

SE Methoden der Usability Evaluation

Szenariobezogene Logfileanalyse



Marko Pilop & Henryk Plötz

[\(pilop|ploetz\)@informatik.hu-berlin.de](mailto:(pilop|ploetz)@informatik.hu-berlin.de)

<http://www.informatik.hu-berlin.de/~pilop/Logfile-Analyse/>

WS 2004/2005

9. Februar 2006 (Revision: 1.10)

Humboldt-Universität zu Berlin
Institut für Psychologie
Dozent: Prof. Dr. Hartmut Wandke

Zusammenfassung

Diese Ausarbeitung ist im SE Methoden der Usability Evaluation zum Vortrag Szenariobezogene Logfileanalyse entstanden. Dieses Dokument, die Vortragsfolien, unsere selbst geschriebenen Programme und weiterführende Informationen sind auf der Webseite: <http://www.informatik.hu-berlin.de/~pilop/Logfile-Analyse/> verfügbar.

Das erste Kapitel widmet sich der Geschichte und Herkunft der vorgestellten Methode. Sie wird eingeordnet erklärt und bewertet.

Das zweite Kapitel stellt unsere eigentliche Evaluationsaufgabe vor. Es wird auf unser Vorgehen, Komplikationen und deren Lösungen eingegangen. Anschließend werden die gewonnenen Daten ausgewertet und eine Interpretation versucht.

Das dritte Kapitel stellt ein weiteres Beispiel der Logfileanalyse anhand von Webserver-Logdateien vor.

Inhaltsverzeichnis

1	Einführung	5
1.1	Geschichtliches zu Logfiles	5
1.2	Klassifikation von Evaluationsmethoden	5
1.2.1	Subjektive Verfahren	5
1.2.2	Objektive Verfahren	5
1.2.3	Analytische Verfahren	6
1.3	Grundlagen der Methode	6
1.3.1	Was sind Logfiles?	6
1.4	Warum Logfiles?	6
1.5	Bewertung der Methode	7
1.5.1	Vorteile der Methode	7
1.5.2	Nachteile der Methode	8
1.5.3	Verbesserungsvorschläge	8
2	Analyse der Fernbedienung	9
2.1	Versuchsaufbau	10
2.1.1	Simulationssimulation	10
2.2	Eingabe	11
2.3	Verarbeitung	14
2.4	Ausgabe	14
2.5	Auswertung	14
2.5.1	Aufgabe 1	14
2.5.2	Aufgabe 2	16
2.5.3	Aufgabe 3	16
3	Analyse von Webserverlogs	21
3.1	Aufbau von Webserverlogs	21
3.2	Beispiel	22
3.3	Ausblick	23
4	Zusammenfassung	25
	Literatur	26

1 Einführung

1.1 Geschichtliches zu Logfiles

Auf den ersten Blick scheinen Logfiles und die damit verbundenen Analysemethoden noch nicht sehr alt zu sein, da sie auf Computertechnik basieren. Jedoch deutet die Analogie Logfile → Logbuch auf eine bereits etablierte Benutzung in der Schiffsführung hin. Sinn und Zweck von Logfiles ist es, das Benutzerverhalten konstant¹ zu erfassen, um bei einer nachfolgenden Auswertung Rückschlüsse ziehen zu können.

1.2 Klassifikation von Evaluationsmethoden

Wo die Logfile-Analyse bei den verschiedenen Evaluationsmethoden angesiedelt ist, sieht man in der folgenden Klassifizierung der Verfahren:

1.2.1 Subjektive Verfahren

Subjektive Verfahren knüpfen unmittelbar an die Beurteilung durch den Benutzer an. Es werden „weiche“ Daten, wie zum Beispiel *bequem, angenehm, klar* oder *einsichtig* gewonnen.

1.2.2 Objektive Verfahren

Bei den objektiven Verfahren sollen die subjektive Einflüsse ausgeschaltet werden, wenn quantitative, statistisch abgesicherte Daten, wie zum Beispiel *Ausführungs- und Lernzeiten* oder *Fehlerraten* erhoben werden. Das Ziel dabei ist die Beobachtung der tatsächlichen Benutzung der Anwendung.

Stufen der Objektivierung

Ohne technische Unterstützung bleibt uns die Anfertigung von Beobachtungsprotokollen, die in der Regel strukturiert vorliegen und die zu Beobachtenden Sachverhalte vorgeben. Dabei kann sich der Beobachter passiv verhalten, oder Fragen zur näheren Abklärung bestimmter Sachverhalte, in Form eines Beobachtungsinterviews, stellen.

Bei der Evaluation von software-ergonomischen Fragestellungen spielt die technische Beobachtung eine große Rolle. Dabei wird die Beobachtung von einem System übernommen, dass alle Interaktionsschritte der Benutzer aufgezeichnet. Dieses Verfahren ist als

¹mehr oder weniger

Logfilerecording bekannt. Ein so aufgenommenes Logfile kann durch Statistikprogramme ausgewertet werden.

1.2.3 Analytische Verfahren

Die analytischen Verfahren sind zwischen den subjektiven und objektiven Verfahren angesiedelt. Es handelt sich dabei um eine leitfadensorientierte Evaluation durch Experten.

1.3 Grundlagen der Methode

Bei der Logfile-Analyse geht es um die Auswertung einer Datenquelle, die meist in Form einer Datei² vorliegt. Diese Datei ist eine maschinenlesbare Aufzeichnung des Benutzerverhaltens, zum Beispiel *Mausklicks* oder *Tastenschläge* und des *Systemzustands*, bei der jeder Eintrag einen möglichst eindeutigen *Zeitstempel* besitzt.

1.3.1 Was sind Logfiles?

Logfiles bilden also Folgen von Momentaufnahmen von Systemzuständen ab, die aus *Ereignissen*, *Zuständen* und *Zeiten* bestehen. Die Logfiles können zeit- und/oder ereignisgesteuert geschrieben werden. So werden bei zeitgesteuerten Logfiles in regelmäßigen Intervallen die Daten geschrieben, während bei den ereignisgesteuerten nur beim Eintreten von Ereignissen diese protokolliert werden.

Anhand der Logfiles ist auch das Replay, also ein wiederholtes Abspielen, möglich.

Gemessen werden in der Regel einfache Verhaltensdaten, wie *Zeiten* und *Fehler*, aber auch komplexe Verhaltensdaten, wie *Lernzeiten*, der *Umgang mit Fehlern*, oder benutzte *Strategien*.

Logfiles bieten die Möglichkeit, Benutzer- bzw. Systemverhalten *automatisch* zu Erfassen.

Einige Beispiele für Logfiles sind System- / Fehlerprotokolle und Server-Logfiles, wie in Abbildung 1.1 dargestellt oder Flugdatenschreiber, die als sogenannte Blackbox Daten zur Rekonstruktion einer Fehlerursache liefern.

1.4 Warum Logfiles?

Logfiles sind eine Notwendigkeit, da die Beobachtung durch den Untersucher, der ein Protokoll anfertigt, träge ist. Es ist ihm nicht möglich, eine schnelle Abfolge von Aktionen zu erkennen. Da der Beobachter nicht alles aufzeichnen kann, ist das Protokoll ungenau. Man bräuchte mehr Beobachter die verteilt protokollieren, oder könnte die Auswertung später anhand von Videoaufzeichnungen vornehmen. Diese Verfahren können allerdings die Kosten der Evaluierung in die Höhe treiben.

²engl. File

```
pille@schnattel:~$ cat /var/log/syslog
Mar 19 23:10:01 [/usr/sbin/cron] (root) CMD (test -x /usr/sbin/run-crons && /usr/sbin/run-crons )
Mar 19 23:10:01 [/etc/cron.hourly/dgndns.update] updating DgndNS
Mar 19 23:10:01 [dgndnsupdate] Your ipaddress is not changed, skipping update...
Mar 20 07:05:01 [/usr/sbin/cron] (root) CMD (/etc/mgetty+sendfax/new_voice_scan)
Mar 20 14:24:30 [kernel] ReiserFS: hdd1: using ordered data mode
Mar 20 14:24:30 [kernel] ReiserFS: hdd1: journal params: device hdd1, size 8192, journal first block 18, max trans len 1024, max batch 900, max commit age 30, max trans age 30
Mar 20 14:24:30 [kernel] ReiserFS: hdd1: Using r5 hash to sort names
Mar 20 14:25:02 [kernel] ReiserFS: hdd1: found reiserfs format "3,6" with standard journal
Mar 20 14:25:04 [kernel] ReiserFS: hdd1: using ordered data mode
Mar 20 14:25:04 [kernel] ReiserFS: hdd1: journal params: device hdd1, size 8192, journal first block 18, max trans len 1024, max batch 900, max commit age 30, max trans age 30
Mar 20 14:25:04 [kernel] ReiserFS: hdd1: Using r5 hash to sort names
Mar 20 16:58:39 [kernel] usb 1-3: new high speed USB device using ehci_hcd and address 3
Mar 20 16:58:40 [kernel] Initializing USB Mass Storage driver...
Mar 20 16:58:40 [kernel] scsi0 : SCSI emulation for USB Mass Storage devices
Mar 20 16:58:46 [scsi.agent] disk at /devices/pci0000:00/0000:00:10.3/usb1/1-3/1-3:1.0/host0/target0:0:0:0:0:0
Mar 20 16:58:46 [kernel] SCSI device sd1: 30070080 512-byte hdwr sectors (20004 MB)
Mar 20 16:58:46 [kernel] sd1: assuming drive cache: write through
Mar 20 16:58:46 [kernel] sd1: <?usb-storage: queuecommand called
Mar 20 16:58:46 [kernel] sd1
Mar 20 18:38:26 [sshd] Did not receive identification string from 61.11.11.120
Mar 20 18:56:45 [sshd] reverse mapping checking getaddrinfo for static11-120.dsl-num.eth.net failed - POSSIBLE BREAKIN ATTEMPT!
Mar 20 18:56:51 [sshd] Invalid user patrick from 61.11.11.120
Mar 20 18:56:51 [sshd] reverse mapping checking getaddrinfo for static11-120.dsl-num.eth.net failed - POSSIBLE BREAKIN ATTEMPT!
Mar 20 18:56:57 [sshd] Invalid user patrick from 61.11.11.120
Mar 20 18:56:57 [sshd] reverse mapping checking getaddrinfo for static11-120.dsl-num.eth.net failed - POSSIBLE BREAKIN ATTEMPT!
Mar 20 18:56:57 [sshd] Last output repeated 3 times -
Mar 20 18:57:23 [sshd] Did not receive identification string from 61.11.11.120
Mar 20 20:45:06 [pppd] LCP terminated by peer
Mar 20 20:45:06 [pppd] Modem hangup
Mar 20 20:45:06 [pppd] Connection terminated.
Mar 20 20:45:06 [pppd] Connect time 1440.0 minutes.
Mar 20 20:45:06 [pppd] Sent 2058915103 bytes, received 983488893 bytes.
Mar 20 20:45:11 [pppd] Starting link
Mar 20 20:45:11 [pppd] Serial connection established.
Mar 20 20:45:11 [pppd] Connect: ppp0 <-> /dev/pts/0
Mar 20 20:45:14 [pppd] PAP authentication succeeded
Mar 20 20:45:14 [opt/firewall/firewall] firewall script started
Mar 20 20:45:55 [/usr/local/sbin/dgndns.update] updating DgndNS
Mar 20 21:00:02 [/etc/cron.hourly/dgndns.update] updating DgndNS
Mar 20 21:00:02 [dgndnsupdate] Connecting to members.dgndns.org...
Mar 20 21:00:02 [dgndnsupdate] succeeded!
Mar 20 21:00:02 [dgndnsupdate] Update good and successful, IP updated.
Mar 22 04:14:41 [sshd] Invalid user subase from 220.95.215.148
Mar 22 04:14:44 [sshd] Invalid user master from 220.95.215.148
Mar 22 04:14:47 [sshd] Invalid user account from 220.95.215.148
Mar 22 04:14:50 [sshd] Invalid user backup from 220.95.215.148
Mar 22 04:14:53 [sshd] Invalid user server from 220.95.215.148
Mar 22 04:14:56 [sshd] Invalid user adam from 220.95.215.148
Mar 22 04:15:00 [sshd] Invalid user alan from 220.95.215.148
Mar 22 04:15:03 [sshd] Invalid user frank from 220.95.215.148
Mar 22 04:15:06 [sshd] Invalid user george from 220.95.215.148
```

Abbildung 1.1: SysLog-Ausgabe des Linux-Betriebssystems

Logfiles sind eine Methode, den Beobachtungsprozess nach definierten Parametern zu automatisieren, indem die relevanten Gesichtspunkte z.B. programmgestützt aufgezeichnet werden.

Logfiles werden zur Unterstützung bei der Entwicklung, z.B. Fehlersuche, oder der Evaluation von Prototypen verwendet, indem das Benutzer- und Systemverhalten beim Bearbeiten von prototypischen Aufgaben aufgezeichnet wird. Auch bei der Langzeit-Evaluation in realisierten Systemen werden Logfiles verwendet.

Durch die Analyse von Logfiles kann man versuchen, das beobachtete Verhalten zu erklären, oder zukünftiges Verhalten in vergleichbaren Situationen vorherzusagen.

1.5 Bewertung der Methode

1.5.1 Vorteile der Methode

Mit der Logfile-Analyse kann die Erfassung des Benutzerverhaltens, ohne Eingriff durchgeführt werden, wodurch sich der Benutzer weniger beobachtet fühlt und sich auf die Aufgabe konzentrieren kann. Da eine Einflußmöglichkeit durch Dritte ausgeschlossen

wird, kann ein hohes Maß an Objektivität erreicht werden. Außerdem ist es eine ökonomisch günstige Lösung, da sich die Personenanzahl weitgehend bei gleichbleibenden Kosten skalieren läßt.

1.5.2 Nachteile der Methode

Die Logfile-Analyse liefert als Ergebnis keine direkten Informationen zu Ursachen bestimmter Bedienhandlungen. Die Analyse von Fehlern ist schwer und meist nur in Verbindung mit anderen Methoden der Usability-Evaluation klärbar. Eine Deutung von Ereignissen im Logfile ist oft kompliziert, da Verhaltensweisen, Situationen und Aufgaben identifiziert werden müssen. Es ist so z.B. nicht eindeutig, was lange Wartezeiten des Benutzers bedeuten.

Außerdem existiert kein einheitliches Logfile-Format, daß für die Weiterverarbeitung geeignet ist. Jeder Hersteller hat seinen eigenen proprietären Standard, der nicht kompatibel zu Analysewerkzeugen ist. XML wird zwar in den letzten Jahren aus Austauschformat favorisiert, jedoch scheitern einige Hersteller am Einhalten des Standards, oder es werden Daten ohne semantischen Zusammenhang abgelegt und dadurch wertlos. Im Prinzip muß daher die Automatisierung erst entwickelt werden, bevor sie genutzt werden kann.

Einen weiteren Nachteil stellen die durch das Logging verbrauchten Ressourcen dar. Wie überall existiert ein Trade-Off zwischen Zeit und Geld. Spielt Geld keine Rolle, so kann von einer Automatisierung abgesehen werden und die Arbeit wird durch mehr Personal erledigt.

Das Logging selbst verbraucht Systemressourcen wie *Rechenzeit* und *Speicherplatz* und das Datenaufkommen muß bewältigt werden. Speziell bei einer Langzeitevaluation müssen die Daten reduziert bzw. komprimiert werden.

Diese Aspekte müssen schon bei Entwicklung von Logfiles berücksichtigt werden.

1.5.3 Verbesserungsvorschläge

Um die Nachteile der Methode zu korrigieren, sollten subjektive Kenngrößen mit erfasst werden. Generell macht die Kombination mit anderen Methoden, wie z.B. *Videokonfrontation* oder *Lautes Denken* viel Sinn. Ursachen, oder Beweggründe der Benutzer können so nachträglich erklärt werden und helfen bei der Analyse der Daten.

2 Analyse der Fernbedienung

Unsere Aufgabe war die Usability Evaluation einer Fernbedienung für einen Videorekorder. Zu diesem Zweck wurde uns ein Windows-Programm mit einer eingeschränkten Simulation von Fernbedienung, Fernseher und Videorekorder zur Verfügung gestellt, welches ein Logfile in einem XML-ähnlichen Format erzeugt.

Mit dem Hinblick auf eine mögliche Auswertung gab es mit dem Logfile jedoch einige Probleme:

- Der Zeitstempel für Ereignisse war unbrauchbar, da er nur Stunden, Minuten und Millisekunden enthielt. Die Idee das Binary¹ zu patchen um die Millisekunden durch Sekunden zu ersetzen kam leider zu spät². Da die Millisekunden wegen der fehlenden Sekunden unbrauchbar waren, war nur eine minutengenaue Auswertung möglich.
- Auch bei geöffneter Klappe werden Ereignisse für Knöpfe aufgezeichnet die nur bei geschlossener Klappe erreichbar sind. Genauer gesagt sind alle Knöpfe die nur bei geschlossener Klappe bedienbar sind auch bei geöffneter Klappe zu benutzen, bis auf einen, an dessen Stelle ED/EW steht. Dadurch ist es bei der Auswertung der Daten nicht möglich einfach nur aus den Daten des Logfiles auf die bedienten Knöpfe zu schließen, sondern man muss zusätzlich (mindestens) den Zustand der Klappe einbeziehen, der natürlich nur implizit als Ereignisse für „Klappe wird geöffnet“ und „Klappe wird geschlossen“ im Logfile enthalten ist.
- Das von der Simulation generierte Dateiformat hat eine oberflächliche Ähnlichkeit mit XML, ist jedoch nicht notwendigerweise well-formed³, was die Verarbeitung mit einem normalen XML-Parser erschwert oder unmöglich macht. Das schwerwiegendste Problem in dieser Hinsicht ist, dass das Logfile unter Umständen Umlaute enthält⁴, diese aber nicht UTF-8-kodiert sind und die Datei auch keine XML-Deklaration mit Angabe der verwendeten Zeichencodierung enthält.

Die Simulation selbst war auch nicht ohne Auffälligkeiten:

¹das Simulationsprogramm

²für die Auswertung. Die von uns korrigierte Version steht auf der Website

<http://www.informatik.hu-berlin.de/~pilop/Logfile-Analyse/> zum Download bereit

³eine Anforderung an XML-Dokumente, syntaktisch korrekt zu sein

⁴Das ist zum Beispiel im Monatsnamen in der Datumsangabe der Fall. Die Namen dort kommen aus der Systemumgebung, was neben ‚January‘ und ‚Januar‘ auch ‚Jänner‘ ermöglicht. Selbiges gilt für ‚März‘ etc. Übrigens ist die Verwendung von lokalisierten Monatsnamen nicht gerade hilfreich beim automatischen Parsen des Datums.

- Wenn man am Anfang Kanäle runterschaltet geht der ausgewählte Kanal offenbar in den negativen Bereich. Wenn man erst einen Kanal hoch und dann runterschaltet verhält es sich normal (1 → 8). Für Testpersonen die zuerst runterschalten ist das natürlich verwirrend, und muss zudem in der Auswertung beachtet werden.
- Wenn man ein Timerprogramm editiert, werden die vorherigen Einstellungen des Timers vergessen und Standardwerte wiederhergestellt. Das ist ebenso für Benutzer und Auswertung unschön, und sicher nicht im Original enthalten. Das bewirkt zusätzliche Probleme für unerfahrene Anwender: Wenn der Benutzer zuerst nur die Hälfte des Timerprogramms eingibt (zum Beispiel Sendepfad und Tag) und dann versehentlich die Programmierung verlässt, muss er diese Einstellungen wiederholen wenn er das Programm erneut bearbeitet.
- Die Liste mit Programmen die für die Anzeige der Programmnamen in der Aufnahmeprogrammierung verwendet wird ist nicht die Liste aus der Sendertabelle. Das dürfte den Anwender auch mehr verwirren, wenn er zunächst erfolgreich die Sendertabelle umgestellt hat, dann aber eine andere Reihenfolge beim Erstellen von Timerprogrammen vorfindet.

2.1 Versuchsaufbau

Um möglichst viele Versuchspersonen in möglichst kurzer Zeit an dem Versuch teilnehmen zu lassen, entschieden wir uns, eine Webseite als Kommunikationsmittel einzusetzen und deren Adresse an Bekannte und Freunde weiterzugeben.

Die Webseite enthielt eine Version des Simulationsprogrammes zum herunterladen, Instruktionen für die Durchführung des Versuchs zusammen mit den Versuchsaufgaben, sowie ein Formular um die erzeugten Logfiles nach abgeschlossenem Versuch wieder zu uns zurückzuschicken. Insgesamt haben 12 Versuchspersonen ihre Logfiles eingesandt.

2.1.1 Simulationssimulation

Da das Logfile viele der Zustandsdaten nicht enthielt, die für eine ordnungsgemäße Auswertung der gesammelten Daten notwendig wären, haben wir uns so etwas wie eine Simulation der Simulation geschrieben⁵. Das System besteht aus mehreren Komponenten, alle in Python geschrieben:

1. Die Daten aus dem Logfile werden eingelesen und nach besten Kräften geparkt. Dabei entsteht unter anderem eine geordnete Liste mit Ereignissen die Tastenbetätigungen der Versuchsperson repräsentieren.
2. Die so gewonnenen Ereignisse werden nach und nach an einen Programmteil übergeben der ein Modell des internen Zustandes der Simulation führt. Der Zustand dieses Modells wird entsprechend den einkommenden Ereignissen verändert.

⁵verfügbar unter <http://www.informatik.hu-berlin.de/~pilop/Logfile-Analyse/>

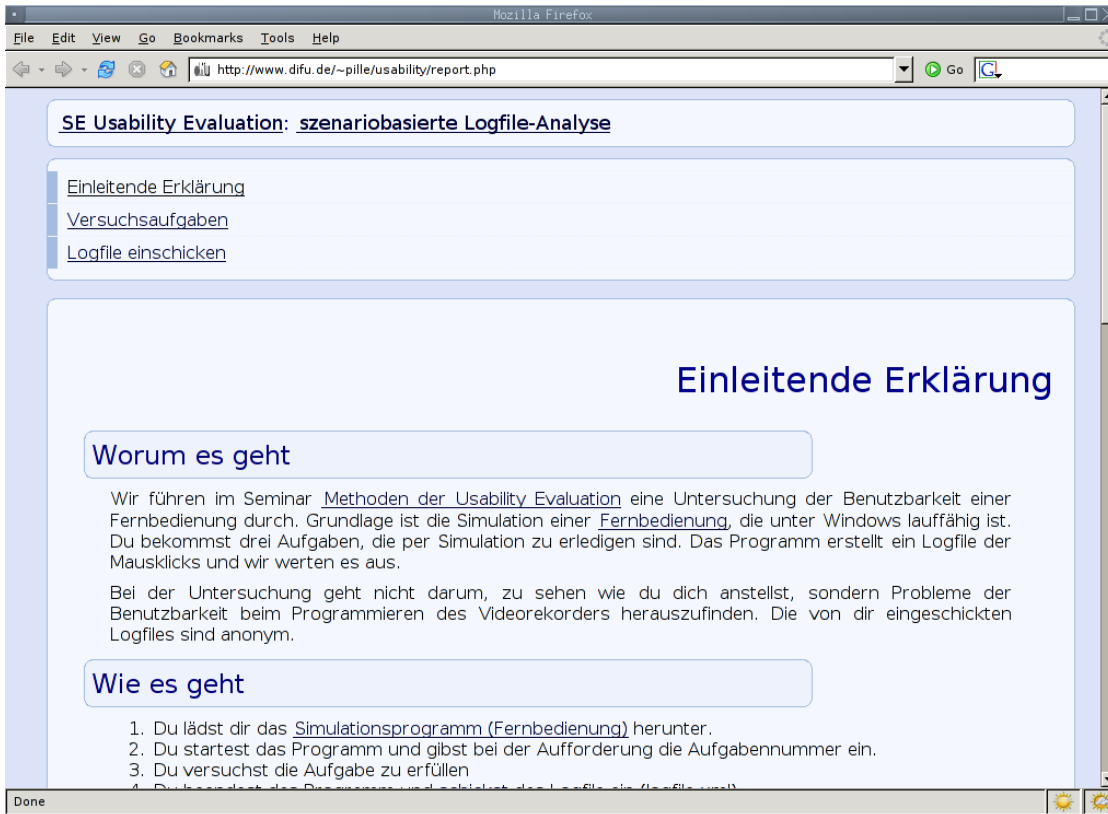


Abbildung 2.1: Die Webseite für die Versuchspersonen

3. Eine Komponente modelliert die verschiedenen Aufgabenszenarien und beobachtet den Zustand des Modells der Simulation. Auf diese Art gewinnt sie Einsicht in den Fortschritt den der Benutzer bei der Lösung der Aufgabenstellung macht und kann auch feststellen, wann und ob die Aufgabe gelöst wurde.

Auf diese Art können viele der Unzulänglichkeiten des Logfiles auf relativ elegante Art umgangen werden: Probleme mit der Zeichenkodierung zum Beispiel betreffen nur die erste Komponente und können dort lokal behandelt werden, während Probleme mit der Simulation an sich (etwa die negativen Kanalnummern) in der zweiten Komponente verarbeitet werden können, ohne die dritte Komponente zu betreffen.

2.2 Eingabe

Ein Beispiel für ein generiertes Logfile ist in Abbildung 2.3 zu sehen. Man erkennt dort sehr schön einige der zweifelhaften Features des Logfiles, wie das Datums- und Uhrzeitformat oder auch die offenbar bedeutungslosen Elemente „Level“ und „Meaning“.

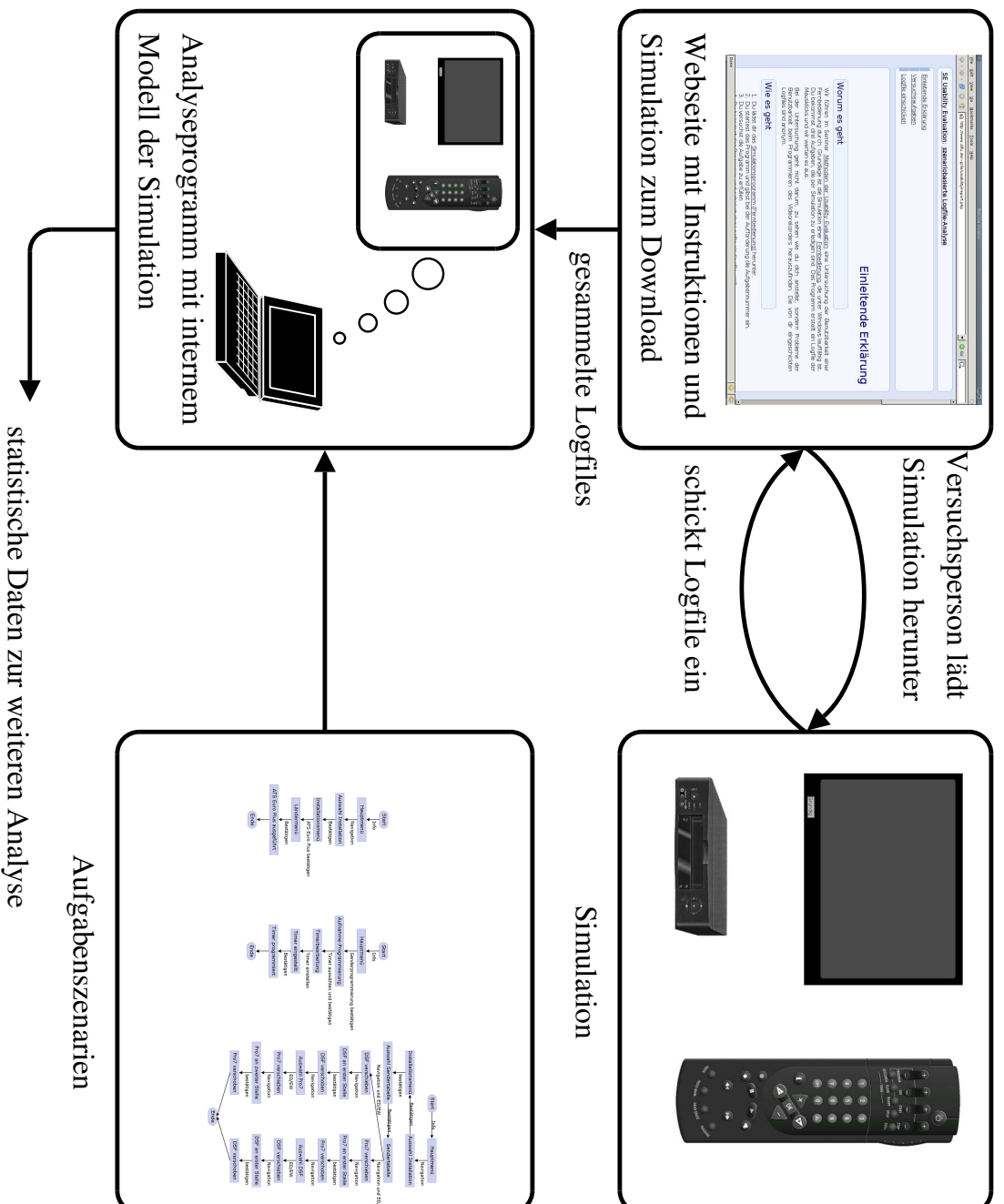


Abbildung 2.2: Der schematische Versuchsaufbau

```

<Logfile>
  <Prototype>Videorecorder</Prototype>
  <Date>Mittwoch, Januar 12, 2005 um 10:07 PM</Date>
  <Testnumber>42</Testnumber>
  <Execution>Versuchsleiter</Execution>
  <Data>
    <DataSet>
      <DataSource>Simulation</DataSource>
      <EventSource>B_TV</EventSource>
      <Event>Click</Event>
      <TimeStamp>22:07:700</TimeStamp>
      <Level/>
      <Meaning/>
      <Item>
        <Name>Zustand</Name><Type>String</Type><Value>ON</Value>
      </Item>
    </DataSet>
    <DataSet>
      <DataSource>Simulation</DataSource>
      <EventSource>B_Info</EventSource>
      <Event>Click</Event>
      <TimeStamp>22:08:710</TimeStamp>
      <Level/>
      <Meaning/>
      <Item>
        <Name>xxx</Name><Type>xxx</Type><Value>xxx</Value>
      </Item>
    </DataSet>
    :
  </Data>
</Logfile>

```

Abbildung 2.3: Beispiel für ein generiertes Logfile

2.3 Verarbeitung

Den Versuchspersonen wurden drei Aufgaben gestellt:

1. Sie sollten die automatische Senderbelegung mit ATS Euro Plus ausführen. Dazu wurde ihnen nur gesagt dass es da irgendwo eine Funktion mit diesem Namen gibt, sie diese finden und aktivieren sollten.
2. Sie sollten eine Aufnahme zu einem vorgegebenen Zeitpunkt durchführen lassen.
3. Sie sollten die Sender in der Sendertabelle in eine vorgegebene Reihenfolge bringen.

Um die Lösungen auswerten zu können haben wir für jede dieser Aufgaben einen abstrakten Lösungsweg als Graph mit Zuständen und Übergängen erstellt. Knoten in diesem Graphen sind Zustände der Simulation bzw. der Interaktion des Benutzers mit der Simulation, etwa „Videorekorder zeigt Hauptmenü an“ oder „Benutzer stellt Startzeitpunkt ein“. Die Kanten der Graphen sind mit Tastenbetätigungen beschriftet. Unsere drei Graphen sind in Abbildung 2.4 zusammengestellt.

Zur automatischen Auswertung wurden die Graphen so umformuliert, dass sie von unserem Python-Programm interpretiert werden konnten. Die dritte Komponente in der Simulationssimulation hat dann – wie oben angedeutet – diese Graphen benutzt, um den Fortschritt der Aufgabenlösung zu verfolgen (also den Knoten im Graphen an dem sich der Benutzer gerade befindet) um diverse Statistiken, etwa über korrekte⁶ und inkorrekte Tastenbetätigungen.

2.4 Ausgabe

Eine beispielhafte Ausgabe unseres Auswertungsprogrammes ist in Abbildung 2.5 zu sehen.

Man sieht dort die Tastenbetätigungen (im wesentlichen alles was mit B_ anfängt) des Benutzers, sowie die Auswirkungen auf den internen Zustand. Die eckigen Klammern enthalten die Position des Cursors in den Menüebenen, falls das Menü aktiv ist. Die Zahlentripel am Ende der Auswertung sind jeweils *Minimum*, *Mittelwert* und *Maximum* und hauptsächlich bei der Auswertung von mehreren Logfiles relevant.

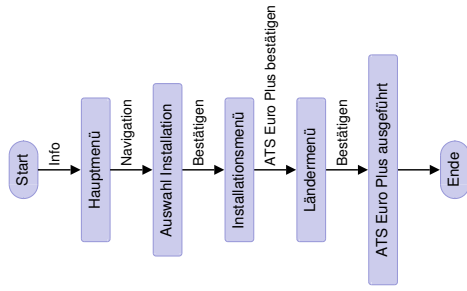
2.5 Auswertung

Wir haben Logfiles von 12 Personen erhalten, eine Übersicht zeigt Tabelle 2.1.

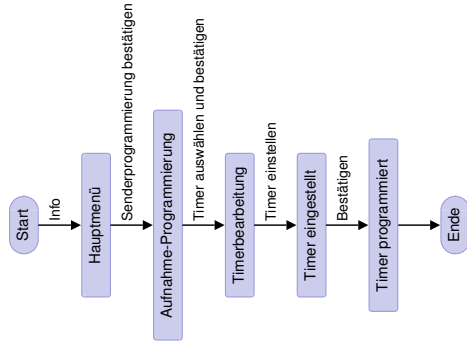
2.5.1 Aufgabe 1

Alle Versuchspersonen haben Aufgabe 1 gelöst, die auch die einfachste Aufgabe war. Abbildung 2.6 zeigt die aufsummierten Tastenbetätigungen der Versuchspersonen.

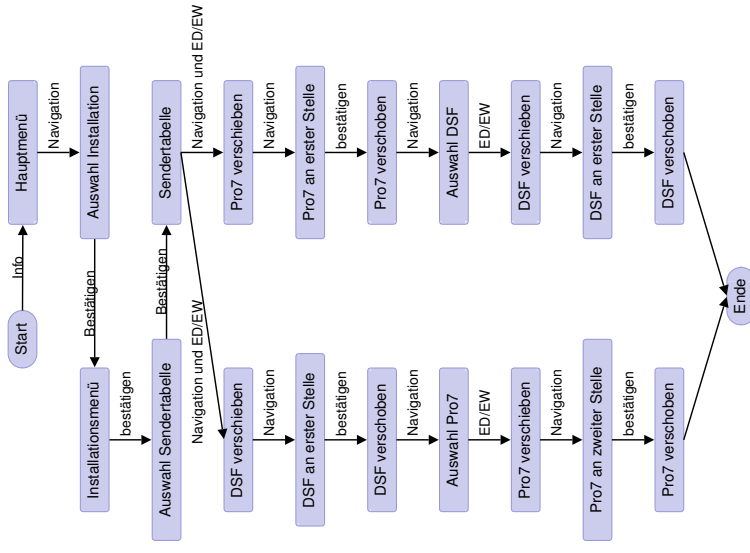
⁶der Lösung dienliche



Aufgabe 1



Aufgabe 2



Aufgabe 3

Abbildung 2.4: Die Problemlösungsgraphen für die drei Aufgaben

Aufgabe	Vollst. Lösungen	Durchschn. Zeit [min]
1	12	3
2	10	4
3	8	4

Tabelle 2.1: Übersicht über die abgegebenen Logfiles

Man sieht zum Beispiel, dass eine Person in Zustand 1 erhebliche Probleme hatte den weiteren Weg zu finden (160 Fehlbetätigungen), sich aber im Generellen die Anzahl der Fehlbetätigungen mit dem Fortschreiten des Versuchs verringert.

2.5.2 Aufgabe 2

Bei Aufgabe 2 (Abbildung 2.8) ist auffällig dass für den Schritt im Zustand 3 besonders viele zielführende Tastenbetätigungen nötig waren, aber das war zu erwarten, da das die Phase der Bearbeitung des Timerprogramms ist und der Videorekorder⁷ nur inkrementelle Änderungen zulässt, aber keine direkte Eingabe.

In diesem Zustand gibt es aber auch sehr viele fehlerhafte Tastenbetätigungen, da nicht alle Versuchspersonen das Prinzip der getrennten Programm-/Datum-/Start-/Stop-/Plus-/Minus-Tasten sofort oder überhaupt verstanden haben und häufig mit den Kanal- bzw. Lautstärkewahl-tasten versucht haben die Programmierung zu beeinflussen.

2.5.3 Aufgabe 3

Bei Aufgabe 3 gab es zwei mögliche Lösungen: Entweder man verschiebt zuerst DSF und dann Pro7, oder man macht es umgekehrt. Die Verlaufsglyphen (Abbildung 2.8) können das leider nicht vollständig korrekt darstellen. Das macht aber auch nichts, denn die zweite Variante hat keine der Versuchspersonen gewählt.

Man sieht viele Personen in Zustand 5 hängen, d.h. sie hatten wenig Schwierigkeiten die Sendertabelle zu finden, dann aber extreme Probleme, mit dem Verschieben von DSF zu beginnen. Das deutet darauf hin, dass die Benutzbarkeit hier stark eingeschränkt war, da nicht alle Benutzer die ED/EW-Tasten auf Anhieb gefunden⁸ haben, bzw. die Taste überhaupt fanden oder die Existenz der Klappe wahrnahmen.

⁷zumindest dessen Simulation

⁸dazu musste man die Klappe öffnen

```

1 Doing ../logs/LogFiles/217.231.29.151-1105881221-2-logfile.xml-toolong
2 Logfile from Videorecorder created by Versuchsleiter for test 1 on 2005-01-16 14
   :07:00
3 B_Power []
4 VCR is now ON
5 B_TV []
6 TV is now ON
7 B_Info []
8 Entering top-level menu
9 B_MRight [0]
10 Top-level menu: one down
11 B_OK [4]
12 Entering Installation
13 B_OK [4, 0]
14 Entering ATS Euro Plus
15 B_OK [4, 0, 0]
16 Executing ATS Euro Plus
17 -----
18 Task completed
19 -----
20 B_PLeft []
21 B_PRight []
22 One program up, now at 2 (____)
23 B_PRight []
24 One program up, now at 3 (Pro7)
25 B_PRight []
26 One program up, now at 4 (____)
27 B_PRight []
28 One program up, now at 5 (RTL)
29 B_PRight []
30 One program up, now at 6 (DSF)
31 B_PRight []
32 One program up, now at 7 (____)
33 B_PRight []
34 One program up, now at 8 (ZDF)
35 B_PRight []
36 One program up, now at 1 (ARD)
37 B_TV []
38 TV is now OFF
39 B_Power []
40 VCR is now OFF
41
42 Accumulated progress:
43   State 0: 3 right, 0 wrong
44   State 1: 1 right, 0 wrong
45   State 2: 1 right, 0 wrong
46   State 3: 1 right, 0 wrong
47   State 4: 1 right, 0 wrong
48 11 events after completion
49
50 Summary:
51   Task 1: 1 complete out of 1, times 1 1.00 1
52
53 Top buttons for task 1: B_PRight   (8 8.00 8)
54                       B_OK       (3 3.00 3)
55                       B_TV       (2 2.00 2)
56                       B_Power    (2 2.00 2)
57                       B_TV       (2 2.00 2)

```

Abbildung 2.5: Ausgabe der Simulationssimulation für eine Lösung der ersten Aufgabe

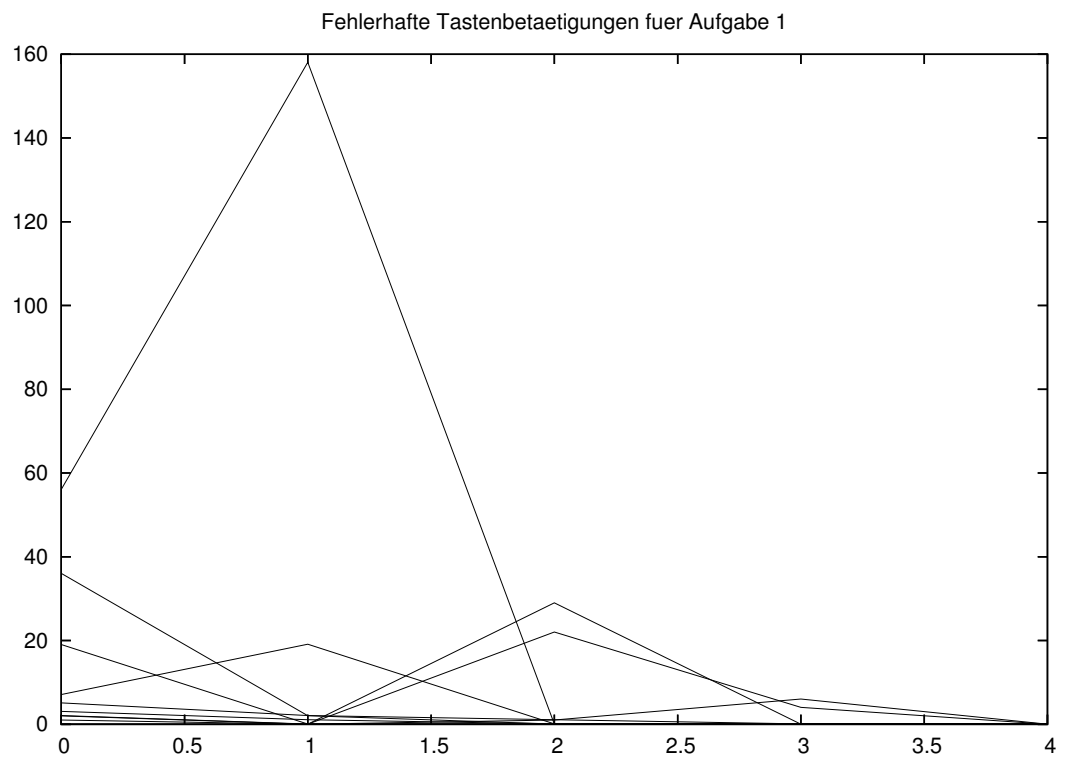
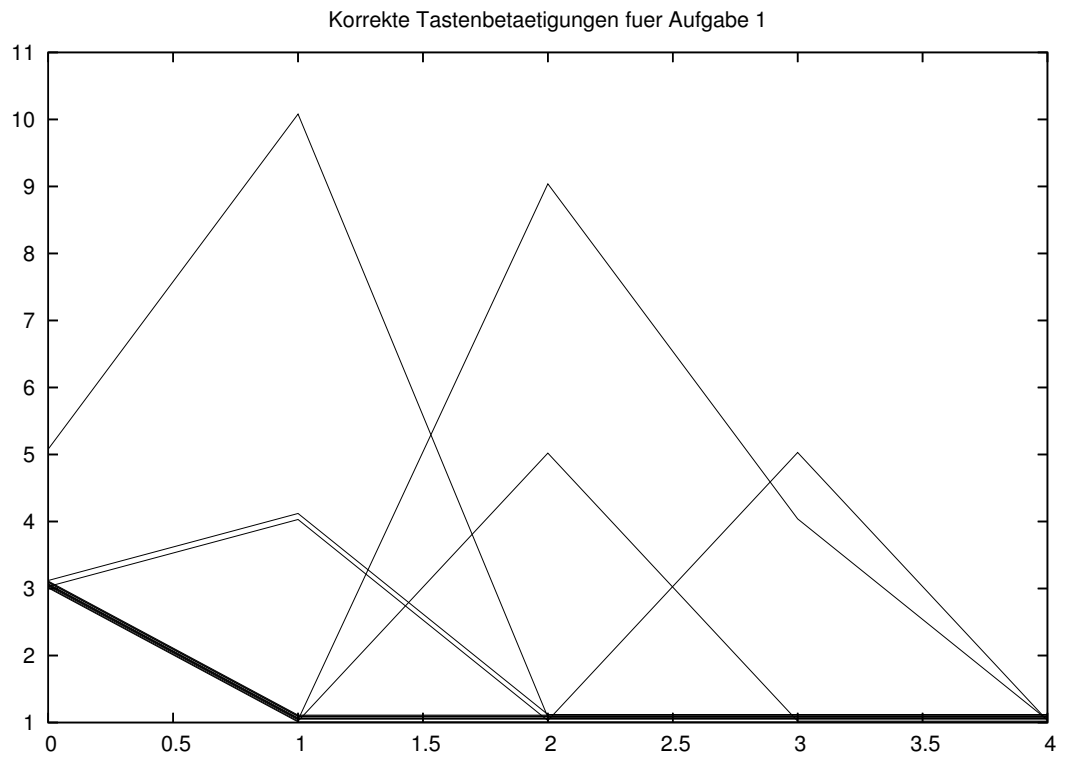


Abbildung 2.6: Tastenbetätigungen für Aufgabe 1

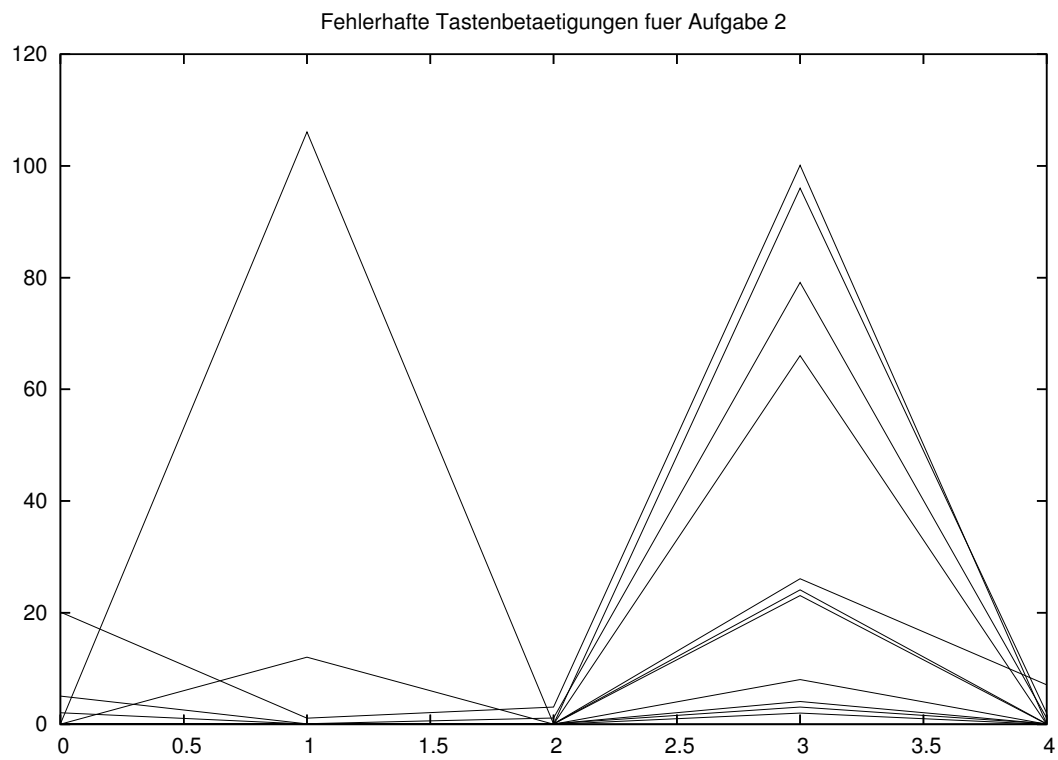
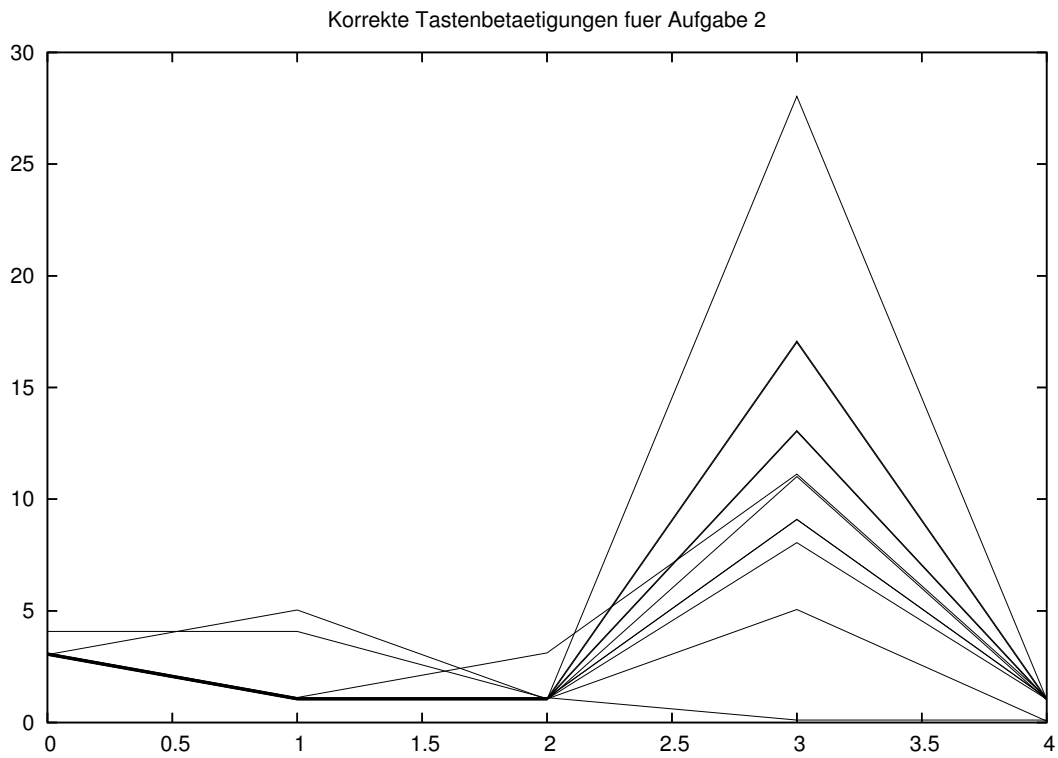


Abbildung 2.7: Tastenbetaetigungen fuer Aufgabe 2

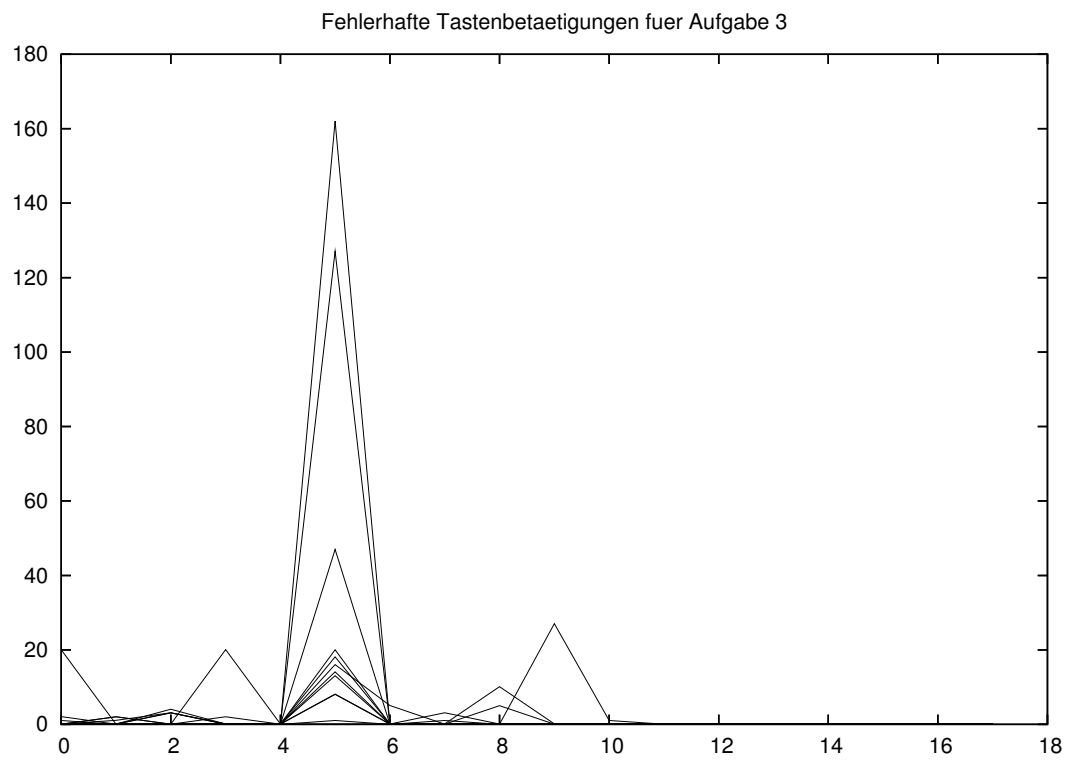
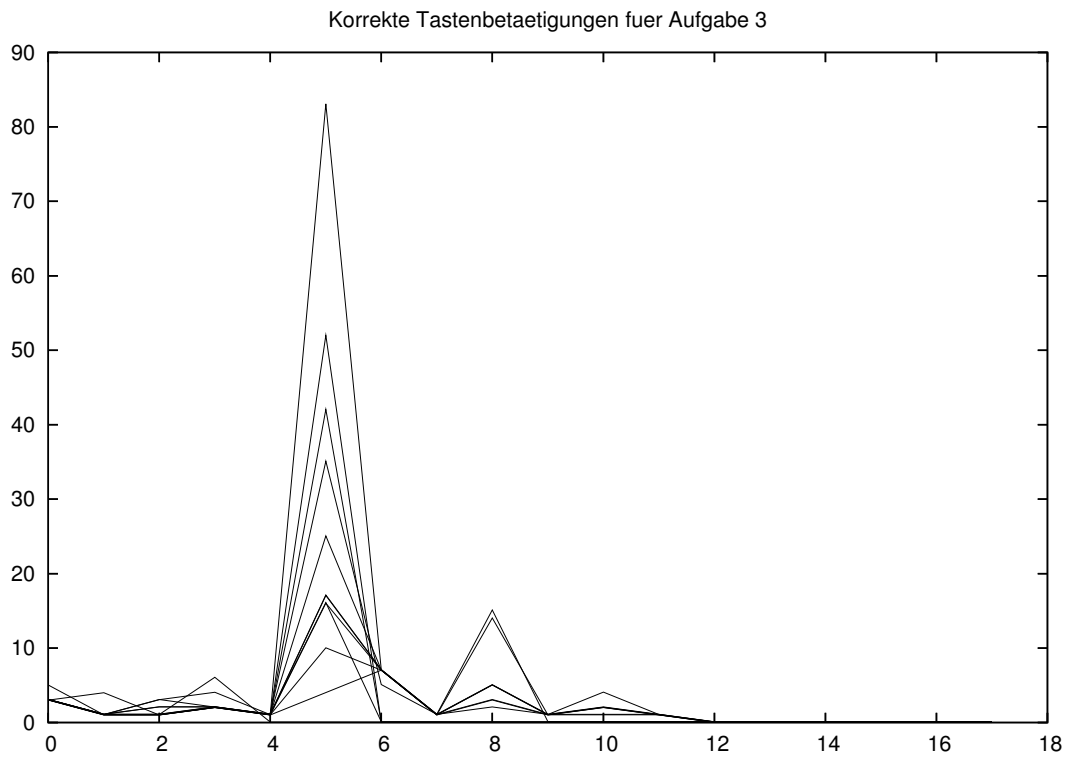


Abbildung 2.8: Tastenbetaetigungen fuer Aufgabe 3

3 Analyse von Webserverlogs

Wer sich im Netz bewegt, sollte sich bewußt sein, daß er Spuren hinterlässt. Diese lassen sich auf jeden Fall beim Server, aber auch auf dem Weg dahin protokollieren. Bei Webservern bietet sich diese gute Möglichkeit an, die Spuren in Logfiles zu nutzen, um Statistiken über Besuche zu erzeugen.

Allerdings sollte man bei der Interpretation der Statistiken beachten, daß viele erhobene Informationen unzuverlässig sind und die gesamte Auswertung dadurch ungenau werden lassen.

3.1 Aufbau von Webserverlogs

Wird eine WWW-Seite aufgerufen, so erzeugt der Webserver der die Anfrage bearbeitet einen entsprechenden Eintrag in seinem Logfile. Das Format sieht dabei folgendermaßen aus:

```
remotehost rfc931 authuser [date] "request" status bytes
```

Die Bedeutung der einzelnen Felder ist in der unten stehenden Tabelle 3.1 aufgeführt:

remotehost	IP des Besuchers
rfc931 authuser	Benutzername zur Authentisierung (optional)
[date]	Datum und Uhrzeit der Anfrage
"request"	Anfrage des Besuchers
status	HTTP-Status
bytes	Größe des transferierten Dokumentes

Tabelle 3.1: Erklärung der Felder einer Webserver-Logzeile

Zu diesem Standard existiert auch noch eine Erweiterung, die zusätzlich zwei weitere Felder definiert (siehe Tabelle 3.2):

```
remotehost rfc931 authuser [date] "request" status bytes "referer" "agent"
```

Die beiden neuen Felder beherbergen folgende Informationen:

referer Letzte Anfrage des Besuchers
agent Browserkennung des Besuchers

Tabelle 3.2: Erklärung der erweiterten Felder einer Webserver-Logzeile

3.2 Beispiel

Nachstehend ist ein Auszug des Webserverlogfiles der Website <http://www.sozialestadt.de> vom Januar 2005 bestehend aus neun Einträgen zu sehen:

```
1 207.68.146.55 - - [01/Jan/2005:00:00:21 +0100] "GET /robots.txt HTTP/1.0" 200 308  
   "-" "msnbot/0.3 (+http://search.msn.com/msnbot.htm)"  
2 207.68.146.55 - - [01/Jan/2005:00:00:22 +0100] "GET /praxisdatenbank/externe-  
   datenbanken/ HTTP/1.0" 200 8860 "-" "msnbot/0.3 (+http://search.msn.com/msnbot  
   .htm)"  
3 207.46.98.76 - - [01/Jan/2005:00:00:35 +0100] "GET /fr/suche/ HTTP/1.0" 200 5153 "  
   "-" "msnbot/0.3 (+http://search.msn.com/msnbot.htm)"  
4 207.46.98.62 - - [01/Jan/2005:00:02:01 +0100] "GET /veroeffentlichungen/endbericht  
   /6.phtml HTTP/1.0" 200 12646 "-" "msnbot/0.3 (+http://search.msn.com/msnbot.  
   htm)"  
5 207.46.98.76 - - [01/Jan/2005:00:02:17 +0100] "GET /en/veroeffentlichungen/  
   zwischenbilanz/1-socially-integrated-city-appraisal.shtml HTTP/1.0"  
   200 171462 "-" "msnbot/0.3 (+http://search.msn.com/msnbot.htm)"  
6 217.230.114.243 - - [01/Jan/2005:03:40:24 +0100] "GET /sozialestadt-intro.swf HTTP  
   /1.1" 206 114950 "-" "Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; Win 9x  
   4.90; DT; FunWebProducts)"  
7 217.230.114.243 - - [01/Jan/2005:03:40:35 +0100] "GET /welcome.phtml HTTP/1.1"  
   200 2987 "-" "Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; Win 9x 4.90; DT  
   ; FunWebProducts)"  
8 217.230.114.243 - - [01/Jan/2005:03:40:35 +0100] "GET /images/portal-deutsch.gif  
   HTTP/1.1" 200 416 "-" "Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; Win 9x  
   4.90; DT; FunWebProducts)"  
9 217.230.114.243 - - [01/Jan/2005:03:40:35 +0100] "GET /images/portal-english.gif  
   HTTP/1.1" 200 401 "-" "Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; Win 9x  
   4.90; DT; FunWebProducts)"
```

Das erste Feld mit der IP-Adresse des Besuchers kann unter Umständen Aussagekräftiger werden, wenn man versucht sie in einen Namen aufzulösen. Dies gelingt nicht bei allen Adressen, wie der Versuch zeigt:

```
1 207.68.146.55 - - [01/Jan/2005:00:00:21 +0100] "GET /robots.txt HTTP/1.0" 200 308  
   "-" "msnbot/0.3 (+http://search.msn.com/msnbot.htm)"  
2 207.68.146.55 - - [01/Jan/2005:00:00:22 +0100] "GET /praxisdatenbank/externe-  
   datenbanken/ HTTP/1.0" 200 8860 "-" "msnbot/0.3 (+http://search.msn.com/msnbot  
   .htm)"  
3 msnbot.msn.com - - [01/Jan/2005:00:00:35 +0100] "GET /fr/suche/ HTTP/1.0"  
   200 5153 "-" "msnbot/0.3 (+http://search.msn.com/msnbot.htm)"  
4 msnbot.msn.com - - [01/Jan/2005:00:02:01 +0100] "GET /veroeffentlichungen/  
   endbericht/6.phtml HTTP/1.0" 200 12646 "-" "msnbot/0.3 (+http://search.msn.com  
   /msnbot.htm)"  
5 msnbot.msn.com - - [01/Jan/2005:00:02:17 +0100] "GET /en/veroeffentlichungen/  
   zwischenbilanz/1-socially-integrated-city-appraisal.shtml HTTP/1.0"  
   200 171462 "-" "msnbot/0.3 (+http://search.msn.com/msnbot.htm)"  
6 pD9E672F3.dip.t-dialin.net - - [01/Jan/2005:03:40:24 +0100] "GET /sozialestadt-  
   intro.swf HTTP/1.1" 206 114950 "-" "Mozilla/4.0 (compatible; MSIE 5.5; Windows  
   98; Win 9x 4.90; DT; FunWebProducts)"
```

```

7 pD9E672F3.dip.t-dialin.net - - [01/Jan/2005:03:40:35 +0100] "GET /welcome.phtml
  HTTP/1.1" 200 2987 "-" "Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; Win 9x
  4.90; DT; FunWebProducts)"
8 pD9E672F3.dip.t-dialin.net - - [01/Jan/2005:03:40:35 +0100] "GET /images/portal-
  deutsch.gif HTTP/1.1" 200 416 "-" "Mozilla/4.0 (compatible; MSIE 5.5; Windows
  98; Win 9x 4.90; DT; FunWebProducts)"
9 pD9E672F3.dip.t-dialin.net - - [01/Jan/2005:03:40:35 +0100] "GET /images/portal-
  english.gif HTTP/1.1" 200 401 "-" "Mozilla/4.0 (compatible; MSIE 5.5; Windows
  98; Win 9x 4.90; DT; FunWebProducts)"

```

Offenbar können die ersten beiden Einträge nicht aufgelöst werden. Die der Zeilen drei bis fünf werden zu `msnbot.msn.com` aufgelöst. Damit sind sie als Roboter eines Suchmaschinenbetreibers einzustufen. Ebenfalls zu diesem Ergebnis kommt man, wenn man sich die `agent`-Bezeichnung der ersten fünf Einträge ansieht. Sie deuten alle auf die gleiche Suchmaschine von MSN hin.

Die restlichen Einträge stammen höchstwahrscheinlich von „echten“ Besuchern, der den MS Internet Explorer unter Windows 98 zu benutzen scheint und sich über eine DSL-Leitung der Telekom eingewählt hat.

Zur automatischen Analyse der Webserverlogs existiert unter anderem das Programm *AWStats*, von dessen Übersichtsseite (Abbildung 3.1) man statistische Daten der Besucher einsehen kann. Die Informationen beziehen sich auf die Browser und Betriebssysteme der Besucher. Über das GEOIP-Plugin kann dabei das Herkunftsland und die Stadt, bzw. Region des Surfers herausgefunden werden. Auch wird versucht beliebte Ein- und Ausstiegsseiten herauszufiltern. Mit diesen Informationen können Wegdesigner und Administratoren die Seite besser an die Besucher anpassen. Auch lassen sich selten bis gar nicht genutzte Inhalte identifizieren.

3.3 Ausblick

Jenseits der beschränkten und unzuverlässigen Informationen, die ein Besucher beim Webserver hinterlässt, ist es möglich das Logging direkt im Browser des Benutzers durchzuführen. Durch einen, für Versuche modifizierten Browser, können mehr und genauere Informationen gewonnen werden, wie zum Beispiel genauere Zeiten und Klicks innerhalb des Dokuments.

Desweiteren gibt es Programme zur Visualisierung und Auswertung von „Mausspuren“¹, mit denen zusätzlich die Mauspfade als Informationsquelle aufgezeichnet werden.

¹siehe dazu den Vortrag und die Ausarbeitung von Arne Hoxbergen und Konrad Hilde

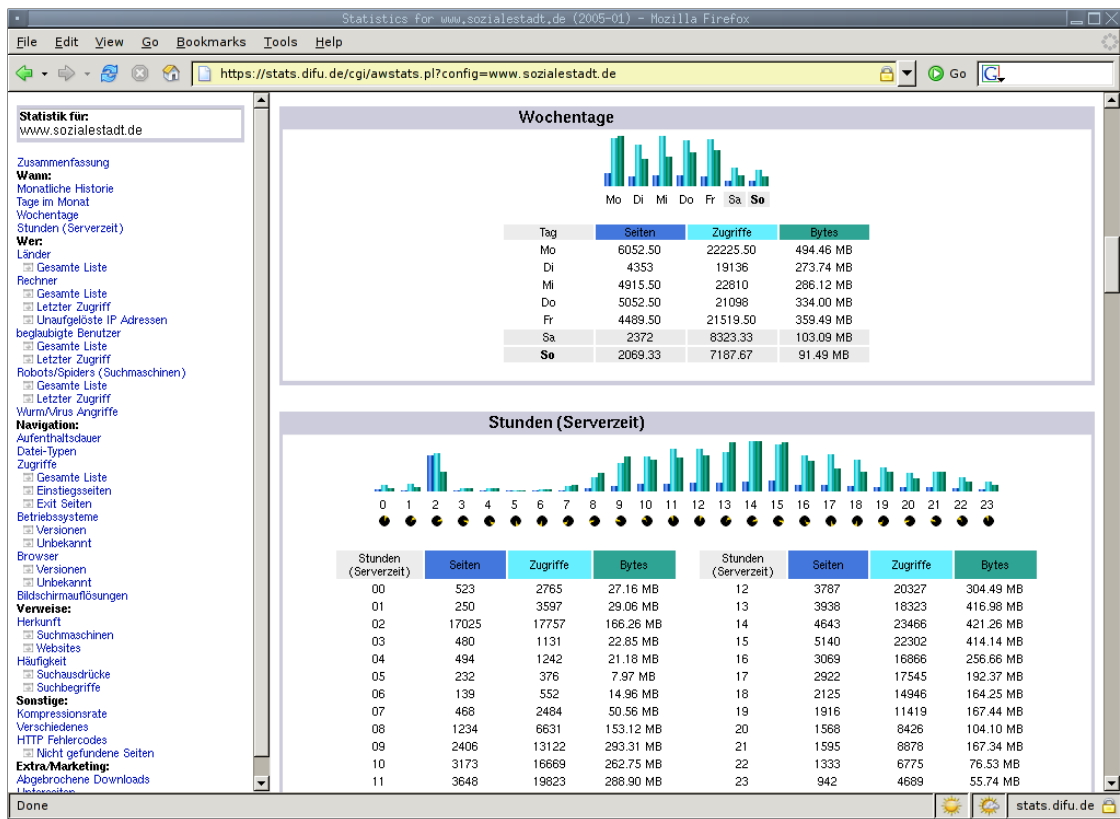


Abbildung 3.1: AWStats - der Seite <http://www.sozialestadt.de> vom Januar 2005

4 Zusammenfassung

Insgesamt läßt sich über die Methode der Logfileanalyse sagen, dass sie sehr gut in Beziehung auf *Objektivität*, *Reliabilität* und *Validität* ist. Allerdings ist sie bei alleiniger Anwendung nur wenig Aussagekräftig. Um diese Schwierigkeit zu überwinden, sollte man sie mit anderen Methoden kombinieren.

Als besonderer Vorteil sticht die gute Skalierbarkeit hervor. Ist die Versuchsumgebung einmal entworfen, lassen sich beliebig viele Versuchspersonen einsetzen. Dadurch kann die Logfileanalyse sehr ökonomisch eingesetzt werden.

Diese Methode zeigt sich sehr interdisziplinär. Um die Auswertung zu erleichtern, können Mustererkennungstechniken angewandt werden.

Literatur

- Heide, B. H. auf der. (1993). Welche software-ergonomischen evaluationsverfahren können was leisten? In K.-H. Rödiger (Ed.), *Software-Ergonomie '93, Berichte des German Chapter of the ACM* (S. 157–171). Stuttgart: Teubner.
- Jordan, P. W. (1998). *An introduction to usability*. London: Taylor & Francis.
- Rauterberg, M., Spinas, P., Strohm, O., Ulich, E., & Waeber, D. (1994). *Benutzerorientierte Software-Entwicklung* (Band 3). Zürich, Schweiz: vdf, Hochschulverlag an der ETH Zürich.