validate an XML document against a schema. The tool is able to verify whether the document conforms to constraints expressed in the schema.

Many other standards in connections with the XML standard are existing. *Extensible Stylesheet Language* (XSL) for example is a technology describing how to format or transform the data in an XML document to another format. The output format can either be again in an XML format or not. An XML document is being transformed with a XSL document to an output format. This process is also called *XSL Transformation* (XSLT). With XSL one XML document can also be transformed into several different viewable output formats in a similar way than *Cascaded Style Sheets* CSS is applied to HTML for rendering purposes. Some browsers like the actual Mozilla Firefox browser are able to render XML with XSL thus implementing XSLT and creating a viewable output. The upcoming Internet Explorer version 6 in opposite to the actual Internet Explorer version 5, is expected to implement this standard too.

Another part of the XSL definition is the special *XSL-Formatted Objects* (XSL-FO) format used in layouting XML documents. There are tools available for transforming XML documents with XSL-FO into PDF, *PostScript* (PS) or other layout formats. Despite the powerful possibilities provided by XSL it was not necessary for the FUSSY application to use XSL transformations.

XML Path Language

XML Path Language (XPath) is a query language for addressing portions of an XML document. The syntax is similar to the syntax used for *Uniform Resource Locators* (URLs) or path notations to filenames in Linux or Unix systems. The notation is has many defaults for common cases. The simplest kind of path expression takes a form such as *"/A/B/C"*, which selects *C* elements who are children of *B* elements. These *B* elements are again children of the XML root element *A*.

Several different so called axis can be defined. The axis define how the XML treeshould be navigated while querying elements. The default axis is the child axis. Many other axes are available including attributes, descendant, parent and so on. An abbreviation for using the very important attribute axis is *"@"*. Another abbreviation for using the descendant axis is *"//"*. One short example using these two axes is *"//@id"* which selects all the attributes named id anywhere in the document.

It also possible to use conditions. Conditions are written in square brackets. Many other expressions are supported. Arithmetic and boolean operators, several string manipulation functions and aggregation operations. The XPath Language is used in different crucial places of the FUSSY application and thus very important for understanding FUSSY and being able to extend the functionality for example with new analysis charts.

5.5 Building FUSSY

FUSSY uses the Apache Software Foundation [2005a] Ant build tool for the building process. Ant has similarities to the common *Make* tool used in Unix and Linux environments. Ant itself is a Java application and also primarily intended for use with Java development processes. Ant has many advantages. *Make* tools are shell-based therefore platform independence is difficult to achieve. Ant already has many supported functionalities. Ant is a cross platform tool with XML as format for configuration and for defining the build process.

One nice feature supported by Ant is the iterative compilation of Java files. During the compile process Ant compiles only necessary Java files. A Java file with an already up to date corresponding *.class* file is not compiled twice. Ant is a very advanced extensible and customisable build tool. Despite of the several hundred already built-in tasks Ant can be extended with new tasks. The already available tasks cover areas like archiving, compiling, deployment, documentation, logging, file handling, revision control systems, mailing, testing and many others. Nowadays Ant build files are widespread particularly in the Open Source Java development community. Many IDEs including Eclipse support the integration of Ant in their building process.

The default name for the files for describing the build procedure is *build.xml*. Therefore these files are normally called build files. Build files have so called targets with defined dependencies between them. Every target contains actions. These actions, called Ant tasks, will be executed if the specific target is called directly or by means of the defined dependency.

5.5.1 Ant Installation

For being able to develop, compile and build FUSSY distributions an installation of the JDK and Ant is required. The JDK can be downloaded from `http://java.sun.com`. The Ant binary installation can be downloaded from `http://ant.apache.org/bindownload.cgi`. The Ant binary distribution consists of the following directory layout [Apache Software Foundation, 2005b]:

```
  ant
   +--- bin  // contains launcher scripts
   |
   +--- lib  // contains Ant jars plus necessary dependencies
   |
   +--- docs // contains documentation
   |      +--- ant2    // a brief description of ant2 requirements
   |      |
   |      +--- images  // various logos for html documentation
   |      |
   |      +--- manual  // Ant documentation (a must read ;-)
   |
   +--- etc // contains xsl goodies to:
              //   - create an enhanced report from xml output
              //     of various tasks.
```

For an installation at least the *lib* and *bin* directories are necessary. The *bin* directory must exist in the PATH variable and the ANT_HOME environment variable should be set to the directory where Ant is installed. On Unix like and some Windows operating systems the Ant wrapper scripts can guess the ANT_HOME variable though it is recommended to manually set the variable. The JAVA_HOME environment variable should also be set to the directory where the JDK is installed. This is needed for JDK functionalities such as the important *javac Ant task* for compiling Java files to *.class* files. It this is not done the following warning message is shown when Ant is executed:

```
C:\Programme\apache-ant-1.6.5\bin>ant
Unable to locate tools.jar.
Expected to find it in C:\Programme\Java\j2re1.4.2_09\lib\tools.jar
```

Also the JAVA_HOME environment variable must be set accordingly to the place where the Java Runtime Environment is installed. The contents of this variable can be queried under Linux with the *set* connected with the *grep* command. In Windows the *set* command is available however no *grep* command is provided. The *set* command causes much output hence in Windows the existence of the JAVA_HOME variable can be checked more comfortable with the *echo* command.

```
gmodes@amilo:~/fussy$ set |grep JAVA_HOME
JAVA_HOME=/usr/lib/j2sdk1.4-sun

C:\Dokumente und Einstellungen\gmodes>echo %JAVA_HOME%
C:\j2sdk1.4.2_09
```

How an environment variable can be set under Linux or Windows is described in Section 5.2. More verbose installation instructions can be found in the Ant manual [Apache Software Foundation, 2005b]. If an execution of the *ant* command gives the following output the installation including setting the necessary environment variables is successful. Ant states not finding the *build.xml* file which is an expected error in this place.

```
C:\Dokumente und Einstellungen\gmodes>ant
Buildfile: build.xml does not exist!
Build failed
```

5.5.2 Fussy Source Code Extraction

To build the FUSSY from source the *fussy-src.zip* has to be extracted. This file is part of the normal FUSSY installation in located in the root directory. This source file is extracted directly in the actual directory. After this extraction the FUSSY directory has the following tree structure and contents (some output is removed):

```
gmodes@amilo:˜/fussy$ tree
.
|-- INSTALL
|-- LICENSE
|-- LICENSE.JFreeChart
|-- LICENSE.iText
|-- README
|-- anvilexample
|   |-- 2004-11-18_01.anvil

...

|   ‘-- usage-spec.xml
|-- build.xml
|-- fussy
|-- fussy-src.zip
|-- fussy.bat
|-- fussy.jar
|-- fussy.zip
|-- lib
|   |-- itext-1.2.jar
|   |-- jcommon-1.0.0-pre2.jar
|   ‘-- jfreechart-1.0.0-pre2.jar
‘-- src
    |-- fussy
    |   |-- FussyConstants.java
...

    |   |-- XmlUtil.java
    |   ‘-- chart
    |       |-- FussyChartUtil.java
    |       ‘-- FussyDatasetUtil.java
    ‘-- script
        |-- fussy
        |-- fussy.bat
        ‘-- runbuild

6 directories, 50 files
```

5.5.3 Using Ant

In this *fussy-src.zip* file the whole *src* directory tree including start scripts are included. In this extracted *fussy-src.zip* file the necessary *build.xml* for Ant is included too. The FUSSY *build.xml* has actually the following targets available.

- The **init** target creates the build and build classes directory used for the compile process.

- The **compile** target compiles all Java source files into the *build/classes* directory. The necessary libraries are also copied into the *build/lib* location. Afterwards example Anvil files, license and other files and scripts are are copied to their defined locations.
- The **dist** target at first creates the dist directory. In this *dist* directory the single *fussy.zip* file containing all generated files from the build is created. This file also contains the *fussy-src.zip* file which in turn contains all Java source files and the Ant *build.xml*.
- The **javadoc** target generates the FUSSY javadoc API documentation out of the annotated source file documentations. In Java this is the standard form of API documentation. After executing this target the documentation can be found in the *javadoc* directory in the HTML format.
- The **clean** target removes all created files and directories.

Ant targets can have dependencies and furthermore a default target can be specified. In the FUSSY *build.xml* the *compile* target depends on the *init* target and finally the *dist* target depends on the *compile* target. The default target is the *dist* target thus if the Ant application is called without parameter the *dist* target is called. Due to the defined dependencies in this case the *init*, *compile* and *dist* target are called automatically in this order.

```
gmodes@amilo:~/fussy$ ant
Buildfile: build.xml

init:
    [mkdir] Created dir: /home/gmodes/fussy/build/classes

compile:
    [javac] Compiling 24 source files to /home/gmodes/fussy/build/classes
     [copy] Copying 23 files to /home/gmodes/fussy/build
     [copy] Copying 1 file to /home/gmodes/fussy
      [jar] Building jar: /home/gmodes/fussy/build/fussy.jar

dist:
    [mkdir] Created dir: /home/gmodes/fussy/dist
      [zip] Building zip: /home/gmodes/fussy/build/fussy-src.zip
      [zip] Building zip: /home/gmodes/fussy/dist/fussy.zip

BUILD SUCCESSFUL
Total time: 3 seconds

gmodes@amilo:~/fussy$ ls -lh dist
insgesamt 2,6M
-rw-r--r--  1 gmodes gmodes 2,6M 2005-09-13 22:38 fussy.zip
```

Every Ant target prints out a short summary while executing different build processes. This build process creates the *build* directory structure, compiles all source files and creates the *fussy.jar* in this directory. Finally the *fussy.zip* containing all necessary files is created in the *dist* directory. With the *javadoc* Ant target the FUSSY API documentation in HTML format is created out of information contained in the Java source files.

```
gmodes@amilo:~/fussy$ ant javadoc
Buildfile: build.xml
```

```
javadoc:
    [mkdir] Created dir: /home/gmodes/usage/usage/javadoc
  [javadoc] Generating Javadoc
  [javadoc] Javadoc execution
  [javadoc] Loading source files for package fussy...
  [javadoc] Loading source files for package fussy.chart...
  [javadoc] Constructing Javadoc information...
  [javadoc] Standard Doclet version 1.4.2_05
  [javadoc] Building tree for all the packages and classes...
  [javadoc] Building index for all the packages and classes...
  [javadoc] Building index for all classes...
  [javadoc] Generating /home/gmodes/usage/usage/javadoc/stylesheet.css...

BUILD SUCCESSFUL
Total time: 9 seconds
```

With the *clean* Ant target all created files and directories can be removed. It is not necessary to execute this target if just a recompilation should take place. Ant automatically detects changed files and only recompiles these files. If no changes are made nothing will be compiled or created.

```
gmodes@amilo:~/fussy$ ant clean
Buildfile: build.xml

clean:
   [delete] Deleting directory /home/gmodes/fussy/build
   [delete] Deleting directory /home/gmodes/fussy/dist
   [delete] Deleting: /home/gmodes/fussy/runbuild

BUILD SUCCESSFUL
Total time: 0 seconds
```

The *build.xml* defines the build process. As example an XML code snippet defining the *dist* target is shown and explained. Ant commandos are called Ant tasks. Ant tasks are XML elements with other elements or attributes used as parameters.

```
        <target name="dist" depends="compile"
        description="generate the distribution" >
                <!-- Create the distribution directory -->
                <mkdir dir="${dist}"/>
                <!--Generate a single zip file from src -->
                <zip destfile="${build}/fussy-src.zip">
                        <fileset dir="." casesensitive="yes">
                                <include name="src/**"/>
                                <include name="build.xml"/>
                        </fileset>
                </zip>
                <!--Generate the distribution zip -->
                <zip destfile="${dist}/fussy.zip">
                        <fileset dir="${build}" casesensitive="yes">
                                <exclude name="classes/**"/>
                        </fileset>
                </zip>
        </target>
```

In the first line the dependency of the *compile* target is defined. The Ant file system tasks have similarities with Linux/Unix commands. The Ant task for creating directories is called *mkdir*. At first the *dist* directory is created. This is only done if necessary, if the directory already exists nothing will be done.

With the *zip* task a ZIP file can be created or updated. With the contained *fileset* task the files to be compressed can be defined. The first *zip* Ant task creates the *fussy-src.zip* containing all source files and the *build.xml* file. The second *zip* tasks creates the distribution *fussy.zip* file.

The Ant build system eases the implementation of a platform independent build process of software, particularly in the Java domain. It is not necessary anymore to write script and batch files for the build process more than once for different operating systems. Everywhere the same XML build file can be used. Due to the advanced capabilities of Ant it would even be possible to use the Ant framework not only for the build process. Ant scripts could also be used for the startup of applications thus avoiding the parallel Linux and Windows start files.

Nevertheless FUSSY does not use Ant for starting the application. This would require additional overhead since Ant has to be already properly installed on the system. For simplification of the installation process two startup scripts are provided and no additional software besides JRE is actually necessary for running the FUSSY application.

5.5.4 Development

For further development of the application an IDE is recommended. The author actually uses the Eclipse IDE (3.5.1). There are also other Open Source and proprietary IDEs available. Of course it is also possible, especially with the advanced Ant building tool, just using a common text editor for coding in Java. IDEs offer many functionalities including refactoring, integration of revision control systems, debugging and many other mechanisms. A simpler and quicker development of applications is often possible with IDEs.

While changing Java source files normally the application has to be started very often for debug and test purposes. Several different possibilities exist starting the application during development. One possibility is to start it within the used IDE. In eclipse this is easy possible, startup parameters can be passed, the virtual machine can be chosen and much more. Also debugging is supported very well by the Eclipse IDE.

Another possibility is using the *runbuild* script located in the FUSSY installation root path. This script is copied from the *src/script* directory to the FUSSY installation directory when the *build* Ant target is called. This script uses the JFreeChart and iText libraries from the *lib* directory but the FUSSY classes are taken directly out of the *build/classes* directory instead if using the classes contained in the *fussy.jar* file. If this script is called from other locations it is necessary to set the contained FUSSY_HOME variable. The Ant *clean* task removes this file so changes are lost in this case.

While developing also the normal build process can be used for running FUSSY. After changing the necessary Java source files Ant can be used for building a new *fussy.zip* file. With this file an installation is done and the application can be started the common way. Obviously this process is not straightforward thus not recommended.

5.6 Internal Structure

Anvil files from different coding sessions are imported into the FUSSY application. To be able to generate summarised results these files are put together. This is achieved with an internally created data structure. Every new imported file is appended at the end. Finally this structure contains all the data from the imported files.

In the FUSSY development the common Sun Microsystems [2005a] Code Conventions for the Java Programming Language are used. Class names start with a capital letter and method names are written lowercase. If class or method names consist of several words these words are put together every word starting with a capital first letter (such an example for a method name is *addAnvilXml-Document*, an example for a class name is *TrackElement*). The used Anvil coding specification file is attached in Appendix A. An extensive documentation of the FUSSY API is included in Appendix B.

5.6.1 Startup

When the programme is started with the start script or batch file the *main* method from the *Main* class is called at first. A version and general usage message is always printed to standard out. Afterwards the passed (if any) command line options are parsed with help of the *Option* class. Therefore the *parseArgs* from the *Option* class is called.

This *parseArgs* method in the *Option* class creates and returns a new *option* object. The available command array with the command line parameters is evaluated in a loop and the corresponding parameters in the option object are set. An *option* object is returned at the end of the method initialised with the passed startup command line parameters. If special options like showing the help message or the license are passed, the further execution of the application will be prevented via setting a flag in this option object. When a resolution option is used the passed parameters from this option are parsed. Error messages will be printed if these parameters are not numbers or out of the valid range.

It is possible to define a Anvil file directory. When a passed argument does not start with a hyphen it is used as directory argument. At first it is checked if this directory exists and whether Anvil files are contained in this directory else an error is shown immediately. After this option parsing the default values are set into the *option* object. If no files should be saved the show option is activated and when no directory is given the actual active directory is used as already stated in the Manual Section 5.3.

5.6.2 Taxonomy

As next action the most important object, the *taxonomy* as instance of the *Taxonomy* class is created. All Anvil files in the given Anvil directory are parsed with an XML parser and the created documents are added to this *taxonomy* object. All Anvil tracks from this file are added into the *taxonomy*. Anvil stores start and end time with seconds from the beginning of the coding session in primary tracks. Span tracks reference to primary elements. To be able to combine several different Anvil files an offset is always kept up to date. All newly added files are appended internally to the end of the already created structure. The *taxonomy* creates one object for every track once and adds single track elements into these tracks while adding new Anvil files considering offsets.

Every imported Anvil file contains several coded tracks. Single tracks are extracted one after another from every Anvil file by means of XPath queries during this import process. The tracks contains track elements. Every track element has a start time, end time and an index. Span tracks reference to given indices of their corresponding primary tracks. Index and time offsets are considered while adding new files. Another implicitly executed task is to create on the fly *TrackElementSummary* objects if an element name is occurring for the first time respectively adding the new track element to an already existing *TrackElementSummary* object. Thus a summary of the elements is created on the fly and used in different contexts during the later analysis process.

When the initial import of all Anvil files is done and the internal structure containing the contents of all these files is created a post processing is started. This is initiated with a single call of the *postProcessAddReferencedAttributes* method of the one and only *taxonomy* object. Attributes are added for the feature group, application and the tool track. The algorithm iterates over all elements of the corresponding track and finds the concurrent track elements of the other parallel tracks. The name of these elements are added. One example is the feature group track were also the name of the active application, tool, activity, goal and if applicable the used browser name is added. Such a feature group entry in the internal XML representation looks like the following snippet.

```
<featuregroup name="Library data view"
              start="3359.2825"
              end="3368.568"
              time="9.285645"
              index="157"
              application="TUG-Online"
              tool="University Management System"
              browser="MS Internet Explorer"
              activity="Asset management [Sachgüter, Inventar]"
              goal="Library management"/>
```

An application and a tool element in general spans several feature group elements. Hence in the application and tool tracks only activity, goal and if applicable browser name is added because more than one element could match in the feature group track and so the relation is not unique.

5.6.3 Waiting Time Handling

The application track is a secondary span track which references to the feature group primary track. An application can therefore only change if the feature group also changes at this point. Even if the waiting track had been a span track it would be possible to have another application element in the primary track as the application is just waiting for. This is possible if the feature group changed, the waiting application would change however the application would stay the same.

In the specification the waiting track is not defined as another span track but as another primary track hence being able to independently start and stop waiting within an ongoing feature group and application element. Thus it is possible while waiting for one application already another or even none application is active. While non-computer work is happening no computer application is coded. In Figure 5.5 such an example is shown in the Anvil Annotation Board. The SAP application has a longer waiting time and during this waiting time another non-computer related work is done.

A direct comparison of waiting times from the different applications is already possible when only information from the waiting track is used. More used applications will most likely have more waiting times in general, this has to be considered. One idea is to examine the waiting time in relation to the application time. A possible solution is to just divide the waiting time of an application with the corresponding application time and thus gaining a relative value. The waiting time is the duration of the considered element in the waiting track.

Questions arise what amount of time should be taken for application time. In the Figure 5.5 the time for SAP in the application track is shorter than the SAP waiting time. Often the user knows in advance which functionalities of applications normally cause long waiting times. If a long waiting process is expected by the user the application often is quickly changed or non-computer tasks are accomplished just like in this example. The application time is only very short in relation to the waiting time moreover the relative value of waiting time divided by application time would be greater than 100% here.

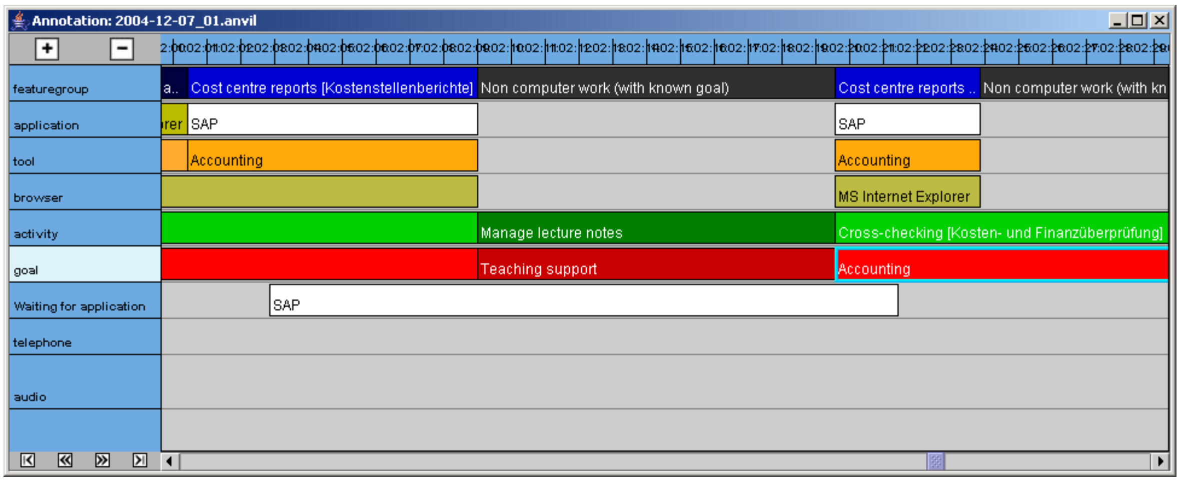


Figure 5.5: If an application caused long term waiting meanwhile other work would be done. Another application could be opened or just a non-computer work is being carried out. Here an example with the SAP application is shown.

So the concept of productive waiting time is developed. Section 6.4 in Chapter 6 provides necessary definitions and descriptions concerning different waiting times. The algorithm in the *handleProductiveWaiting* of the *taxonomy* object iterates over all waiting track elements and for every waiting track element the algorithm is applied. At first all application track elements contained within the time segment of the waiting element are acquired. Only the application elements with the same application as being waited for are taken into account. Also application elements are matched when only one time, either the start or the end time is inside the time frame of the waiting track element. For this elements only the appropriate contained time part is taken. These matching application track elements will be subtracted from the waiting time. The remaining result of this repeating substraction equals the productive waiting time for this application.

This productive waiting time is stored in the application summary elements which are instances from the already mentioned *TrackElementSummary* class. It is not possible to assign these described different time durations to single application elements because in general multiple elements are considered for the calculation of the productive waiting time. Thus the handling of the productive waiting time only is possible in summary data structures. The count of different elements is also stored in these summary elements. The count of the elements is shown if the FUSSY application is called with the debug option.

5.6.4 XML Structure

Out of the internal data structure consisting of *TrackElement* and *TrackElementSummary* objects an XML document can be created. While the XML document is again a data structure consisting of Java objects it is very easy to export this structure. When FUSSY is called with the debug option the internal XML document is stored into a file. All generated charts use data from this XML structure. This structure is shown and explained now.

```
?xml version="1.0" encoding="UTF-8"?>
<usage>
  <featuregrouptrack>
```

```
    <summary count="2066" time="40858.668">
      <featuregroup name="Browse content" time="3632.71"
                    waitingtime="0.0" count="300"/>
      ...
      <featuregroup name="Modify files and directories .../>
    </summary>
    <featuregroup name="Data management"
                  start="0.0" end="27.14283" time="27.14283"
                  index="0" application="MS Outlook" tool="Emails"
                  activity="Filing [Ordner-/Kartei- und Dateiverwaltung]"
                  goal="Office management"/>
    <featuregroup name="Non computer work (with known goal)".../>
    ...
  </featuregrouptrack>
  <applicationtrack>
    <summary count="836" time="24952.383">
      <application name="SAP" time="989.2949" productivewaitingtime="304.85352"
                   waitingtime="739.4238" count="47"/>
      ...
    </summary>
      ...
      <application name="MS Word"
                   start="15720.27" end="15729.841" time="9.571289"
                   index="290"
                   activity="Exam administration" goal="Teaching support"/>
      ...
  </applicationtrack>
  <tooltrack>
    ...
    <tool name="Accounting"
          start="19456.123" end="19549.408"
          time="93.28516" index="362"
          browser="MS Internet Explorer"
          activity="Cross-checking..." goal="Accounting"/>
    ...
  </tooltrack>
  <activitytrack>
  ...
  </activitytrack>
  <goaltrack>
  ...
  </goaltrack>
  <browsertrack>
  ...
  </browsertrack>
  <waitingtrack>
    <summary count="176" time="1388.4271">
      <waiting name="SAP" time="739.42755"
               productivewaitingtime="304.85352"
       waitingtime="739.4238" count="87"/>
    </summary>
    ...
  </waitingtrack>
  <telephonetrack>
  ...
  </telephonetrack>
</usage>
```

The *usage* is the single XML root element. For every track a corresponding XML element such as *featuregrouptrack* exists. The first element in such a track element is a *summary* XML element containing summarised data from the single track elements. In the *summary* for every distinct feature group name exactly one element with aggregated summary data is contained.

Single *featuregroup* elements follow after the track summary element. These feature group track elements are in temporal sequence starting with the first element from the first imported Anvil file and ending with the last element of the last imported Anvil file. The XMLelements of the other tracks have a similar structure. Some elements contain referenced attributes like the feature group elements contain their respective coded activities and goals. Secondary track elements have start and end times already in seconds instead of index numbers referencing to primary track elements.

5.6.5 Chart Creation

The *getUsageOutputDocument* method from the *taxonomy* object creates an XML document with a structure as just described in the previous Section 5.6.4. All charts generated by FUSSY use this XML document as data basis.

The *FussyChartUtil* class is the helper class for creating charts. All charts are created with a single method call of this class. The *createBarChartFromSingleSummaryNode* method is used if the data for the chart to create is directly taken from the summary elements. One example is the XPath query *"//waitingtrack/summary"*. The resulting elements are the summary elements of the waiting track.

The *createBarChartFromNodeList* is used when applying XPath queries on all elements of a track. In FUSSY this method is used if some kind of limitation is necessary and not all elements from a given track should be used for the chart creation process. Summary elements can not be used in this case because these elements already have summarised all data from a track. One example of such an XPath query is *"/usage/featuregrouptrack/featuregroup[@name != 'Non computer work (with known goal)']"*. This query selects all elements from the feature group track except elements with the name *Non computer work (with known goal)*. Thus only computer related feature groups are used.

Both described method calls for creating charts have additional passed parameters. The most important parameters are the chart title, a threshold, the chart type and the resolution to use. The threshold is applied to the result values. If result values are below this threshold no single bars for these values are created, instead all these values are cumulated and one bar named *Others* is created.

JFreeChart Library

The *FussyChartUtil* uses internally the JFreeChart library by Gilbert and Morgner [2005]. This is a very powerful library being able to create many different kind of charts. JFreeChart is distributed under the LGPL license. The *createBarChartFromSingleSummaryNode* and *createBarChartFromNodeList* methods for themselves internally use the *FussyDatasetUtil* class for creating the necessary data structure used by the JFreeChart library out of the node structure from the XPath query. The data structure to create is an instance of the *DefaultCategoryDataset* for all supported charts by FUSSY. The chart type parameter is an internal FUSSY constant and is used for deciding which type of dataset has to be created with help of the *FussyDatasetUtil* class. The different chart data may include separate or accumulated waiting times, non-computer times and others. Additionally the passed thresholds are handled here.

FUSSY in the actual version uses two different types of charts provided by the JFreeChart library. These are the *BarChart* and the *StackedBarChart* type respectively. In a StackedBarChart the result bars consist of several distinct parts thus being able to show additional proportional information in every result bar. This is used amongst others in the FUSSY charts showing the non-computer proportion of goals and activities such as shown in Figure 6.2 and Figure 6.4 of the Chapter 6.

JFreeChart in the used version 1.0.0-pre2 is not able to handle displaying an overall sum of the different parts of a bar in a *StackedBarChart*. Hence in extension to the JFreeChart library a specific class the *FussyStackedBarRenderer* is created for providing this feature. *FussyStackedBarRenderer* is a inner class of the *FussyChartUtil* class and therefore has no distinct file. The *FussyChartUtil* class also handles font type and size, label descriptions, margins, colour definitions and other necessary code defining the layout of the charts.

5.6.6 Chart Viewing and Storage

The generated chart is an instance if the *JFreeChart* class. Directly after creation of every single chart the *handleChart* method from the *Main* class is called. With help of this method viewing and storing of the charts is implemented. The *handleChart* method itself uses other utility methods. The *showChart* method from the *FussyChartUtil* class provides the code and functionality for creating a new window on the screen for every chart. The application is automatically exited when closing the last window. FUSSY can also save the generated charts in the PNG bitmap image format. This is implemented with the *saveChartAsPNG* utility method from the *ChartUtilities* class which is already included in the JFreeChart library.

The PDF export functionality is available with the *printChartToPDF* method from the *FussyChartUtil*. The PDF export is implemented with help of the Open Source iText PDF library by Lowagie and Soares [2005]. iText can generate PDF files on the fly. The iText library can be used with either the *Mozilla Public License* or alternatively the LGPL license. As already stated in Section 5.3 the PDF format is a very good format for storing the charts created by FUSSY thus the recommended file storage format.

5.6.7 Performance

The used hardware is a laptop with Pentium-M 1.7 GHz CPU and 512 MB RAM. The import of all 12 Anvil files created in the FUSS-B study so far and the creation of the internal datastructure lasts around 2.5 seconds. The additional storage of PDF files also takes around 2.5 seconds thus having an overall running time of approximately 5 seconds in this case.

The saving of PNG images with a resolution of 1024 by 768 pixel is considerable slower and lasts around 13 seconds resulting in an overall running time of approximately 15.5 seconds. If the results are shown on the computer screen this lasts around 5.5 seconds. Hence the overall running time here is circa 8 seconds.

5.7 Summary

The FUSSY application meets the requirements defined in Section 5.1. It is easy to extend the application with new analysis charts. The JFreeChart library already has many built-in chart types and features available and will further improve in the future. Due to the implementation in the Java programming language FUSSY is a cross platform application. With the Ant build process no IDE is necessary for the generation of an executable distribution out of the source files.

FUSSY is quite quick, as presented in the previous Section 5.6.7. Having results as PDF available in a few seconds is definitely adequate. Even if much more data has to be analysed in the future progression of this study the performance will be sufficient. Some ideas for future FUSSY improvements are outlined in Section 7.3.

Chapter 6

Results of the FUSS-B Study

This chapter presents the results of the FUSS-B study referring to the available data so far. Due to the FUSS-B study is newly designed there only have been a very small number of test participants yet. Therefore a relatively small amount of video data is available up to now for analysis purposes as outlined in Section 4.1.5 of Chapter 4. Hence it is not possible to create any statistically firm statements out of the coded and analysed data. This presented results should be rather seen as proof of concept of the overall study design and implementation. This is a shared chapter and written together by Sandra Brückler and Günter Modes.

All observed and analysed test participants are administrative assistants in an educational environment. The resulting charts contained in this chapter are altogether created with the FUSSY application. This application is created as part of this study. Informations regarding the FUSSY application including a manual are found in Chapter 5. In some of the generated diagrams a threshold is used from the FUSSY application while creating the chart. The data below this threshold is not shown in a separate column but the values are internally summed up and presented in the *Others* column of the charts.

6.1 Goals

Goals correspond to the main scopes of work. An actual task is embedded in a goal. Due to the gathered observation data is limited to administrative assistants so far, the expected goals are related to teaching areas. Figure 6.1 shows the distribution of the different goals. This first chart includes only computer working times. *Teaching Support* (27.7%) is unsurprising the most important goal. However the *Public Relations* goal is on the second place with 24% of the overall time. This is mainly caused by one test participant who had to write, print and create Christmas cards. This task lasted several hours and was coded with this *Public Relations* goal.

The *Infrastructure* goal is used for work concerning the general infrastructure of the institute. In general this goal will not have such a high proportion of 12.9%. One test participant had to analyse the utilisation of lecture halls and generate charts. This was a vast and long task. This task is coded with the *Infrastructure* goal hence this goal got such an importance.

The non-computer times are included in Figure 6.2 and their respective proportions are shown for every goal. Although with the non-computer durations added also *Teaching Support* and *Public Relations* stay in front of the other goals. *Staff Support* is now on the third position instead of *Infrastructure*. The *Staff Support* goal has the highest proportion regarding the proportion of non-computer work in relation to computer work for this goal. There is only around 29% computer time but 71%

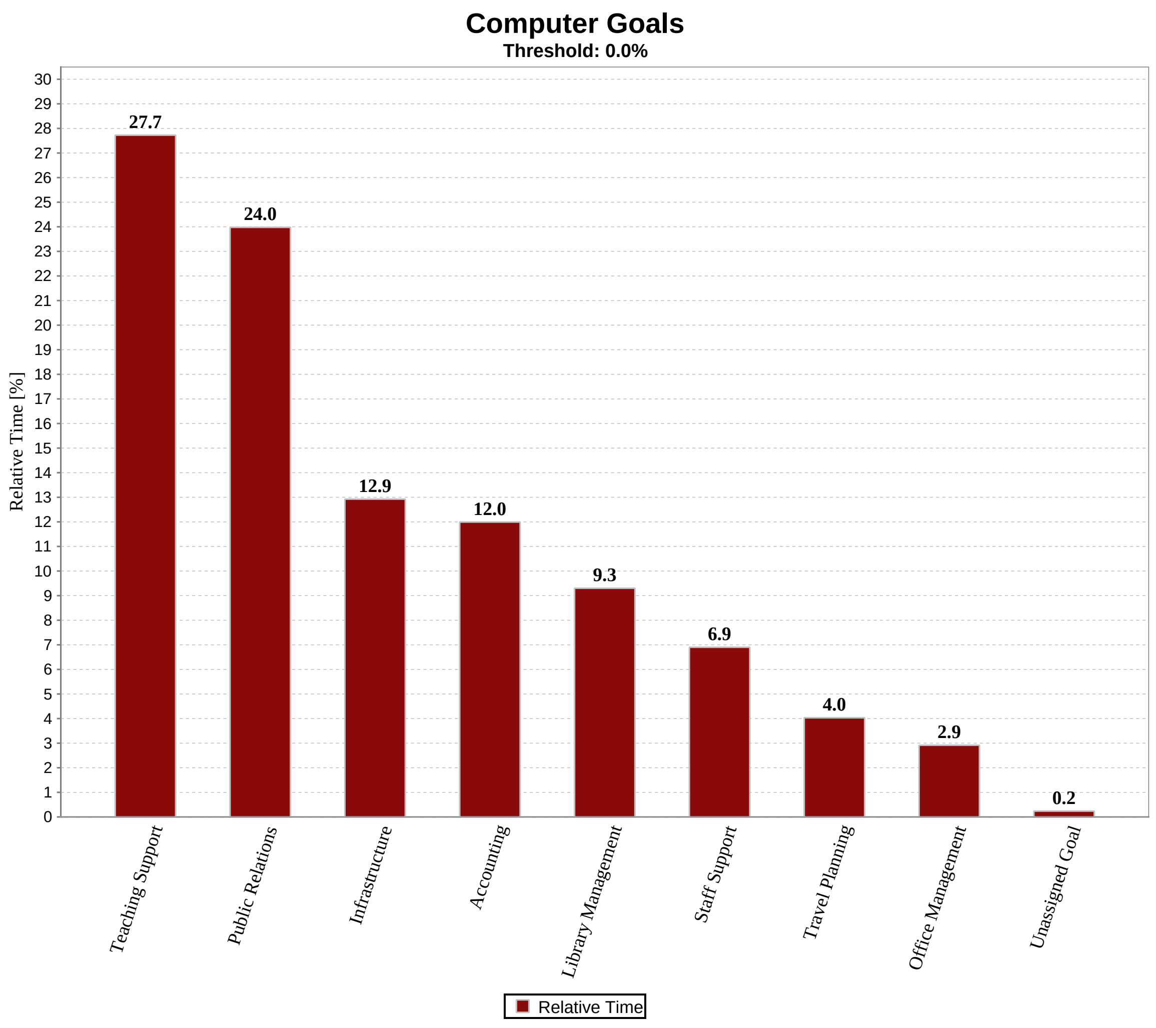


Figure 6.1: The Usage of the Different Goals. Only Computer Working Times are Included in this Chart.

non-computer time within this goal. All the other goals except for the negligible *Unassigned Goal* have less non-computer than computer work thus indicating the importance of the computer as tool for administrative assistants.

6.2 Activities

The analysis of the different activities is outlined in Figure 6.3. *Room Administration* is first with 17.3% followed by *Advertising And Marketing* (14.9%). *Room Administration* is normally not really a very common activity. The reason for such a prominent position of this activity is the same as already stated in the previous Section 6.1 regarding the *Infrastructure* goal. One test participant had to analyse the utilisation of lecture halls. This task is coded with *Room Administration* as activity and *Infrastructure* as goal respectively.

The *Advertising And Marketing* activity has also a very high value with 14.9%. The reason for

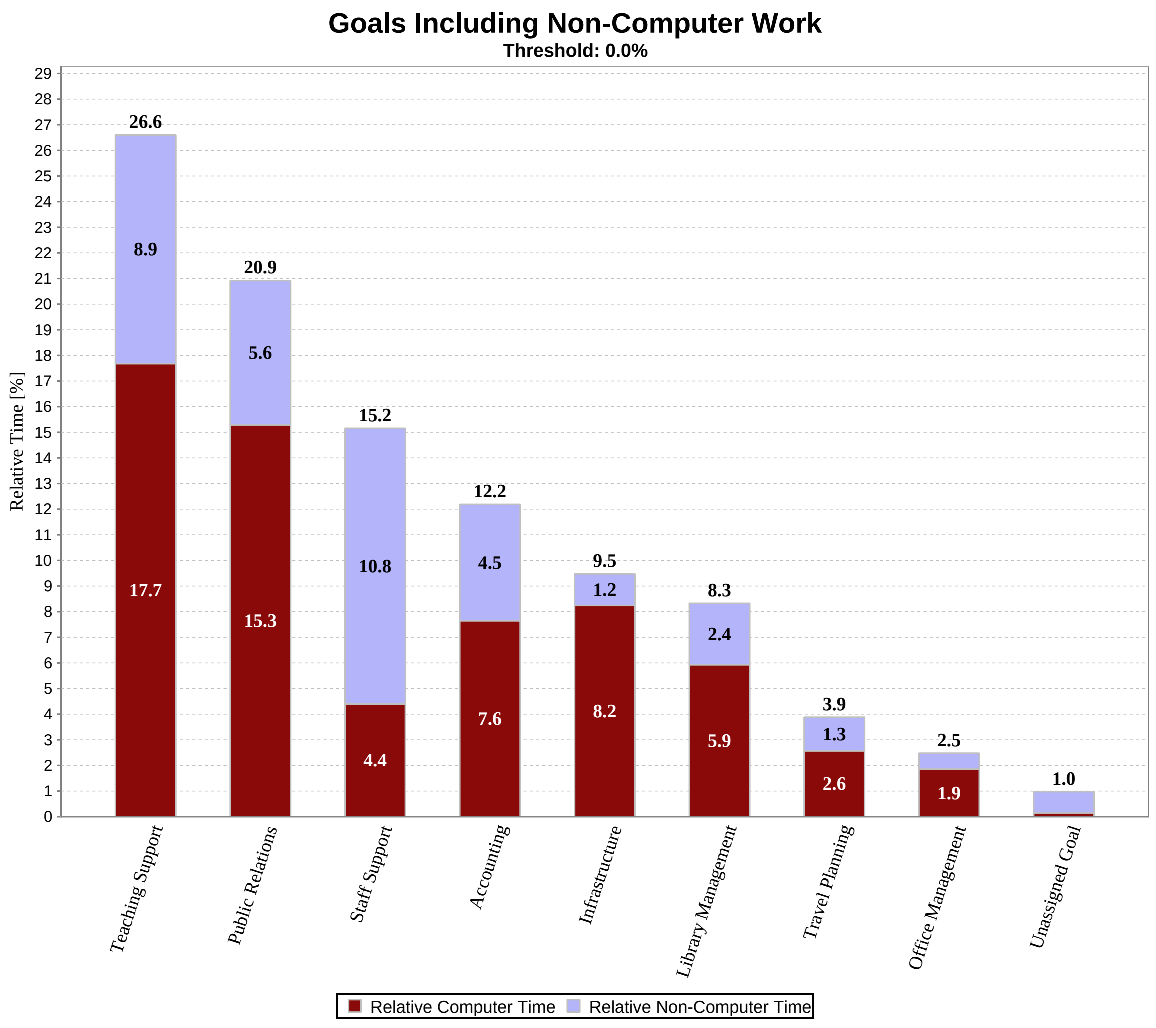


Figure 6.2: The Different Goals Including the Non-Computer Work.

this is already described in the previous section too. The user had to do a lengthy task concerning the creation and printing of Christmas cards. This task was coded with the *Advertising And Marketing* activity and *Public Relations* goal.

As expected *Course Administration* with 13.4% and *Exam Administration* with 8.2% are very common activities, mainly included in the *Teaching Support* goal, with high proportions. This *Course Administration* activity includes the overall management of the courses in the online *University Management System* of the university. Also for the two accounting activities *Asset Management* and *Cross-Checking* a high investment in time is expected. Surprisingly the *Register, Unregister And Recruit Staff* activity is on the fifth place with 10% computer time proportion. This is introduced from one participant writing a serial email for an advertisement of a vacancy including an extensive email address acquisition process.

Figure 6.4 includes non-computer work additionally to the computer related tasks. Due to *Advertising And Marketing* also included some non-computer work, putting the Christmas cards into envelopes, this activity is here at the first position overtaking the *Room Administration* activity which

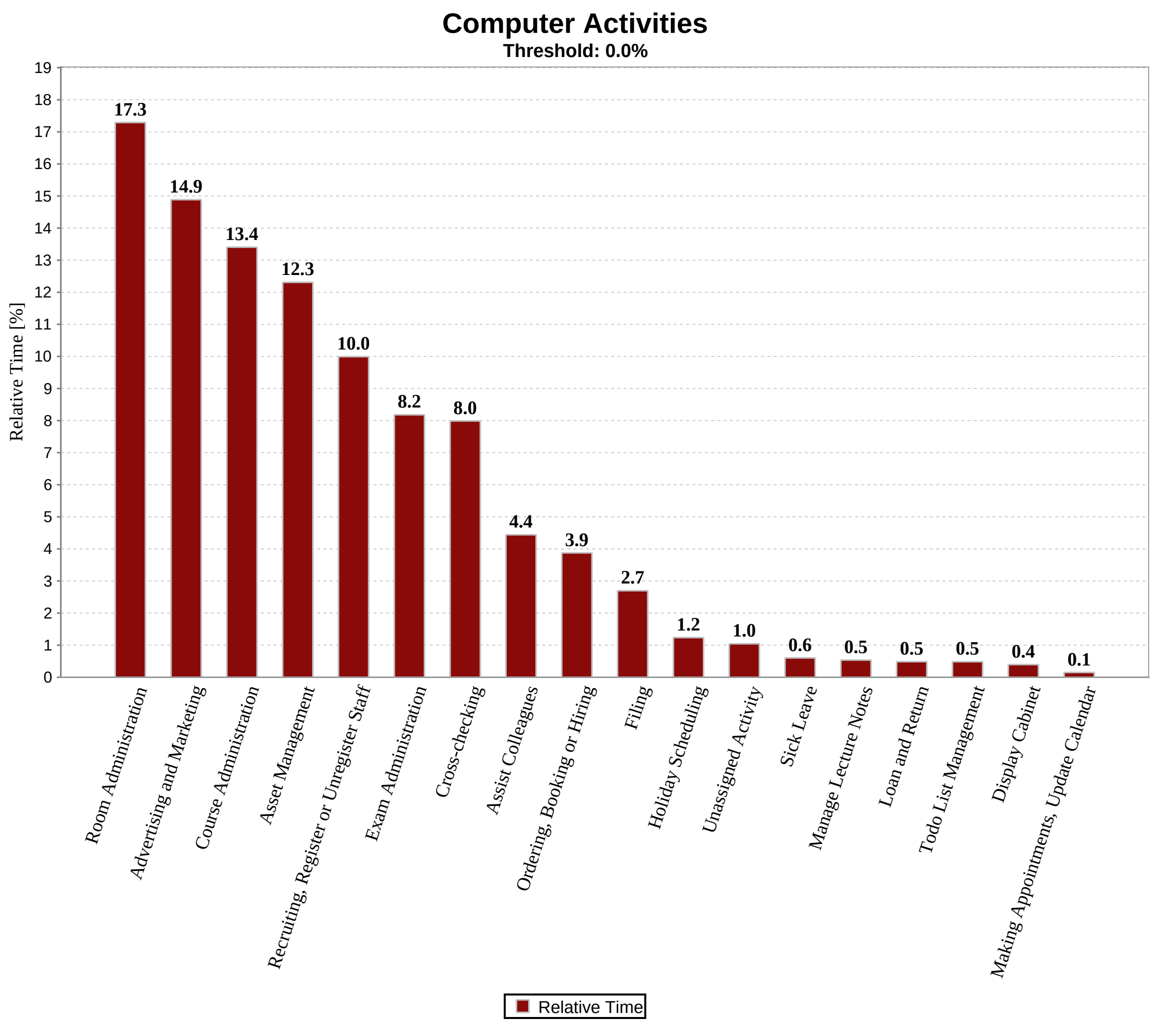


Figure 6.3: Computer Activities.

has not such a high non-computer fraction. The most important activities contain mainly computer times hence the same reasons for the positions are valid as in the computer activities chart.

Assist Colleagues is an exception which proves the rule with almost 69% of non-computer activity. This activity is mainly verbal oriented, therefore this finding is quite understandable. Also *Cross-Checking* has a higher proportion of non-computer in relation to the activity time than the other accounting activity *Asset Management*. This activity involves necessary manual work with invoices and other written documents which is the reason of this finding.

The *Student Support* activity only contains non-computer time. Although with more statistical data this activity most likely will never have exact 100% non-computer work, *Student Support* often consists of providing information or paper forms handling and will often have a high non-computer proportion.

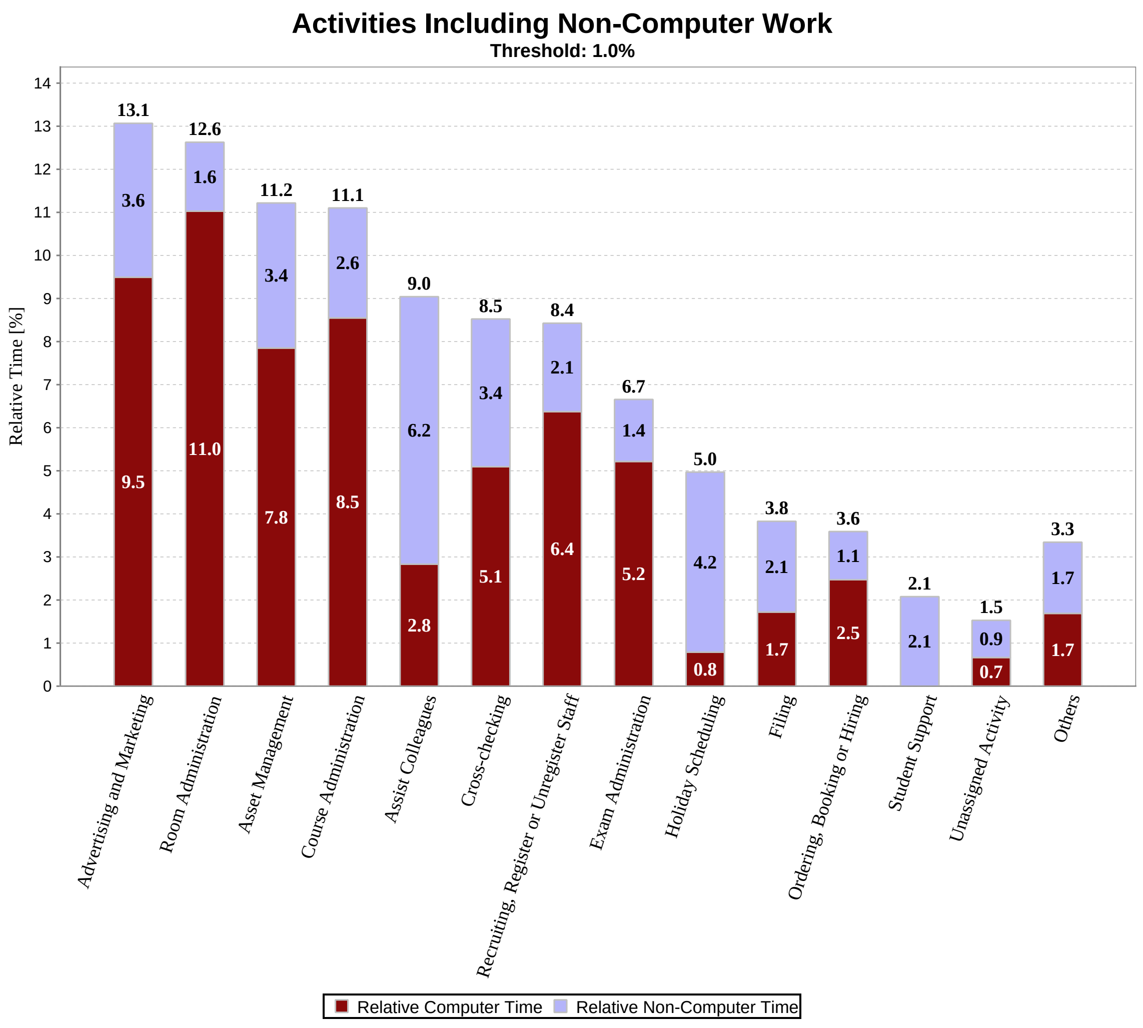


Figure 6.4: Activities Including Non-Computer Work.

6.3 Tools and Applications

Tools are collective names of different types of applications used for a specific tasks as word processing or writing emails. So has the *Email* tool different specific application instances such as *MS Outlook*, *The Bat* or *Pegasus Mail*. These programs are specific software applications all of them are able to create emails.

Certain software products directly correlate with specific Applications. For these tools only one specific application is used by the participants. Figure 6.5 demonstrates the *University Management System* (27.5%) and *Spreadsheet* (24.5%) tool being the leading tools in respect of their usage time. In the application usage diagram (Figure 6.6) the percentages for these tools almost exactly correspond to *MS Excel* (24.4%) and *TUG Online* (27.5%) respectively.

With other applications such a correlation does not exist. The *Wordprocessing* tool on the third place for example has a percentage of 22.7% as opposed to *MS Word* having a much higher value of 26.9% in the application usage diagram. The actual MS Office Suite can be configured for using MS

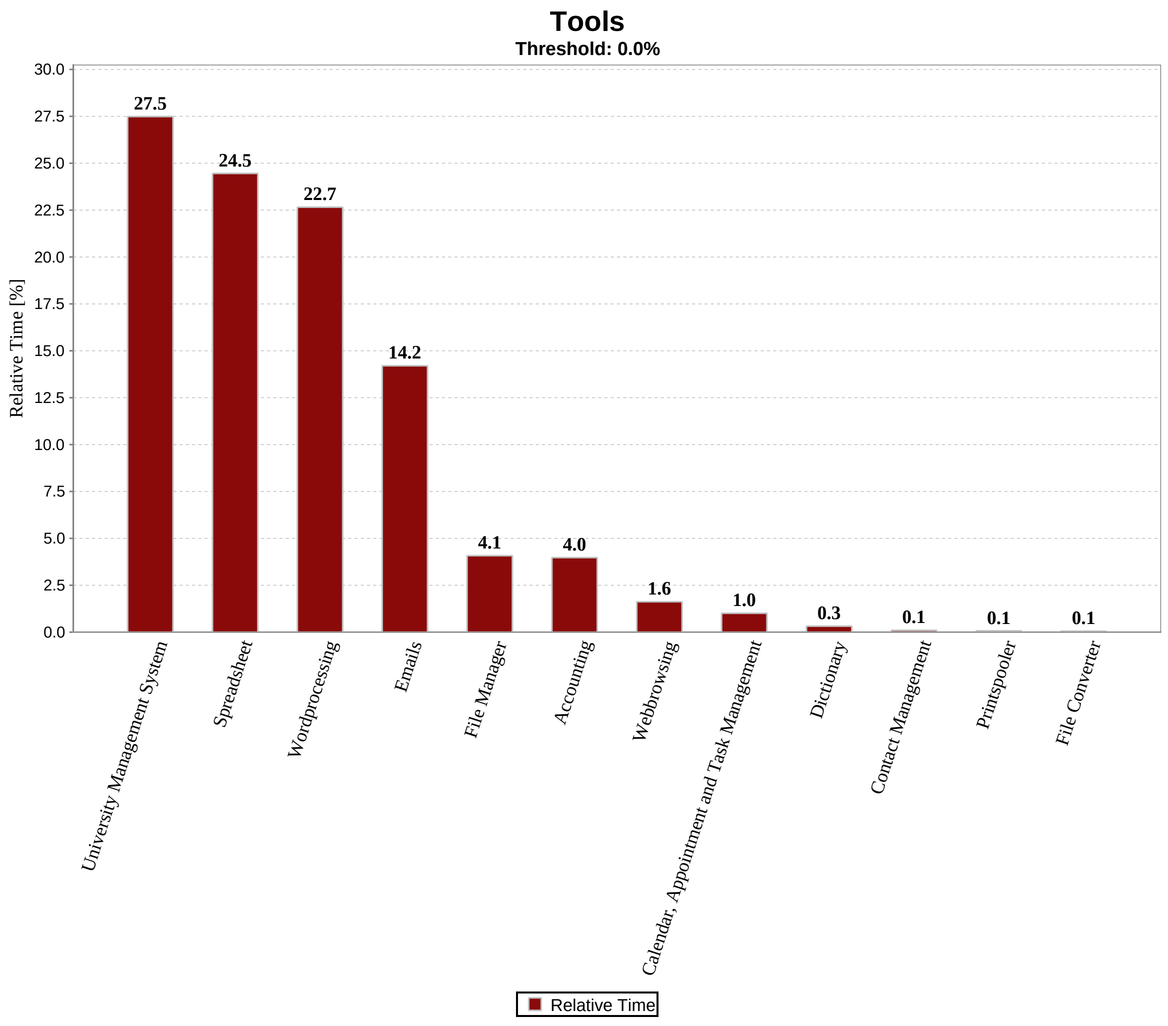


Figure 6.5: The Different Used Tools.

Word as default mail editor. One test participant used *MS Word* as application for creating emails. It was decided to code in this case *MS Word* as application and *Email* as corresponding tool. Thus a part of the *Email* tool percentage also contributes to the *MS Word* application value raising *MS Word* to the second place regarding the usage time in the application result chart.

The *Email* tool width 14.2% is split up into different email applications (*MS Outlook*, *Pegasus Mail* and *The Bat* and *MS Word*). These distribution values can not immediately required because the *MS Outlook* application is not only included in the *Email* tool but also in the *Calendar, Appointment and Task Management* tool therefore not all of the *MS Outlook* time is reflected in the *Email* time. Hence the distribution of the different applications included in the *Email* tool is shown in an own Figure 6.7.

The *File Manager* tool corresponds to the *MS Windows Explorer* application with both having 4.1% proportion in both diagrams. The *Accounting*(4.0%) tool has again a direct correlation with the *SAP* (4.0%) application. Due the *Acrobat* (2.6%) application (the full version of Acrobat, not the Acrobat Reader) is also able to create documents, it is coded with the tool *Wordprocessing*.

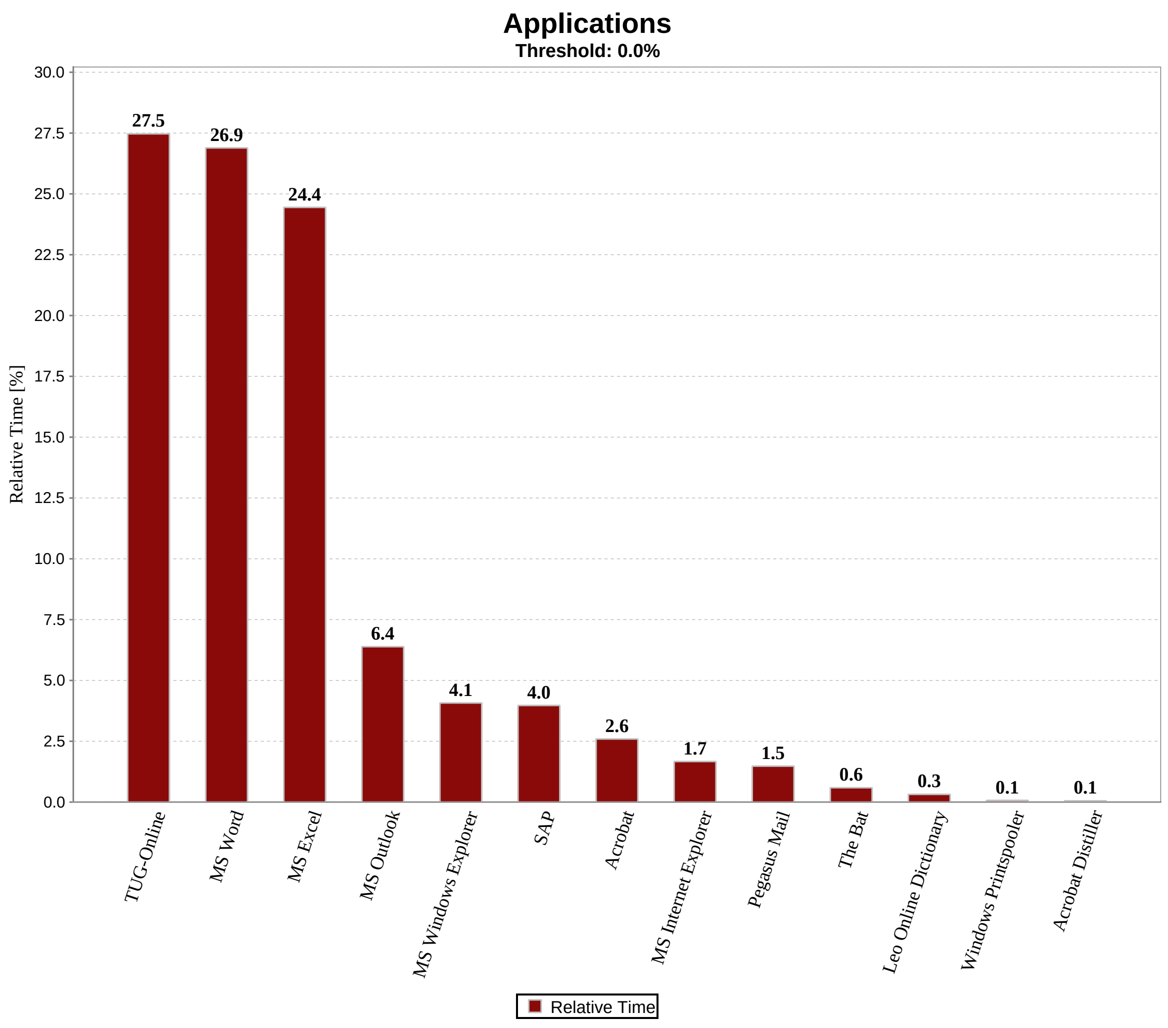


Figure 6.6: The Different Used Applications.

The *MS Internet Explorer* application with 1.7% is the portion of time where this application is used as *Webbrowsing* tool only. Due to this browser is also used for the *TUG-Online* and other online applications like *Leo Online Dictionary* this time does not reflect the overall usage of this webbrowser. In the coding guidelines it is defined to code online applications similar to normal applications and tools, hence the *TUG-Online* application is coded as tool *University Management System* (and not *Webbrowsing*) and as application *TUG-Online*. The only coding difference to an offline application is the parallel already coded browser track, while the online application is active. Taking this separate browser take into account the overall percentage of web browser use is 35.46%. The different used online applications in the web browser in relation to the overall web browser time can be observed in Figure 6.8.

The leading *TUG-Online*, *MS Word* and *MS Excel* have almost the same percentage values each and together a sum of around 80% as it is also shown in Figure 6.6. The more or less corresponding tools regarding the already mentioned correlations *University Management System* , *Spreadsheet* and *Wordprocessing* have together a percentage of circa 75%. These are quite expected results for admin-

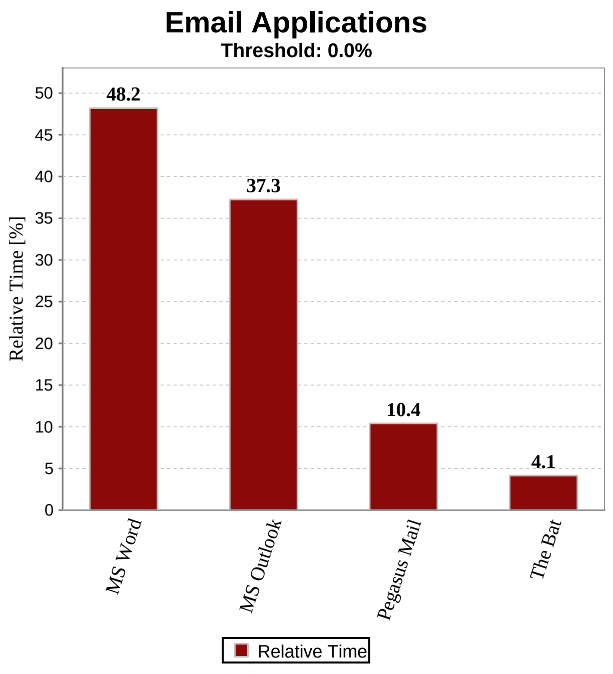


Figure 6.7: The Different Applications Used as Email Tool.

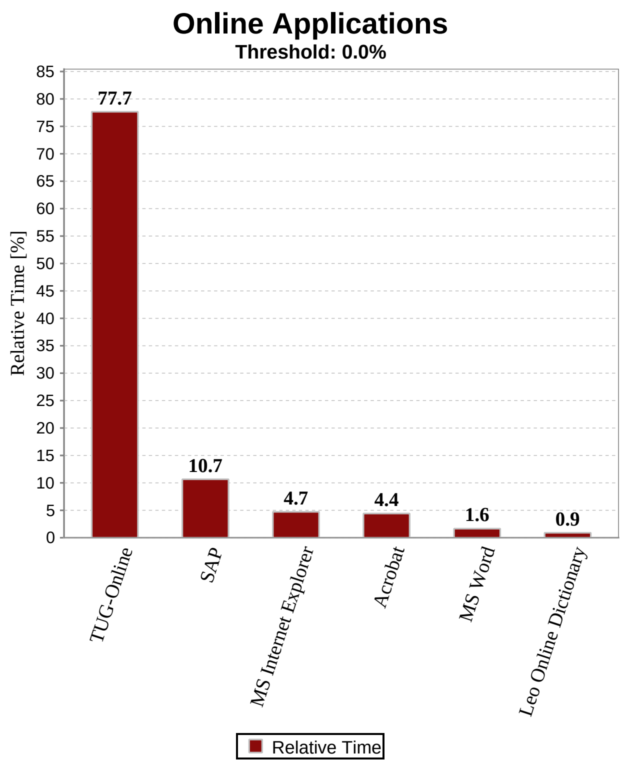


Figure 6.8: The Different Online Applications Used in the Web Browser.

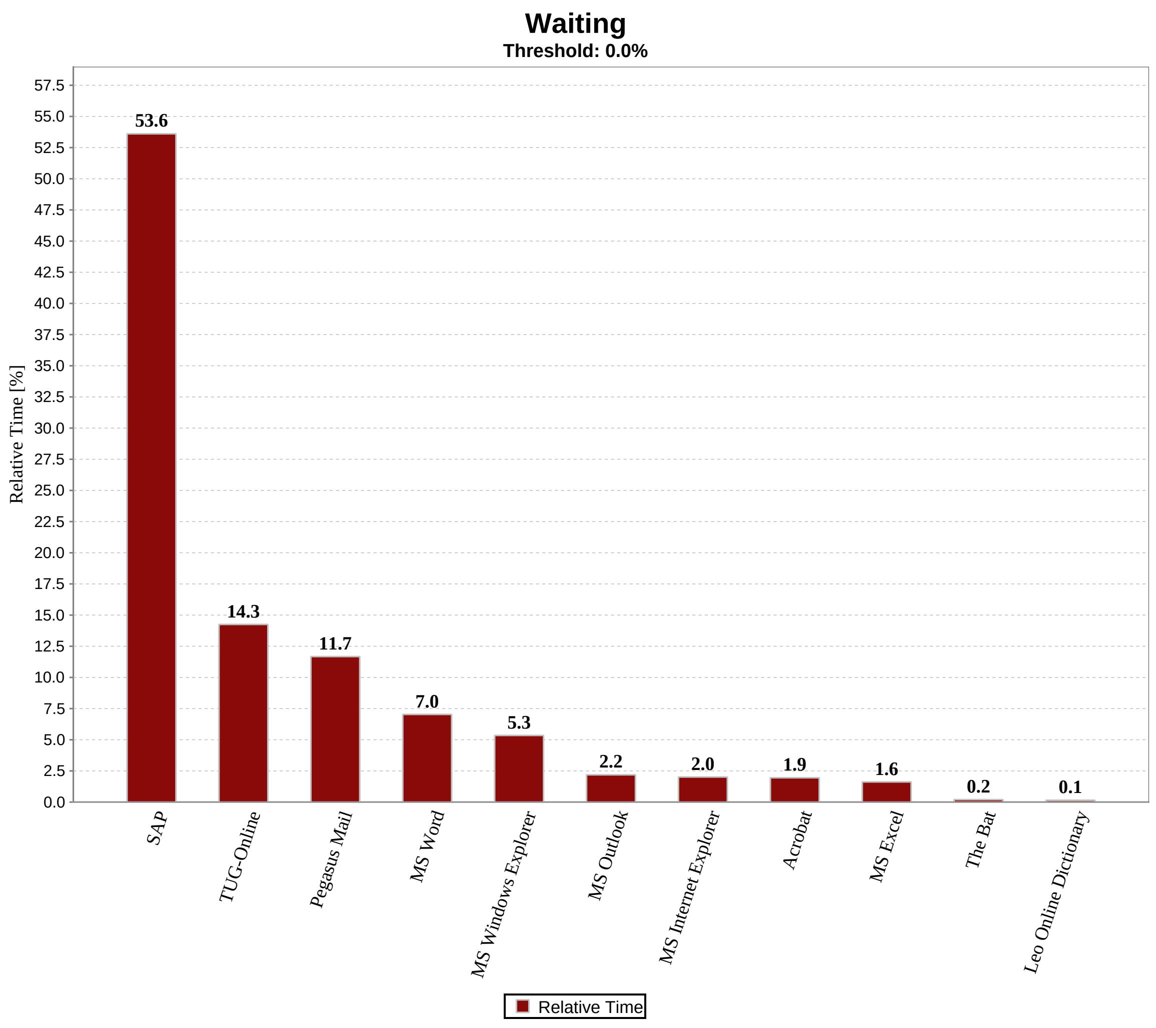


Figure 6.9: Waiting Times of the Different Applications.

istrative assistants. University management systems, word processing and spreadsheet applications are quite common applications in this role. Also the *Email* tool on the fourth place of the used tools is not very surprising. An unexpected result is the very low value of the *Calendar, Appointment and Task Management* tool. A higher value was initially expected by the FUSS-B team.

6.4 Waiting Time Analysis

Waiting time is defined as the time period the user waits for an application to complete a task meaning the application is not responsive during this time. Often but not always the operating system shows a specific waiting cursor in this case. The analysis of different waiting times may be more an usability instead of an usage issue nevertheless it is done as part of this FUSS-B study. A direct comparison of waiting times from the different applications is presented in Figure 6.9. The values are relative to the overall waiting time of all applications. For the creation of this chart the summary information of the independently coded waiting track is used.

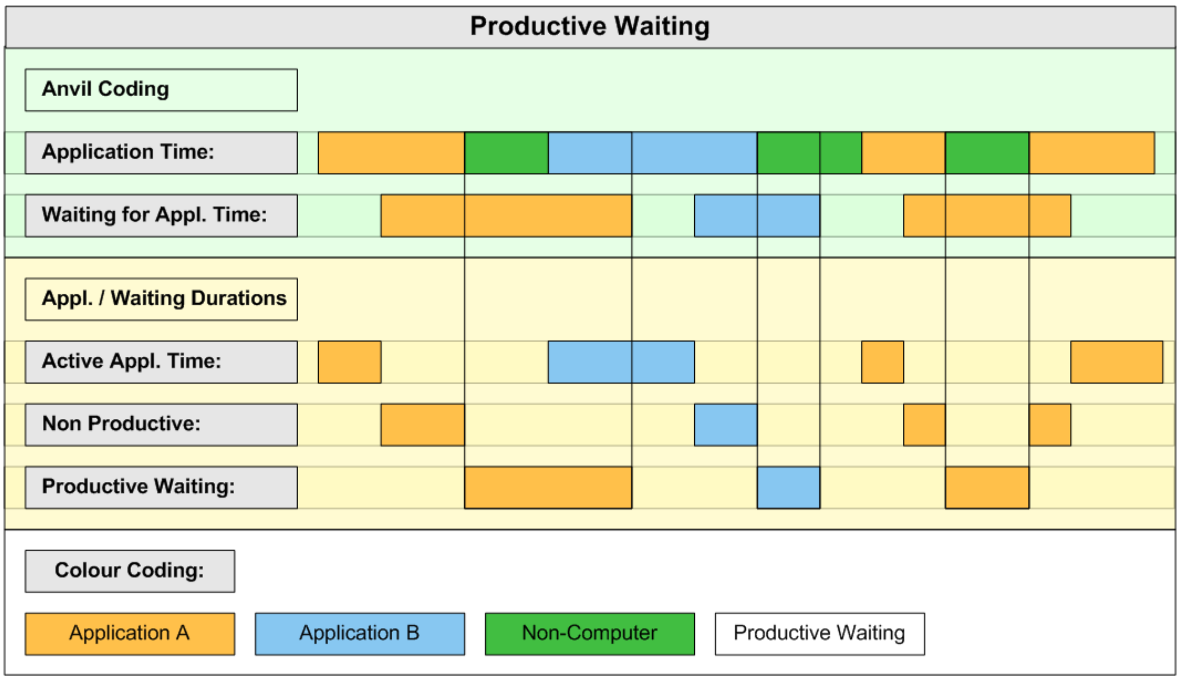


Figure 6.10: This Draft Outlines the Definition of Active Application, Non-Productive and Productive Time.

The chart reveals that particularly the *SAP* application produces the longest waiting time with a percentage of 53.6% percent in relation to the overall waiting time. *SAP* is running by means of a web client, this is the main cause for the long waiting periods. Another application that sometimes forces the user to wait is the *TUG-Online* application with 14.3%. *Pegasus Mail* (11.7%), *MS Word* (7.0%) and *Internet Explorer* with 5.3% follow next.

During waiting it is possible to have another application active. Due to non-computer work is coded with no computer application it is even possible actually having no application coded while waiting for another application. Sometimes the user changes the foreground application or performs non-computer tasks because a long waiting process is expected. Such an Anvil coding example is shown in Figure 5.5 of Chapter 5. To handle this situation the concept of productive waiting time is developed.

6.4.1 Definitions

A schema outlining the situation and following definitions is presented in Figure 6.10. The summarised time from start to end of all application elements in the application track is called the overall application time or just application time regardless whether a waiting takes place or not. The same definition is used for the overall waiting time of an application. The productive application time is the application time were no waiting for this application takes place. If a waiting takes place during the same application is coded in the application track this will be called non-productive application time or just non-productive time because the application is not responsive. If no waiting takes place during the application time this is named the active application time.

Non-productive application time is synonymous to non-productive waiting time thus this time is just named non-productive time. An example is outlined in the Figure 6.10. Two applications *A* and *B* are shown in drafted Anvil coding tracks. Also non-computer durations are shown in the application track, although in an actual Anvil coding these non-computer times are only represented in feature groups. Other tracks do not have any influence hence only the application and waiting tracks are drafted in this figure. In Section 5.6.3 the algorithm of FUSSY handling the productive waiting time is described in detail.

For the author it is important to repeat the statement from Section 2.3 in this place. It is explicitly not a goal of this study to observe the efficiency or productivity of the test participants. The labelling of the non-productive application time is intuitively done and is not intended to be used for any analysis comparing different test participants. It is normal to wait in front of an application while an application is not responsive to user input. It will be even very advanced from users if they decide to do something other work while waiting for an application. In many cases it is technically not possible to change to another application because the computer blocks all user input, in other cases this is possible.

To shortly summarise, it should also be a goal to minimise all forced waiting times including productive waiting time to be able to use the computer as an efficient tool. Of course it is better to do some anyway necessary non-computer work while waiting for a computer task to finish instead of just waiting. Though it would be even better for users to be able to decide for themselves when the time has come doing non-computer related tasks.

6.4.2 Application And Waiting Time

In Figure 6.11 the application waiting times showing the portion of productive and non-productive time are presented. The application having the most productive waiting time is SAP with 22.1% productive waiting time. *MS Word* has 5% and *MS Windows Explorer* has 3.6% productive waiting time respectively. What do these values mean? One conclusion is that the users already know which and when specific applications will have a long waiting time and the higher the portion of productive waiting time to the remaining waiting time the better the users bypass the annoying situation and do some other task.

The high waiting values from the *MS Windows Explorer* were caused by a network problem during one recording session. Waiting times from the *MS Windows Explorer* application normally do not have a noticeable impact. The *MS Word* waiting times are caused again by the already mentioned Christmas cards task. The used serial letter function of *MS Word* caused waiting time. Putting the Christmas cards into envelopes, during this waiting, was the accordant productive time. Many functionalities of *SAP* cause more or less waiting time every time used. These functionalities are known by the users thus the high productive waiting time of SAP is caused.

Another interesting chart including the applications time and the different waiting times can be observed in Figure 6.12. The non-productive waiting times are already contained in the overall application time (therefore this time is also already contained in Figure 6.6) however the productive waiting times are added to the application time. This is quite obvious when observing the Figure 6.10 containing the different waiting time definitions. This chart reveals some interesting findings. Only the waiting times from *SAP* and *Pegasus Mail* are relevant regarding the application time. *MS Word* and also the *TUG-Online* and all the other waiting times are quite negligible.

A not yet considered influence is that more used applications will most likely have more waiting times in general meaning the waiting time has to put in relation to the application time. At first the obvious solution wold be just dividing the waiting time of an application with the corresponding application time and thus gaining a relative value. Considering again Figure 6.10 questions arise what

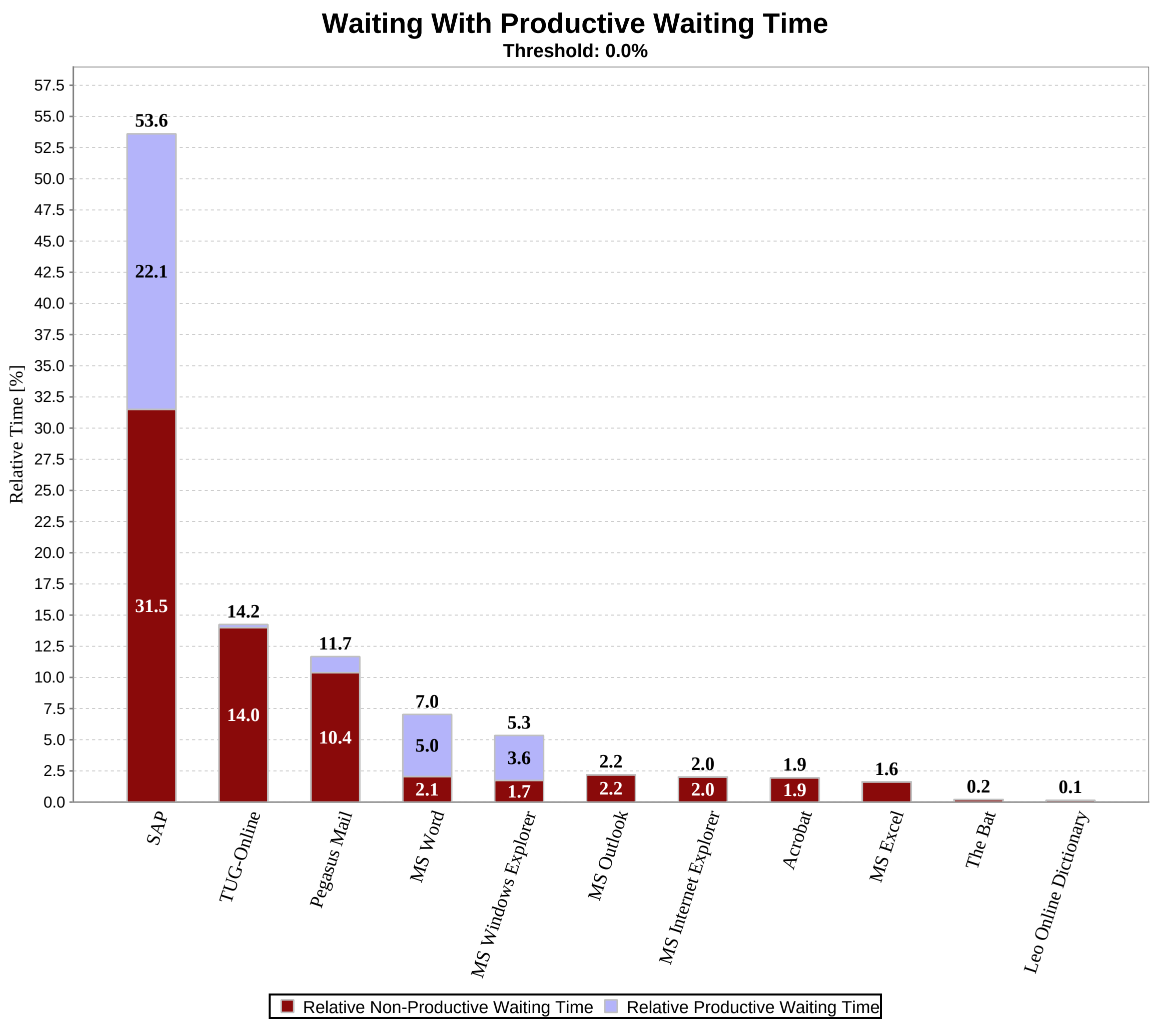


Figure 6.11: Waiting Time of the Different Applications with Distinguishable Productive Waiting Time.

amount of time should be taken for application time. The application time could be shorter than the waiting time therefore the relative value of waiting time divided by application time could be greater than 100%. Hence it is decided to calculate other relative waiting times. At first the waiting time in relation to the sum of application and productive wait time is calculated instead meaning although another application is active the application is nevertheless assumed to be coded.

A second relative waiting time is also calculated. In fact the non-productive time in relation to the application time thus having a rating how long the user really waited in front of an actual foreground application. Considering Figure 6.10 both of these values are always less or equal 100%.

The result chart containing these two relative waiting values for each application can be observed in Figure 6.13. *SAP* is at first with 57,1% relative waiting time in relation to application and productive time. The second relative value is 43.9% meaning although the users already are doing many other works while waiting for the *SAP* application the users actively nevertheless wait almost 44% of the application time waiting for a response.

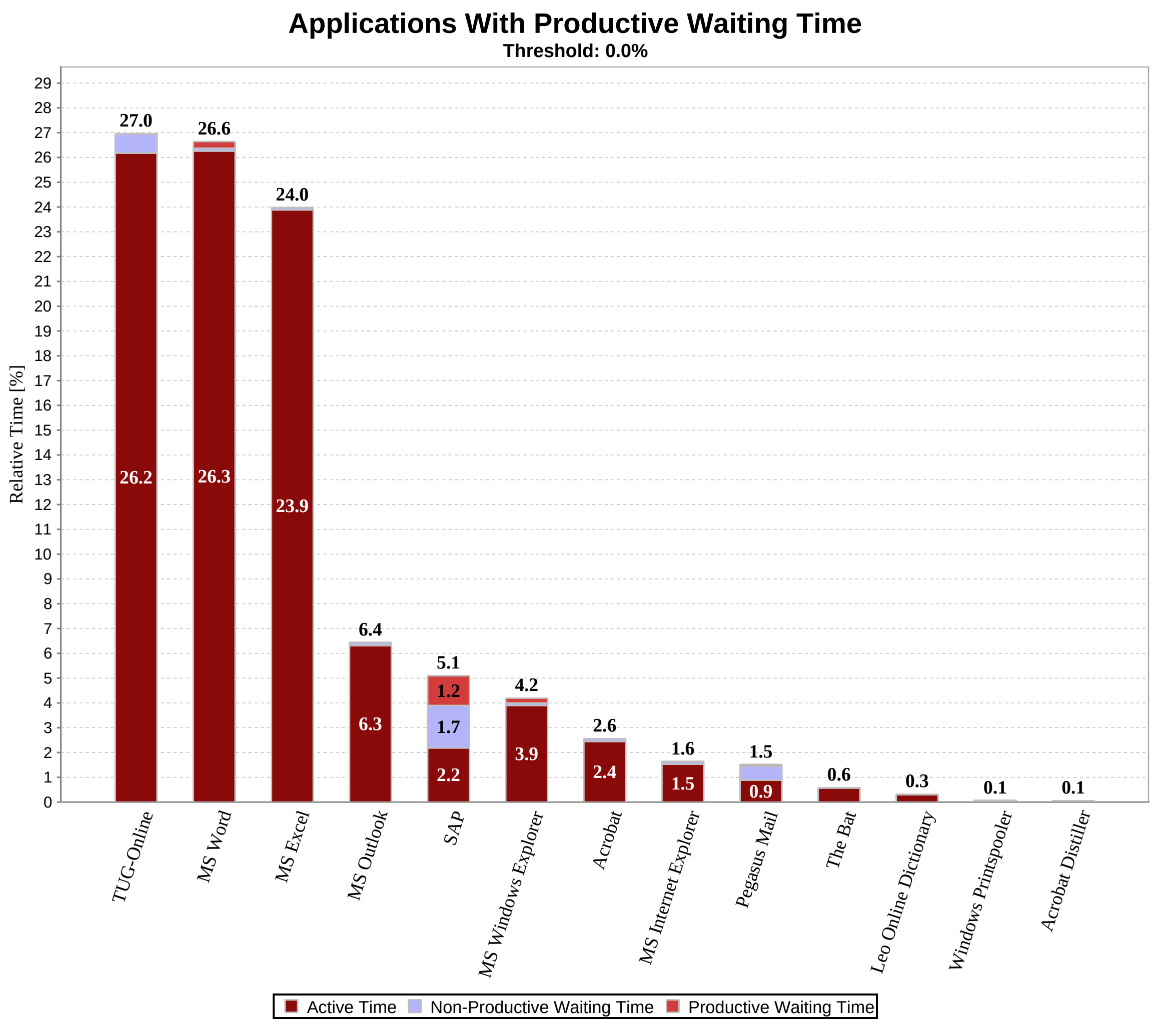


Figure 6.12: Applications With Included Waiting and Productive Waiting Time.

Pegasus Mail is on second place with 41.7% and 38.9% non productive wait respectively. This is a distorted value because only one test participant uses this application. This user started a few times the application for checking emails and immediately exited it afterwards. The waiting times directly reflect the startup times. No actual usage of this application took place. As already mentioned the high waiting values from the *MS Windows Explorer* were caused by a network problem and are therefore also not representative. The waiting values from the *MS Internet Explorer* with 6.7% both relative values are very small regarding the fact being a web browser application with network latency issues. All other relative waiting values for the applications are below 5%.

6.5 Feature Groups

The different features from desktop applications are grouped together into so called feature groups. In general most of these feature groups are not connected with a special application. An example of a feature group is *Browse Content* which can happen in a word processing application as well as

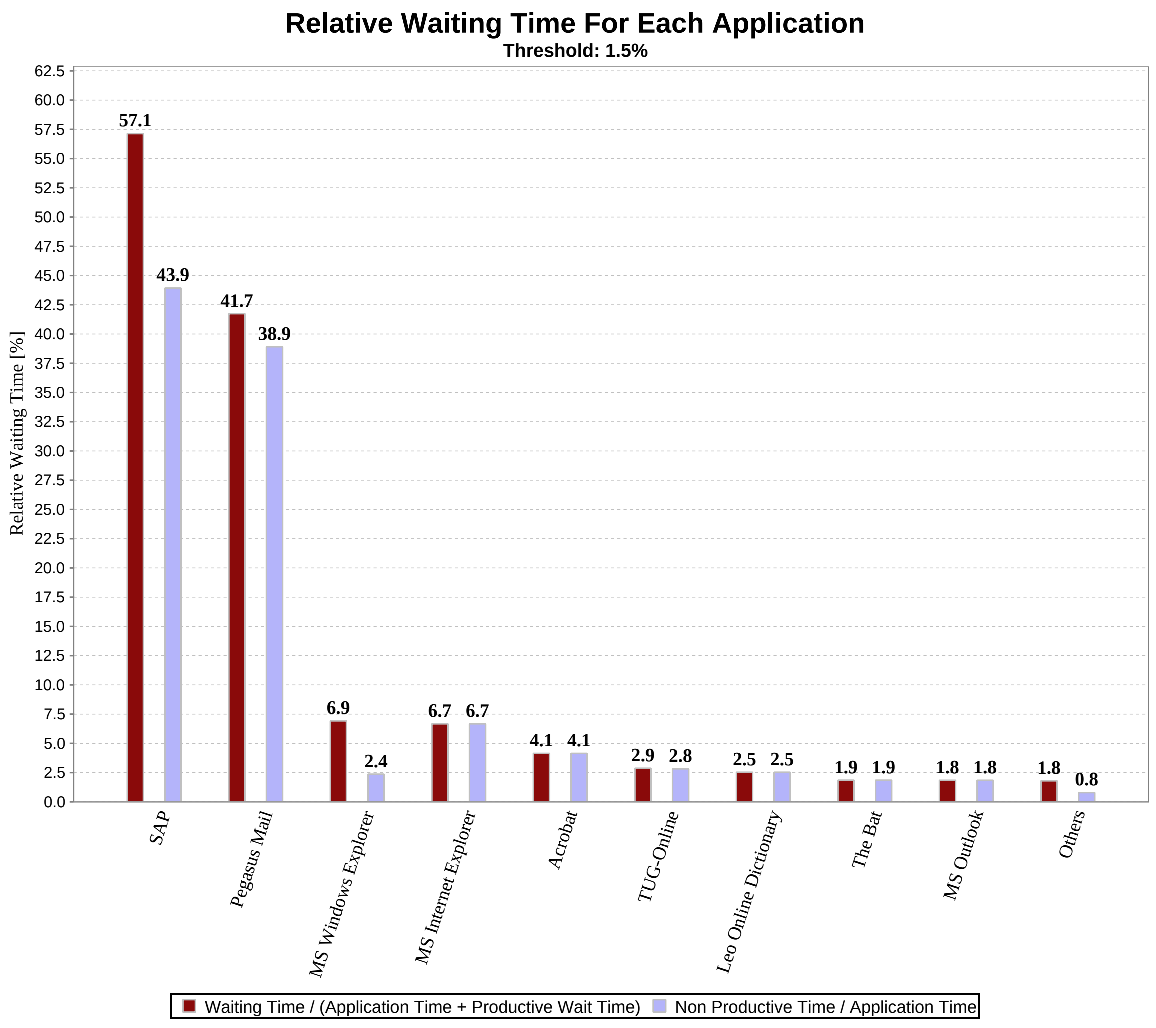


Figure 6.13: Relative Waiting Time For Each Application

in a spreadsheet or web browser software. With the special *Non-Computer Work* feature group it is possible to distinguish between tasks accomplished on a computer and other tasks which are executed without computer support. *Non-Computer Work* is coded if the goal of this non-computer work is known. If the goal is unknown then another special feature group named *Away from workplace and unknown or just unknown goal* or just *Away from Workplace* is taken during the coding process.

Figure 6.14 includes also non-computer and away times. In this case the *Non-Computer Work* feature group is leading with 35%. This is an interesting value giving evidence that over a third of all tasks are not computer related. Due to the interests of this study are mainly in computer related tasks another chart is created. Figure 6.15 shows the time distribution of the different feature groups while only using the computer thus removing the *Non-Computer Work* feature group. The order of feature groups stays of course the same. The respective percentages have changed.

Text Creation And Formatting with almost 30.7% and *Browse Content* with 14.6% are the two leading feature groups. With a distance *Serial Letters* (6%) followed by *Course Data View* (5.6%) and *Course data Management* are next. All other feature groups are below 5%. It is observable that

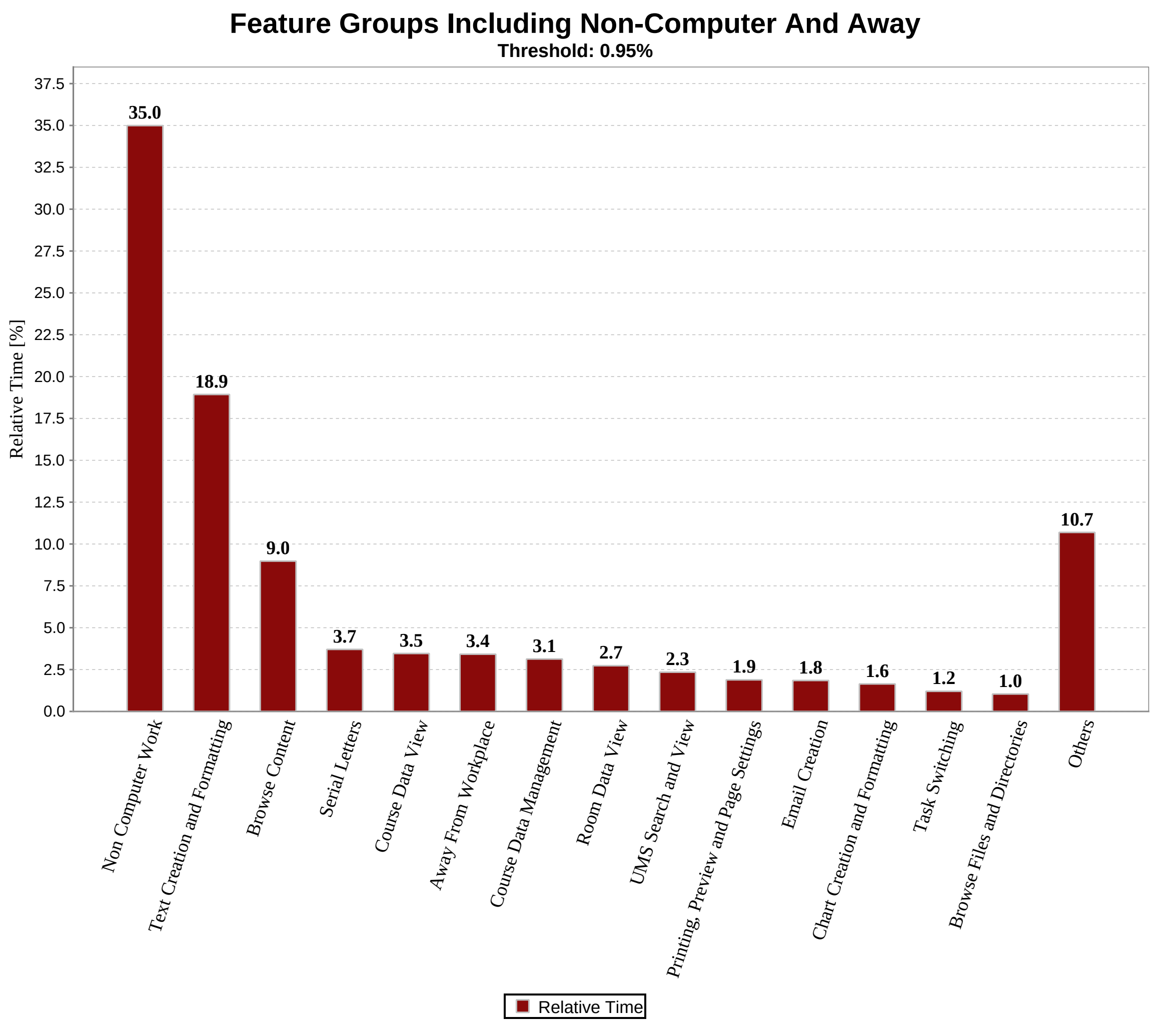


Figure 6.14: Feature Groups Including Non-Computer And Away.

there is a large spectrum of different feature groups and the sum of all feature groups above 5% is only 62% in opposite to all previous charts. Even the sum of all feature groups below 1.5% is 15.8%.

It is quite expected to have *Text Creation And Formatting* and *Browse Content* in front. *Serial Letters* probably has a very prominent position another time caused by the Christmas cards task of one test participant. The next feature four groups are all *University Management System* related. *Course Data Management*, *Room Data View* and *UMS Search and View* together have a quite high percentage of 18.9% regarding the distributed feature groups. To summarise there are no surprising findings regarding the feature groups analysis.

Countable Feature Groups For some of the coded elements the duration is not interesting instead the analysis is done non temporal and only the occurrence count is considered. These feature groups are *Program Options*, *Save*, *Task Switching* and *Using Help*. Their respective count values are shown in Table 6.1. Of course it is not always possible to notice every occurrence of these feature groups. If the user saves a file for example with the common *CTRL-S* shortcut it is impossible to

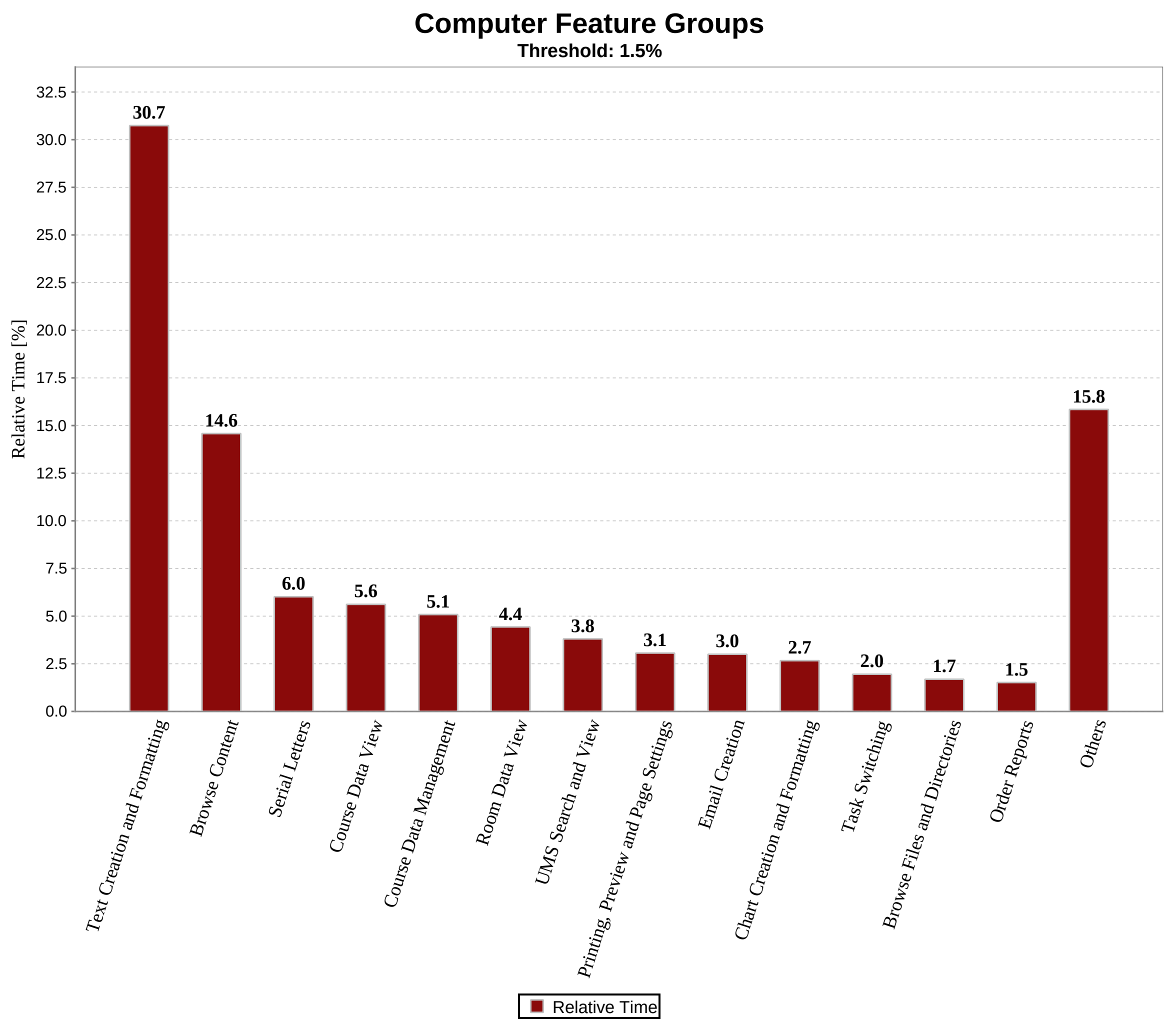


Figure 6.15: Computer Related Feature Groups.

notice this event with the setup of the FUSS-B study. On the other side if the user instead decides to save the file with the save button in the toolbar or with choosing the save command from the application menu it will be possible for the coders to notice these action events. If an exact analysis of such events was necessary the study design could be extended for example with a keylogger application.

The *Task Switching* feature group is named in full length *Task Switching (taskbar or ALT-TAB dialog)*. To summarise the definitions in the coding guidelines [Brückler, 2005, Appendix A] regarding this feature group:

- This feature group will be only coded if the duration of this task switching is more than one second.
- The *Task Switching* group is coded if an application is chosen in the task bar or with the *ALT-TAB* shortcut. If several applications are closed or minimised one after another this overall procedure is also coded as The *Task Switching*.

Count Feature Group	Number of Occurrences	Percent in Relation to Overall Counts
Task Switching	69	4.26%
Save	43	2.65%
Program options	13	0.80%
Using Help	6	0.37%

Table 6.1: Countable Feature Group Occurrences.

- While coding the *Task Switching* feature group the tool is always coded using *File Manager* and the application to use is the *Windows Explorer*.

6.6 Discussion

For being able to guess the goal in the first recording sessions the users where asked to give a hint for the goal they are trying to achieve. This was dine with a document for a spreadsheet application containing the assumed goals. It was observed in these first sessions that the users where seldom able to assign unique goals if they started a new goal. Other issues were also observed. Sometimes the users presented the goal in the middle of the course because they forgot it before. The FUSS-B finally decided to drop the idea of showing the actual goal by the user. For coding this task a specific goal was defined, the *Usage* goal. The overall percentage of this goal is 0.98% with included non-computer tasks and 1.6% when only considering computer related tasks. For the further analysis and creation of the charts this goal is ignored by means of suitable XPath queries.

Due to the very small number of test participants and data to analyse it is not possible to create any statistically firm statements in any way. In several places unexpected findings are explained with one single special task an user has done. That is way, as already mentioned at the beginning of this chapter, all of the presented results should be rather seen as proof of concept of the overall study design and implementation instead of an attempt where statistical evidences or correlations could or should be found.

The considerably large and for the users obviously annoying waiting times from the *SAP* application are mainly caused because only the web client version is actually used. Even worse the main server is not located in the university in Graz but in Vienna and all users authenticate and connect with a separate *Virtual Private Network* (VPN) connection. One immediate finding of this study although more usability than usage related is the very bad usability regarding response time of the observed used *SAP* client. It is not acceptable observing an everyday work important application more than half of the time (relative waiting times of 57%) not responsive. Of course the users try to bypass the problems and try to include other tasks in the waiting periods but usability guidelines state that an application should react as quick as possible meaning at maximum a few seconds, normally below a second. The *SAP* application with the actually used installation has often waiting periods up to 30 seconds, sometimes even lasting several minutes (!).

Although online applications running in a browser will never achieve the usability and response time of good client applications, the *TUG-Online* application shows what is possible. The server is located in-house and very quick. A relative waiting time below 3% was achieved. This is a very good value for a web application.

Chapter 7

Outlook

“ The best way to predict the future is to invent it.”

[Alan Curtis Kay.]

It is an ongoing development to use highly portable usability laboratories. In opposite to common fixed locations for usability testing field observations are easily possible with portable solutions. Usage studies implicitly need portable equipment. Real users can be observed while performing their daily tasks in their common environment and context.

Although hardware based mobile solutions are available particularly the software based solutions gain more and more attention lately [Spolsky, 2005]. A more or less complete usability laboratory can be cost-effective simulated by means of a standard PC hardware, a webcam, a microphone and the necessary software. The software may be a proprietary all-in-one software or a self designed solution consisting of several different applications such as presented in this thesis.

The proposed setup seems to be adequate for the continuation of the FUSS-B in the field with more participants being observed and analysed. Nevertheless there are many improvement ideas regarding the different phases.

7.1 Possible Recording Improvements

One shortcoming of the actual recording solution presented in Section 4.1 and Section 3.6.3 is the necessary installation of the VNC server software on the User PC. Additionally administrative privileges are required for this installation.

7.1.1 Installation Avoidance

TeamViewer by Rossmanith GmbH [2005] is another VNC based software. TeamViewer requires no installation and no administrative rights. TeamViewer is Open Source software licensed under the GPL. In order to bypass firewall issues the DynGate proprietary router software is offered commercially. A possibility to directly start the clientside VNC server application from a *Compact Disc Read-Only Memory* (CD-ROM) could be possible with the TeamViewer software. It is recommended to evaluate this software for being able to avoid any installation of software thus minimising the impact on the User PC.

7.1.2 Connection Alternatives

Another limitation of the actual setup is the necessary connection to the LAN. Several solutions are possible. Theoretically a second PCI network card could be installed on the User PC. Due to the necessary hardware change this idea is dropped. An adaption of this idea is to use USB based network cards such as wireless or Bluetooth adapters and build a second network for being able to connect to the VNC server of the User PC without changing the normal LAN settings. Due to the newly created network with an own IP address range no valid LAN IP address is necessary anymore. Another alternative is to use direct USB to USB connections or other forms of cable connection like firewire, serial or parallel direct cable connections with software simulated private networks.

7.1.3 Video and Audio Streaming

One drawback of the FUSS-B recording design is the connection of the webcam and microphone to the Recorder PC hence the Recorder PC can only have a given distance to the user regarding the possible and available cable lengths. To overcome this problem the video of the webcam and the audio could be streamed by the User PC over the network. The obvious shortcomings with this solution are the increased network traffic and the necessary performance impact on the User PC. Additional software has to be installed or at least started on the User PC.

7.1.4 Recording Automation

On the Recorder PC it is necessary to manually start the used software applications and to arrange the windows sizes and positions. If the screen recording with Camtasia is started, the region for recording and the file save location will have to be chosen too. It should be evaluated how these procedures can be automated. A *Software Development Kit* for controlling Camtasia programmatically is available. Another problem is the two GB file size limit from Camtasia. A very powerful Recording PC may be able to already store the videos in XviD format instead of TSCC thus avoiding the problem.

Also other screen recording software applications may be evaluated. There are many alternatives available. The Windows Media Encoder 9 and the Wink (supports no audio recording) software are both available as freeware and can save screen recordings in different formats.

Also Linux Open Source solutions are available including *vncrec* and *vnc2swf* for directly recording VNC sessions. These recorded sessions can be converted to XviD or other codecs with the *transcode* software for example. [Washko, 2004; de Alwis, 2004]

7.1.5 Visionary Recording Hardware

One visionary all-in-one tapeless recording system is hardware based using a setup like shown in Section 3.2. This hardware could have some buttons in to front to control play, record and other options. With a *Liquid Crystal Display* (LCD) the actual state is shown. All of the necessary hardware is built into a mini barebone system in the size of a common amplifier component for an audio system. No screen, keyboard or mouse are necessary. Interfaces for the connection of the User PC monitor signal, a webcam and a microphone exist. Some software is written for controlling the different applications for screen recording, video and audio recording. The VisionRGB-PRO card in connection with splitter could be used for capturing the monitor signal.

Probably the implementation of such a solution is not as simple in the real world as it seems in theory. Challenges like for example space in the bare bone box, performance and stability of the overall system and insufficient or no *Application Programming Interfaces* (APIs) for controlling the necessary applications could lead to non-trivial problems.

According to the previous ideas instead of using a hardware based setup also a more software based design is possible. The direct connection of the monitor signal from the User PC is replaced with some other kind of connection (LAN, USB, wireless, ..) (see Section 7.1.2) and additional software running on the User PC (maybe started from a CD-ROM) is required.

7.2 Coding Tool Ideas

The Anvil software used for the coding process fulfilled most of the expectations of the FUSS-B team. Anvil is described in detail in Section 3.4.6. It has a highly intuitive user interface thus is easy to learn. It is a cross-platform application, uses established standards (XML) and is extensible with a plugin API.

A minor drawback is the sometimes not perfectly synchronised audio track of the used AVI XviD file. Anvil has smaller bugs when editing annotations particularly using the extend feature of track elements. Thus the most requested feature would be an always available undo function.

The coding process requires normally several passes of watching the video. Due to the audio information and coding accuracy it is necessary for the first pass to watch the video in realtime. However this would not always be necessary for the other passes. Therefore one useful feature would be the possibility to play the video two or three times the normal speed. Actually it is only possible to slow down the video playback speed.

Despite the satisfaction with the actually used Anvil application, the upcoming SignStream version 3 should be evaluated anyway. Up to four synchronised video files are allowed which could change the necessary combined recording of screen and video hence avoiding the large files. The actual version 2.2 application is only running on Macintosh operating systems however the not yet available version 3 is ported to Java and should run on many platforms. SignStream version 3 will be released under an Open Source license

7.3 Proposed FUSSY Improvements

The FUSSY software is the first iteration of the analysis software developed for being able to generate results for this study out of imported Anvil files. Software can always be improved thus some ideas for improving the FUSSY software are given.

FUSSY will print out warnings if the coded data conflicts with the defined coding taxonomy or if consistency problems of the data are detected. This warnings show the position in the internally used XML however do net reflect the original position in one of the imported Anvil files. This should be improved.

An installer for FUSSY would ease the installation process. An installer application should be chosen for the creation of an installation package. For example the *Ant Installer Generator* (Antigen) takes an Ant script and is able to create graphical installers from Ant scripts.

FUSSY actually only supports command line options. It is necessary to change the Java code if a new type of chart should be created. In future FUSSY should use a configuration file for being able to define many more options. The different file names of the generated charts, their title, their thresholds, the used resolution for every single chart, the used XPath queries for creation of the charts, the used labels, font sizes are some examples of proposed supported options in this configuration file. The XML format is recommended as configuration file format.

The used taxonomy respectively the used coding specification for FUSSY is hardcoded. FUSSY should have a freely defineable configuration probably including informations in the Anvil specification file. FUSSY could be implemented as Anvil plugin for being able to create charts right out of the Anvil application. The necessary restructuring of the FUSSY software and the provided advantages should be evaluated.

7.4 FUSS-B Study Future

The relatively small sample of three administrative assistants, everyone working on an institute of the University of Technology in Graz, is not statistically significant. Nevertheless some interesting findings particularly regarding waiting time could be shown. At first this FUSS-B study should continue with more participants to be able to generate statistically firm results.

Furthermore the FUSS-B study is not intended to be a standalone study. This developed framework is able to conduct different kind of usage studies also regarding other business areas such as designers, management, system administrators, programmers and others. New taxonomies will have to be (iteratively) developed for these other studies. The framework does not only support usage studies, also usability issues can be taken into consideration. It is even possible to add new tracks and annotations to already existing Anvil codings afterwards thus being able to analyse completely different aspects than considered initially in the study design.

Chapter 8

Concluding Remarks

In this thesis the design and implementation of the *Framework for Usage Studies of Software in Business Environments* (FUSS-B) usage study was developed. This thesis covered particularly the technical aspects. Chapter 2 introduced the study goal. The main goal is to reveal how users actually work in their actual working environment in their common environment including different task contexts.

In Chapter 3 evaluations were conducted and finally a software based recording was chosen. The VNC application in connection with Camtasia Recorder stores the screen and video from the user using a separate PC connected to the user's network. The Anvil software was picked for the coding stage. It was decided to implement the FUSSY software for the analysis.

After having outlined the FUSS-B study design in Chapter 4 a highly portable laptop based solution was presented for the recording. An overview about all necessary software for the study including the Anvil application for coding was presented. Afterwards this implemented analysis tool named FUSSY was presented in Chapter 5. The installation, a manual, the internal design and the used libraries are shown. The development and building process with the Ant tool was considered too. FUSSY shows created charts on screen or stores them in PDF or PNG image format. These created results were presented in Chapter 6. Due to the available data no statistically firm statements could be made nevertheless interesting results such as the the annoying waiting time of the online SAP application or the distribution of non-computer related tasks could be acquired.

In the Chapter 7 the ongoing development concerning portable usability laboratories was emphasised. Improvements in the recording stage possibly enabling further minimisation of the impact on the observed user, particularly regarding the software on the User PC were proposed. New useful, however not yet implemented features for Anvil were recommended. This thesis concluded with ideas and recommendations for potential future FUSS-B developments based on this introduced setup for recording, coding and analysis.

Appendix A

Anvil Coding Specification

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<annotation-spec>
  <doc>
    This annotation is for the FUSS-B usage study.
  </doc>

<!-- **************** HEAD ********************** -->
 <head>
  <valuetype-def>
   <valueset name="featuregroupType">
     <value-el color="#0000FF">Asset reporting[Anlagenbestandsuebersicht]</value-el>
     <value-el color="#ffaa70">Away from workplace and unknown or just unknown goal</value-el>
     <value-el color="#0000D8">Browse content(search, view, scrolling and read-only)</value-el>
     <value-el color="#00009F">Browse files and directories</value-el>
     <value-el color="#0000E8">Calculation</value-el>
     <value-el color="#0000E8">Calendar data management</value-el>
     <value-el color="#0000E8">Calendar data view</value-el>
     <value-el color="#0000DF">Chart creation and formatting</value-el>
     <value-el color="#aaaaFF">Close and minimize</value-el>
     <value-el color="#0000D8">Complete a form (including sending)</value-el>
     <value-el color="#0000CF">Cost centre reports [Kostenstellenberichte]</value-el>
     <value-el color="#0000C8">Course data management</value-el>
```

```
<value-el color="#0000C8">Course data view</value-el>
<value-el color="#0000BF">Customer order data management [Kundenauftrag ausgehend]</value-el>
<value-el color="#0000BF">Customer order data view [Kundenauftrag ausgehend]</value-el>
<value-el color="#0000B8">Data management</value-el>
<value-el color="#00009D">Desktop/Startmenu view or customize</value-el>
<value-el color="#0000AF">Downloading files</value-el>
<value-el color="#0000A8">Drawing</value-el>
<value-el color="#00009F">Email creation (including sending)</value-el>
<value-el color="#00009F">Email view</value-el>
<value-el color="#000098">Exam data management (dates and marks)</value-el>
<value-el color="#000098">Exam data view (dates and marks)</value-el>
<value-el color="#00008F">Get additional informations (e.g. hotels)</value-el>
<value-el color="#000088">Importing and embedding data (e.g. images)</value-el>
<value-el color="#00007F">Input start, destination and settings</value-el>
<value-el color="#000078">Invoice data management [Faktura verwalten]</value-el>
<value-el color="#000078">Library data management</value-el>
<value-el color="#7777bb">Library data view</value-el>
<value-el color="#00006F">Modify files and directories (includes new, move, delete, rename and copy)</value-el>
<value-el color="#000068">Navigation (e.g. bookmarks, history)</value-el>
<value-el color="#2F2F2F">Non computer work (with known goal)</value-el>
<value-el color="#aaaaFF">Open and new</value-el>
<value-el color="#000058">Order data management [Bestellwesen]</value-el>
<value-el color="#000058">Order data view [Bestellwesen]</value-el>
<value-el color="#00004F">Order reports [Auftragsberichte]</value-el>
<value-el color="#000048">Presentation preview</value-el>
<value-el color="#00003F">Printing, preview and page settings</value-el>
<value-el color="#00003F">Printspooler view and management</value-el>
<value-el color="#000038">Program options</value-el>
<value-el color="#00002F">Publication data management</value-el>
<value-el color="#00002F">Publication data view</value-el>
<value-el color="#000028">Receiving goods [Wareneingang]</value-el>
<value-el color="#00001F">Reservation data management [Reservierungen verwalten]</value-el>
<value-el color="#00001F">Reservation data view [Reservierungen]</value-el>
<value-el color="#000018">Room data management (reservation, calendar of events [Terminkalender d. Raumes])</value-el>
<value-el color="#000018">Room data view (e.g. room search, view, calendar of events [Terminkalender d. Raumes])</value-el>
<value-el color="#aaaaFF">Save</value-el>
<value-el color="#00000F">Search address</value-el>
<value-el color="#8080FF">Search files and directories (with searchcriteria)</value-el>
<value-el color="#7F7FFF">Serial letters</value-el>
<value-el color="#6F6FFF">Spell checking or grammar checking</value-el>
```

```
    <value-el color="#5F5FFF">Staff data management [Personalverwaltung: Reisekosten, Krankmeldung, Urlaubsscheine]</value-el>
    <value-el color="#7F7FFF">Student data management</value-el>
    <value-el color="#7F7FFF">Student data view</value-el>
    <value-el color="#00009B">Task Switching (taskbar or ALT-TAB dialog)</value-el>
    <value-el color="#4F4FFF">Text creation and formatting</value-el>
    <value-el color="#3F3FFF">UMS search and view (e.g. persons, courses,...)</value-el>
    <value-el color="#2F2FFF">Using help</value-el>
    <value-el color="#2F2F2F">Usage</value-el>
  </valueset>

  <valueset name="toolType">
    <value-el color="#ffaa08">Accounting</value-el>
    <value-el color="#ffaa10">Calendar, Appointment and Task Management</value-el>
    <value-el color="#ffaa60">Car Reservation System</value-el>
    <value-el color="#ffaa20">Contact Management</value-el>
    <value-el color="#ffaa20">Dictionary</value-el>
    <value-el color="#ffaa00">Document Viewer</value-el>
    <value-el color="#ff7718">Emails</value-el>
    <value-el color="#ffaa68">Fax</value-el>
    <value-el color="#ffaa00">File Converter</value-el>
    <value-el color="#ff4400">File Manager</value-el>
    <value-el color="#ffaa58">Hotel Reservation System</value-el>
    <value-el color="#ffaa00">Image Editing</value-el>
    <value-el color="#ffaa80">Other Tool</value-el>
    <value-el color="#ffaa40">Presentations</value-el>
    <value-el color="#ffaa30">Printspooler</value-el>
    <value-el color="#ffaa48">Routing</value-el>
    <value-el color="#ffaa00">Spreadsheet</value-el>
    <value-el color="#ffaa00">Telephone Book</value-el>
    <value-el color="#ffaa00">University Management System</value-el>
    <value-el color="#ffaa30">Webbrowsing</value-el>
    <value-el color="#ffaa28">Wordprocessing</value-el>
 </valueset>

 <valueset name="applicationType">
    <value-el color="#CCCC00">Acrobat</value-el>
    <value-el color="#CCCC00">Acrobat Reader</value-el>
    <value-el color="#CCCC00">Acrobat Distiller</value-el>
    <value-el color="#CCCC00">Corel Photopaint</value-el>
    <value-el color="#CCCC00">Herold Online Telephone Book</value-el>
```

```
   <value-el color="#CCCC00">Leo Online Dictionary</value-el>
   <value-el color="#CCCC00">Mozilla Firefox</value-el>
   <value-el color="#999900">Mozilla Thunderbird Emails</value-el>
   <value-el color="#EEEE00">MS Excel</value-el>
   <value-el color="#BBBB00">MS Internet Explorer</value-el>
   <value-el color="#777700">MS Outlook</value-el>
   <value-el color="#EEEE00">MS Powerpoint</value-el>
   <value-el color="#FFFF00">MS Windows Explorer</value-el>
   <value-el color="#AAAA00">MS Word</value-el>
   <value-el color="#ffaa00">Online Car Reservation</value-el>
   <value-el color="#ffaa00">Online Hotel Reservation</value-el>
   <value-el color="#FFFF00">Online Route Planing</value-el>
   <value-el color="#CCCC00">Pegasus Mail</value-el>
   <value-el color="#CCCC00">Photoshop Pro</value-el>
   <value-el color="#aaaa77">SAP</value-el>
   <value-el color="#CCCC00">The Bat</value-el>
   <value-el color="#eeee99">TUG-Online</value-el>
   <value-el color="#CCCC00">Windows Printspooler</value-el>
 </valueset>

<valueset name="browserType">
   <value-el color="#CCCC44">Mozilla Firefox</value-el>
   <value-el color="#BBBB44">MS Internet Explorer</value-el>
   <value-el color="#ddddbb">Opera</value-el>
 </valueset>

<valueset name="activityType">
   <value-el color="#00FF00">Advertising and marketing</value-el>
   <value-el color="#00FF00">Asset management [Sachgüter, Inventar]</value-el>
   <value-el color="#00EF00">Assist colleagues</value-el>
   <value-el color="#00E800">Budget planning</value-el>
   <value-el color="#00DF00">Cash administration</value-el>
   <value-el color="#00D800">Course administration</value-el>
   <value-el color="#00CF00">Cross-checking [Kosten- und Finanzüberprüfung]</value-el>
   <value-el color="#00C800">Display cabinet [Schaukasten]</value-el>
   <value-el color="#00BF00">Exam administration</value-el>
   <value-el color="#00B800">Filing [Ordner-/Kartei- und Dateiverwaltung]</value-el>
   <value-el color="#00AF00">Holiday scheduling [Urlaubsplanung]</value-el>
   <value-el color="#00A800">Homepage content management</value-el>
   <value-el color="#009F00">Inventory management [Verbrauchsmaterial]</value-el>
```

```
    <value-el color="#009800">Invoice processing [Fakturierung]</value-el>
    <value-el color="#008F00">Loan and return</value-el>
    <value-el color="#008800">Making appointments, update calendar</value-el>
    <value-el color="#007F00">Manage lecture notes</value-el>
    <value-el color="#007800">Official invitation (for Quästur)</value-el>
    <value-el color="#006F00">Order management [Bestellabwicklung]</value-el>
    <value-el color="#006800">Ordering, booking or hiring</value-el>
    <value-el color="#005F00">Organise catering</value-el>
    <value-el color="#005800">Organise equipment</value-el>
    <value-el color="#004F00">Prepare proceedings [Material/Unterlagen aufbereiten]</value-el>
    <value-el color="#004800">Prepare travel documents (maps, routing)</value-el>
    <value-el color="#003F00">Publication management</value-el>
    <value-el color="#003800">Receiving goods [Wareneingang]</value-el>
    <value-el color="#002F00">Recruiting, Register or Unregister staff [Rekrutierung, An-Abmeldung Mitarbeiter]</value-el>
    <value-el color="#002800">Registration</value-el>
    <value-el color="#001F00">Request travel approval [Reiseantrag (Quästur)]</value-el>
    <value-el color="#001800">Room administration</value-el>
    <value-el color="#000F00">Search and price comparison</value-el>
    <value-el color="#000800">Sick leave [Krankmeldung]</value-el>
    <value-el color="#000100">Student support</value-el>
    <value-el color="#ffaa00">Todo list management</value-el>
    <value-el color="#ffaa00">Travel expensive report</value-el>
    <value-el color="#ffaa00">Unassigned activity</value-el>
  </valueset>

<valueset name="goalType">
    <value-el color="#FF0000">Accounting</value-el>
    <value-el color="#F80000">Event organisation</value-el>
    <value-el color="#B80000">Infrastructure</value-el>
    <value-el color="#EF0000">Library management</value-el>
    <value-el color="#E80000">Office management</value-el>
    <value-el color="#DF0000">Public relations</value-el>
    <value-el color="#D80000">Staff support [Mitarbeiterunterstützung]</value-el>
    <value-el color="#CF0000">Supplies and inventory</value-el>
    <value-el color="#C80000">Teaching support</value-el>
    <value-el color="#BF0000">Travel planning</value-el>
    <value-el color="#B80000">Unassigned goal</value-el>
    <value-el color="#2F2F2F">Usage goal</value-el>
  </valueset>
  </valuetype-def>
 </head>

<!-- ************************* BODY ******************** -->
 <body>
  <track-spec name="featuregroup" type="primary" color-attr="type">
    <doc>
     Feature groups
    </doc>
    <font color="white" size="12"/>
    <attribute name="type" valuetype="featuregroupType">
      <doc>
        The type of the used feature group.
      </doc>
    </attribute>
  </track-spec>

  <track-spec name="application" type="span" ref="featuregroup" color-attr="type">
   <doc>
     Applications
    </doc>
    <attribute name="type" valuetype="applicationType">
      <doc>
        The type of the used application.
      </doc>
    </attribute>
  </track-spec>

  <track-spec name="tool" type="span" ref="featuregroup" color-attr="type">
    <doc>
     Tools
    </doc>
    <attribute name="type" valuetype="toolType">
      <doc>
        The type of the used tool.
      </doc>
    </attribute>
  </track-spec>

  <track-spec name="browser" type="span" ref="featuregroup" color-attr="type">
 <doc>
   Browser
  </doc>
  <attribute name="type" valuetype="browserType">
    <doc>
      The used browser.
    </doc>
  </attribute>
</track-spec>

<track-spec name="activity" type="span" ref="featuregroup" color-attr="type">
  <font color="white" size="12"/>
 <doc>
   Activities
  </doc>
  <attribute name="type" valuetype="activityType">
    <doc>
      The type of the activity.
    </doc>
  </attribute>
</track-spec>

<track-spec name="goal" type="span" ref="featuregroup" color-attr="type">
  <font color="white" size="12"/>
 <doc>
   Goals
  </doc>
  <attribute name="type" valuetype="goalType">
    <doc>
      The defined goal.
    </doc>
  </attribute>
</track-spec>

<track-spec name="Waiting for application" type="primary" color-attr="application">
  <attribute name="application" valuetype="applicationType">
  </attribute>
  <attribute name="featuregroup" valuetype="featuregroupType">
  </attribute>
</track-spec>
```

```
  <track-spec name="telephone" type="primary">
  </track-spec>

  <track-spec name="audio" type="waveform" height="1.5"/>

</body>
</annotation-spec>
```

Appendix B

FUSSY API Documentation

B.1 fussy

Package fussy

This package contains almost all classes for the FUSSY application. This application is used for the FUSS-B usage study for creating charts out of Anvil XML files.

The main method from the Main class is used for starting the application. The passed options are evaluated and stored in an object which is an instance of the Options class. Also the classes for handling the different tracks and their contained track elements are contained in this package. For applying the coding scheme primarily the Taxonomy class is used.

Utility classes for creating charts are contained in the fussy.chart package.

Interfaces

FussyConstants	All FUSSY specific constants are contained in this class.

Classes

Main	This is the FUSSY Main class containing the main method for starting.
Options	With the help of this class the passed options are stored.
PrimaryTrack	The used class for all primary tracks.
ReferenceTrackElement	The ReferenceTrackElement is used in span tracks as track element.
SpanTrack	This class is used for all span tracks.
Taxonomy	All data from the imported files is handled with the help of this class.
TrackElement	All track elements are instances of this class.
TrackElementSummary	Object instances of this class gather summarisation data of all elements with the same name.
WaitingElement	A special extension of the TrackElement class used for waiting tracks.
WaitingTrack	The class for handling the waiting track.
XmlErrorHandler	With this class XML parsing errors are handled and shown.
XmlUtil	This class provides common utility methods regarding the handling of XML files and XML data structures.

Exceptions

FussyException	All occurring errors are handled by means of this FussyException or other derived classes.
OptionException	The OptionException is internally used in the case of invalid options.

B.1.1 FussyConstants

Interface fussy.FussyConstants

```
fussy.FussyConstants
```

public interface fussy.FussyConstants
All FUSSY specific constants are contained in this class.

Variables

`public static final String` **`NON_COMPUTER_WORK_FEATUREGROUP`**

The ”Non computer work (with known goal)” String

`public static final String` **`PRODUCTIVE_WAITING_TIME`**

The ”productivewaitingtime” String

`public static final String` **`TIME`**

The ”time” String

`public static final int` **`TYPE_APPL_WAITING`**

A stacked bar chart type with included waiting

`public static final int` **`TYPE_APPL_WAITING_WITH_PRODUCTIVE`**

A stacked bar chart type with included waiting and productive waiting

`public static final int` **`TYPE_NORMAL`**

The normal chart type with one column and an aggregation value above.

`public static final int` **`TYPE_STACKED_WAITING`**

The stacked bar chart type with waiting included.

`public static final int` **`TYPE_STACKED_WAITING_RELATIV`**

A normal bar chart type with two relative waiting time columns

`public static final int` **`TYPE_WITHNONCOMP`**

A stacked bar chart type with included non-computer time.

`public static final String` **`WAITING_TIME`**

The ”waitingtime” String

B.1.2 FussyException

Class fussy.FussyException

```
java.lang.Object
     |
     +----java.lang.Throwable
               |
               +----java.lang.Exception
                         |
                         +----fussy.FussyException
```

public class fussy.FussyException
`extends java.lang.Exception`

All occurring errors are handled by means of this FussyException or other derived classes.

Constructors

`public FussyException(String message)`

Constructs a new exception with the passed string.

Parameters:

`message` The error message.

B.1.3 Main

Class fussy.Main

```
java.lang.Object
      |
      +----fussy.Main
```

public abstract class fussy.Main
`extends java.lang.Object implements FussyConstants`

This is the FUSSY Main class containing the main method for starting. This class uses other utility classes. The most important utility classes are the Taxonomy and the FussyChartUtil class.

Methods

`public static Options` **`getOptions`**`()`

Get the used options, these are either the default values or defined by command line parameters.

Returns: The actually used Options object.

`public static void` **`main`**`(String args[])`

This is the entry point of the FUSSY application.

Parameters:

`args` The given startup parameter (if any).

`public static void` **`writeWarning`**`(String warning)`

Write a warning if applicable. Warnings are only written if not suppressed by means of a given option.

Parameters:

`warning` The String to be written as warning.

B.1.4 OptionException

Class fussy.OptionException

```
java.lang.Object
      |
      +----java.lang.Throwable
              |
              +----java.lang.Exception
                      |
                      +----fussy.FussyException
                              |
                              +----fussy.OptionException
```

public class fussy.OptionException
`extends FussyException`

The OptionException is internally used in the case of invalid options.

Constructors

`public OptionException(String message)`

Constructs a new exception with the passed string.

Parameters:

`message` The error message.

B.1.5 Options

Class fussy.Options

```
java.lang.Object
     |
     +----fussy.Options
```

public class fussy.Options
`extends java.lang.Object`

With the help of this class the passed options are stored. Default values are also handled here.

Constructors

`public Options()`

Creates a new Option object.

Methods

`public File[] getAnvilFiles()`

All handled Anvil files are returned.

Returns: All Anvil files as File[]

`public int getHeight()`

Get the achtual value for height.

Returns: The height is returned.

`public int getWidth()`

Get the actal width value.

Returns: The width is returned.

`public boolean isDebug()`

Check if debug is enabled.

Returns: True if debug is activated.

`public boolean isExecuteFussy()`

Should the application execute normally or not. Fussy will exit if help, license or version is shown.

Returns: If false is returned the application will stop the further execution.

`public boolean isSavePDF()`

Should images in PDF format be saved?

Returns: True if PDF will be saved.

`public boolean isSavePNG()`

Should images in the PNG format be saved?

Returns: If true images will be saved.

`public boolean isShowChart()`

Show the created charts on screen?

Returns: If true the charts are shown on screen.

`public boolean` **`isSuppressWarnings`**`()`

Check if warnings shiould be suppressed.

Returns: True if warnings should be suppressed.

`public boolean` **`isUseTime`**`()`

Should time or percent be used as units.

Returns: True if time should be used.

`public static Options` **`parseArgs`**`(String args[]) throws OptionException`

Parse the command array which is passed to the application (if any).

Parameters:

`args` The arguments from the command call

Returns: An option object with initialised values also considering default values.

Throws: `OptionException` - Is thrown if an invalid option is given.

`public static void` **`printUsage`**`()`

Print the supported options.

`public void` **`setDebug`**`(boolean debug)`

Set debug active or not.

Parameters:

`debug` Set true to enable debug.

`public void` **`setExecuteFussy`**`(boolean executeFussy)`

Should the application execute normally or not. Fussy will exit if help, license or version is shown.

Parameters:

`executeFussy` If false the application will exit.

`public void` **`setHeight`**`(int height)`

Set the height.

Parameters:

`height` The height to set.

`public void` **`setResolutionStr`**`(String res) throws OptionException`

The passed resolution string is parsed, width and height are set. If the string is invalid an exception with a correspondig error message is thrown.

Parameters:

`res` The resolution string given as parameter.

Throws: `OptionException` - This exception is thrown if the resolution string is invalid

`public void` **`setSavePDF`**`(boolean savePDF)`

Should images in PDF format be saved?

Parameters:

`savePDF` Set to true for having PDF files saved.

`public void` **`setSavePNG`**`(boolean savePNG)`

Should images in the PNG format be saved?

Parameters:

`savePNG` Set to true if images should be saved.

`public void` **`setShowChart`**`(boolean showChart)`

Show the created charts on screen?

Parameters:

`showChart` True if the charts should be shown.

`public void` **`setSuppressWarnings`**`(boolean suppressWarnings)`

Set if warnings shiould be suppressed.

Parameters:

`suppressWarnings` True if all warnings should be suppressed.

`public void` **`setUseTime`**`(boolean useTime)`

Set if time or percent should be used as units.

Parameters:

`useTime` True if time should be used.

`public void` **`setWidth`**`(int width)`

Set the width.

Parameters:

`width` The width to set.

`public String` **`toString`**`()`

Create a string containing the actual options.

B.1.6 PrimaryTrack

Class fussy.PrimaryTrack

```
java.lang.Object
     |
     +----fussy.PrimaryTrack
```

public class fussy.PrimaryTrack
`extends java.lang.Object`

The used class for all primary tracks. Most of the functionality of the track handling is contained in this class. The track elements and summary elements are also stored in this class.

Constructors

`public PrimaryTrack(String xmlTrackName, String xmlElementName)`

Constructs a new primary track.

Parameters:

`xmlTrackName` The XML track element name.

`xmlElementName` The element name.

Methods

`public void` **`addTrackElement`**`(TrackElement element, Element el) throws FussyException`

Method for adding a track element. While adding the element the different track summaries are created (if necessary) and used on the fly.

Parameters:

`element`

Throws: `FussyException` - Is thrown if an error occurred.

`public TrackElement` **`createTrackElement`**`(String name, Element el) throws FussyException`

The track specific element is created with this method.

Returns: The specific TrackElement for this track.

Throws: `FussyException` - Is thrown if an error occurred.

`public void` **`createTrackElementsXml`**`(Node top)`

The track elements XML structure is created.

Parameters:

`top` An xml track element.

`public void` **`createTrackSummaryXml`**`(Node top)`

This method is used for creating the summary XML elements.

Parameters:

`top` The top element of the summary element values.

`public void` **`debugSummary`**`()`

Print out a summary of all track summary elements. This method only prints output after createTrackSummaryXml was called !

`protected boolean` **`emitNonActiveWaitingTime`**`()`

Used if the internal XML is created. The default value is false.

Returns: true if the waiting time element should be included.

`public float` **`getEndTimeFromIndex`**`(int index)`

Helper method for gaining the end time of a given element. First the element with the passed index is acquired, then the end time of this element is returned.

Parameters:

`index` The index of the concerning element.

Returns: The wanted time in seconds.

`public float` **`getStartTimeFromIndex`**`(int index)`

Helper method for gaining the start time of a given element. First the element with the given index is acquired, then the start time of this element is returned.

Parameters:

`index` The index of the concerning element.

Returns: The wanted time in seconds.

`public TrackElement` **`getTrackElement`**`(int index)`

Acquire the element with the given index.

Parameters:

`index` The index for the wanted element.

Returns: The wanted element.

`public TrackElement` **`getTrackElementForTime`**`(float time)`

A track element is searched for a given point in time.

Parameters:

`time` The point in time for which an element is searched for.

Returns: A TrackElement at a given time is returned if available, else null is returned.

`public List` **`getTrackElementsForTimeInterval`**`(float starttime, float endtime)`

Method for gaining all track elements in a given intervall. TrackElements that are not fully contained are also included in the returned list.

Returns: A list of TrackElements contained in the given interval (the TrackElements that are not fully contained are also included) is returned.

`public int` **`getTrackIndexOffset`**`()`

Get the actual index offset for this track. Used if several Anvil files are put together.

Returns: The actual index offset

`public TrackElementSummary` **`getTrackSummaryElement`**`(String name)`

The track summary element with the passed name should be acquired.

Parameters:

`name` The (unique) name of the summary element.

Returns: The track summary element with the given name.

`public float` **`getTrackTimeOffset`**`()`

Get the actual time offset in seconds for this track. Used if several Anvil files are put together.

Returns: The actual time offset in seconds.

`public String` **`getXmlTrackElementName`**`()`

The used name for the XML track element is returned.

Returns: The name for the XML track element.

`public String` **`getXmlTrackName`**`()`

The used name for the track XML element is returned.

Returns: The name for the track XML element.

`public Iterator` **`iteratorTrackElements`**`()`

An iterator for all track elements can be acquired.

Returns: The iterator for all track elements.

`protected void` **`setEmitNonActiveWaitingTime`**`(boolean emitNonActiveWaitingTime)`

Set to true if waiting times should be considered in the internal XML for this track. The default value is false.

Parameters:

`emitNonActiveWaitingTime` True if waiting time should be saved in the intermediate XML.

`public void` **`setTrackElementClassName`**`(String trackElementClassName)`

Define the full qualified class name to use when creating new track elements. The default value for primary tracks is ”fussy.TrackElement”.

Parameters:

`trackElementClassName` The trackElementClassName to use for creating new track elements.

B.1.7 ReferenceTrackElement

Class fussy.ReferenceTrackElement

```
java.lang.Object
     |
     +----fussy.TrackElement
               |
               +----fussy.ReferenceTrackElement
```

public class fussy.ReferenceTrackElement
`extends TrackElement`

The ReferenceTrackElement is used in span tracks as track element. This class extends the TrackElement class.

Constructors

`public ReferenceTrackElement(String name)`

Creates a new track element with a given name.

Parameters:

`name` The name for the track element to create.

`public ReferenceTrackElement()`

Creates a new track element.

Methods

`public int` **`getEndIndex`**`()`

Get the end index.

Returns: The end index of this element.

`public int` **`getStartIndex`**`()`

Get the start index.

Returns: The start index of this element.

`public void` **`setEndIndex`**`(int endIndex)`

Set a new end index.

Parameters:

`endIndex` The new end index.

`public void` **`setStartIndex`**`(int startIndex)`

Set a new start index.

Parameters:

`startIndex` The new start index.

B.1.8 SpanTrack

Class fussy.SpanTrack

```
java.lang.Object
     |
     +----fussy.PrimaryTrack
               |
               +----fussy.SpanTrack
```

public class fussy.SpanTrack
`extends PrimaryTrack`

This class is used for all span tracks. In addition to the PrimaryTrack also reference elements are handled here.

Constructors

`public SpanTrack(PrimaryTrack referenceTrack, String xmlTrackName, String xmlElementName)`

Constructs a new span track.

Parameters:

`referenceTrack` The corresponding reference track for this span track.

`xmlTrackName` The track name used in the internal XML structure.

`xmlElementName` The track element name used in the internal XML structure.

Methods

`public void` **`addReferenceTrackElement`**`(ReferenceTrackElement element, Element el) throws FussyException`

A reference element can be added with this method. While adding the element actual times for start, end and sum time are calculated out of the included start and end indices of the element.

Parameters:

`element` A reference element

Throws: `FussyException` - Thrown if an error occurred.

`public PrimaryTrack` **`getReferenceTrack`**`()`

The stored reference track can be acquired with this method.

Returns: The referenced track.

B.1.9 Taxonomy

Class fussy.Taxonomy

```
java.lang.Object
     |
     +----fussy.Taxonomy
```

public class fussy.Taxonomy
`extends java.lang.Object`

All data from the imported files is handled with the help of this class.

Constructors

`public Taxonomy()`

If a Taxonomy object is created with this constructor the used internal structures are initialised.

Methods

`public void addAnvilXmlDocument(Document anvildoc) throws FussyException`

This method handles the import of an Anvil XML annotation file. If called several times the offsets are considered and one structure is created.

Parameters:

`anvildoc` The new file to add.

Throws: `FussyException` - An exception is thrown if an error occurred.

`public Document getUsageOutputDocument()`

Create the internal XML datasstructure containing data from all imported files.

Returns: The created XML structure containing all data.

`public void postProcessAddReferencedAttributes()`

This method should be called after all files are imported. Necessary cross references are added and the different waiting times are handled.

B.1.10 TrackElement

Class fussy.TrackElement

```
java.lang.Object
     |
     +----fussy.TrackElement
```

public class fussy.TrackElement
`extends java.lang.Object`

All track elements are instances of this class.

Constructors

`public TrackElement()`

Creation of a new track element.

`public TrackElement(String name)`

Creation of a new track element with a given name.

Methods

`public void addAdditionalXmlAttribute(String xmlattr, String value)`

This method is used for adding additional attributes to be contained in the XML data structure.

Parameters:

`xmlattr` The name of the xml attribute to add.

`value` The value of this newly created XML attribute.

`public void emitXml(Element xmlel)`

Method for setting the data of this object into different attributes of the given XML element.

Parameters:

`xmlel` The object to fill with data.

`public float` **`getEndTime`**`()`

Get the end time.

Returns: The end time in seconds.

`public Integer` **`getIndex`**`()`

Get the index of this element.

Returns: The index of this element.

`public String` **`getName`**`()`

Get the name of this element.

Returns: The name of this element.

`public float` **`getStartTime`**`()`

Get the start time.

Returns: The start time in seconds.

`public float` **`getTime`**`()`

Get the time duration of this element (end-start)

Returns: The time duration of this element.

`public void` **`setEndTime`**`(float end)`

Set the end time of this element.

Parameters:

`end` The new end time.

`public void` **`setIndex`**`(Integer index)`

Set the index of this element.

Parameters:

`end` The new index value.

`public void` **`setName`**`(String name)`

Set the name of this element.

Parameters:

`name` The new name for this element.

`public void` **`setStartTime`**`(float start)`

Set the start time of this element.

Parameters:

`start` The new start time.

`public void` **`setTime`**`(float time)`

Set the time duration of this element (end-start)

Parameters:

`time` The time duration which should be set.

`public String` **`toString`**`()`

Create a string containing start, end, time, index and name data.

B.1.11 TrackElementSummary

Class fussy.TrackElementSummary

```
java.lang.Object
     |
     +----fussy.TrackElementSummary
```

public class fussy.TrackElementSummary
`extends java.lang.Object`

Object instances of this class gather summarisation data of all elements with the same name.

Constructors

`public TrackElementSummary(String key)`

Create a new summary element with the given key.

Parameters:

`key` The key (=name) of this summary element.

Methods

`public void` **`addApplicationWaitingTime`**`(float waiting)`

Add a application waiting time.

Parameters:

`productiveWaiting` The time to add in seconds.

`public void` **`addProductiveWaitingTime`**`(float productiveWaiting)`

Add a productive waiting time.

Parameters:

`productiveWaiting` The time to add in seconds.

`public void` **`addTrackElementWithEqualKey`**`(TrackElement element) throws FussyException`

An element with the same key is added.

Parameters:

`element` The element to add.

Throws: `FussyException` - This exception is thrown if the key of this object is not identical to the name of the given object.

`public float` **`getApplicationWaitingTime`**`()`

Get the aggregated application waiting time.

`public int` **`getCount`**`()`

Get the count of the already contained elements.

Returns: The actual element count.

`public String` **`getKey`**`()`

Get the key of this summarisation element.

Returns: The key of this object.

`public float` **`getProductiveWaitingTime`**`()`

Get the aggregated productive waiting time.

Returns: The aggregated productive waiting time in seconds.

`public float` **`getTotalTime`**`()`

Get the total time duration of all contained elements.

Returns: The overall time of all elements.

`public String` **`toString`**`()`

Create a string containing debug data.

B.1.12 WaitingElement

Class fussy.WaitingElement

```
java.lang.Object
     |
     +----fussy.TrackElement
               |
               +----fussy.WaitingElement
```

public class fussy.WaitingElement
`extends TrackElement`

A special extension of the TrackElement class used for waiting tracks.

Constructors

`public WaitingElement(String name)`

Create a new waiting track element.

Parameters:

`name` The name of this waiting element. It is waited for applications thus this is always an application name.

Methods

`public void` **`emitXml`**`(Element xmlel)`

This method extends the overwritten method with setting the additional featuregroup XML element.

Parameters:

`xmlel` An XML element.

`public String` **`getFeatureGroupName`**`()`

Get the feature group name of this waiting element.

`public void` **`setFeatureGroupName`**`(String featureGroupName)`

Set the feature group name of this waiting element.

Parameters:

`featureGroupName` The feature group name to set.

B.1.13 WaitingTrack

Class fussy.WaitingTrack

```
java.lang.Object
     |
     +----fussy.PrimaryTrack
               |
               +----fussy.WaitingTrack
```

public class fussy.WaitingTrack
`extends PrimaryTrack`

The class for handling the waiting track.

Constructors

`public WaitingTrack(String xmlTrackName, String xmlElementName)`

Constructs a new waiting track.

Parameters:

`xmlTrackName` The XML track element name.

`xmlElementName` The element name.

Methods

`public TrackElement` **`createTrackElement`**`(String name, Element el)`

Create a new waiting element. This overwritten method addionally handles the feature group being waited for.

B.1.14 XmlErrorHandler

Class fussy.XmlErrorHandler

```
java.lang.Object
     |
     +----fussy.XmlErrorHandler
```

public class fussy.XmlErrorHandler
`extends java.lang.Object implements org.xml.sax.ErrorHandler`

With this class XML parsing errors are handled and shown.

Constructors

`public XmlErrorHandler()`

Creates a new instance of XmlErrorHandler

Methods

`public void` **`error`**`(SAXParseException ex)`

Print out an error message.

Parameters:

`ex` The exception containing the error message.

`public void` **`fatalError`**`(SAXParseException ex) throws SAXException`

Print out a fatal error message.

Parameters:

`ex` The exception containing the error message.

`public void` **`warning`**`(SAXParseException ex)`

Print out a warning message.

Parameters:

`ex` The exception containing the error message.

B.1.15 XmlUtil

Class fussy.XmlUtil

```
java.lang.Object
     |
     +----fussy.XmlUtil
```

public abstract class fussy.XmlUtil
`extends java.lang.Object`

This class provides common utility methods regarding the handling of XML files and XML data structures.

Methods

`public static Element` **`createChildElement`**`(Node parent, String element_name)`

Creates a child element in an XML data structure.

Parameters:

`parent` The parent element of the new element being created.
`element_name` The name of the new element being created.

Returns: The created XML element.

`public static void` **`debugPrintData`**`(Document doc)`

Print the contents of an XML daata structure on standard out.

Parameters:

`doc` The root node of the XML structure.

`public static String` **`domToString`**`(Node node)`

Convert an XML data structure into an XML string.

Parameters:

`node` The root node of the XML structure.

Returns: The string containing the data in serialised form.

`public static String` **`getFirstChildNodeValue`**`(Element el)`

The value of the first child node is acquired.

Parameters:

`el` An XML element.

Returns: The value of the first child node.

`public static Element` **`getFirstElement`**`(Node node)`

Get the first element of an XML node.

Parameters:

`node` The given node.

Returns: The first element of this node.

`public static float` **`getFloatAttribute`**`(Element ele, String attrname)`

Get an attribute as Float object.

Parameters:

`ele` An XML element.

`attrname` The attribute name.

Returns: The value of the wanted attribute of the XML element, 0 will be returned if the attribute is empty or missing.

`public static int` **`getIntAttribute`**`(Element ele, String attrname)`

Get an attribute as int value.

Parameters:

`ele` An XML element.

`attrname` The attribute name.

Returns: The value of the wanted attribute of the XML element.

`public static Integer` **`getIntegerAttribute`**`(Element ele, String attrname)`

Get an attribute as Integer object.

Parameters:

`ele` An XML element.

`attrname` The attribute name.

Returns: The value of the wanted attribute of the XML element.

`public static Element` **`getSecondElement`**`(Node node)`

Get the second element of a node.

Parameters:

`node` The given node.

Returns: The second element of this node.

`public static String` **`getSecondElementFirstChildNodeValue`**`(Element el)`

The value of the first child node of the second element is acquired.

Parameters:

`el` An XML element.

Returns: The value of the first child node of the second element.

`public static Document` **`newDocument`**`()`

Create a new empty XML document.

Returns: A new root node of an empty XML structure.

`public static Document` **`parse`**`(File f) throws ParserConfigurationException, SAXException, IOException`

Read and parse an XML file and create a XML data structure.

Returns: The root node of the created XML data structure.

`public static final void` **`writeDocToFile`**`(Document doc, String tempfilename) throws FussyException`

Serialise an XML daata structure to a file.

Parameters:

`doc` The root node of the XML structure.

`tempfilename` The path and name of the file being created.

`public static final String` **`writeDocToTempFile`**`(Document doc) throws FussyException`

Write an XML document to a temporary file.

Parameters:

`The` root node of an XML data structure.

Returns: The name of the saved file.

B.2 fussy.chart

Package fussy.chart

This package contains two classes for the creation and handling of charts.

The JFreeChart and the iText library are used by the classes of this package.

Classes

FussyChartUtil	An utility class for the creation of charts.
FussyDatasetUtil	A helper class for transforming the existing XML data structures into JFreeChart data structures.

B.2.1 FussyChartUtil

Class fussy.chart.FussyChartUtil

```
java.lang.Object
     |
     +----fussy.chart.FussyChartUtil
```

public abstract class fussy.chart.FussyChartUtil
`extends java.lang.Object implements fussy.FussyConstants`

An utility class for the creation of charts. This class uses the JFreeChart and iText libraries.

Methods

`public static JFreeChart` **`createBarChartFromNodeList`**`(NodeList nodelist, String title, float threshold, String rangeAttr, int type, boolean isUseTime)`

Create a chart out of an XML nodelist data structure.

Parameters:

`nodelist` The nodelist containing the data.
`title` The title of the chart.
`threshold` The threshold to apply.
`rangeAttr` Data values of this attribute name are used for creating the chart.
`type` The type of the chart (see FussyConstants).
`isUseTime` True if time should be used instead of percent.

Returns: The created chart.

`public static JFreeChart` **`createBarChartFromSingleSummaryNode`**`(Node node, String title, float threshold, int type, boolean isUseTime)`

Creates a chart out of a summary node of the XML data structure.

Parameters:

`node` The node containing the data.
`title` The title of the chart.
`threshold` The used threshold.
`type` The type of the chart (see FussyConstants).
`isUseTime` True if time should be used instead of percent.

Returns: The created chart.

`public static void` **`printChartToPDF`**`(JFreeChart chart, int width, int height, String fileName)`

Stores a chart into a PDF file. PDF is a vector format thus the width and height values are only used for layout purposes.

Parameters:

`chart` The chart to save.
`width` The width used for the layout.
`height` The height used for the layout.
`fileName` The file name of the file to be created.

`public static void` **`showChart`**`(JFreeChart barchart, int width, int height)`

Shows a bar chart on the screen.

Parameters:

`barchart` The bar chart to show on the screen.
`width` The width used for the layout.
`height` The height used for the layout.

B.2.2 FussyDatasetUtil

Class fussy.chart.FussyDatasetUtil

```
java.lang.Object
     |
     +----fussy.chart.FussyDatasetUtil
```

public abstract class fussy.chart.FussyDatasetUtil
`extends java.lang.Object implements fussy.FussyConstants`

A helper class for transforming the existing XML data structures into JFreeChart data structures.

Methods

`public static DefaultCategoryDataset` **`createSummaryDefaultCategoryDatasetFromNodeList`**`(NodeList nodelist, float threshold, String sumAttr, int type, boolean isUseTime)`

This methods creates DefaultCategoryDataset data structures for the JFreeChart library out of XML data structures.

Parameters:

`nodelist` The nodelist XML structure containing the data.
`threshold` The threshold to apply.
`sumAttr` The name of the attribute to be shown in the chart.
`type` The type of the chart (FussyConstants).
`isUseTime` True if time should be used instead of percent.

Returns: The created data structure.

Bibliography

Abreu, P., Coloos, J., Mahesa, L. and Minich, T. (2005). *Welcome to virtualdub.org! - virtualdub.org*. `http://www.virtualdub.org/`. 38, 54

Ames, M. (2003). *Distributed Mentor Project 2003*, CHI 2004. `http://www.ocf.berkeley.edu/~morganya/research/dmp/report.html`. 15

Andrews, K. (2004). *Writing a Thesis: Guidelines for Writing a Master's Thesis in Computer Science*, Graz University of Technology, Austria. `ftp://ftp2.iicm.edu/pub/keith/thesis/thesis.zip`. ii

Apache Software Foundation (2005a). *Apache Ant - Welcome*. `http://ant.apache.org/`. 75

Apache Software Foundation (2005b). *Apache Ant User Manual*. `http://ant.apache.org/manual/index.html`. 75, 76

Aufrecht, J. (2005). *Setup Instructions for Tapeless Usability Lab*, Joel S Aufrecht Consulting. `http://aufrecht.org/usability-setup.html`. 15

Biobserve GmbH (2005). *BIOBSERVE-Products overview*. `http://www.usability.biobserve.com/products/index.html`. 12

Black, J. (2002). *BusinessWeek online: Usability Is Next to Profitability*, The McGraw-Hill Companies Inc. `http://www.businessweek.com/technology/content/dec2002/tc2002124_2181.htm`. 2

Brignull, H. (2004). *Morae vs Visual Mark - Battle of the Software-Based Usability Video Labs*. `http://www.id-book.com/harry/morae_vs_visualmark.htm`. 17

Brückler, S. (2005). A Usage Study of Desktop Users: Coding Scheme and Analysis. Master's thesis, University of Technology Graz, Institute for Information Systems and Computer Media (IICM), A-8010 Graz, Austria. ii, 1, 3, 4, 6, 54, 55, 102

Byrne, M. D. (2001). A day in the life of ten WWW users. 5

Byrne, M. D., John, B. E., Wehrle, N. S. and Crow, D. C. (1999). *The tangled Web we wove: a taskonomy of WWW use*. In CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 544–551, New York, NY, USA (1999). ACM Press. 5

Catledge, L. D. and Pitkow, J. E. (1995). *Characterizing browsing strategies in the World-Wide Web*. Computer Networks and ISDN Systems, 27(6):1065–1073. 5

Cockburn, A. and McKenzie, B. (2000). What Do Web Users Do ? An Empirical Analysis Of Web Use. Technical report, University of Canterbury, Department of Computer Science, Christchurch, New Zealand. 4, 5

Cunningham, H. G. (2005). *Welcome Visitors*, Cunningham & Cunningham, Inc., Portland, Oregon. `http://c2.com/cgi-bin/wiki?WelcomeVisitors`. 38

Datapath Limited (2005). *PCI Plug-in RGB Capture Card from Datapath*. `http://www.datapath.co.uk/visRGBPRO.htm`. 11

de Alwis, B. (2004). *Screen Capturing and Playback Using VNC*. `http://www.cs.ubc.ca/~bsd/vncrecording.html`. 105

Diamond Bullet Design (2005). *Usability Glossary: usage study*, Diamond Bullet Design. `http://www.usabilityfirst.com/glossary/term_725.txl`. 3

DivX, Inc. (2005). *DivX, Inc.: Welcome to DivX, Inc.*, DivX, Inc. `http://www.divx.com/corporate/`. 24

DonationCoder.com (2005). *ScreenCasters - DonationCoder.com*. `http://www.donationcoder.com/Reviews/Archive/ScreenCasting/`. 7, 16

Dybkjær, L., Berman, S., Kipp, M., Olsen, M. W., Pirrelli, V., Reithinger, N. and Soria, C. (2001). *Survey of Existing Tools, Standards and User Needs for Annotation of Natural Interaction and Multimodal Data*, ISLE Natural Interactivity and Multimodality Working Group Deliverable D11.1. `http://isle.nis.sdu.dk/reports/wp11/`. 18

Eclipse (2005). *Eclipse*. `http://www.eclipse.org`. 35

Extron Electronics (2005). *Emotia Xtreme*. `http://www.extron.com/product/product.asp?id=emotiaxtreme&subtype=39&view=over#tabs`. 10

Famatech, Inc. (2005). *Radmin - PC Remote Control Software*, Famatech, Inc. `http://www.radmin.com/`. 13

Fisher, C. and Sanderson, P. (1993). *Exploratory sequential data analysis: traditions, techniques and tools*. SIGCHI Bull., 25(1):34–40. 6, 55

Fisher, C. and Sanderson, P. (1996). *Exploratory sequential data analysis: exploring continuous observational data*. interactions, 3(2):25–34. 6, 55

Gilbert, D. and Morgner, T. (2005). *www.jfree.org - JFreeChart*, Object Refinery Limited. `http://www.jfree.org/jfreechart/index.php`. 85

Grabner, M., Kollmann, H. and Andrews, K. (2005a). Different Techniques of Usage and Usability Studies. Technical report, Institute for Information Systems and Computer Media (IICM), A-8010 Graz Austria. 5

Grabner, M., Kollmann, H. and Andrews, K. (2005b). Software Usage Studies. Technical report, Institute for Information Systems and Computer Media (IICM), A-8010 Graz Austria. 2, 6

Grünwald, S. (2003). *The Influence of the Internet and Open Source Software on the Strategy and the Business Model of Enterprises in the Digital Net Economy*. PhD thesis, University of Technology Graz, A-8010 Graz Austria. 4

Kipp, M. (2003). *Anvil 4.0, Annotation of Video and Spoken Language*. Graduate College for Cognitive Science. 34

Kipp, M. (2004a). *Anvil - The video annotation research tool*, German Research Center for Artificial Intelligence (DFKI). `http://www.dfki.de/~kipp/anvil/`. 6, 31, 56

Kipp, M. (2004b). *Anvil - The video annotation research tool*, German Research Center for Artificial Intelligence (DFKI). `http://www.dfki.de/~kipp/anvil/faq.html#length`. 34

Lowagie, B. and Soares, P. (2005). *iText, a Free Java-PDF Library: Home Page*, Adolf Baeyensstraat 121, 9040 Gent, BELGIUM. `http://www.lowagie.com/iText/`. 86

MacLaughlin, D., Neidle, C. and Greenfield, G. (2000). *SignStream User's Guide, Version 2.0. American Sign Language Linguistic Research Project, Report Number 9*. Boston University, Boston, MA, report number 9 edition. `http://www.bu.edu/asllrp/reports.html#RPT9`. 28

Mangold Software & Consulting GmbH (2005). *INTERACT leading the way forward in behavioural observation*. `http://mangold.de/english/intoverview.htm`. 27

Marcus, A. (2002). *The ROI of Usability*, User Experience Magazine. `http://www.upassoc.org/usability_resources/usability_in_the_real_world/roi_of_usability.html`. 2

Mayo, E. (1933). *The human problems of an industrial civilization*. MacMillan, New York, NY, USA. 54

München, L. (2005). *Das Projekt LiMux - Die IT-Evolution bei der Landeshauptstadt*, Landeshauptstadt München. `http://www.muenchen.de/Rathaus/dir/limux/89256/index.html`. 4

Neidle, C. (2000). SignStream: A Database Tool for Research on Visual-Gestural Language. Technical Report Report Number 10, Boston University, Boston, MA. `http://www.bu.edu/asllrp/reports.html#RPT10`. 28

Neidle, C. (2005). *SignStream Screenshots*, Boston University, Boston, MA. `http://www.bu.edu/asllrp/signstream/screenshots.html`. 29

Neidle, C., Sclaroff, S., Zhuravlova, I., Boyd, J., MacLaughlin, D., Lee, R. G., Foelsche, O., Greenfield, D., Bahan, B. and Kegl, J. (2004). *SignStream :: American Sign Language Linguistic Research Project :: Boston University*, Boston University, Boston, MA. `http://www.bu.edu/asllrp/signstream/`. 28

Netopia, Inc. (2005). *Timbuktu Pro Remote Control Software*, Netopia, Inc. `http://www.netopia.com/software/products/tb2/`. 13

Network World, I. (2005). *Morae: Usability testing for the masses*, Network World, Inc. `http://www.networkworld.com/reviews/2005/020705revmorae.html`. 17

Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann (original hardcover edition published by AP Professional). ISBN 0-12-518406-9. 3

Noldus Information Technology bv. (2004). *Noldus - The Observer*, Noldus Information Technology bv. `http://www.noldus.com/site/doc200401012`. 18, 19

RealVNC Ltd (2005). *About RealVNC*, RealVNC Ltd. `http://www.realvnc.com/`. 15

Rossmanith GmbH (2005). *TeamViewer Free Desktop Sharing*. `http://www.teamviewer.com/`. 104

Sanderson, P. (1997). *MacSHAPA*, University of Illinois. `http://www.aviation.uiuc.edu/institute/acadprog/epjp/macshapa.html`. 27

Sellen, A. J., Murphy, R. and Shaw, K. L. (2002). *How knowledge workers use the web.* In CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 227–234, New York, NY, USA (2002). ACM Press. 4

Sem, S. (2005). *XviD.org :: Home of the XviD codec*, Michael Militzer. `http://www.xvid.org`. 24

Spolsky, J. (2005). *Usability Testing with Morae.* `http://www.joelonsoftware.com/articles/UsabilityTestingwithMorae.html`. 104

Sun Microsystems, I. (2005a). *Code Conventions for the Java Programming Language*, Sun Microsystems, Inc. `http://java.sun.com/docs/codeconv/`. 81

Sun Microsystems, I. (2005b). *Java Technology*, Sun Microsystems, Inc. `http://java.sun.com/`. 70, 71

Symantec Corporation. (2005). *Symantec pcAnywhere*, Symantec Corporation. `http://enterprisesecurity.symantec.co.uk/products/products.cfm?productID=326&PID=na&EID=0`. 13

TechSmith Corporation (2005a). *Camtasia Studio - Screen Recording for Demos and Training*, TechSmith Corporation. `http://www.techsmith.com/products/studio/default.asp?lid=CamtasiaStudioHome`. 15, 16

TechSmith Corporation (2005b). *Morae: A Complete Usability Testing Solution for Web Sites and Software*, TechSmith Corporation. `http://www.techsmith.com/products/morae/default.asp?lid=MoraeHome`. 6, 28

The Utah Education Network (2005). *Video Conferencing Glossary*, Utah State Office of Education and the Utah System of Higher Education. `http://www.uen.org/delivery/ivc_glossary.shtml`. 9

Trienes, R., Jansen, J., Carvalhais, J., Fujão, C., Magalhães, M., Serranheira, F. and Simoés, A. (1998). Observing ergonomics: The Observer Video-Pro in RSI research. Technical report, Noldus Information Technology b.v., Wageningen, The Netherlands and Department of Ergonomics, University of Lisbon, Lisbon, Portugal. `http://www.noldus.com/events/mb98/abstracts/trienes.htm`. 19

Tukey, J. (1977). *Exploratory Data Analysis.* Addison-Wesley. ISBN 0201076160. 6

Udell, J. (2005). *Jon Udell: All about screencasting*, InfoWorld. `http://weblog.infoworld.com/udell/2005/02/21.html`. 7

Users First, LLC (2004). *VisualMark Portable Observation Lab.* `http://www.usersfirst.com/index.jsp?page=products`. 17

Washko, D. S. (2004). *Creating Animated Screenshots on Linux.* Linux Gazette, 102. 105

Wien (MA 53) (2005). *WIENUX-Tag: Wiener Lösung für Open Source*, Stadt Wien. `http://www.magwien.gv.at/vtx/vtx-rk-xlink?SEITE=020050705010`. 4

Wikipedia (2005a). *http://en.wikipedia.org/wiki/Usability*, Wikimedia Foundation, Inc. `http://en.wikipedia.org/wiki/Usability`. 3

Wikipedia (2005b). *Non-linear editing system*, Wikimedia Foundation, Inc. `http://en.wikipedia.org/wiki/Non-linear_editing_system`. 11

Wikipedia (2005c). *Object-oriented programming*, Wikimedia Foundation, Inc. `http://en.wikipedia.org/wiki/Object-oriented_programming`. 72

Wikipedia (2005d). *Screencast*, Wikimedia Foundation, Inc. `http://en.wikipedia.org/wiki/Screencast`. 7

Wikipedia (2005e). *Wiki*, Wikimedia Foundation, Inc. `http://en.wikipedia.org/wiki/Wiki`. 37

Wikipedia (2005f). *XviD*, Wikimedia Foundation, Inc. `http://en.wikipedia.org/wiki/XviD`. 24

Woods, D. K. and Fassnacht, C. (2005). *Transana v2.05*, University of Wisconsin, Madison, WI, USA. `http://www.transana.org/`. 6, 31

Zhuravlova, I., Boyd, J., Lee, R. G., Neidle, C. and Schlang, M. (2004). *SignStream Development*, Boston University, Boston, MA. `http://www.bu.edu/asllrp/signstream/development.html`. 30

Zhuravlova, I., Boyd, J., Lee, R. G., Neidle, C. and Schlang, M. (2005). *SourceForge.net: Project Info - SignStream*, Boston University, Boston, MA. `http://sourceforge.net/projects/signstream/`. 30