FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

The Imperfect Remote Control

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

im Rahmen des Studiums

Information & Knowledge Management

eingereicht von

Stefan Linecker
Matrikelnummer 0326868

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer/in: Ao.Univ.-Prof. Mag. Dr. Margit Pohl

Wien, 15.06.2013

(Unterschrift Verfasser/in)

(Unterschrift Betreuer/in)

Erklärung

Stefan Linecker
Hörzing 5
4912 Neuhofen

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

_____ _____

(Ort, Datum) (Unterschrift)

Kurzfassung

Die klassische Infrarot-Fernbedienung ist ein weitverbreitetes, lange erprobtes Gerät. Mit dem Aufkommen von Interaktiven Fernsehen und den damit verbundenem Angebotswachstum von Inhalten und Features droht die klassische Fernbedienung jedoch an ihre Grenzen zu stoßen. Die allgemeine Zielsetzung dieser Arbeit ist zur Benutzerzufriedenheit von Interaktivem Fernsehen beizutragen – dieses Ziel wird verfolgt, indem untersucht wird, ob Tastendrücke auf der Fernbedienung signifikante Rückschlüsse über die Gestaltung der Fernbedienung als solches und der damit verbunden Benutzerschnittstellen am Fernsehgerät zulassen. Eine Methode aus dem usability testing namens remote logging wird zur Datenbeschaffung verwendet, die eigentliche Analyse der Daten erfolgt unter Zuhilfenahme von Methoden aus der Informationsvisualisierung. Die zentralen Erkenntnisse dieser Arbeit bestehen aus drei speziell angepassten Visualisierungsmethoden und einer Auswahl an Mustern, die durch Anwendung eben dieser Visualisierungsmethoden gefunden wurden.

Abstract

The traditional infrared remote control is a widespread and time-tested device but, with the advent of interactive television and the accompanying growth of available content and features, it might reach its limits. The overall aim of this work is to contribute to customer satisfaction of interactive television by exploring if the data of button presses in such a system is useful for gaining significant insights for the design of remote controls and interaction television user interfaces. It does so by applying a usability method called remote logging and information visualization techniques are used to analyze the gathered data. The central findings of this work consist of three purpose-build visualization techniques and a variety of patterns found using these methods.

Contents

List of Figures

List of Tables

List of Listings

This Page Intentionally Left Blank.

Chapter 1 : Introduction

1.1 Motivation

Television remote controls are one of the typical technical artifacts found in most of today‘s households. They were invented over 60 years ago and continuously adapted to the needs of the current technologies. These were videocassette recorders in the eighties, satellite television in the nineties and DVDs shortly after the millennium. With the emerging technology of interactive television, the traditional remote control might reach its limits. The user must be enabled to interact with a dynamically growing volume of content and features while still expecting an intuitive, easy to use interface.

Too often, the usual approach for coping with increasing complexity is rather technocratic. Adding a few more buttons here, a few more menu options there. This has already made the remote a rather crowded device and the overall experience of watching television far too complex. Programming the VCR, and the frustration this yields, is one of the quasi-classical examples in usability literature.

In my belief, there are two key insights to tackle this problem. First, modern consumer electronics isn‘t about isolated devices anymore but about interplaying services and experiences. If you want to build something pleasant, you should not focus on the design of a single device, but on the design of a whole service [Shostack 1993, Mager 2009] and the role of this service in a larger ecosystem. Second, there might not be a one-stop solution for reducing the complexity of the television experience but there might be a promising process. Interactive television is a virtual product, built upon software. This potentially enables rapid iteration and extensive usability testing and can ultimately lead to an evolutionary design process, where different features and designs can be implemented, evaluated and refined continuously [Rampersad 2010] and efficiently.

This work is all about the second point and its focus lies on the *evaluation* of remote controls. Thereby, the overall aim of this work is to contribute to customer satisfaction of interactive television by developing guidance for the design of remote controls and the corresponding Graphical User Interfaces found in interactive television applications today.

1.2 State of the Art

Interactive television is one of today's hot topics and besides broadcasters and companies in respective industries, scientific institutions in the areas of human-computer interaction, audience research and media research contribute to this field. An interesting summary on the history of interactive television can be found in [Jensen 2008].

Usability in general, and particularly the assessment of navigational features play an important role in research of interactive television. Kunert [Kunert 2009] gives an excellent overview of design guidance in this field and, at the same time, mentions some very important structural problems: research in this area is often bound to specific environments (a specific broadcaster's offerings, designated middleware, etc.) and a lot of research is hidden behind cooperate walls or at least hard to validate scientifically.

Cooper [Cooper 2008] provides an interesting overview of the interactive television user experience and the role of the remote control within this context, so does [Darnell 2008] and [Epelde et al. 2011]. A variety of authors have criticized the usability of traditional remote controls [Daly-Jones 2003, Eronen 2004] and there is a variety of research on alternative input methods like speech recognition [Roibás et al. 2005], the use of everyday objects [Aroyo 2007] and gesture recognition [Freeman et al. 1994, Kim et al. 2004]. One of the most established events (and in the form of its proceedings, also one of the most interesting publications on the topic) is the European Conference on Interactive Television (or short EuroITV[1]).

In opposite to the relatively new topic of interactive television, usability testing is a well known, scientifically covered [Nielsen 1994, Shneiderman 1986] and accepted (yet still highly dynamic) territory. The methods of *remote logging* or *log file analysis* [Andrews 1998, Valdman 2001], which were used in this work belong to the standard repertoire of usability testing. Visualization techniques for log file data were described by [Guzdial et al. 1994] and the capturing of user interaction in interactive television environments, just like it will be described during the course of this work, was also described in [Teixeira et al. 2009].

[1] http://www.euro-itv.org/ (Retrieved 2012/12/31)

1.3 Research Question

As mentioned, the overall aim of this work is to contribute to customer satisfaction of interactive television. The thesis tries to achieve this by visualizing remote control button presses. Since the design space of a traditional remote control is rather limited, the questions during the early stages of research typically sounded something like: *Which physical buttons have to reside on the remote control?* or *How should the button layout look like?* It became clear soon that for answering those simple questions, more questions have to be asked, namely *Which methods can be used to visualize the collected data?* and *Which phenomena or patterns can be observed through the visualizations?* and *Which recommendations for the design of remote controls can be inferred?* Those questions were of practical nature, determined by the starting point (the log file data) and aim (to give some remote control design guidance) of the work. All those individual questions are surpassed by the following, core question of this thesis, namely:

> *Is the data of button presses in an interactive television system useful to gain insights for the design of remote controls?*

1.4 Overview of Thesis

The thesis is split into two major parts. The first part contains all the theoretical foundations while the second part is all about *data*. Using a real-world interactive television system and a usability testing method called *remote logging*, I will try to reveal ways for the semi-automatic evaluation of remote control usability.

The theoretical part starts with *Chapter 2* where the remote control is in the center of attention. This chapter starts with a section about the history of the remote control, followed by a comparison with other consumer electronic devices. After that, the features of the traditional remote control and its alternatives are evaluated. The chapter ends with some remarks and the description of the remote control used throughout this work.

Chapter 3 is all about interactive television. After a short introduction to the term itself and some of the most important interactive television services, there is a section about all the changes this technology brings, followed by a section about the paradoxes of interactive television. Last but not least the specific interactive television system is roughly sketched.

Graphical user interfaces reshape the interactive television user experience. and *Chapter 4* is all about that. The chapter starts with a brief introduction

about graphical user interfaces in the area of television. This is followed by a section on the advantages of those graphical user interfaces and another section with some innovative examples. At the end of this chapter, the graphical user interface which was used throughout the experiments is introduced.

The final theoretical chapter, *Chapter 5*, is about Usability Engineering. After usability and usability testing is discussed, there is a section about the core methodology for the following evaluations that‘s called *remote logging*. The chapter ends with an overview of the study setup.

In contrast to the former chapters, *Chapter 6* is totally data centric. It‘s all about the Physical Layout of the remote control. Two specific properties, namely button count and button placement, are covered in detail. Furthermore, two visualization techniques for those properties are presented.

Chapter 7 is all about the interplay between the physical layout of the remote control and the virtual user interface on the television screen. The method of investigation hereby is the search for patterns in the log data. As with the prior chapter, dedicated visualization methods are presented that help understand the data.

This work ends with *Chapter 8,* where you can find a highly condensed summary of the conclusions.

Chapter 2 : Remote Control

2.1 Evolution of the Remote Control

The idea of controlling things remotely is as old as mankind[1] and while it would be worthwhile to speculate about the origin of this concept, the relevant timeframe for this work only starts at the end of the 19th century. Back then, the very first experimental remote control devices were built and used for controlling vehicles and vessels. Technical developments continued, but it took the remote control until World War II to reach *critical mass*. Many of the basic research in this area is attributed to military purposes and funding.

The commercial availability of remotes in the area of television started in the fifties. One of the first products was the *Zenith Space Commander 300*, developed by Austrian-born Robert Adler. A thoughtfully engineered and beautiful device, now popular among collectors of vintage electronics. The *Space Commander 300* only had two buttons, one for zapping around the channels and one for choosing the volume. One button per dimension was enough to navigate through the cycle. This sounds a bit odd today, but with the limited number of states and the very expensive hardware back then, this was perfectly convenient. The very first remote controls used small hammers and aluminum rods to generate high-frequency tones. These devices were purely mechanic and in contrast to their modern counterparts, required no battery at all. Following these very early models ultrasound was used for signal transmission, which dominated the market for the next twenty years. The remote control in its contemporary form, using integrated circuits and infrared, was born in the eighties.

Back in the early days of the remote, it was a real challenge to the engineers. The technology was new, the components were costly and there were no industry standards. Speaking of today, the challenges have changed. Han Kohar, Director of User Interface Design at Philips Design states that *„The increasing number of functions on the TV has caused remotes to become crowded and complex. People struggle to find what they are looking for*[2]*."* The main concern isn't about technology anymore but about user experience.

---

[1] Where „controlling remotely" is not meant as a technical term but as a general concept. For instance, the invisible forces of the ancient deities were referred to as *actio in distans*.

[2] http://www.design.philips.com/philips/sites/philipsdesign/about/design/designnews/newvaluebydesign/march2010/less_is_more.page (Retrieved 2010/10/04)

Figure 2.1 shows the evolution of the remote control. It starts with the *Zenith Space Commander 300* in the sixties. As mentioned above, this remote had two buttons. One for zapping around the channels and one for choosing the volume. The following *NEC RD-104* is typical for the eighties. It added a power button, mute and numeric keys. The *Grundig TP720* added additional buttons for teletext, as well as color and direction keys. This dates back to the nineties. Last but not least, the fairly recent *Sony RM-ED012* features some additional buttons for playback, recording and settings, exceeding a total button count of fifty.

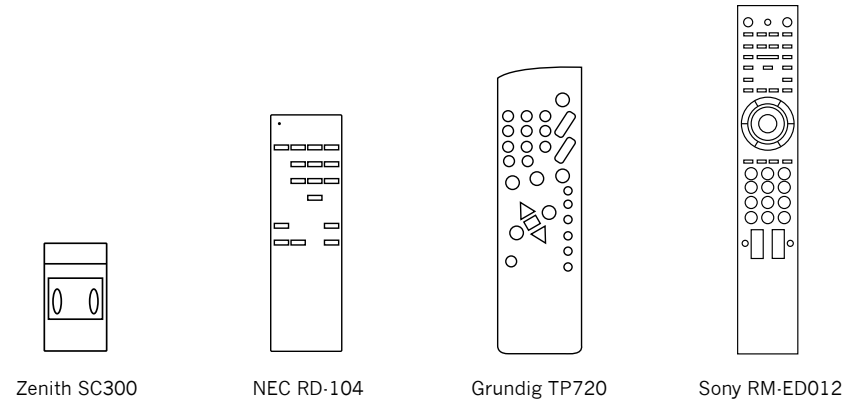


Figure 2.1: Evolution of the remote control

Strictly speaking, the above figure is trimmed to support my argument. Anyhow, the examples are pretty illustrative and the remote control is indeed getting glut. Imitating Jacob Nielsen[1], I counted the total number of buttons in my living room. Spread among five different remote controls, I counted[2] 167 buttons. Nielsen even counted more[3]. In [Cooper 2008], William Cooper states that *"the remote control has been a significant innovation in the television user experience that we now take for granted. However, experience has shown that even the design and use of particular buttons on the remote control*

---

[1] Jacob Nielsen's Alertbox, June 7, 2004: Remote Control Anarchy. (http://www.useit.com/alertbox/20040607.html, retrieved 4.10.2010)

2 *Fisher REM-540* controlling the stereo equipment with 15 buttons, *Humax RS-591K* controlling the digital satellite receiver with 36 buttons, *Grundig Tele Pilot 160 C* controlling the television set with 35 buttons, *Xoro HSD 8420* controlling the DVD player with 45 buttons and an unknown *Philips* remote controlling the VCR with 36 buttons. The total button count is 167.

[3] 239 buttons spread among six remote controls.

can have a significant impact on the usability of an interactive application." If every single button counts, one might conclude that with every additional button, reaching proper usability gets tougher. This fits perfectly with the former say of Han Kohar where the words *crowded* and *complex* are ultimately used in a very close connection.

2.2 Comparison to other devices

It's interesting to compare the remote control with other devices in the area of consumer electronics. I counted the buttons on several of those devices. Additionally, I also counted the buttons on some remote controls and made up the following tables.

Device	*Button Count*
Logitech M90 Mouse	3
Canon Digital IXUS 60	12
Sony Playstation 3 Controller	16
Nokia 6300 Mobile Phone	21
Apple Wireless Keyboard	78

Table 2.1: Button count of some CE devices

Device	*Button Count*
Bang & Olufsen Beo4	36
Philips RCPF03E08B	38
TiVo DVR Remote	39
Artemy Lebedev BBK RC026	43
Microsoft MCE Remote	44
UPC Digital Remote	44
Samsung MF59-00298A	47
Sony PS3 Remote	51
Sony RM-ED012	53
LG 42LT75	62

Table 2.2: Button count of some recent remote controls

The mean button count of *Table 2.2* is 46. The considered mobile phone, a device with comparably many navigational needs, has less then half the buttons. This is hardly surprising since the mobile phone is a menu driven device and its buttons are, for the most part, general purpose.

I think it‘s valuable to compare the remote to other devices which have to deal with ever increasing feature count. The modern car [Damiani et al. 2009] is an interesting example. It features a broad range of computerized systems for navigation, communication, entertainment and driver assistance. All these systems compete for user attention, a fact for which car manufacturers get both fame and shame. The current in-car user interface is a mixture of both highly specialized and generic interface elements and every manufacturer has its very own design philosophy. Despite these differences, there are prominent and commonly used principles, one of it being the consideration of context [Bellotti et al. 2005]. Parking Assist Systems, Lane Change Assistants or Navigation Systems are perfect examples. They stay quiet all the time, but in the moment of need, they come up. In opposite, the remote control does not reflect context. It always has the same buttons, whether they are functional at the very moment or not. This lack of feedback yields many usability problems.

Another prominent device is the mobile phone. Undeniably, the mobile phone gets new features at a very high rate. Unlike the car or the remote control, the mobile phone has never suffered from excessive button count. This is a simple matter of size. Too small buttons or too large dimensions are both unthinkable in the highly competitive mobile phone market. Therefore, most of the features of the mobile phone aren‘t directly accessible via buttons but through the display.

Comparing the remote control to the car and the mobile phone helps to recognize one very interesting difference. Unlike the phone or the car, remote controls aren‘t high tech, high price devices. They are an accessory to the television and were often only considered a parasitical cost factor.

2.3 Features of the Remote Control

Edsger W. Dijkstra once said that *„Simplicity is prerequisite for reliability“*. In my believe, cost constraints and the subsequent restriction to „simple“ hardware aren‘t bad things per se. Despite the relative simplicity, the remote control has many features and qualities. We might not explicitly notice this features anymore but we miss them if they suddenly disappear. Hereafter, I will try to summarize these features, assuming a traditional[1] but well designed remote control.

[1] With „traditional“, I mean a button-only remote without additional hardware features.

1. *Remotes are robust.*

   They typically survive a fall from the coffee table, a device with a fancy touch screen possibly won‘t. Fragile devices require lots of attention, but especially in the living room you don‘t want to look after your devices but after your guests or family. Also, remotes are quite resistant to dirt and liquids and since remote controls are also cheap (wholesale prices are typically beyond ten dollars), it isn‘t that much of a fiasco to replace one. Another convenient thing is that remotes only require little maintenance. All you have to do is change the battery once a year. No charging stations, no software updates, no boot time.

2. *Remotes are universal.*

   A very important design constraint with remote controls is *„one size fits all“*. And it isn‘t only about size but about every single perceivable dimension. Human perception is not deterministic and the way I perceive something does not imply that all others perceive the same. Roger Whitehouse uses the term *perceptual fingerprint* to describe the uniqueness of everybody's own perception [quoting Norman 1988]. Well designed remote controls have to work for a broad set of perceptual fingerprints, without any restrictions to age, acuity or handedness. Depending on the actual design, traditional remote controls fulfill these requirements quite well.

3. *Remotes are supportive.*

   The size and shape of the remote fits our hands. The materials of the body and the buttons support haptic perception[1]. These things follow well understood and documented laws [Kern 2008; ISO 9241]. The button layout is in a logical manner (e.g. [Wertheimer 1923]). All in all, the traditional remote control is a really convenient device. If it is well designed it can be used one-handed, without looking at it and even without light. One could say that if it is a good one, it works *without thought* [Krug 2005].

2.4 Alternative input methods

Thinking about the pros and cons of the traditional remote control, the dominance of this device might have an additional cause in the lack of good alternatives. According to Pablo Cesar, the possible alternatives are *„extension of traditional remote controls, augmentation of everyday objects such as*

[1] Remotes like the Samsung BN59-00545A even feature Braille markings next to the most important buttons.

pillows or paper, and repurposing of other personal devices such as mobile phones." [Cesar et al. 2008, p. 2]. Deviant from Cesar, I did not try to categorize the possible alternatives but I had a closer look at the four alternatives which I personally thought of being the most promising.

1. *Touchscreen remotes*
   One rather obvious possibility to increase the navigational power of the remote control is to add technical features. Prominent examples are all kinds of multidimensional knobs and displays but as with the mobile phone the current trend goes towards touch sensitive displays. Touchscreen remotes replace the physical buttons of the traditional remote control with virtual buttons and controls. This adds flexibility and scalability since button layout then is a matter of software. Unfortunately touchscreen remotes come at a very high price. Virtual buttons eliminate all haptic navigation features. Your fingers don't get any feedback, making it almost impossible to find the right button without looking at the remote. Additionally touchscreen remotes aren't as robust as traditional remotes. Just think about fatty fingers. Last but not least, another crucial thing is price. The recently introduced *Samsung RMC30C2*, an advanced touchscreen remote control, costs more then two hundred dollars.

2. *Software*
   Another possibility is to use other devices to take over the job of the remote control. Thereby, every device with networking capabilities has the potential to become the replacement [Lo et al. 2005]. More and more manufacturers start providing particular software for different platforms. The most prominent example might be the iPhone application *Remote*[1] for remotely controlling an Apple TV. This can be quite convenient, since browsing through the next weeks of programming can be achieved much easier with suitable software on the computer screen then on the television. The problem is that software solutions in this context usually only work in addition to the traditional remote control. If the mobile phone is gone or the laptop is off, you need a dedicated, reliable device to take over control.

3. *Speech*
   Speech input uses human speech for specifying commands or for interacting with a dialog-based menu system [Wittenburg et al, 2006]. Speech input has already proven its effectiveness in some real world

[1] http://www.apple.com/itunes/remote/
(Retrieved 2010/10/04)

scenarios but it is questionable if this technology is appropriate for the living room. There are indicators for the potential of this technology [Berglund et al. 2004] but their are some problems. The first problem is the placement of the microphone. If you put it into the television set, noise levels are low and filtering gets tough. If you put it in the remote control or another dedicated device, interaction probably requires somewhat unnatural behavior. Speech recognition works best with a closed sets of commands and options, where technical finesses and artificial intelligence are able to increase the hit ratio, even with sloppy speech. With interactive television, you have a multilingual environment with lots of proper names, made-up words, slang and unlimited input. This is difficult terrain. Additionally, speaking to a computer might simply make us feel somewhat uncomfortable and a high performance microphone in an internet-enabled device in the middle of the living room may raise concerns about privacy and surveillance.

4. *Gestures*
   With gesture recognition, hand gestures are used to remotely control the television [Freeman et al. 1994]. Thereby cameras continuously capture the viewer and image processing techniques are used to detect commands in the captured pictures. Unfortunately, gesture recognition fails due to some very basic things, the most important being that it requires direct sight to the television set or to an input camera respectively. Further disadvantages include that the gesture interactions have to be learned first.

2.5 Conclusions

At the beginning of this chapter, we had a look at the evolution of the remote control and saw that, if it continues like that, the traditional remote control might reach its limits soon. The sheer number of buttons is exceeding a reasonable quantity. This diminishes the original qualities of the remote and frustrates the user. With the emerging technology of interactive television (which will be introduced in the next chapter) and its many features, things will even get worse. Adding buttons will not be a feasible solution anymore.

We then had a look at the alternatives to the remote control. All of them are somewhat promising but none of them seems to completely replace the traditional remote control in the near future.

If you need more navigational power but you can neither expand the traditional remote control, nor are there any reasonable alternatives to the remote control, you have to solve the problem elsewhere. The usual solution is to move functionality from the physical body of the remote control to the screen of the television set. Graphical User Interface elements are already widely used within the area of television but only with the dynamic, networked nature of interactive television do they unfold their full potential. Those Graphical User Interfaces will we covered in *Chapter 4*.

The important questions on the design of the remote control and the corresponding user interfaces then typically are things like: Which physical buttons should the remote have? How to organize the buttons? Should I solve this with a menu item or a dedicated button? And so forth. These questions sound trivial at the very first moment but they are key issues concerning the user experience of interactive television. I will try to bring up many of these matters in the course of this work.

2.6 Study setup

Throughout this thesis, the following remote control layout was used.

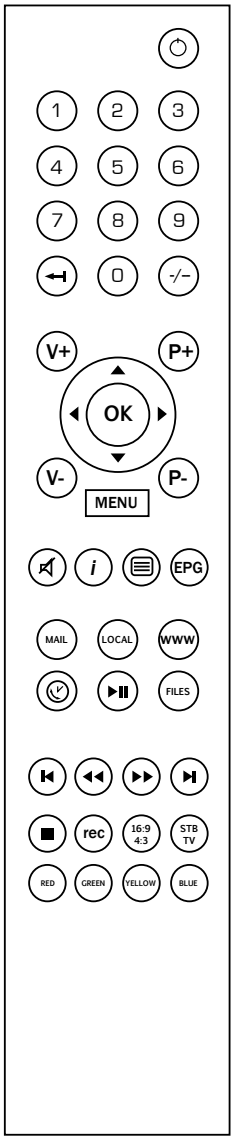


Figure 2.2: Remote control layout used for study

It‘s a custom remote control specially developed for the interactive television system I had access to. The average button count of the former remote controls used to be 46, this has 42. Concerning both button count and functionality, this remote should be pretty representative and comparable to many other ones.

Chapter 3 : Interactive Television

3.1 Interactive Television defined

This work arose from the need of reducing the complexity of remote controls. Throughout the document both the increasing complexity and the possible design solutions to cope with this complexity have their origins in the emerging technology of interactive television.

Interactive television is a fairly recent term. Technologies and services in this area are currently evolving rapidly and so do the definitions. On the one hand, interactive television is used for *„any television or video programming that incorporates enhanced content or some style of user interactivity“* [Karyn 2005, p.1]. Voting for candidates or answers in quiz shows is a classical example for such interactivity. On the other hand, interactive television *„is also used as an umbrella term to cover the convergence of television with digital media technologies“* [Karyn 2005, p.1]. Video on Demand[1] would be a typical example for the later definition.

Interactive television has its origins in the former definition, that‘s where „*interactive“* comes from[2]. Meanwhile, the second definition is more important and interactive television is frequently used when people are primarily talking about media convergence and media distribution, rather then interactivity per se. Please take this into consideration.

Many times, the enabling technology behind interactive television is IPTV, which stands for Internet Protocol Television. With IPTV, content is delivered using broadband internet connections rather then satellite or radio frequency signals and television is not coming over the air anymore but over the internet. Currently, many TV manufacturers are equipping their sets with internet capabilities, enabling features like weather widgets or video streaming, where additional content is retrieved via the web. These television sets still use traditional reception techniques for television content and IPTV for additional content, combining two very different worlds. When speaking about interactive television, the degree of interactivity is highly dependent on the underlying technology[3]. Throughout this work, IPTV is assumed to be this technology.

---

[2] Some authors distinguish different levels of interactivity, e.g. [Ruhrmann et al. 1997]

[3] Simple forms of interactivity can also be realized without the „cloud“, e.g. by bundling all possible alternatives with every dynamic choice.

Returning to the definition of interactive television, it is sufficient for this text to think about it by terms of its characteristics. Following Kunert, a good way to approach this is a comparison [Kunert 2009, p.19] between interactive television and traditional television. I will utilize the characteristics of IPTV for this comparison.

1. *Traditional television is standardized.*

   Due to the very nature of the medium, the same signal is broadcasted to all recipients. In contrast, IPTV relies on packet-switched networks for signal distribution, where every chunk of information typically contains the unique address of the receiver. Thus, every recipient could potentially get its very own data stream and therefore its very own programming.

2. *Traditional television is time dependent.*

   At any given time all recipients receive the same content simultaneously. As stated above, with IPTV, every recipient could get its very own data stream. This enables time independence. Content can be buffered centrally and is then further distributed on behalf of a specific user at any time.

3. *Traditional television is one way.*

   Terrestrial, satellite or cable television all have one central data source and many peripheral data sinks. Data always flows from the source to the sinks. This is completely reversed with interactive television. There, every client has the capabilities both to receive and transmit information. It is no problem if downstream rates are far beyond upstream rates (as it is with ADSL) but some kind of return path is an absolute necessity for true interactivity.

Beside these three very important characteristics, the significance of this work is closely tied to the associated experience. In this text, watching television is understood as a lean back, low interaction activity in a typical living room environment. It is further assumed that a traditional remote control is the primary component of physical interaction.

3.2 Interactive television services

After defining interactive television, let‘s now take a look at some of the common interactive television services. Without making claims of being complete, the following listing describes some of those.

- *Television*
  The main difference between IPTV and things like *Apple TV*[1] or *Roku*[2] is television. Streaming boxes usually don‘t bring along live television since they are mostly used in addition to some cable service. With IPTV, live television and live radio are integral parts of the experience. Thereby television is extended by features like trick play and instant recording.

- *Timeshift*
  The digital and centralized nature of content distribution allows centralized ring buffers. This enables so called time-shifting, where content is recorded to a digital „archive“ and can be viewed at a later time.

- *Personal Video Recorder*[3] *(PVR)*
  This basically is the same as a VCR but recording takes place centrally and in a digital manner. Unlike VCRs, nPVRs are fully integrated in the overall system, which greatly simplifies the associated user experience.

- *Interactive Program Guide*
  Interactive program guides (IPG) are the online version of the TV guide. They display scheduling information in an interactive, explorable manner.

- *Internet on television*
  The networked nature of interactive television enables access to all kinds of internet services like social networks, news sites or web radio. This might work via a general-purpose web browser or via specialized applications.

- *Video-on-Demand*
  This is the virtual counterpart to the video rental shop. Instead of a

[1] http://www.apple.com/appletv/

[2] http://www.roku.com/

[3] Respectively *networked* Personal Video Recorder (nPVR)

physical store, it consists of an online archive of digital video content where consumers can browse and rent movies or TV shows.

- *Local Services*
  IPTV network operators have far more knowledge of their customers than broadcasting stations. This enables all kinds of personalized and localized offerings.

3.3 Stakeholders and their motives

One might say that interactive television had kind of a hard start. Its history already contains dead standards[1], bankruptcies and many overstated forecasts and expectations[2]. In my belief, usability problems aren't always caused by bad design or lack of technical ingenuity but by other forces. Time to market, statutory provisions, standard wars, or software patents all might have negative influences on the usability of interactive television. I think that it's very valuable to take a look at the big picture, to identify the mayor stakeholders in this area and examine their motives.

Public institutions play a very important role here. First of all, they determine copyright legislation. Those copyright laws have vast influence on the whole area of interactive television and some services entirely depend on nifty nuances of those laws. In addition, public institutions often invest in the expansion of broadband infrastructure. Interactive television clearly requires broadband internet connections to work. Vice versa, large investments in broadband infrastructure are often justified by the requirements of interactive television. The larger goal here is to bring television from air to ground [Negroponte, p.20]. Bandwidth in the air is physically limited and therefore the most expensive. With broadband cables, this typically is just a matter of investment - if you need more bandwidth, you have to dig more fiber. The public institutions usually have the monopoly of regulating frequency spectrums. If they were able to relocate television from air to ground, they would be able to resell those very expensive and limited frequency spectrums to mobile service providers.

---

[1] The most prominent one being MHP.

[2] http://www.koreatimes.co.kr/www/news/biz/2008/11/123_32839.html (Retrieved 2010/10/06)
http://www.presseportal.de/pm/16952/1077562/capgemini (Retrieved 2010/10/06)
http://www.adlittle.de/1169.html?&no_cache=1&view=252 (Retrieved 2010/10/06)

The commercial stakeholders include whole industries. Semiconductor manufacturers try to sell System-on-a-Chip solutions, computer makers try to sell servers, storage systems and networking equipment. Software vendors offer operating systems for servers and embedded systems, all the middleware and licenses for video and audio compression.

In addition to technology, interactive television also requires content. The obvious stakeholders in this area are movie studios, record companies, publishing houses, and so forth. In fact, everybody who wants to monetize digital assets has a clear interest in interactive television becoming one more channel of distribution.

3.4 Changes with interactive television

Interactive television yields features that fundamentally change the way of the television experience. There are no widely accepted design patterns yet, so it requires a lot of effort to build systems that use all those new functionalities in a convenient manner. I had a look at the common features and services of interactive television and tried to identify the core changes with respect to the user, resulting in the following list.

1. *It's all about software.*
   Modern television sets essentially are *„displays filled with tons of memory and lots of processing power"* [Negroponte 1996, p. 46]. In this sense, they are much more like a computer then a television. In the area of television, many things which were traditionally realized with specialized hardware are now getting *„computerized"* and replaced by their digital successors. HF reception is replaced by IP networks, the PAL encoding system is replaced by MPEG streams, OSD encoders are replaces by pixel planes. This transition isn't only happening with consumer electronics, the real revolution is happening centrally. Instead of a small number of broadcasting stations, a loosely coupled and potentially infinite network of content providers takes over the supply of content and services. All in all, the television experience isn't about an isolated physical artifact anymore but about a massive software ecosystem.

2. *It's interactive.*
   One might think about two kinds of interactivity. The first one, which I will further refer to as *„implicit interactivity"*, includes all user interactions which are necessary to reach a certain goal, e.g. all steps to find a specific movie. Thereby the interactions are triggered by user

demand and are not a goal in itself. I call the second kind of interactivity *„explicit interactivity"*. Thereby, user interaction isn't triggered by user demand but by contextual causes. Examples include the participation in quiz shows, opinion-polls, contextual interactive ads and contextual betting. Such interactions sometimes contradict the lean-back manner of watching television. If explicit interactivity is used unwisely, it feels just like pop-ups on webpages.

3. *Time is a primary navigational dimension.*
   One very prominent feature-category with interactive television is time-shifting. This includes features like catch-up TV or smart pause where you view a broadcasted program at a later time. Time then is a primary navigational dimension. Just like choosing the channel, you also choose the time. This is one of the toughest features to communicate but people who master this functionality tend to use it really frequently[1]. Time-shifting is the new zapping.

4. *It's connected.*
   Modern interactive television is connected to both local and global data networks. Local networks enable the interplay with other consumer equipment, especially for the purposes of media sharing and home automation. Amongst other things, global data networks enable all kinds of internet-based services. For example, current television sets with network access already provide access to online video platforms like *YouTube*. Internet access is also necessary if you want to build a marketplace for device specific applications like the *Apple AppStore for iPhone*[2]. Another very interesting use case is so called *„social television"* [Cesar 2008]. This describes the enrichment of the television experience by social interactions. Presence awareness systems, context sharing and messaging systems are all part of this category. Last but not least, the connected television enables contextual metadata. Actual content can be linked to other recourses via URLs, CRIDs[3], geo tags, etc.

---

[1] http://www.comcast.com/About/PressRelease/PressReleaseDetail.ashx?PRID=1000b
(Retrieved 2010/10/04)

[2] http://www.apple.com/iphone/features/app-store.html
(Retrieved 2010/10/04)

[3] Content Reference Identifiers (CRIDs) are globally unique identifiers for television programs, just like URLS are globally unique identifiers for webpages.

5. *Radical increase of features.*

   Today the television experience is spread amongst a number of devices, frequently including television sets, satellite dishes, personal video recorders, and DVD players. This is the source of many usability problems. All those devices behave so differently, each and every device has its own remote control and if something is broken, it‘s hard to locate the error. Modern interactive television has the potential to converge most of the typical consumer electronics in the living room into one device. This would decrement the number of different devices and unify user interactions. The television sets‘ main purpose then would not be television anymore but being the primary visual interface to all sorts of digital media. The television set usually has a premium location in the living room, and often is the best display, concerning both size and resolution, in the whole home. Therefore it should not be a surprise if people want to access their digital assets using the television set. These assets might include digital music libraries, digital photo archives, web pages, bought, rented or recorded video content, web radio stations, and so on. Add all those features to the television functionalities of interactive television and you have a really powerful device.

6. *Radical increase of content.*

   With the radical increase of features comes the radical increase of content. Internet connectivity alone adds an almost infinite source of content. Another huge source is catch-up TV. If you have 40 channels and no backlog, you have 40 choices. If you have 40 channels, an average of 25 shows per day[1] and a one day backlog you have 1000 choices[2]. Another feature which significantly increases the available content is video-on-demand. Imagine having the content of a whole movie rental store at your fingertips. Since fast, centralized storage is always getting cheaper, such archives have nearly no technical limitations. Consider Amazon VoD, which already offers 50.000[3] movies. IMDb[4] lists 1.671.500[5] movies. In my opinion, it‘s just a

---

[1] 40 channels and 25 shows per day per channel are both really defensive estimates, real world values might be significantly higher.

[2] 40 channels x 25 shows x 1 day = 1000 choices

[3] As of September 2nd, 2010.

[4] The Internet Movie Database, now a subsidiary company of Amazon.
http://www.imdb.com/
(Retrieved 2010/10/04)

[5] http://www.imdb.com/stats
(Retrieved 2010/10/04)

matter of time to have access to all of them. With such massive amounts of data, search and recommendation engines become extremely important.

All those features and changes require user interaction. Especially with new services where the functionality is not already known by the user, it is important to take great care about design. The users should have the chance to get a clear mental model of the service without to much cognitive cost. Users don‘t want to read manuals, they want instant gratification and the opportunity for safe exploration. While this section contained the descriptions of all those changes, Section 4.3 contains some actual examples how these changes can be implemented.

3.5 The paradox of choice

At the very heart of interactive television lies a glaring contradiction: Television isn‘t about interactivity but all about *inactivity*. Interactivity merely is a necessity for the larger goal of relaxation and entertainment.

With interactive television, the number of features and the quantity of content is increasing radically. We should be happy about that much freedom of choice and autonomy but frequently we are not. The interpretation of information requires attention but when watching television we usually don‘t want to concentrate. We don‘t want to waste brain cycles, we want to watch television *without thinking*. The sheer quantity of choices brings along the burden of decision making. Barry Schwarz, author of the identically-named book calls this *„The Paradox of Choice“* [Schwarz 2003]. He explains that with an increasing number of options, customer satisfaction must not automatically increase but might actually suffer. People are overwhelmed by too many options and feel uncomfortable about it. He further describes two very interesting consequences of this problem. First, too much choice makes us paralyzed. We are that overwhelmed that we choose not to choose. Second, with too much choice it gets hard to feel satisfied. One always thinks about opportunity costs and all the other choices. The importance of balancing functionality and ease of use can‘t be rejected and it‘s necessary to incorporate those ideas into the design of interactive television services. Practically this means that one should only present feasible numbers of choices to the customer, one should really care about separating the important things from the unimportant and one should always think in a user-centered manner. Many interactive television services might require serious filtering through *search* and/or *recommendations* (whether user-generated, automatic, semi-automatic or editorial).

3.6 Study Setup

The data for this work was provided by *Ocilion IPTV Technologies GmbH*, a company which I was personally affiliated with for several years. It was recorded at a real-world installation in *Ried im Innkreis* in *Austria*. A simplified version of the overall system is illustrated in *Figure 3.1*. At the top, you can see the two basic content sources which are internet and satellite. While the connection to the internet is two-way, satellite reception is one-way. Data from or to the internet is filtered through a firewall that itself is connected to the IPTV middleware. Satellite signals are received via DVB-S/S2 digital video receivers and then forwarded to the middleware via the content network. The content network might also connect storage arrays or other sources of local content archives. The middleware undertakes all kinds of processing and then forwards the desired data to the Set-Top-Boxes, where it is decoded or rendered and finally transmitted to the television set.

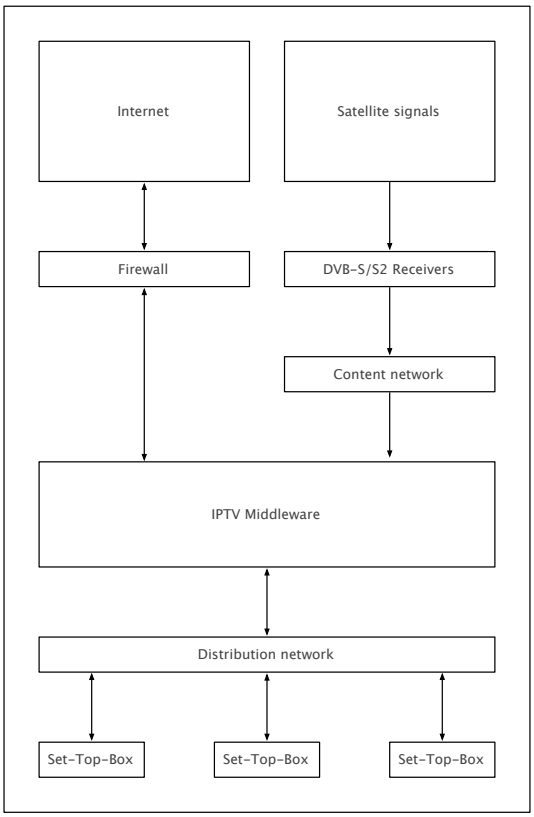


Figure 3.1: Simplified overview of the used IPTV system

The log data (which will be explained later) was recorded at the Set-Top-Box and sent back to the IPTV Middleware where it was archived. At a later time, it was downloaded from there and analyzed offline.

Chapter 4 : Graphical User Interfaces

4.1 Graphical User Interfaces in Television

The previous two chapters were about remote controls and interactive television. We saw that interactive television introduces many interesting features but the remote control might reach its limits when it is about operating all those new features in a user-friendly manner.

As stated in *Chapter 3*, the alternatives to the traditional remote control aren‘t that promising in the short term. We need more navigational power and have to decide if we want to realize this navigational power on the remote control or on the television. As we saw, upgrading the remote control with additional features might be harmful to its original qualities and brings more drawbacks than benefits. The logical consequence is to move some complexity to the television screen and to use graphical user interfaces. Graphical user interfaces are well known for being the successor of the text-oriented command-line interface and the current de-facto standard for the visual part of human-computer interaction. A graphical user interface offers a visual representation of computer programs using a simple and consistent visual language. Rather than typing (and remembering) commands, you might input information by manipulating graphical elements. Such elements might mimic physical artifacts like buttons and switches. If well designed, graphical user interfaces provide a relatively intuitive way to interact with a given computer system, even for novice or untrained users.

In the area of personal computing, graphical user interfaces bundle a multitude of techniques including windows, icons, menus, direct manipulation, etc. [Myers 1998]. In opposite to the computer, graphical user interfaces on the television usually don‘t use a mouse. Frequently they even aren‘t called graphical user interfaces and the term „on-screen display“ is used instead. This has historical reasons and dates back some twenty or thirty years. Back then, video cassette recorders came into market, and manufacturers started to use Vacuum Fluorescent Displays for displaying timing information. This displays were placed at the front of the device itself. The following figure shows a basic version of such a display.

Figure 4.1: Basic Vacuum Fluorescent Display

Unfortunately, such displays were rather expensive and the amount of information which could be displayed was limited, so manufacturers soon started looking for alternatives. They found a solution in superimposing graphical elements onto the television picture, which was then called *on-screen display*. The following figure shows such an on-screen display, more exactly an on-screen display for the timer programming of a video cassette recorder.

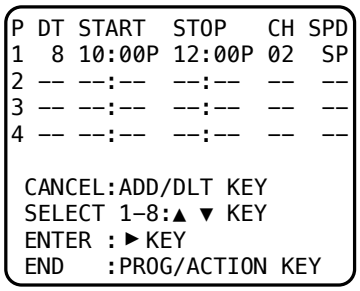


Figure 4.2: Video cassette recorder on-screen programming

The use of graphical user interfaces in the area of television isn‘t limited to interactive television. Quite contrary, graphical user interfaces are already widely used in current television equipment.

Modern Set-Top-Boxes are based on highly integrated system-on-chip solutions. Such units integrate a multitude of subsystems into a single integrated circuit. This might include video and audio decoders, security processors, various memory blocks, general purpose processors and of course a graphics subsystem. Many recent system-on-chip solutions, like the *Sigma SMP8656*[1] or the *Intel CE4100*[2] support OpenGL ES, a branch of the OpenGL 3D graphics API specifically aimed towards embedded systems. These systems have sufficient capacity to implement advanced graphical user interface.

---

[1] SMP8650 Series brochure
http://www.sigmadesigns.com/uploads/documents/SMP8650_br.pdf
(Retrieved 2010/09/10)

[2] Brochure not published as of time writing, brochure of predecessor available at
http://download.intel.com/design/celect/downloads/ce3100-product-brief.pdf
(Retrieved 2010/09/10)

4.2 Advantages of Graphical User Interfaces

In my belief, it‘s good to keep the remote control as simple as possible and implement everything else using the graphical user interface on the television screen. This leaves the original qualities of the remote control untouched while exploiting the advantages of the graphical user interface. If you have a setting where the remote control only has the most basic functionalities and is finely tuned to cooperate with the graphical user interface, there are some nice advantage:

- *Little cost of the remote control*
  If the remote control stays simple it also stays cheap. A graphical user interface is a virtual product, it costs (almost) nothing to reproduce, even if very sophisticated. Once developed, digital artifacts scale effortlessly. In contrast, an elaborate remote control would cost quite something and these costs would accrue with each and every unit.

- *Great prototyping capabilities*
  Software systems provide much better prototyping capabilities than hardware systems. Modifications can potentially be implemented, tested and deployed in a matter of hours. The usability evaluation techniques used in the forthcoming chapters will exploit the centralized, virtual nature of interactive television.

- *Flexibility for future developments*
  Using a basic remote control design and a powerful graphical user interface makes one very flexible for future developments. Replacing the remote control of every customer would be a nightmare, adding one option to an on-screen menu is an standard procedure. It‘s easy to update server-based software, it‘s hard to upgrade hardware.

4.3 Design patterns for Interactive Television

Interactive television is a relatively young technology and does not yet have its one visual language. As with any young technology, there is no widely accepted set of design rules. Nevertheless, interactive television did not come into existence out of nothing. It‘s a successor of traditional television and therefore inherits huge parts of the associated experience. Depending on the location, television might have been around for some sixty years and the influences it has on our culture are terrifying. Think about the moon landing of Apollo 11, media coverage of missiles hitting targets in the Persian Gulf War, the Apple „1984“ television commercial

for the original Macintosh or simply the launch of MTV. All those pictures are part of our cultural heritage. We might even know the meanings of such mystical abbreviations as *„V+“* or *„AV 1“*. Interactive television is also a graphical user interface and, therefore, additionally inherits huge parts from there.

If we are looking for guidance on the design of interactive television interfaces, we might use knowledge from traditional graphic design, including the resources on typography (e.g. [Frutiger 1986]), layout (e.g. [Wertheimer 1938]) or color (e.g. [Itten 1961]). We might also use guidance from the area of graphical user interfaces (e.g. [Apple 1992]), interaction design (e.g. [Moggridge 2007]) or usability (e.g. [Lund 1997]). But the television also has its very own rules. Some of them are due to the specific technologies. When designing for interactive television, one has to take care about safe area[1], pixel ratio[2], multiple resolutions[3] and a lot of other things. There is excellent guidance on those technical matters such as [BBC 2005] or [Gawlinski 2003]. Other rules are due to the specific experience. The television is usually controlled using a remote control, the distance between the television set and the viewer is rather high, it‘s a multiuser device, it often is a shared experience and so forth.

Fortunately there is real-world software where all these knowledge is already combined. It‘s interesting to evaluate such software and observe and speculate about the design decisions. I did this[4] with media center software including *Apple Frontrow, Boxee, Plex* and IPTV systems like *AT&T U-Verse TV, BT Vision, Swisscom TV or Verizon FiOS TV* and then picked up the six core changes of *Chapter 3.4* for categorization and tried to identify some common design solutions resulting in the following list of exemplary patterns. They don‘t necessarily fulfill the formal criteria for a design pattern but they are interesting examples. For a more extensive overview of design guidance for interactive television, see [Kunert 2009] or [Lu 2005].

---

[1] Televisions (especially older ones) don‘t show the full picture but cut off a small amount on the borders. The safe area defines an area where graphics and text will be displayed correctly on the majority of television screen. Traditionally these are 10% of the overall resolution for action safe area and 20% of the overall resolution for title safe area.

[2] In fact, pixels on the television screen aren‘t square. They have a ratio of 1.067 x 1.000.

[3] The battle between standard definition (with a resolution of 720x576 pixels) and high definition (with a resolution of 1920x1080) is already won in favor of high definition. Nevertheless you might want to support older television sets.

[4] Partly by using the systems, partly by reading about it.

1. It's all about software

The fact that interactive television is all about software becomes clear in the very first moment of use where the majority of systems display some kind of interactive main menu. In the personal computing world, you typically use a mouse to select an item from the menu. With interactive television, you don't have a mouse, therefore you typically navigate the list with cursor buttons. The following figures show the main menus of *Apple TV* and *Boxee*. They follow a similar strategy since they both split the screen in area for features and an area for content recommendations. Just like the front page of a website, this screen is the central starting point and should be designed with great care.

Figure 4.3: Apple TV main menu[1]

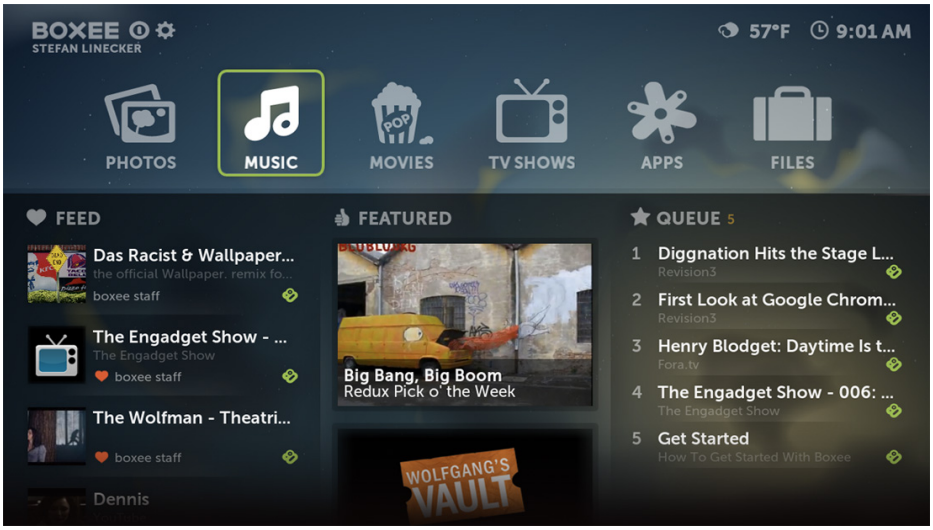


Figure 4.4: Boxee main menu[2]

[1] Courtesy Apple Inc., Source: http://ww.apple.com/pr/products/apple-tv/apple-tv.html (Retrieved 2012/07/05)

[2] Screenshot

2. It's interactive

Throughout this work you might have noticed by skepticism about interactivity. Nevertheless, the sheer amount of content and features makes interactivity a reality. *Figure 4.5* shows an interface with a listing of search results, as promised by the soon to be released Google TV. Search is no cure to information overload but it can clearly decrease the amount of interactions needed when looking for specific content. Another category where interactivity might make sense is gaming (e.g. see *Figure 4.6).*

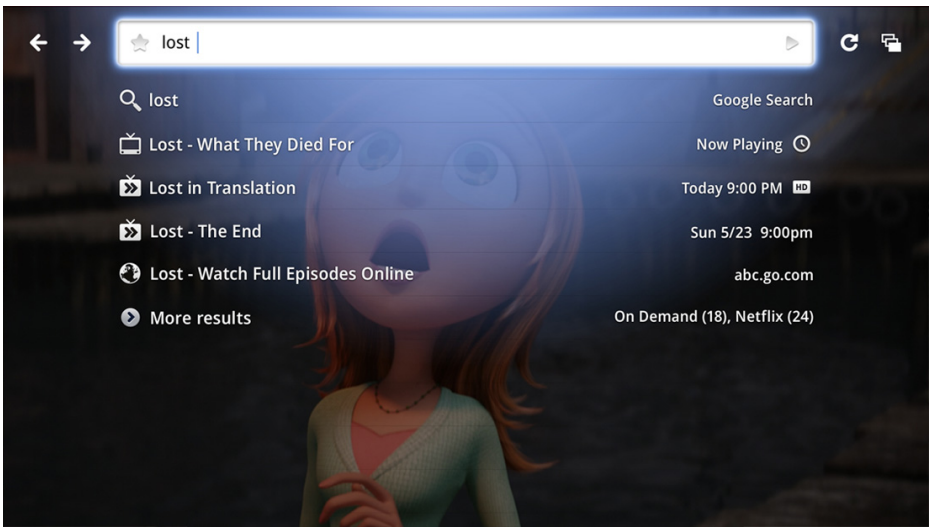


Figure 4.5: Google TV Search[1]

Figure 4.6: Accedo Poker[2]

[1] Courtesy Google Inc., use complies with http://www.google.com/permissions/

[2] Courtesy Accedo Broadband Inc., used with permission.

3. Time is a primary navigational dimension

With interactive television, choosing the time is just as ordinary as choosing the channel. It's best practice nowadays to use satellite stream metadata (e.g. ETSI EN 300 468) to generate interactive program guides like those shown in *Figure 4.7* and *Figure 4.8*. Due to its layout, these program guides are also called *EPG grid* or *EPG matrix*. It's convenient for the user if such a grid incorporates visual indication of the currently running program and the current time (as in *Figure 4.7*). This simplifies perceptibility and ensures intuitive navigation.

Figure 4.7: Finecom EPG grid[1]

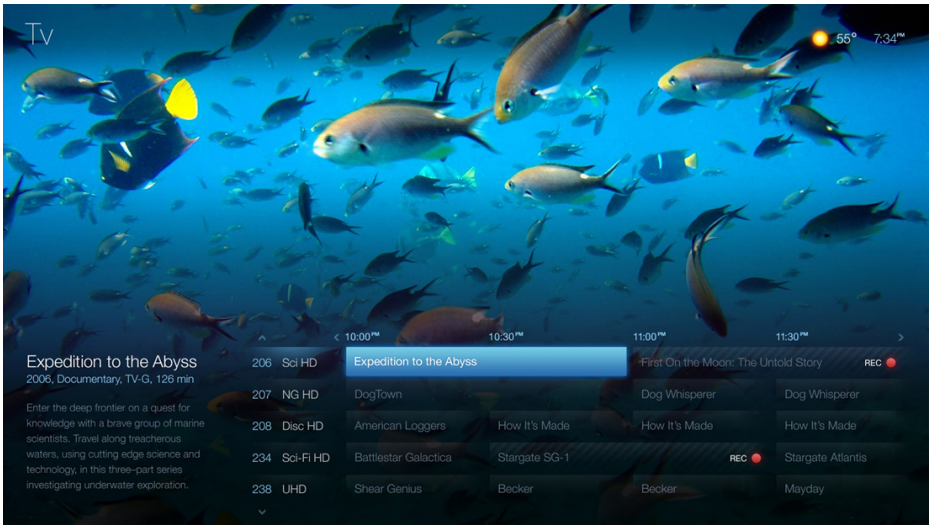


Figure 4.8: XBMC EPG grid mockup[2]

[1] Courtesy Finecom Telecommunications AG, Biel. Used with permission.

[2] Courtesy phyletik.com. Used with permission.

4. It's connected

In the beginnings of interactive television, Set-Top-Box manufacturers often bundled ordinary browsers with their software stacks. Due to limited processing speed and the missing input devices, user experience was lousy. Currently it looks like the users don't want to view ordinary websites on the television but specialized ones. One can then divide the internet experience in three different versions: the 1" version on the mobile phone (with mobile web pages and apps), the ordinary 3" version for the personal computer (with ordinary web pages and full-fledged software) and the 10" version for the television. They share a common backend but the presentation is optimized for the respective platform.

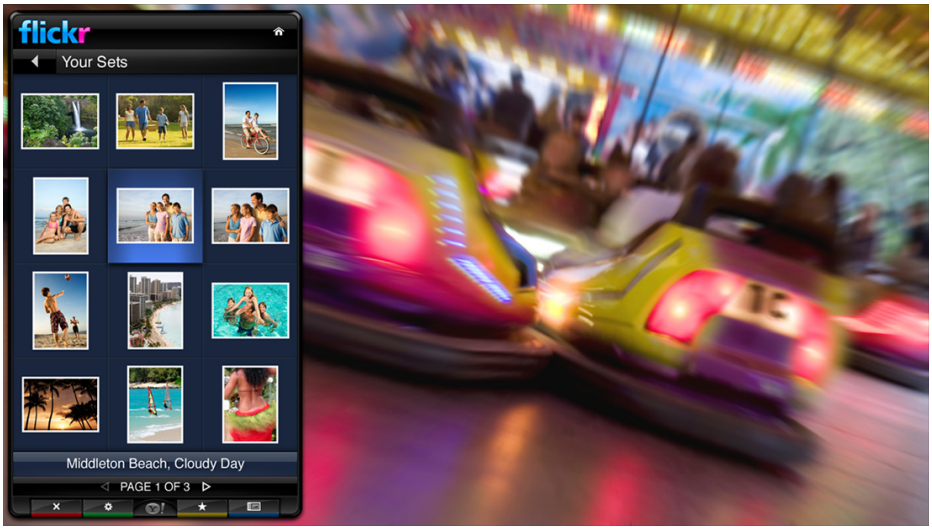


Figure 4.9: Accedo flickr widget[1]

Figure 4.10: Accedo facebook widget[2]

[1] Courtesy Accedo Broadband Inc., used with permission.

[2] Courtesy Accedo Broadband Inc., used with permission.

5. Radical increase of features

Applications for mobile phones nowadays are typically distributed via specialized online marketplaces like the *Apple AppStore* or *Android Market* where third parties can deploy their apps for the respective platforms. Future will show if this concept also works with interactive television. *Figure 4.11* shows such a store from *vudu*. *Figure 4.12* shows the MP3 playback features of *Plex* and should illustrate the quality and user experience of interactive television features. If realized that well, it clearly has the potential to replace individual devices.

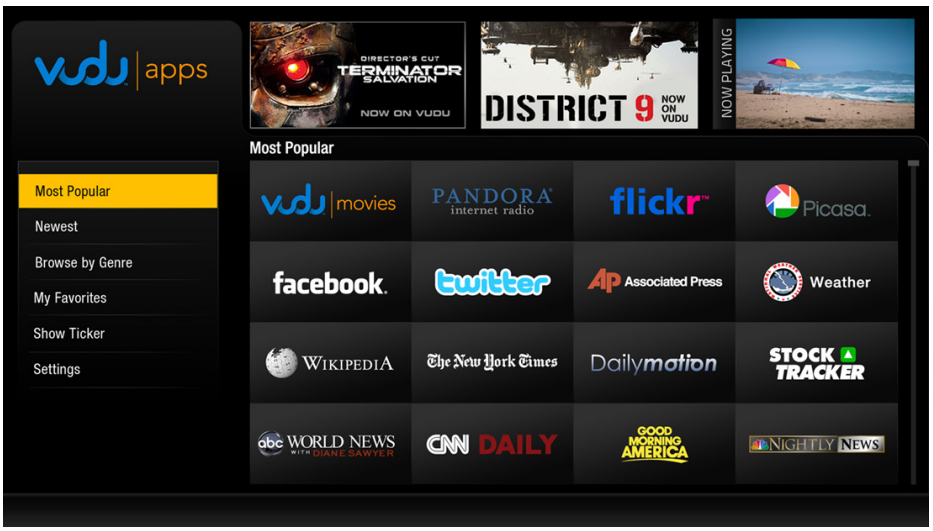


Figure 4.12: Vudu Apps[1]

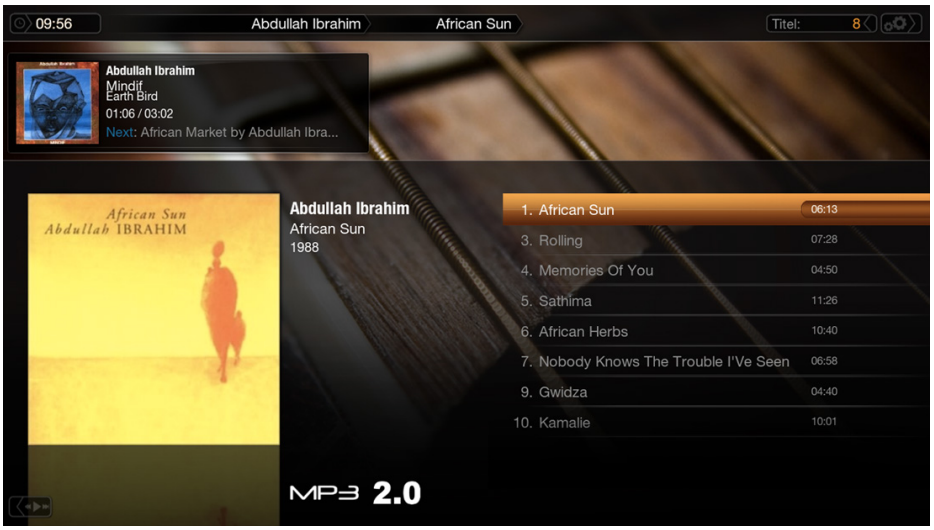


Figure 4.12: Plex MP3[2]

[1] Courtesy Vudu Inc., used with permission.

[2] Screenshot

6. Radical increase of content

As mentioned earlier, Amazon Video-On-Demand currently lists 50.000 videos. Such amounts clearly require smart search features and very good recommendations if you want your customers to find the stuff they are interested in. *Figure 4.13* shows an Amazon Video-On-Demand Screen for an individual movie. The interface incorporates context-sensitive recommendations, just like the original Amazon website. *Figure 4.14* finally shows a movie listing solely represented by cover art. This layout reminds on the shelf in a video rental store.

Figure 4.13: Amazon Video-On-Demand[1]

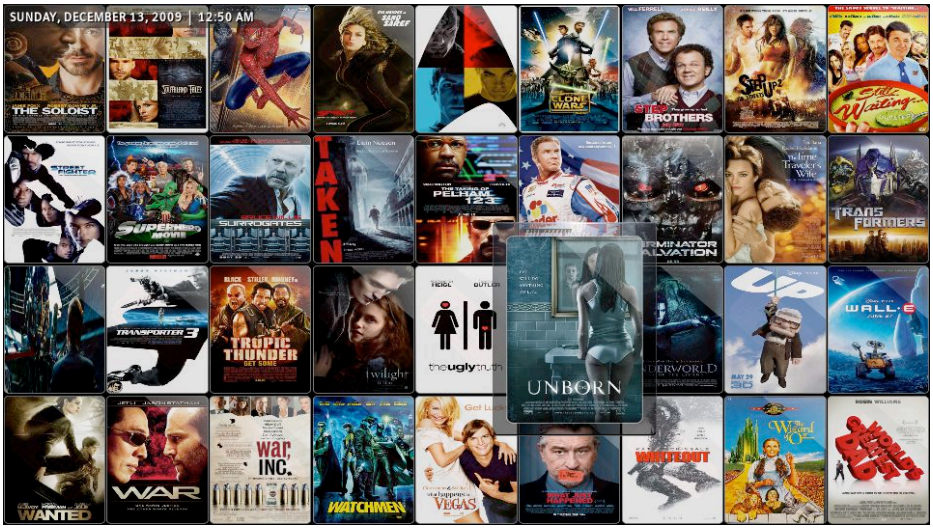


Figure 4.14: XBMC Aero-Kraft Skin[2]

[1] Screenshot

[2] Screenshot

4.3 Study Setup

The following pictures show some exemplary screens of the user interface at the time the log-data was gathered. As a matter of fact, this interface was replaced (see *Figure 4.22* and *Figure 4.23*) soon afterwards, but the following observations should apply to both, new and old, versions and to IPTV interfaces in general.

Figure 4.15: Main Menu

Figure 4.16: Radio Menu

Figure 4.17: Program Table

Figure 4.18: Timeshift Table

Figure 4.19: Live-TV with Overlay

Figure 4.20: PVR Playback

Figure 4.15 shows the *main menu*, the central entry point for the most important features and services. It's the pendant to the index page on websites and should be designed and evaluated with great care. Within the context of interactive television, it makes sense to store the state of the last session when the TV (or set-top-box) is powered off. This means that users aren't automatically brought to the main menu each and every time they turn on their set but only when they navigate there.

Starting from the main menu, the user *browses* his/her way to the menus to (hopefully) end up at the channel or feature he was looking for. *Figure 4.21* shows the first level of the given menu tree, grouped by frequency of use.

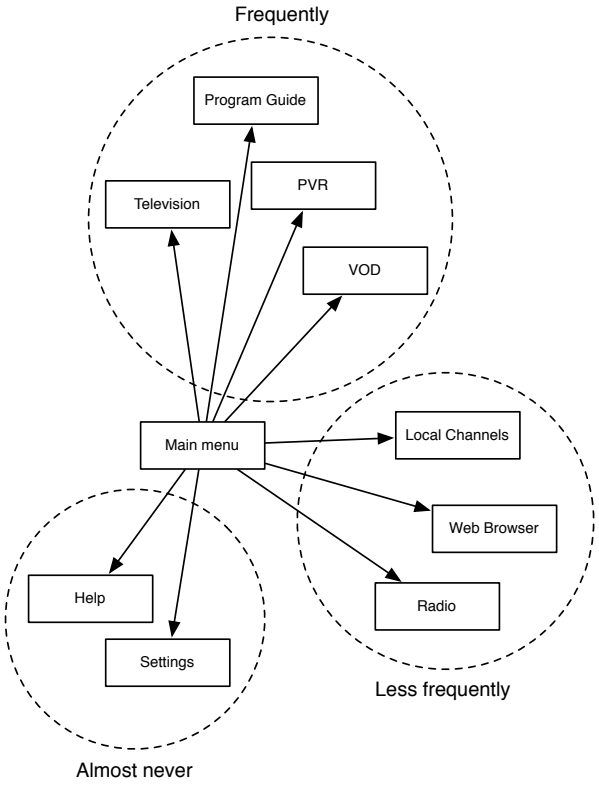


Figure 4.21: Menu tree, depth = 1

Figures 4.16 to *4.20* show some further screens of the user interface. While the selection of screens is not representative and only a snapshot from one given interactive television system at one given time, it should at least illustrate one crucial fact: a large portion of the navigation scenarios in interactive television comes down to content *selection* by browsing through *lists*.

As already mentioned, the screenshots above were already outdated at the time of writing. The user interface was replaced by an overhauled, graphically refined version. While the lifecycle of the whole system might only have started, the user interface typically evolves and gets updated on a regular basis, just like it is the case with websites. Such updates might arrive in small doses or big chunks. The following pictures show some

screens of the updated user interface. *Figure 4.22* shows the new version of the main menu, originally illustrated in *Figure 4.15*. Figure *4.23* shows the EPG matrix, the replacement for the program table (*Figure 4.17*).

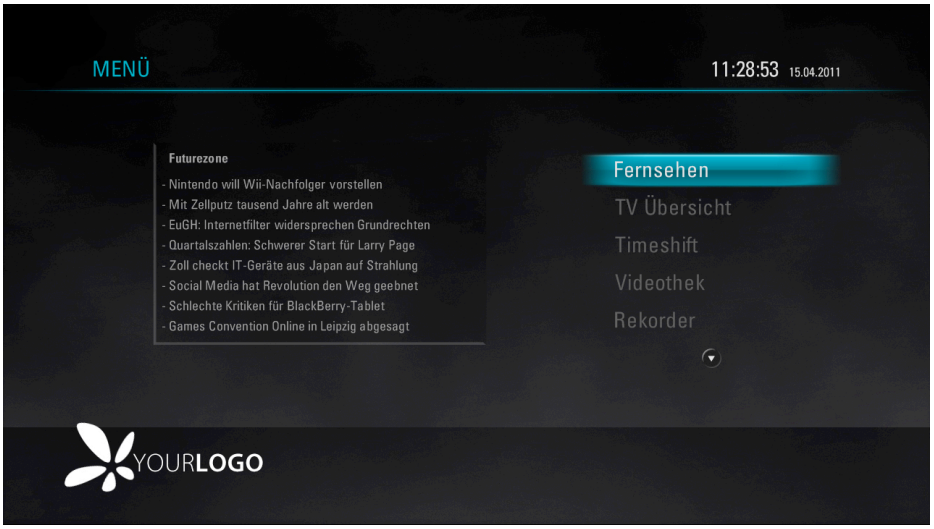


Figure 4.22: Updated main menu

Figure 4.23: Updated EPG matrix

Chapter 5 : Usability Engineering

5.1 Usability

Repeating again, the overall aim of this work is to contribute to customer satisfaction of interactive television by investigating the use of the remote control and the corresponding graphical user interfaces. Customer satisfaction is crucial to the commercial success of a product but it is quite difficult to measure. Additionally, the life cycle of interface-dominated software is rather short and typically consists of many rapid iterations. The classic methods of market research are just to cumbersome for this.

One thing typically highly correlated with customer satisfaction is user-friendliness or *usability*. Usability by itself is in no sense a guarantee for customer satisfaction but it might be seen as a necessary condition. Usability is part of system acceptability, which can be defined as *„the question of whether the system is good enough to satisfy all the needs and requirements of the users and other potential stakeholders“* [Nielsen 1993, p. 25]. Usability basically nails down to the questions how well one can use some given functionality. More formally said *„Usability is the extent to which a computer system enables users, in a given context of use, to achieve specified goals effectively and efficiently while promoting feelings of satisfaction“* [Ivory et al, 2001]. Thereby sufficient principal functionality (which is then called *utility*) is a prerequisite. Usability itself is a combination of learnability, memorability, efficiency of use, error frequency and subjective satisfaction.

There are multiple ways to achieve sufficient usability. So called *Usability Heuristics* provide the wisdom and experience of usability professionals in distilled form, resulting in easy memorizable principles like *„Minimize the users‘ memory load“* or *„Prevent Errors“* [both Nielsen 1993, p.20]. The application of those principles might require some sensitiveness but they provide great value for both the education and daily work of user interface designers. Another way to ensure proper usability is the adaption of the design process. You might try to give the same design task to multiple design teams (which is then called *parallel design*) and/or you might try to let future users participate in the design phase (which is called *participatory design*). Two more keywords commonly used in this context are *prototyping* (or *rapid prototyping*) and *iterative design*. Prototyping is all about creating intermediate versions of an interface in a very cheap and fast manner. The prototype must not support full functionality, nor must it be visually appealing. Its only purpose is to compare or evaluate the principal usability of a given design solution. Iterative design is closely tied to

prototyping. It means nothing more than the implementation of new features one at at time, always followed by a quick evaluation. The purpose of iterative design is the discovery of design problems at an early stage. The umbrella term for all these methods is *usability engineering* and while all mentioned techniques are widely used, the gold standard of usability engineering is *usability testing*.

5.2 Usability Testing

Usability testing [Nielsen 1993, Rubin et al. 2008] itself is a collective term used for the set of methods and techniques suitable for evaluating a product by testing it on genuine users. Usability testing is the opposite of usability inspection where experts *„evaluate a user interface without involving users*[1]*“*. As with any test, usability testing requires thought about representative subjects, adequate test tasks, the test setup, and, last but not least, performance measurements.

Usability testing includes such different methods as *thinking aloud*, *hallway testing* or *eye-tracking*. With thinking aloud, a test subject uses the system and simultaneously talks about his or her thoughts. With hallway testing, you just give the experiment to a random person. Eye-tracking measures the movement of the eye and can determine where your focus is. These are just a few of the methods but all of them fall under the category of user-based testing where *„the essential idea is to base assessment of the usability of a system on observation of users working with the system“* [Bruun et al. 2009]. The usability method used for this word also falls under this category.

Usability testing clearly has its advocates and critics. Nicholas Negroponte, MIT professor and father of the *One Laptop per Child* project once wrote *„I have little respect for testing and evaluation in interface research. My argument, perhaps arrogant, is that if you have to test something carefully to see the difference it makes, then it is not making enough of difference in the first place.“* [Negroponte 1995]. On the other side, Scott Cook, co-founder of Intuit, the inventors of Quicken says *„... we did usability testing in 1984, five years before anyone else... there's a very big difference between doing it and having marketing people doing it as part of their... design... a very big difference between doing it and having it be the core of what engineers focus on.“* Indeed, the effectiveness and validity of usability testing depends heavily on the context. If done *„by rule“* rather than *„by thought“*, usability evaluation might even get harmful [Greenberg 2008]. It‘s also the case that *„Usability*

[1] http://en.wikipedia.org/wiki/Usability_testing (Retrieved 2010/10/04)

findings can vary widely when different evaluators study the same user interface, even if they use the same evaluation technique" [Ivory et al. 2001] and it is often recommended to use several evaluation techniques at once [Dix et al. 1998; Nielsen 1994].

It is important to understand that usability testing is just a method. It can't replace an experienced designer but the same is true vice versa.

5.3 Remote Logging

Throughout the practical experiments in *Chapters 6* and *Chapter 7*, I will use a usability testing method called *remote logging*. This is nothing else than storing the actions of the users (in this case button presses on the remote control) in centralized log files [Andrews 1998, Valdman 2001]. With webpages, it's common nowadays to roll out a new feature to a limited set of users and test whether the feature is accepted. This is done through all kinds of analytics software that basically all fall under the category of remote logging.

Remote logging is a so-called *asynchronous* usability evaluation method since users and evaluators are both spatially and temporally separated. Other remote asynchronous usability testing methods include the *User-reported Critical Incident (UCI)* technique [Bruun et al. 2009] and *unstructured problem reporting*. Both require direct effort from the user, remote logging does not.

Interactive television is an online system and therefore shares one of the biggest advantages of websites: you actually know what your users are doing[1]. A few simple counters in your software can give you much more details then you would get with thousands of dollars worth of market research. With websites, companies basically live from the time we spend on their sites and therefore put a lot of resources into making our stay nice. They know very exactly (at least the successful ones) what is going on and how people use their sites. Paul Graham states „*Software should do what users think it will. But you can't have any idea what users will be thinking, believe me, until you watch them. And server-based software gives you unprecedented information about their behavior. You're not limited to small, artificial focus groups. You can see every click made by every user*" [Graham 2004].

[1] The difficult questions about privacy and data security that arise from this fact aren't discussed in this work.

As you will see, another important technique for the following chapters is information visualization or *„the graphical representation of data“* [e.g. Tufte et al. 1983, Ware 2004]. Information visualization techniques will help us to convert large amounts of equal-looking data into human-readable images.

5.4 Study Setup

Throughout the following Chapters, I use data from 18 voluntary households. For privacy reasons, I have no information whatever concerning those households. I asked for a broad set of different user types and was told that the sample should be somewhat representative. No further information was passed along.

All participants had to *opt-in* for the experiment and were informed what will happen to *their* data. This could potentially have had some influences on user behavior.

In a time period of 38 days, all key presses of those 18 households were recorded, resulting in 90937 logged key presses. This log data was then exported into a plain text file[1], looking similar to the following listing.

```
  ID   | Date       | Time     | Keycode
=======================================
  1242 | 2010-08-30 | 12:51:28 |    4097
  1243 | 2010-08-30 | 12:51:29 |    4097
  1244 | 2010-08-30 | 12:51:33 |    4184
  1245 | 2010-08-30 | 12:51:36 |    4176
  1246 | 2010-08-30 | 12:51:37 |    4176
  1247 | 2010-08-30 | 13:39:41 |    4176
  1248 | 2010-08-30 | 13:39:42 |      13
  1249 | 2010-08-30 | 13:40:25 |    4184
  1250 | 2010-08-30 | 13:40:28 |    4312
```

Listing 5.1: Simplified log file

[1] The anonymized plain text file marks the border from the used IPTV system to my analysis software. In fact, I did *not* write the part of the software which extracts the button presses from the overall signalization messages of the system and generates the plain text files with the logged key entries. This was *by design*, since splitting those two development activities between two people seemed a good idea concerning the given privacy requirements.

The first field, *ID*, was used to differentiate between the individual set-top boxes (which correspond to the individual households in this case). The second and third field, *Date*, respectively *Time*, were used to capture the timestamp of the event. The last field, *Keycode*, was used to store an identifier for the pressed key. In the following chapters, we will omit the numeric representation for the key code and replace it with its name (like *Fast Rewind*) or a symbolic representation (like <<). The 18 households actually used their remote controls rather differently and the total number of button presses, cumulated per set-top box, ranges from less then 100 to more then 10000 (illustrated in the following figure). That's a 100-fold difference.

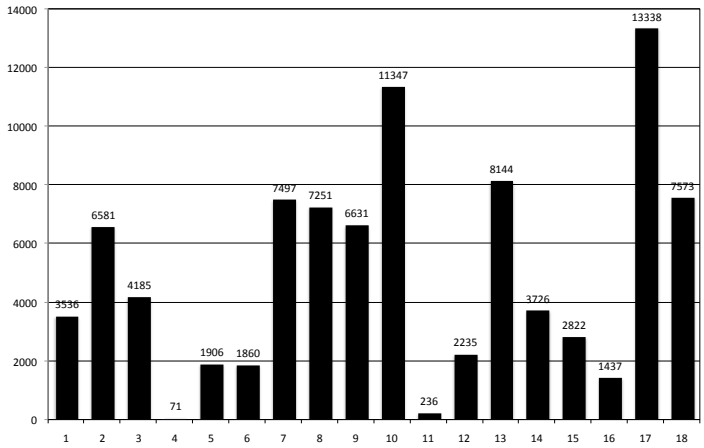


Figure 5.1: Presses per STB

Another interesting thing is the distribution of button presses over the hours of the day, as illustrated in the following figure.

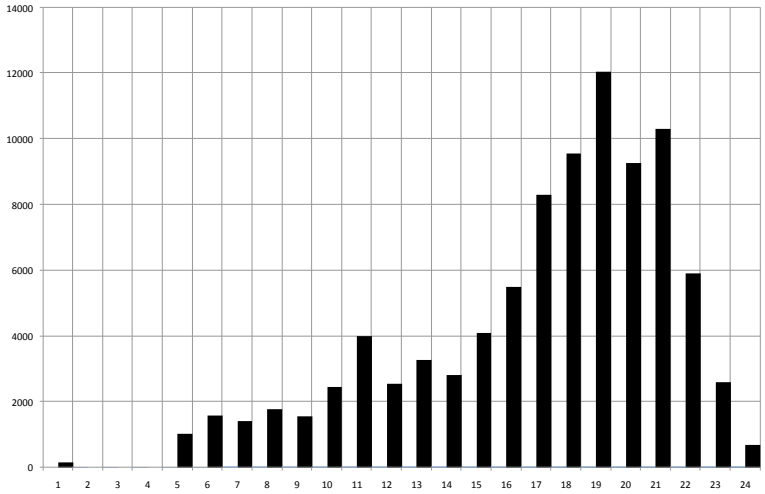


Figure 5.2: Presses throughout daytime

As one would suggest, the number of button presses tends to be very low between 1:00 am and 5:00 am. It then climbs up a bit with a local maximum at noon. The majority of button presses is happening during *prime time*, which is usually referred to as the time period from 7:00 pm to 11:00 pm. All in all, button presses throughout daytime tend to follow the overall usage pattern of television rather closely. For the sake of completeness, I also had a look at the distribution of the button presses over the individual days of the experiment (see *Figure 5.3*).

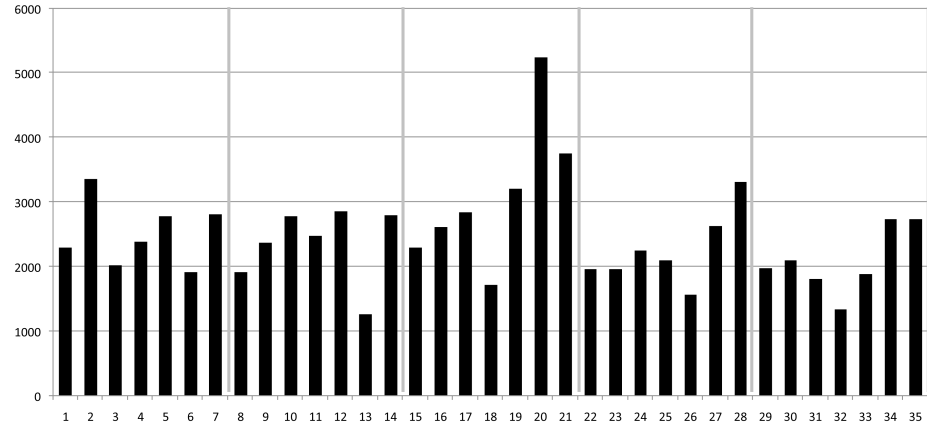


Figure 5.3: Presses throughout the test period[1]

The chart shows one peak, which actually was a Saturday, but it does not suggest a *major* influence of the week day on the number of button presses. If we cumulate the particular week day (see *Figure 5.4*), the picture gets even clearer. No excessive influence is visible.

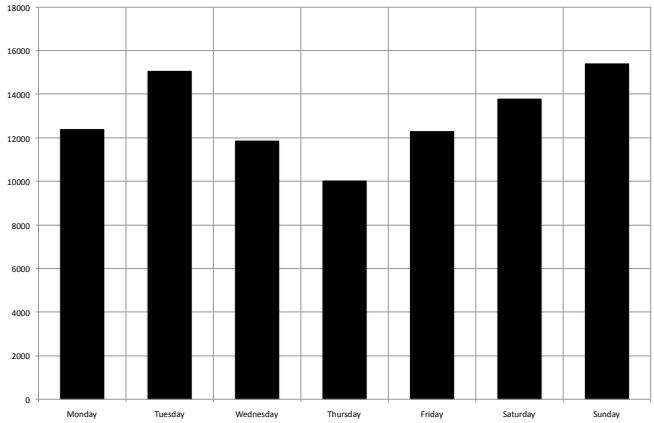


Figure 5.4: Presses throughout weekdays

[1] Only 35 of 38 days shown.

Chapter 6 : Physical Layout

6.1 Button count

The most obvious question concerning the design of a remote control clearly is all about its physical buttons. Which buttons should it have and how should those buttons be arranged? At first, this might seem like a rather far-fetched, trivial question since most of our remote controls tend to look the same. I was involved in the design of remote controls several times and my personal opinion is that it is neither far-fetched nor trivial.

It‘s not *far-fetched* since designing a remote control really comes down to thinking about individual buttons - it‘s all about details[1]. As stated in Chapters 1 and 2, the increase in complexity and competition clearly justifies the investigation into details.

It‘s not *trivial* since those questions about buttons aren‘t really about buttons but about the structure, architecture and quality of a whole product or service. The buttons on the remote control enable the user to interact with the system. Thinking about those simple buttons eventually requires a comprehensive, holistic view of the overall service [Cook et al. 2002; Mager 2009].

The following figure illustrates two extremes for the criterium *button count*.

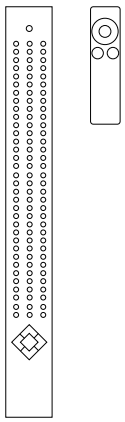


Figure 6.1: Art. Lebedev Studio Pultius[2] and Apple Aluminum Remote

[1] Dieter Rams‘ ten principles to good design include the say „Good design is thorough down to the last detail“ and a famous Charles Eames Quote reads „The details are not the details. They make the design.“

[2] As far as I know, the Pultius only exists as a prototype.

A natural way to approach the importance of individual buttons is to count button presses. I did exactly this using custom Java code, analyzed the log file and generated the following piechart out of it.

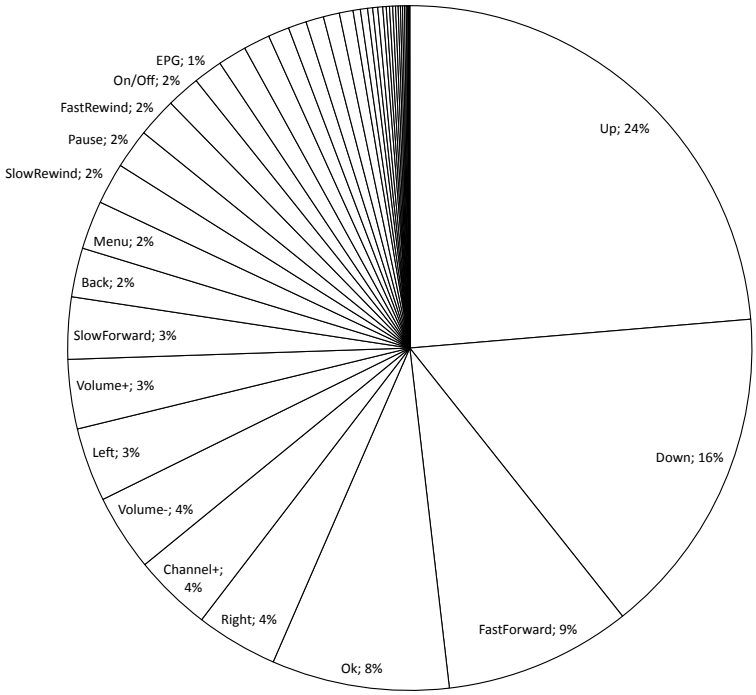


Figure 6.2: Piechart of button presses

It‘s remarkable that the three buttons *Up*, *Down* and *Fast Forward* amount to nearly half of all presses. It‘s also remarkable that none of the number keys are included in the diagram (for layout reasons, I did not label buttons which amount to less then one percent of the overall button count, that‘s where the individual number keys reside).

The importance of the *Up* and *Down* keys can be easily explained since those two keys are used when navigating through all kinds of menus and for zapping through the channels. The importance of the *FastForward* key can also be easily explained: Most people use Time-Shift, and *Fast Forward* thereby is used to fast forward through commercial breaks.

It‘s also interesting to classify the individual keys in functional groups (see Appendix B for details on the functional group classification), resulting in the piechart in *Figure 6.3*.

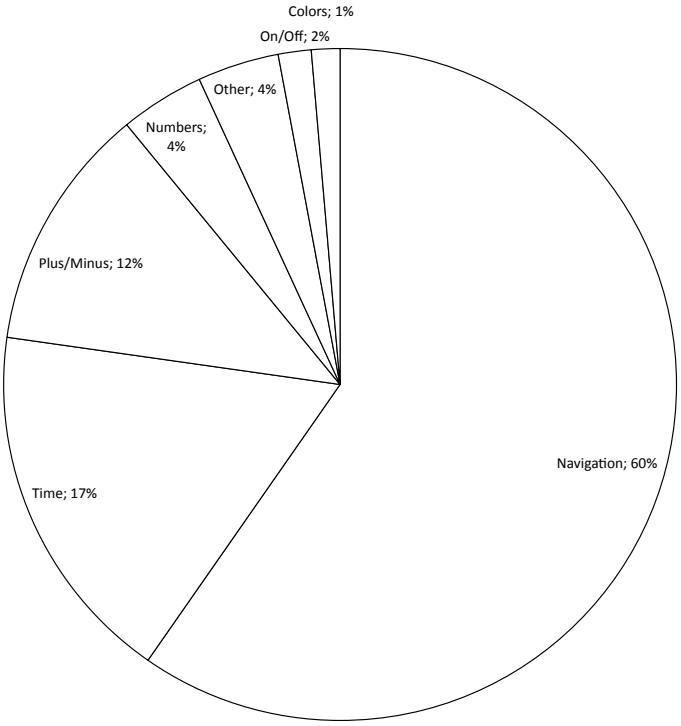


Figure 6.3: Piechart of presses in groups

Here, one can see that the number keys aren't actually that unimportant - they make about 4% of the overall presses. Interestingly, the three top groups, navigation, time and plus/minus amount to 89% of all presses.

Even these simplest counters can be helpful when designing the remote control. Using this data, we could justify the typically central alignment of the navigation keys (those keys are so essential that they have to be in the center). We might also argue that the up/down keys should be more robust (concerning their mechanical properties), since they are pressed ten times more often then some other keys.

For me, those pie charts seemed to be a good starting point. The problem is that they lack an important dimension from the physical remote control, namely the *placement* of the individual buttons.

I tried to overcome this limitation by using a more natural visualization method, illustrated in *Figure 6.4*.

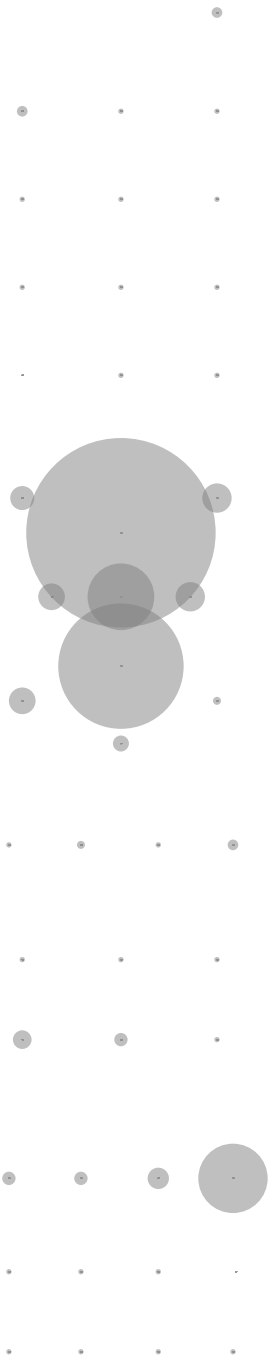


Figure 6.4: Aggregated button presses

Figure 6.4 was constructed by mapping the cumulated presses of the individual buttons of the whole data set to its corresponding x- and y-coordinates. Each button is represented by a filled circle, the size[1] of the circle increasing with button count. As more often a button is pressed, the larger the circle will be. Size then means importance.

Figure 6.5: Overlaid aggregated button presses

[1] I have to confess that the visualization contains a glitch: the button count is used to determine the diameter of the circle. This can be me misleading since, we perceive the area of the circle. One could say that this visualization exaggerates the importance of individual buttons. Since that is exactly what I want to achieve with this image, I remain with this hint.

Figure 6.5 shows the wireframe of the used remote control overlaid with the former visualization. Both images, the puristic one and the overlaid one, provide *specific* value: while the puristic one is perfect for comparison, the overlaid one is better for detailed analysis. The overlaid image also is a better starting point for explaining the construction of the visualization.

In fact, those images are just another view on the data presented in the pie-chart in *Figure 6.2* and while the method is simple, it‘s quite effective for visually justifying the following observations.

Id	*Observation*
1	Navigation keys are used excessively
2	Keys for V+/V-, P+/P- are used excessively
3	Time manipulation keys are used excessively
4	From time manipulation keys, FastForward is used most often
5	Number buttons aren‘t used much
6	Color buttons aren‘t used much
7	All other keys aren‘t used much

Table 6.1: Observations concerning button count[1]

The goal of such observations (at least in the present case) is to find out whether there are usability problems within an interface. It‘s quite interesting to let those images be generated in a first step, make some changes to the interface in a second step and re-generate the images again afterwards for comparison.

While the above images were generated from the entire data set, it might even make more sense (assuming a sufficiently large data set) to restrict or filter the data for individual *screens*, *use cases* or even *users*.

Filtering by user is especially helpful if you want to have a look at the differences of usage. These insights might be used to investigate the adaption of new features. *Figure 6.6* shows such an *individual user* visualization of the 18 households. Many of the observations from *Table 6.1*, which were extracted from the overall visualization, also comply with the majority of individual images. Speaking statistically, we are actually checking if our observations were distorted by individual outliers.

[1] The observations are somewhat similar to [Bernhaupt et al. 2007, p.8]

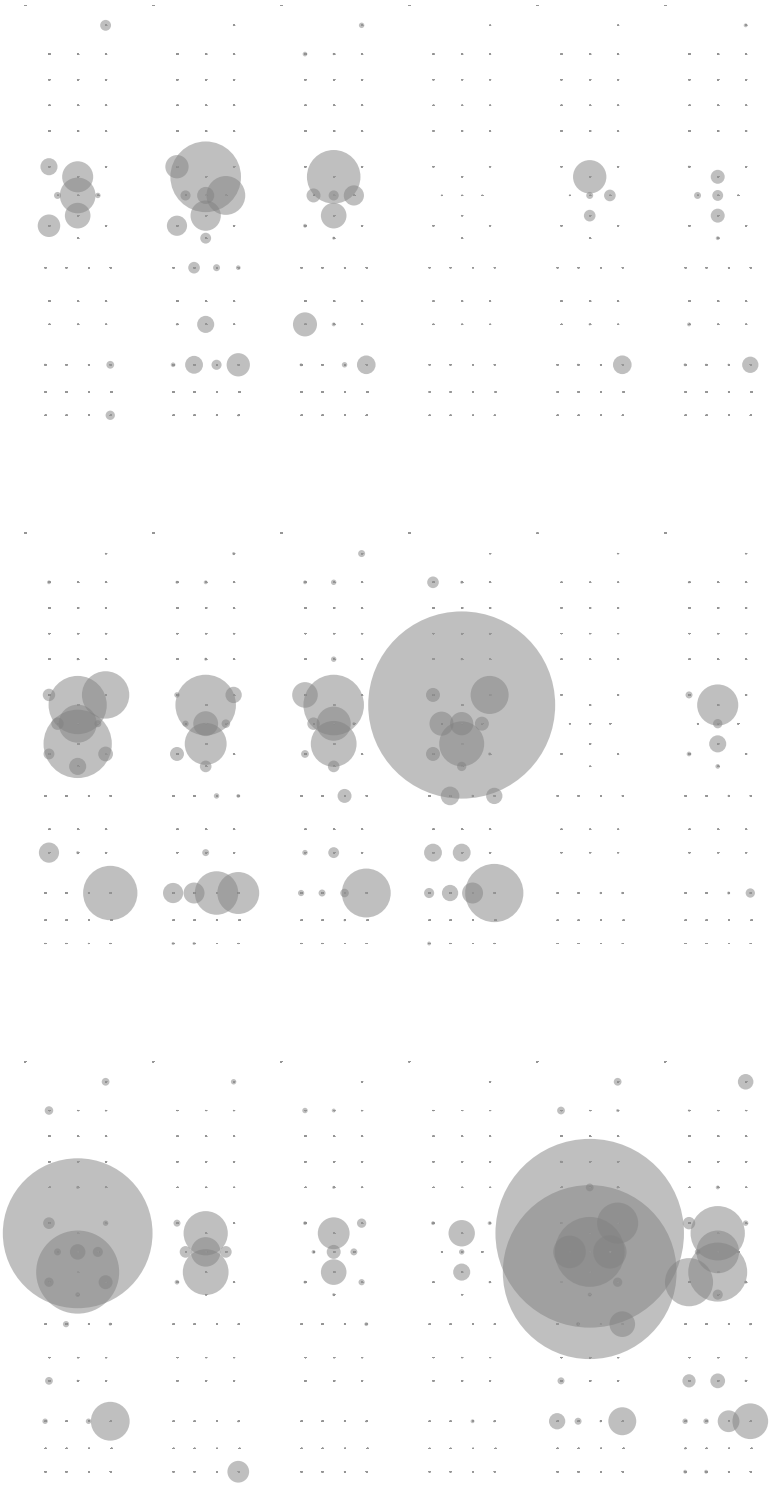


Figure 6.6: Presses per remote

If we have a closer look at the individual images, we can determine if there are common patterns that correspond to the assessments in *Table 6.1*. I did so by simply adding a column to the table where I noted the occurrence of the individual observations.

Id	*Observation*	*Occurrence*
1	Navigation keys are used excessively	all
2	Keys for V+/V-, P+/P- are used excessively	all
3	Time manipulation keys are used excessively	2,8,9,10,17, ...
4	From time keys, FastForward is used most often	2,3,5,6, ...
5	Number buttons aren‘t used much	all
6	Color buttons aren‘t used much	all
7	All other keys aren‘t used much	all

Table 6.2: Validation of observations concerning button count

In addition to underpinning the original observations, we could also use the individual visualizations to make new observations. The ones I found are listed in the *Table 6.3*.

Id	*Observation*	*Occurrence*
8	Some use Pause heavily	7,10,18
9	Some use Timeshift button heavily	3,7,10,18

Table 6.3: Additional observations concerning button count

6.2 Button placement

Equally important to the question which buttons should reside on the physical body of the remote control is the question how those buttons should be laid out.

Apparently, this question can be answered kind of *intuitively*: those buttons that are used together should be located at a similar location. Moreover, the important buttons should be located where the fingers reside, the buttons should reflect the design of on-screen elements and all in all, the button layout should follow a logical, intuitive manner. Despite the subjective manner of the above list, it might provide a good starting point.

All the key presses from the log data can be interpreted as a sequence of key names. If the user first presses the *On* button, followed by the *Down* button three times in a row and afterwards presses the *Ok* button, this sequence would look like the following:

$$(On, Down, Down, Down, Ok)$$

For each pair of adjacent members, one can then count their occurrences, resulting in a set of 3-tuples, each holding the origin key, the target key and the number of occurrences. For the above example this would yield to

$$\{ <On, Down, 1>, <Down, Down, 2>, <Down, Ok, 1> \}$$

The number of occurrences will be high at those combinations that are used regularly and low at those combinations which are used sparely.

This simple technique has an obvious failure: time is not taken into account and events that occurred immediately one after another are weighted the same as events that occurred hours away from each other.

Again, I tried to come up with a method to visualize those sequences in the most natural way. The simplest visualization I found was to draw lines between the button pairs where the thickness of the lines corresponds to the count of their occurrence (see *Figure 6.7*). I called this method *Motion Lines*.

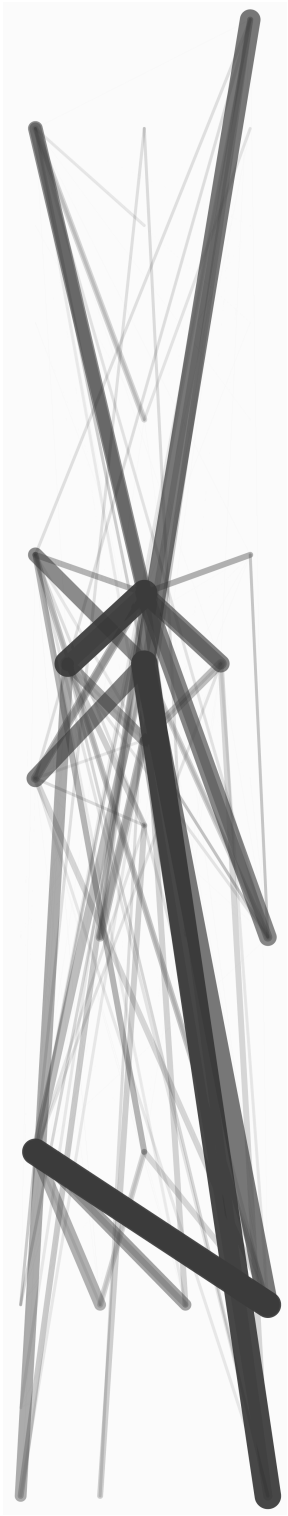


Figure 6.7: Motion Lines of the entire data set[1]

[1] The visual appeal for this was inspired by Burak Arikan's Openstudio Relationships visualisation, available at http://burak-arikan.com/os-relationships. (Retreived 2011/03/09)

As with *Figure 6.4*, this is aesthetically interesting, but we might miss the button overlay. This is where *Figure 6.8* comes into play.

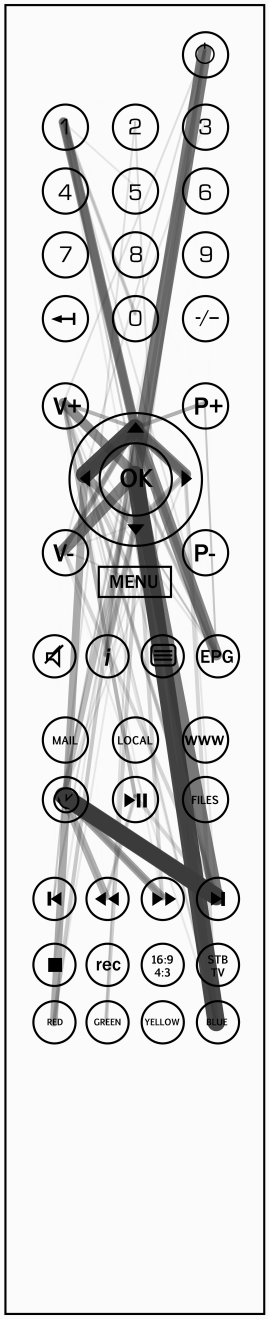


Figure 6.8: Overlaid motion lines of the entire data set

If the user has to go a *way* very often, the line between the two buttons is thick, if the user has to go a way infrequently, the line between the two buttons is thin.

As with the *button count visualization* from Section 6.1, the *button placement visualization* has the goal to simplify observations on the data. The visual interpretation of the given image is simple: *short, thick* lines denote good localization, while *long, thick* lines denote bad localization (buttons that are frequently used in conjunction but placed far of each other). We want to eliminate all lines that are *long and thick*.

Using *Figure 6.8*, the following observations were made:

Id	*Observation*
10	Strong connection between *Ok* and *Blue*
11	Strong connection between *Timeshift* and *Fast Forward*
12	Good overall localization (many thick lines around the central navigation keys are relatively short)
13	*One* is a bit offside
14	*Power on/off* is a bit offside

Table 6.4: Observations concerning button placement

Figure 6.8 was generated from the entire data set. Just like within the previous section, it makes sense to determine the influence of the individual set-top boxes on this image.

Figure 6.9 therefore shows the discussed visualization method applied on each of the eighteen different set-top boxes on its own. As you can see in the 14th sub-image (third row, second column), this instance has a conspicuously bold line from *Ok* to *Blue*, which is also visible in the overall image and documented as *Observation 10*. This might distort the aggregated result but since a relatively strong line between *Ok* and *Blue* can also be found in the 1st sub-image (first row, first column), *Observation 10* might still be considered valid.

Observation 11 is supported by sub-images 3, 6, 7, 9, 10, 13, 15 and 16, which makes this case much clearer. Nothing interferes with observations *12, 13* and *14*.

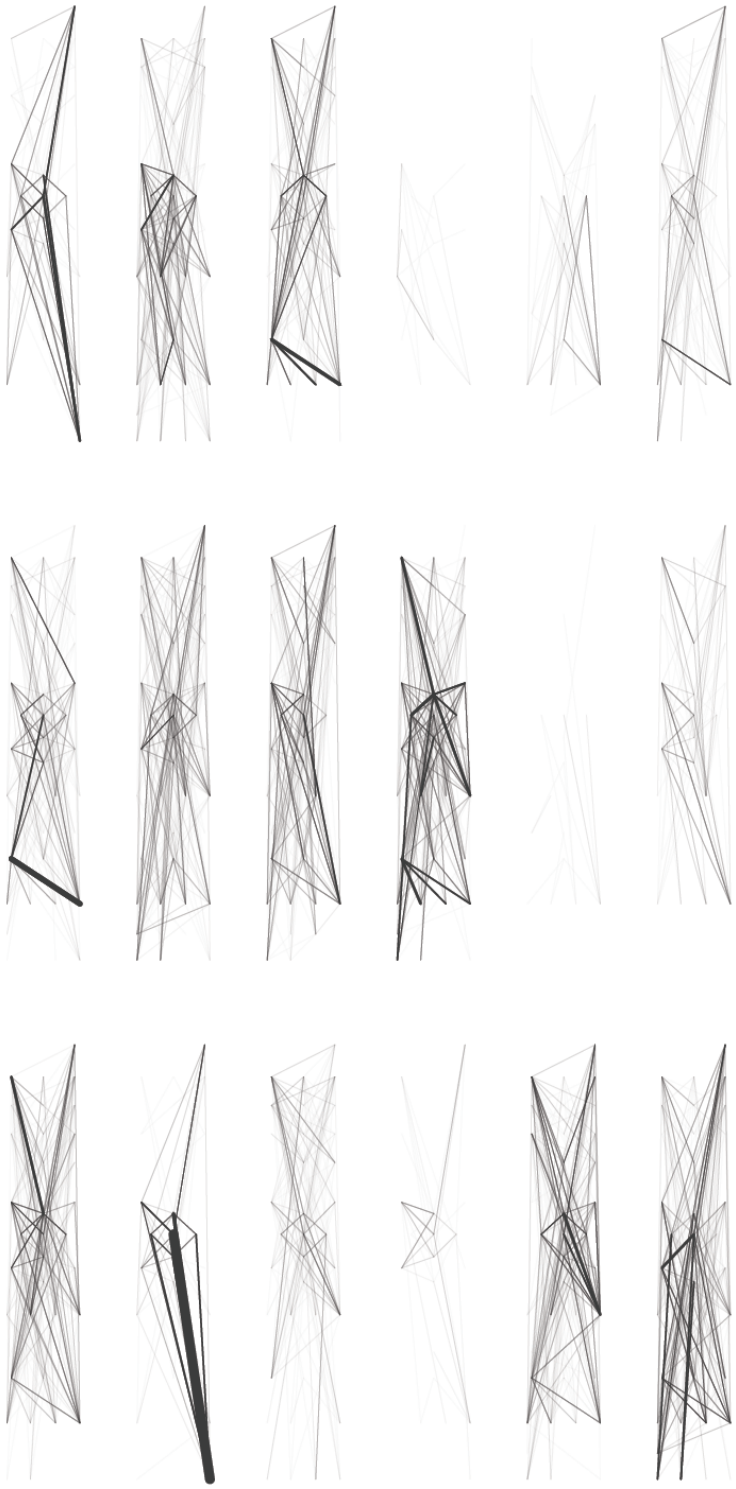


Figure 6.9: Motion lines per set-top box

Chapter 7 : Virtual Layout

7.1 Sessions, Interactions and Events

Till now, this work only considered the static aspects of the remote control, namely button count and button placement. After production and shipment, those properties can‘t be changed and usually remain the same for the whole lifetime of the product. As the remote control comes coupled with a set-top box or television set, those devices will determine this timespan. The typical lifetime of a television set is few years to a decade, the typical lifetime of a set-top box is one to five years.

In contrast to traditional broadcast television, interactive television is highly dynamic, concerning both content representation and the overall user interface. The designer might rearrange different elements of the user interface, change some colors, assign some different shortcuts, etcetera. All those things are realized through software and can be updated on-the-fly. This was not possible with traditional television and those dynamic properties bring modern interactive television closer to the technical attributes of web pages, rather then ordinary television itself.

While the hardware remains the same, the software might get tweaked throughout the whole lifetime of the product. On the one hand, this gives the provider the ability to continuously improve and fine-tune its offerings but on the other hand this is also a burden. The provider has to balance the injection of new features and user interface updates against stability and continuity. Usability evaluation techniques can help to tackle this balancing act.

This chapter will focus on the dynamic interplay between the remote control and the graphical user interface (and thereby leave out many other dynamic aspects). When trying to inquire those dynamic aspects, life gets pretty complicated. Similar to the last chapters, I will *not* try to develop metrics based on rigorous mathematical formalisms but I will rather try to present a view on the data that might enable the investigator to gain insights into user behavior of the system.

Before going into detail, I would like to define the nomenclature used. The given categorization isn‘t included for self purpose but for describing the visual elements used in a few pages and throughout this chapter. In fact, the visual representation was not built upon those concepts but, just the other way around, the categorization was derived from the visualization.

Interactions during the television experience can be categorized into the following three types:

- A *session* is a continuous time window where the customer uses the system. It starts when the television set (respectively the set-top box) is switched on and ends when the television set is switched off. A session isn‘t represented explicitly in the log file but it can easily be derived by scanning each and every line for the corresponding key codes. Sessions can last from several minutes to hours.

- An *interaction* is a time frame where the user interacts actively with the system (and presses buttons which generate log entries). It‘s rather hard to determine the exact borders between interactions if you define interactions by means of user goals. I used a much simpler definition, namely that if no button is pressed for 30 seconds, a new interaction will start when the next button is pressed. This heuristic worked well in my personal experience and throughout the course of this work I found no reason to replace it with a more sophisticated approach. Interactions typically last several seconds to few minutes.

- An *event* is a single button press on the remote control. In contrast to sessions and interactions, events have no duration. They are discrete, punctual events on the timeline. While sessions and interactions are concepts, each and every event is represented by a single line of text in the log file.

A sessions consists of at least one interaction, and an interaction itself consists of at least one event. An event belongs to exactly one interaction and an interaction itself belongs to exactly one session. *Appendix D* provides examples of visualizations (with a method that will be explained shortly) for a variety of typical-length interactions.

The analyzed log data consisted of 790 sessions built upon 10001 interactions built upon 90937 events. The mean total number of presses per set-top box was 5052, the mean daily number of presses per set-top box was 280.

Some further information on the logged data can be found in Section 5.4, *Study Setup* and *Appendix A: Log-File-Format.*

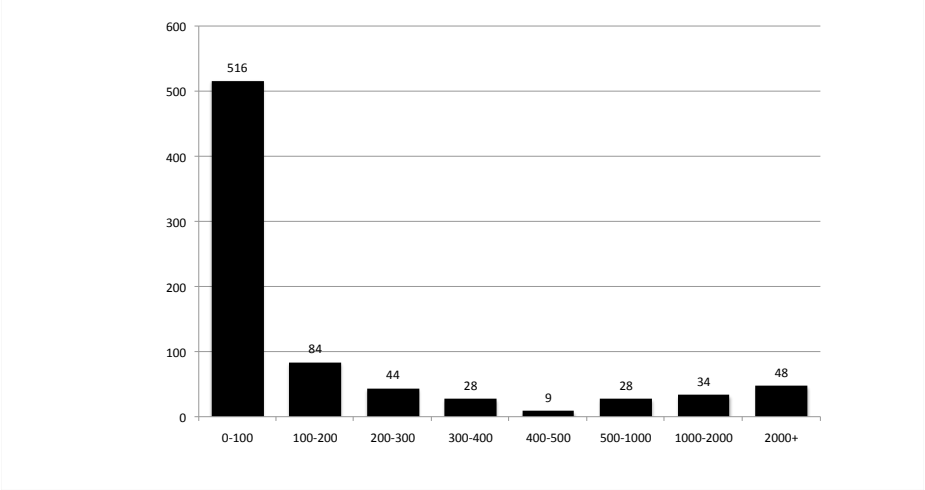


Figure 7.1: Sessions length

Figure 7.1 shows several classes of time intervals (0–100 minutes, 100–200 minutes, etc.) on its x-axis and the corresponding number of interactions that fall in this class on its y-axis. Please note that the intervals get larger towards the right hand-side of the diagram and that the last class is a right-open interval.

Most of the sessions (65%) took between 0 and 100 minutes. 75% of the sessions took less then 200 minutes.

The fact that there are relatively many sessions with a duration beyond 2000 minutes (which are more than thirty hours) suggests that some users do not turn off their set-top boxes. This was possible in the study setting, where set-top box and television set are two distinct devices that are only loosely coupled. Initiatives like HDMI-CEC[1] try to overcome this energy-wasting limitation.

[1] CEC stands for Consumer-Electronic-Control and can be used to interconnect and control electronic equipment like television sets, hi-fi system and set-top boxes. In the past, there have been numerous interoperability issues for devices of different vendors and at the time writing it has yet to be proven if HDMI-CEC can fulfill its promises.

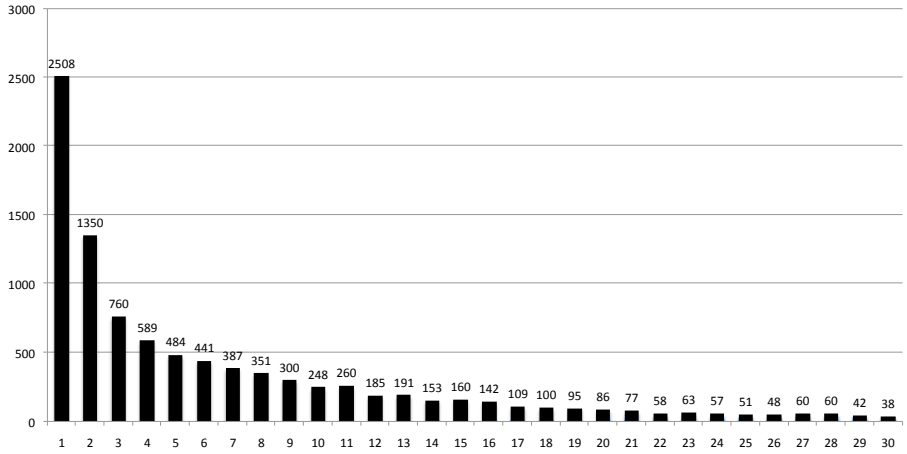


Figure 7.2: Interaction length

Figure 7.2 shows the interaction length (which is equivalent to the number of events it consists of) on its x-asis and the corresponding number of interactions with this length on its y-axis.

As mentioned, an interaction typically lasts from several seconds to few minutes. 95% of all interactions have less then 30 events, 75% of all interactions have less (or equal) then 10 events and more then 50% of all interactions have less then 5 events. In fact, most of the interactions are rather short and the most common interaction length is one event.

The mean duration of interactions was 21 seconds but the longest interaction lasted for nearly 12 minutes.

At the first glance, this curve might look rather boring but it is rather vital: if you ever have a design iteration that influences this curve categorically, odds are high that you have either done something rather good or something rather bad for the usability of the system[1].

[1] This is a good example for the trouble with reasoning in usability testing. We could argue that if the mass centre of the figure moves to the right, the user has to press more buttons to reach its goals, which would be negative. We could also argue that if the mass centre of the figure moves to the right, the user has more joy in using the system and therefore generates more button presses, which would be positive.

7.2 Visualizing Interaction

In its original form, the collected data consists of text-files containing the recorded key presses one event per line (see *Listing 7.1*).

```
  ID   | Date       | Time     | Keycode
=======================================
  1242 | 2010-08-30 | 12:51:28 |    4097
  1242 | 2010-08-30 | 12:51:32 |    4073
  1242 | 2010-08-30 | 12:51:32 |    4074
  1242 | 2010-08-30 | 12:51:33 |    4071
```

Listing 7.1: Simplified log file

This format is optimized for easy storage and handling by a computer. For a human being, it's relatively easy to read this list one-item-at-a-time but it's almost impossible to screen it for patterns (*Appendix A* provides a longer listing).

A logical step to make this data more useful is to utilize a computer program to convert the raw data into a meaningful image. I considered three key requirements when thinking about such a visualization method.

- *Visual representation should be based upon Events, Interactions and Sessions*
  As mentioned, those terms were derived from the visualization and not the other way around. Therefore, it is indeed a bit elusive to call them requirements and I, particularly, want to point out that this was not an a-priori but more of an ad-hoc requirement. As soon as I identified this requirement, I wanted to nail it down for all future iterations, that's why it is here.

- *Information about timing should be embedded*
  The log file contains timestamps but it is impossible to scan those timestamps visually. The desired visualization should incorporate the timely dimension.

- *Re-Use the functional groups of Chapter 6.1 to simplify color mapping*
  As important as the keycode itself is the family the key belongs to. The key family does not have a direct representation in the log file but is derived from the keycode. The same mapping as in prior places in this work should be used. See *Appendix B* for details.

All these consideration and quite an amount of iterations resulted in the following, visually simply, yet highly expressive visualization method.

Figure 7.3: Exemplary visualization of an Interaction

The above picture, which shows a user navigating to teletext page 341, can be *read* in the following way:

„The interaction is initiated by a press on the teletext button. After four seconds, the number button 3 is pressed. After less then one second, the number button 4 is pressed. After one second, the number button 1 is pressed again. For the next 30 seconds, no further key is pressed and the interaction is finished.“

This example, as simple as it is, provides all information necessary to read this class of visualizations. Events (or button presses) are represented by filled rectangles. Their fill color corresponds to the family they belong to. Blue is used for function keys, green is used for numbers and so on. The events are aligned on a horizontal light grey background whereby the spacing between the individual events corresponds to the time that passed by. Interactions are separated by line breaks. The text fields within the events describe the keys, the text fields within the grey background describe the time spans (in seconds). Absolute time is not relevant here, only the amount of seconds something takes is relevant. On the upper left corner, an additional text field holds the unique interaction identifier and the overall length of the interaction in seconds.

I used custom Java code to generate those still images. Groups of forty images each were then put on a canvas and stored as a PDF file. Thereby, one could either choose to sort the insertion by time or interaction length. With the use of vector images, I was able to zoom in if I was interested in a specific text field or detail and zoom out if I wanted to get the big picture. While the visualization was of static nature, my workflow was kind of explorative. The use of a standard PDF viewer and the ability to zoom in and out saved me from writing a more elaborative, interactive visualization code.

Figure 7.4 shows such a PDF file, consisting of a whole series of interactions, each composed of nine events.

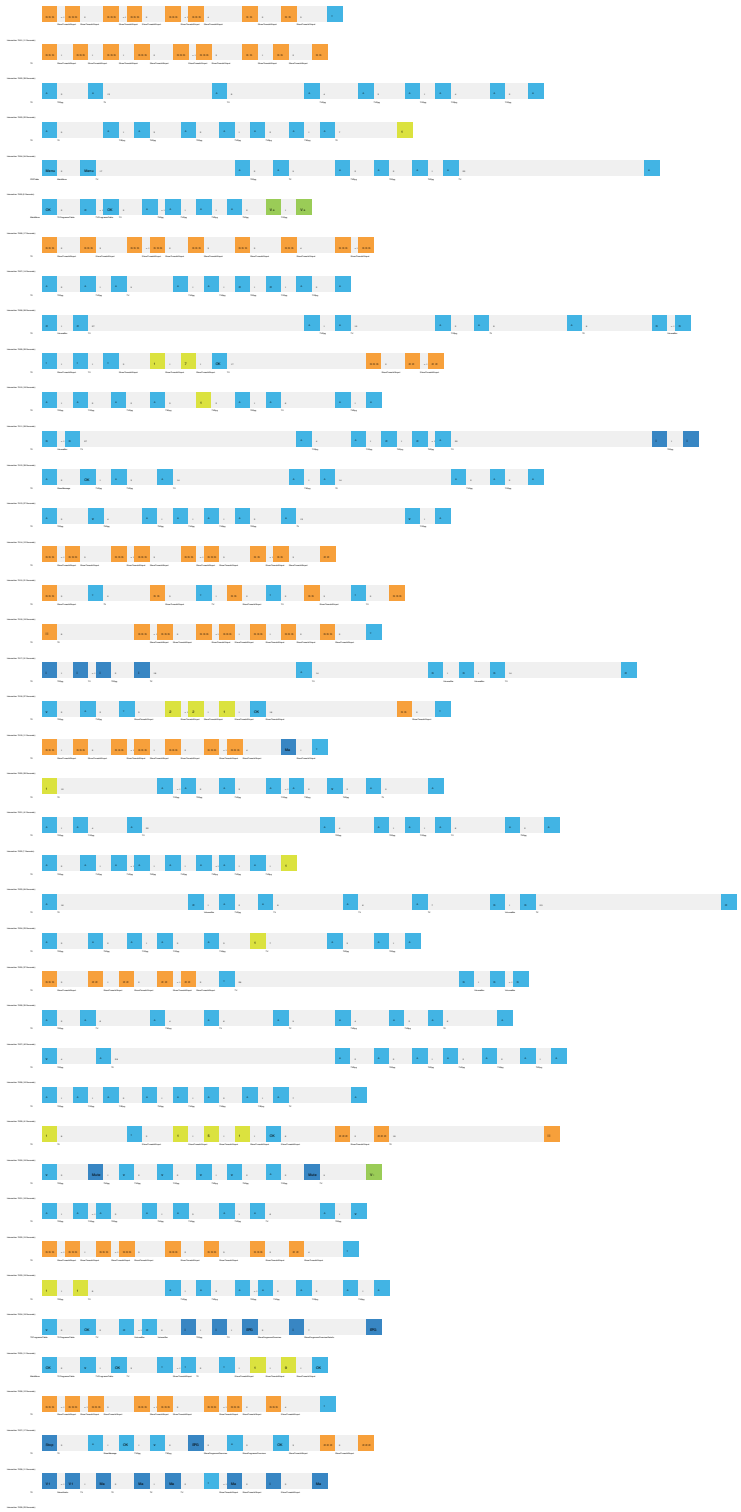


Figure 7.4: Image File

7.3 Interesting Patterns

Equipped with a method to explore the data, I tried to find patterns within it. My approach was rather simple. I just opened different image files next to each other and visually scanned them for patterns. Thereby, I used a variety of off-the-shelf image processing programs to rearrange and resort various parts of the images but my overall workflow has to be considered kind of *unsophisticated*.

If I made an interesting observation, I tried to describe it for myself, noted the original occurrence and tried to validate it against other parts of the dataset.

The following subsections document the most interesting observations I made:

- *Subsection 7.3.1 Categorical Grouping* shows how successive events, very often stay in relationship to each other concerning the family or category they belong to.

- *Subsection 7.3.2 Control Loops* shows how successive events very often manipulate the same dimension, either in the same or in the opposite direction.

- *Subsection 7.3.3 Multi-Event Commands* shows how successive events, very often form a compound command.

- Finally, *Subsection 7.3.4 Error Detection* shows how errors can be detected automatically by analyzing strange interactions.

As a matter of fact, *Categorical Grouping*, *Control Loops* and *Multi-Event Commands* all have a counterpart in a well known concept of computer science, namely within the *Principle of Locality* [Denning 2005] - the theoretical principle behind cache technology[1].

Categorical Grouping might be interpreted as temporal locality (where you access the same entity within small time durations), *Control Loops* might be interpreted as spatial locality (where you access a small number of entities over and over again) and *Multi-Event Commands* might be interpreted as sequential locality (where you access a number of entities in sequential order).

[1] See e.g. http://en.wikipedia.org/wiki/CPU_cache (Retrieved 2012/06/04)

7.3.1 Categorical Grouping

The most obvious and visually comprehensible pattern is that of *categorical grouping*. It's description is simple: successive events do very often belong to the same category (and therefore, share the same color).

Figure 7.5: Categorical Grouping

The above figure shows a large and representative collection of three-event interactions.

Almost all of them contain two successive events with the same color. In fact, less then 3% of the shown interactions do not fulfill this criterium. Additionally, a lot (roughly 60%) of the illustrated interactions only contain events from a single category.

Appendix D contains some examples of interactions built upon more events. Those also confirm the observation of *categorical grouping*. The following figure shows an interaction with 24 events. 23 events out of them form two large blue categorical groups. The example shows a user browsing through the EPG grid, pressing the *Up* button again and again and again.

Figure 7.6: Large categorical groups

It should not be too surprising that buttons with similar functionality are often used in conjunction. If you want to enter a program number, you would have to press some number buttons (which belong to the same button category). If you want to browse the menu, you would have to press some of the cursor buttons (which again belong to the same button category). If you want to find a specific position in a movie, you would have to press some time navigation buttons (which again belong to the same button category) and so forth.

In fact, the classification of buttons into categories and the observation that buttons out of those individual categories are often used in conjunction is somewhat redundant. I nevertheless believe in the value of this observation because the interesting part here isn‘t qualitative but quantitative. The *very* high factor of categorical grouping makes it interesting.

The observation of *categorical grouping* can be used for underpinning the importance of locality in both the design of the physical body of the remote control and the virtual user interface.

7.3.2 Control Loops

Another equally interesting observation is the frequent occurrence of *control loops*. Perfect examples for control loops are channel selection (as shown in *Figure 7.7*), volume selection (as shown in *Figure 7.8*) and time selection (as shown in *Figure 7.9*)

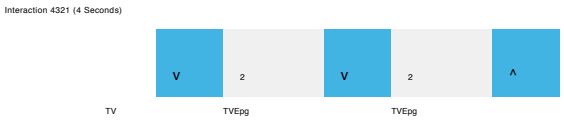


Figure 7.7: Channel selection

Figure 7.8: Volume selection

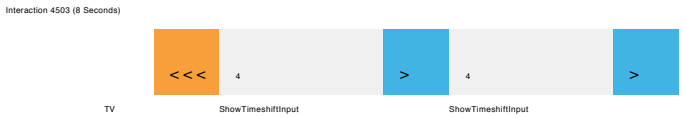


Figure 7.9: Time selection

With control loops, the user presses several buttons in a close series that manipulate the same dimension[1] (like volume, channel, time, cursor position, teletext page number, and so on). Speaking about channel selection (as shown in *Figure 7.7*), the user presses the *Up* and *Down* buttons again and again. Speaking about volume selection (as shown in *Figure 7.8*), the user presses the *V+* and *V-* buttons again and again. Speaking about time selection (as shown in *Figure 7.9*), the user presses the time manipulation keys (*<<<, <<, >>, >>>*) again and again and so forth.

[1] In fact, control loops aren‘t that different from categorical grouping besides the fact that categorical grouping is defined by means of functional groups and control loops rely on the semantics of individual buttons.

Subsequent button presses can manipulate the dimensions in the same direction (e.g. *V+*, *V+*) or in opposite direction (e.g. *V+, V-*). Additionally, subsequent button presses can manipulate the dimension in a more fine-grained or in a more coarse-grained way. *Table 7.1* sums up all of those possibilities[1].

Name	*First key, e.g.*	*Second key, e.g.*
Identical	>>	>>
Opposite	>>	<<
Refine	>>>	>>
Opposite Refine	>>>	<<
Enlarge	>>	>>>
Opposite Enlarge	>>	<<<

Table 7.1: Classes of control loops

It is not uncommon that the speed of control slows down within the interaction. The first events come closely coupled but as finer the adjustment gets the pause between events grows (as you can see in *Figure 7.10*).

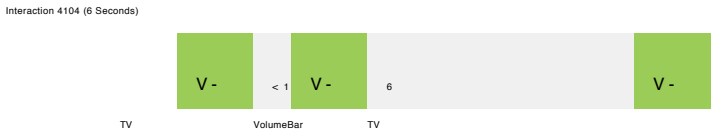


Figure 7.10: Slow down

While comparing lots of control loops for channel, volume and time, an interesting thing showed up, namely that *time* is a very special case. The delays between events within time-manipulating control loops seem to be longer then the delays in volume-manipulating control loops or channel-manipulating control loops. It seems that time is a much more complex dimension then the other two and the cognitive difficulty for us to manipulate it is rather high. Therefore, the navigation aids for the timely dimension (like timelines, clocks, etc.) should be carefully designed and evaluated.

[1] >>> means Fast Forward, >> means Forward, and so on.

The importance of *control loops* is also visible in the examples from *Appendix D* and summed up in *Table 7.2*.

Name	*Number of interactions*	*Number of control loops*
Figure D.1	10	6
Figure D.2	10	9
Figure D.3	10	6
Figure D.4	10	5
Figure D.5	10	9
Figure D.6	10	9
Figure D.7	10	8
Figure D.8	10	9
Figure D.9	10	10

Table 7.2: Occurrence of control loops

In the random sample of different-length interactions, 71 interactions out of 90 interactions (79%) included *control loops*.

7.3.3 Multi-Event Commands

As its name suggests, *Multi-Event Commands* are commands that consist of multiple events. If you have a look at *Figure 7.5*, you can see that some color combinations occur rather frequently. Many of those frequent patterns represent repeated, goal-oriented interactions, some kind of very short scenarios that I simply called *Multi-Event Commands*.

A perfect example for such a *Multi-Event Command* is given in *Figure 7.11*, where the user presses the button *Blue* to enter the EPG grid, followed by the button *Up* to select a program and finally the button *Ok* to select this channel for playback. This *Multi-Event Command* could be called *„Channel selection via EPG grid“*.

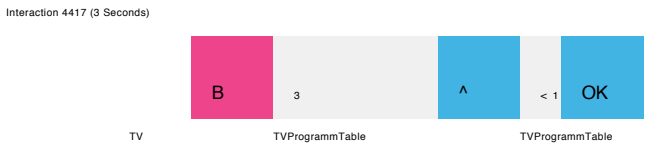


Figure 7.11: Multi-Event Command *Channel selection via EPG grid*

Another *Multi-Event Command* is the following form of channel selection illustrated in *Figure 7.12*. The user presses the „_ _“ button, followed by two presses on the *1* button, followed by a press on the *Ok* button. This *Multi-Event Command* could be called *„Two-Digit Channel Selection“*.

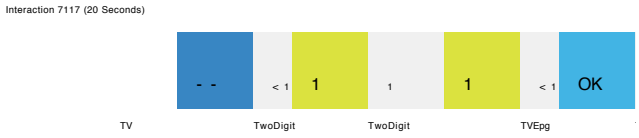


Figure 7.12: Multi-Event Command *Two-Digit Channel Selection*

In fact, *Multi-Event Commands* can be found in many interactions, independent of interaction length. They are important because they represent small, immediate user actions, which is just the place where users either get excited or frustrated about the nuances of interactions. *Multi-Event Commands* are far less frequent[1] then *control loops* and in fact, the 90 interactions from *Appendix D* only contain 13 *Multi-Event Commands* (as shown in *Table 7.3*).

Reference location	*Multi-Event Command*
Figure D.1, Interaction 3	*Channel Selection via Program Table*
Figure D.1, Interaction 5	*Channel Selection via Program Table*
Figure D.1, Interaction 6	*Channel Selection via Program Table*
Figure D.2, Interaction 3	*Show detailed EPG information*
Figure D.4, Interaction 2	*Teletext page input*
Figure D.4, Interaction 3	*Show detailed EPG information*
Figure D.4, Interaction 5	*Show detailed EPG information*
Figure D.4, Interaction 7	*Show detailed EPG information*
Figure D.5, Interaction 8	*Show detailed EPG information*
Figure D.6, Interaction 10	*Confirm Dialog*
Figure D.7, Interaction 1	*Time Input*
Figure D.7, Interaction 7	*Time Input*
Figure D.8, Interaction 3	*Time Input*

Table 7.3: Occurrence of *Multi-Event Commands*

[1] At least if we define that a sequence of commands either is a *control loop* or a *multi-sequence-command* (or none of those) and not both at the same time.

7.3.4 Error Detection

A common method to detect errors is to outsource it to users. This is then called *error reporting* and it has some benefits for both the users and the owners of the system. Within the context of interactive television, error reporting techniques might do more harm than good. User tolerance for failure in the living room is very low. The high subjective value of *quality time* in front of the television set makes something like error reporting impracticable. It might be a good idea to provide something like a hotline for de-escalation of user problems but asking users for error reports is, in my opinion and in the given context, illusive.

This might be different if you have a group of *friendly customers* where you do a long-term test of your new features before you go primetime.

As described in [Akers et al. 2009] *„one approach to reducing the costs of usability testing is to facilitate the automatic detection of critical incidents: serious breakdowns in interaction that stand out during software use.“* Instead of letting the users report their errors explicitly, one tries to find errors by means of analyzing user actions.

Figure 7.13 provides a good example for this. It starts with the user pressing the *Blue* key to enter the EPG grid. With the press of *Ok*, the user leaves the EPG grid again and the selected video stream is displayed. Two seconds into the video, the set-top box is powered off. This whole interaction does not make too much sense.

In fact, each and every interaction that do not make to much sense or feels kind of unnecessary can potentially reveal an error.

Figure 7.13: Breakdown in Interaction

An *error*, in this context, includes some very different types of errors but the focus is on *design errors* that have a negative influence on proper usability. This loose definition of the term *error* is sufficient for this work since whichever error happens, also (implicitly) is a problem concerning usability.

In fact, there are a variety of patterns that potentially indicate error situations, amongst others I found the following:

- Long pauses between events might indicate that the user needs a lot of time to choose amongst the available options. If this happens within the menu, there might be too much menu items. If this happens within another navigational element, the structure of this element might be too complex. If this happens within multi-event commands, either physical button placement or memorability of the combination itself might be bad.

- If the remote control has a *Help* button, both the amount of presses of this button and the context where it was pressed is vital information.

- Repeated and/or random presses on keys that reset state (like *Menu*, or *On/Off*) might indicate that the user got lost. Despite popular believe and sarcasm „Have you tried turning it off and on again?“ is a *proper* question and a hardware reset is a widespread method to tackle usability problems.

- Repeated and/or random presses on keys that wind up the menu tree (like *Home*, *Menu* or *Back*) might indicate that the user can‘t find his/her way in the navigation tree.

- Presses on buttons that do not have functionality assigned in the given context (i.e. dead keys) can indicate usability problems (e.g. inconsistent button usage).

- Presses on buttons that made sense in the contextual dimension but have already reached their bounds (e.g. pressing *Rewind* when a video is already rewound to the very beginning) might indicate usability problems.

- Usage of long menu paths instead of their shortcuts might indicate improper guidance for using those.

All of those factors could be evaluated automatically into something like a list of *key-error-indicators*. Tracking these indicators would be a powerful tool to validate iterations of the user interface.

Chapter 8 : Conclusions

Interactive television is a *hot* topic. In opposite to computer programs or websites, it does not yet have a stable visual language – this provides room for true innovation which is necessarily accompanied by a high risk of failure. Usability evaluation methods can help to identify errors in the area of user interaction, usability, design, etc. The used method of *remote logging* provided interesting insights while *not* interfering with the subject of test at all.

The research question – if the data of button presses in an interactive television system is useful to gain insights for the design of remote controls – can be positively answered. While it is difficult to produce a large amount of unambiguous advice, it is possible to present the data in ways that provide the possibility to find significant indicators for *usability problems* or that help to select a road in the overall design space. While the technical aspects of collecting all the available data was relatively simple, finding ways to densify this data into human-readable images was more of a challenge. Thinking about which conclusions can be drawn from those images is the most difficult part. Correlation is easily mistaken with causality and it is easy to see patterns where there aren‘t. Take for example, *Chapter 6.1*, which is about button count. Using the methods from this chapter, it is possible to identify buttons that are only used rarely. Here comes the important point: if a button is only used rarely, this does not automatically imply that this button is worthless. Just think about the *On/Off* button. It‘s only used rarely but its absence could be daunting.

Information visualization proved to be a cornerstone for understanding the data. In *Chapter 6.1 Button Count*, a method was found for deciding whether a button has a right to exist on the limited, physical body of the remote control or not. It looks like the keys that traditionally have its place on the remote (like *On/Off*, *0, 1, 2*, …, etc.) are well understood and used. The same applies to the general purpose navigation keys. Problems exist with infrequently used buttons that have no clearly designated meaning (like the *info* button) or keys that are simply new (like the *EPG* button). With the current feature set, it indeed is a challenge to design a remote control with less then 40 buttons and it is questionable if purely puristic approaches with minimal button count that offload all functionality to the screen are desirable. The starting point of this work – where precise fine tuning of button count and layout is proposed in favor of radical, minimal redesign – seems to be right (at least as long as no other technological

paradigm replaces the traditional remote control entirely). In *Chapter 6.2 Button Placement*, a method was found for investigating layout issues and indeed minor layout quirks were found.

The methods and patterns described throughout *Chapter 7 Virtual Layout* provide significant possibilities to see things from the user‘s perspective. Especially the commonly occurring *Categorical Grouping* show that the use of the remote control isn‘t a highly sophisticated task with a predefined goal but rather a loose, arbitrary browsing. That‘s just the nature of the contemporary television experience and there should be absolutely no usability penalty (like channel-switching times in the magnitude of several seconds) for this kind of behavior.

Throughout this work, still images were used for investigation. The static nature of the images was necessary to implement the visualization methods in a timely fashion but it clearly limits its use. During the course of this work, I more and more developed the desire to extend my images to dynamic visualizations, enabling me to formulate questions like *What‘s the most common starting key for an interaction when I‘m in the main menu*? This would be an interesting direction for future research.

I would like to end this thesis with some wise words from Cliff Stoll,

> *Data is not information,*
> *Information is not knowledge,*
> *Knowledge is not understanding,*
> *Understanding is not wisdom.*

This Page Intentionally Left Blank.

Acknowledgements

This work would not have been possible without the continuous support of my parents Marianne, Heinrich and Günter and all of my family.

I would also like to thank Ocilion IPTV Technologies[1] for their efforts and support.

All of the software development for this thesis was done using excellent open-source software, namely the Eclipse[2] development environment, the Processing[3] library and the GNU/Linux Operating System.

Finally, I would like to thank Ao. Univ. Prof. Dr. Margit Pohl of the Human Computer Interaction Group of the Technical University of Vienna for her time, supervision and diligent guidance.

[1] http://www.ocilion.com/

[2] http://www.eclipse.org/

[3] http://www.processing.org/

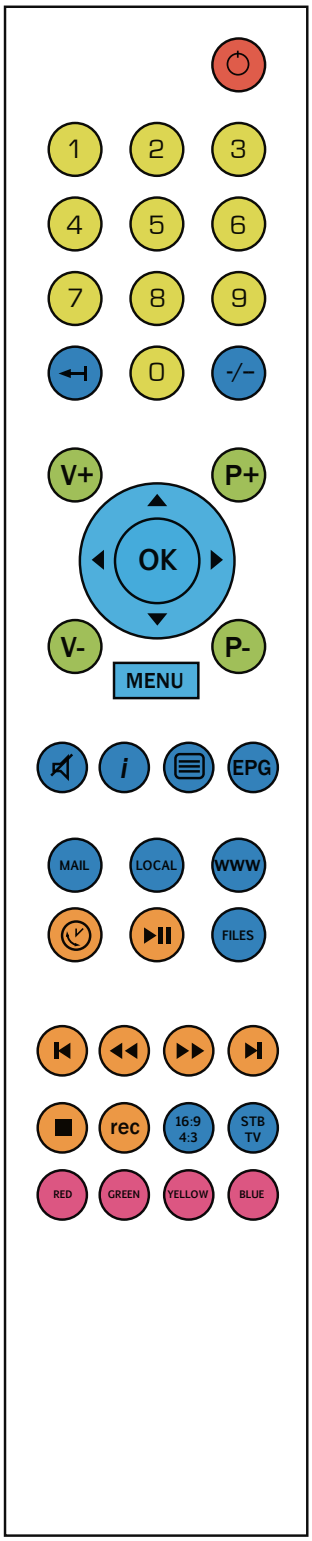
Appendix A: Log-File-Format

STB	Time	Keycode	State
1002	20100902105802000	keycode=4097	state=Standby keyname=key_onoff
1002	20100902105805000	keycode=4176	state=MainMen keyname=key_down
1002	20100902105805000	keycode=13	state=MainMenu keyname=key_ok
1002	20100902105808000	keycode=13	state=TimeshiftTable
1002	20100902105810000	keycode=4171	state=TimeshiftTimeTable
1002	20100902105812000	keycode=4171	state=TimeshiftTimeTable
1002	20100902105814000	keycode=4312	state=TimeshiftTimeTable
1002	20100902105814000	keycode=4312	state=TimeshiftTimeTable
1002	20100902105815000	keycode=4312	state=TimeshiftTimeTable
1002	20100902105816000	keycode=13	state=TimeshiftTimeTable
1002	20100902105817000	keycode=4297	state=ShowTimeshiftInput
1002	20100902105818000	keycode=4297	state=ShowTimeshiftInput
1002	20100902105818000	keycode=4297	state=ShowTimeshiftInput
1002	20100902105819000	keycode=4297	state=ShowTimeshiftInput
1002	20100902105819000	keycode=4297	state=ShowTimeshiftInput
1002	20100902105821000	keycode=4297	state=ShowTimeshiftInput
1002	20100902112827000	keycode=4227	state=TV
1002	20100902112830000	keycode=4226	state=ShowTimeshiftInput
1002	20100902113931000	keycode=4296	state=TV
1002	20100902113932000	keycode=4296	state=VolumeBar
1002	20100902114831000	keycode=4227	state=TV
1002	20100902114834000	keycode=4227	state=ShowTimeshiftInput
1002	20100902114844000	keycode=4227	state=TV
1002	20100902114847000	keycode=4227	state=ShowTimeshiftInput
1002	20100902114849000	keycode=4227	state=ShowTimeshiftInput
1002	20100902114851000	keycode=4227	state=ShowTimeshiftInput
1002	20100902115301000	keycode=4178	state=TV
1002	20100902115302000	keycode=4312	state=MainMenu
1002	20100902115303000	keycode=13	state=MainMenu
1002	20100902115305000	keycode=4171	state=TVProgrammTable
1002	20100902115306000	keycode=4176	state=TVProgrammTable
1002	20100902115307000	keycode=4176	state=TVProgrammTable
1002	20100902115307000	keycode=4176	state=TVProgrammTable
1002	20100902115307000	keycode=4176	state=TVProgrammTable
1002	20100902115308000	keycode=4176	state=TVProgrammTable
1002	20100902115309000	keycode=4176	state=TVProgrammTable
1002	20100902115309000	keycode=13	state=TVProgrammTable
1002	20100902115312000	keycode=4097	state=TV
1002	20100902121411000	keycode=4097	state=Standby
1002	20100902121413000	keycode=4176	state=MainMenu
1002	20100902121414000	keycode=13	state=MainMenu
1002	20100902121415000	keycode=4312	state=TimeshiftTable
1002	20100902121416000	keycode=4312	state=TimeshiftTable
1002	20100902121417000	keycode=13	state=TimeshiftTable
1002	20100902121419000	keycode=4171	state=TimeshiftTimeTable
1002	20100902121421000	keycode=4171	state=TimeshiftTimeTable
1002	20100902121423000	keycode=4171	state=TimeshiftTimeTable
1002	20100902121426000	keycode=4312	state=TimeshiftTimeTable
1002	20100902121426000	keycode=4312	state=TimeshiftTimeTable
1002	20100902121427000	keycode=13	state=TimeshiftTimeTable

```
1002  20100902121453000  keycode=4297   state=TV
1002  20100902121454000  keycode=4297   state=VolumeBar
1002  20100902121644000  keycode=4297   state=TV
1002  20100902121813000  keycode=4227   state=TV
1002  20100902122539000  keycode=4227   state=TV
1002  20100902122645000  keycode=4227   state=TV
1002  20100902122654000  keycode=4227   state=ShowTimeshiftInput
1002  20100902122729000  keycode=4178   state=TV
1002  20100902122730000  keycode=4312   state=MainMenu
1002  20100902122731000  keycode=13     state=MainMenu
1002  20100902122734000  keycode=13     state=TVProgrammTable
1002  20100902122737000  keycode=4097   state=TV
1002  20100903042713000  keycode=4097   state=Standby
1002  20100903042717000  keycode=4176   state=MainMenu
1002  20100903042718000  keycode=13     state=MainMenu
1002  20100903042720000  keycode=4176   state=TimeshiftTable
1002  20100903042721000  keycode=4176   state=TimeshiftTable
1002  20100903042722000  keycode=13     state=TimeshiftTable
1002  20100903042725000  keycode=4171   state=TimeshiftTimeTable
1002  20100903042728000  keycode=4312   state=TimeshiftTimeTable
1002  20100903042728000  keycode=4312   state=TimeshiftTimeTable
1002  20100903042729000  keycode=4312   state=TimeshiftTimeTable
1002  20100903042729000  keycode=4312   state=TimeshiftTimeTable
1002  20100903042730000  keycode=4312   state=TimeshiftTimeTable
1002  20100903042730000  keycode=4312   state=TimeshiftTimeTable
1002  20100903042731000  keycode=4312   state=TimeshiftTimeTable
1002  20100903042731000  keycode=4312   state=TimeshiftTimeTable
1002  20100903042732000  keycode=4312   state=TimeshiftTimeTable
1002  20100903042733000  keycode=13     state=TimeshiftTimeTable
1002  20100903042737000  keycode=4226   state=ShowTimeshiftInput
1002  20100903042739000  keycode=4226   state=ShowTimeshiftInput
1002  20100903042806000  keycode=4297   state=TV
1002  20100903042825000  keycode=4297   state=TV
1002  20100903042826000  keycode=4297   state=VolumeBar
1002  20100903042826000  keycode=4297   state=VolumeBar
1002  20100903044725000  keycode=4296   state=TV
1002  20100903044726000  keycode=4296   state=VolumeBar
1002  20100903044816000  keycode=4297   state=TV
1002  20100903044816000  keycode=4297   state=VolumeBar
1002  20100903044821000  keycode=4297   state=TV
1002  20100903044954000  keycode=4227   state=TV
```

Listing A.1: Raw log file

Appendix B: Remote Control Layout



A special key is the on/off-button since it marks the beginning and end of a session. Upon its infrequent occurrence but high importance it is given a red background.

The three most frequently used groups are those for navigation (light blue), plus/minus (dark green) and time-related (orange).

As mentioned in *Chapter 6*, those three top groups amount to almost 90% of all key presses. Three distinctive colors were used for the top groups since they amount to the vast majority of colored output in the soon to be presented visualization.

The three remaining groups contain all the numbers (light green), the color keys (pink) and all other remaining buttons (dark blue). All of the buttons in this last group aren't used very frequently, which is the common property of all the buttons in this group.

As you can see, group membership is closely related to the positions and distance of the individual buttons. Neighboring buttons regularly belong to the same group.

Figure B.1: Color assignment of key families

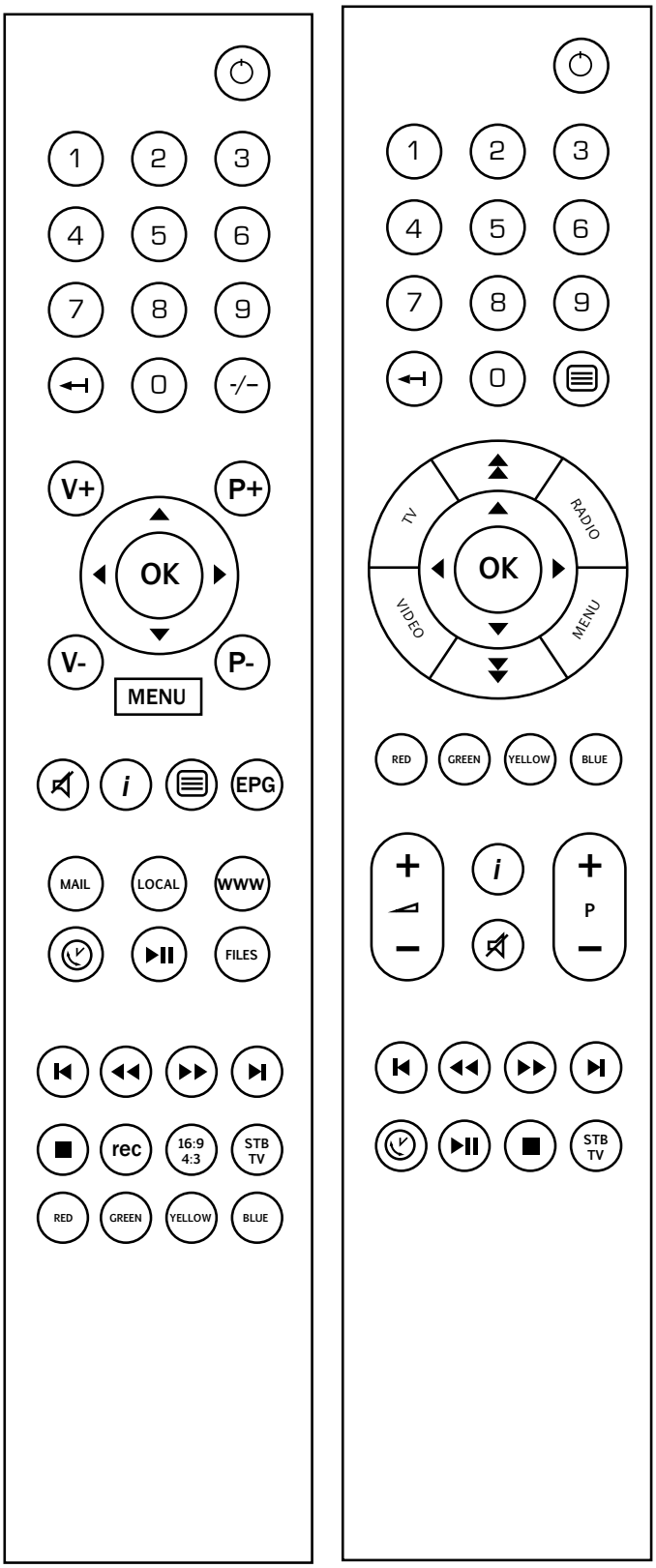


Figure B.2: Remote control iterations

Appendix C: Abbreviations

ADSL	Asymmetric Digital Subscriber Line
CRID	Content Reference Identifier
DSL	Digital subscriber line
FTTH	Fiber to the Home
GUI	Graphical User Interface
HDMI	High Definition Multimedia Interface
HDTV	High Definition Television
IPTV	Internet protocol television
ITU	International Telecommunications Union
MHP	Multimedia Home Platform
MPTS	Multiple Program Transport Stream
NGN	Next Generation Network
OSD	On-screen display
PVR	Personal video recorder
PPV	Pay-per-view
PS	MPEG Program Stream
SoC	System-On-Chip
STB	Set-top box
SPTS	Single Program Transport Stream
TS	MPEG Transport Stream
VoD	Video on Demand
VCR	Video Cassette Recorder
XML	Extensible Markup Language

Appendix D: Visualization Samples

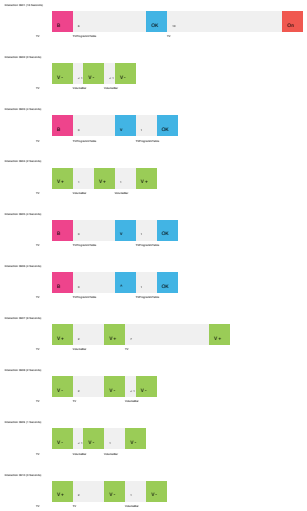


Figure D.1: Sample interactions consisting of three events

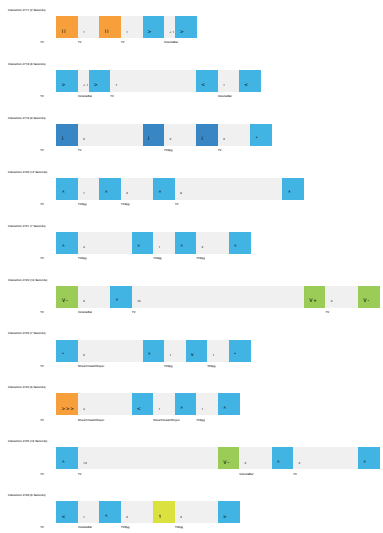


Figure D.2: Sample interactions consisting of four events

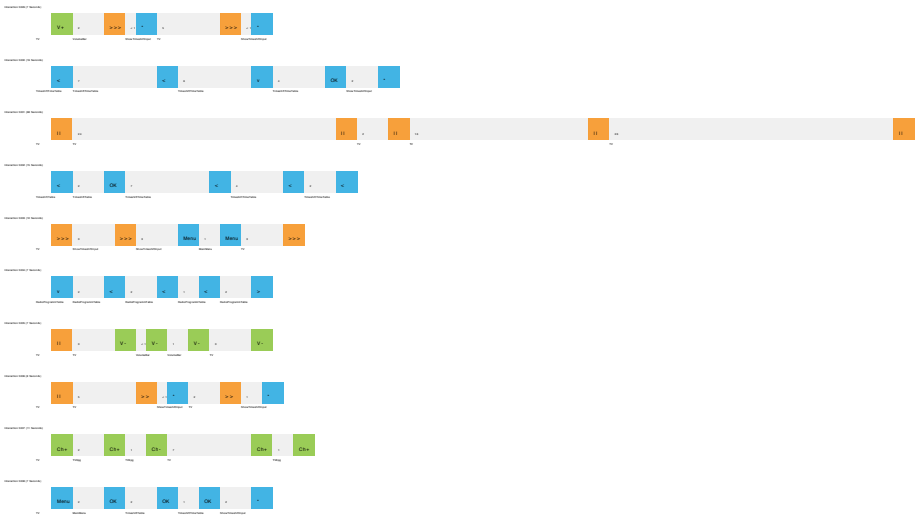


Figure D.3: Sample interactions consisting of five events

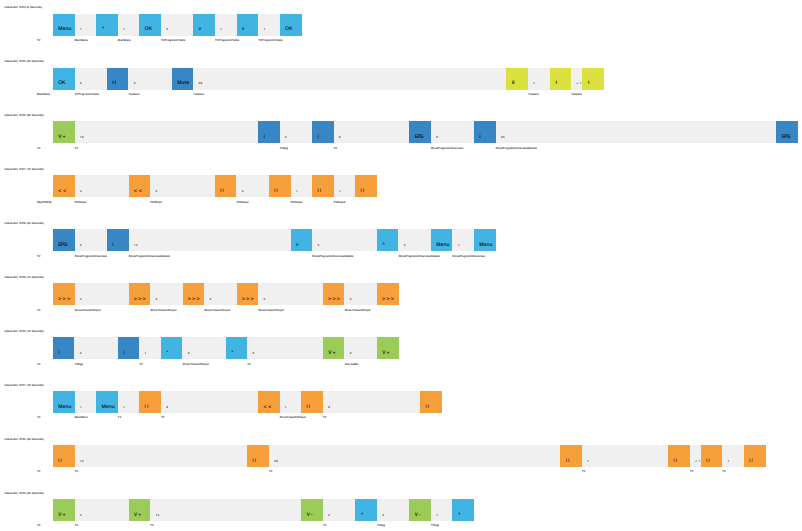


Figure D.4: Sample interactions consisting of six events

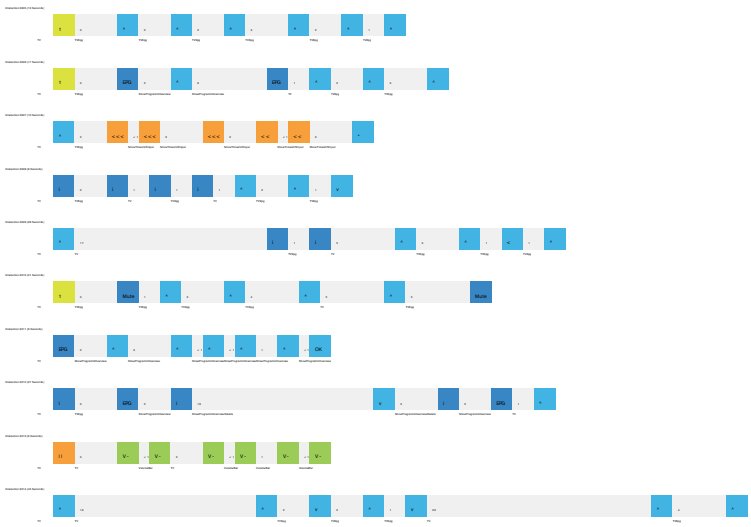


Figure D.5: Sample interactions consisting of seven events

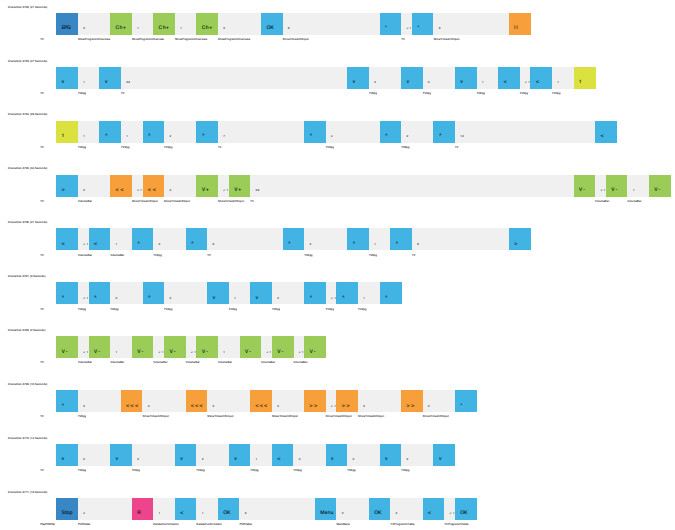


Figure D.6: Sample interactions consisting of eight events

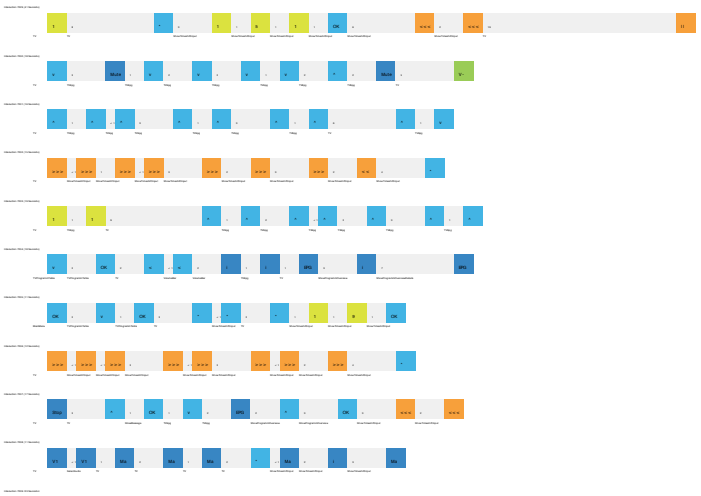


Figure D.7: Sample interactions consisting of nine events

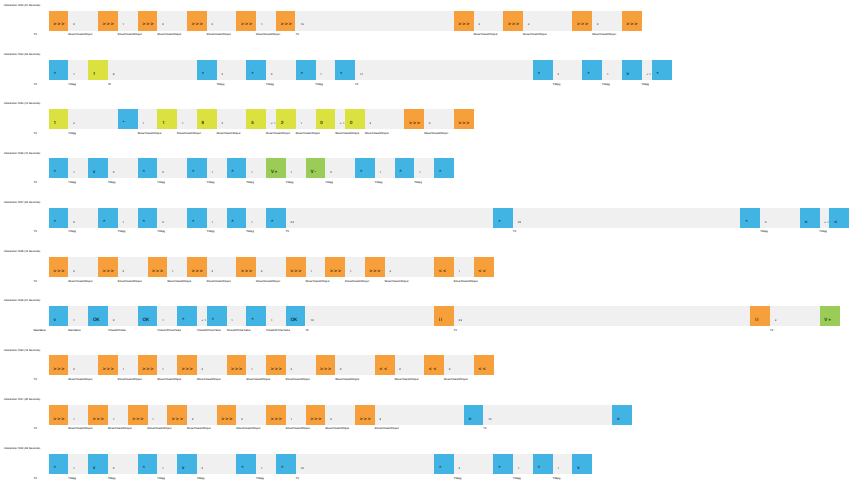


Figure D.8: Sample interactions consisting of ten events

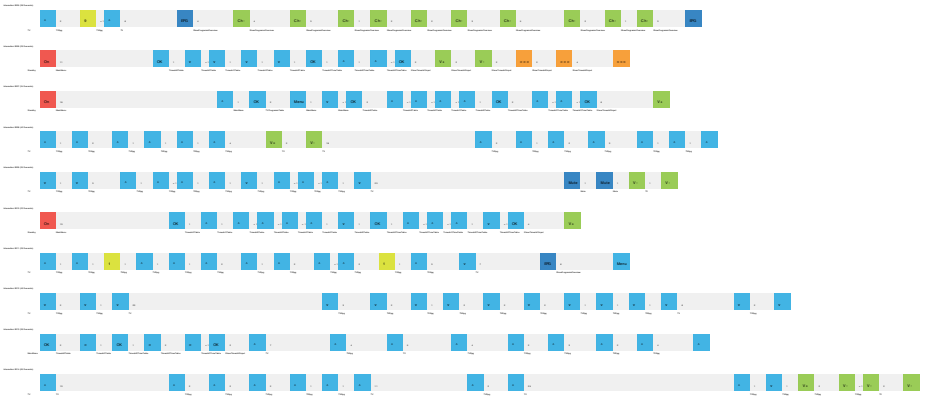


Figure D.9: Sample interactions consisting of fifteen events

Bibliography

[Andrews 1998] Andrews, J.H., 'Theory and practice of log file analysis', 1998.

[Akers et al. 2009] Akers, D.; Simpson, M.; Jeffries, R. & Winograd, T., 'Undo and erase events as indicators of usability problems', CHI '09: Proceedings of the 27th international conference on Human factors in computing systems, ACM, New York, 2009; pp. 659-668.

[Apple 1992] Apple Computer, Inc. ' Macintosh human interface guidelines', Addison-Wesley, 1992.

[Bellotti et al. 2005] Bellotti, F. & Gloria, A.D. & Montanari, R. & Dosio, N. & Morreale, D., 'COMUNICAR: designing a multimedia, context-aware human-machine interface for cars', Cognition, Technology & Work, Volume 7, Number 1, Springer, 2005; pp. 36-45.

[Berglund et al. 2004] Berglund, A. & Johansson, P., 'Using speech and dialogue for interactive TV navigation', Universal Access in the Information Society, Volume 3, Number 3, Springer, 2004; pp. 224-238.

[Bernhaupt et al. 2007] Bernhaupt, R.; Obrist, M.; Tscheligi, M., 'Usability and usage of iTV services: lessons learned in an Austrian field trial', Computers in Entertainment (CIE), Volume 5, Issue 2, ACM, New York, 2007; pp. 6.

[Bernhaupt et al. 2008] Bernhaupt, R.; Obrist, M.; Weiss, A.; Beck, E.; Tscheligi, M., 'Trends in the living room and beyond: results from ethnographic studies using creative and playful probing', Computers in Entertainment (CIE), Volume 6, Issue 1, ACM, New York, 2008; pp. 1-23.

[Bruun et al. 2009] Bruun, A. & Gull, P. & Hofmeister, L. & Stage, J., 'Let your users do the testing: a comparison of three remote asynchronous usability testing methods', Proceedings of the 27th international conference on Human factors in computing systems, ACM, New York, 2009; pp. 1619-1628.

[Cesar et al. 2008] Cesar, P. & Chorianopoulos, K. & Jensen, J.F., 'Social television and user interaction', Computers in Entertainment (CIE), Volume 6, Number 1, ACM, New York, 2008; pp. 1-10.

[Cook et al. 2002] Cook, L.S. & Bowen, D.E. & Chase, R.B. & Dasu, S. and Stewart, D.M. & Tansik, D.A., 'Human issues in service design', Journal of Operations Management, Volume 20, Number 2, 2002; pp. 159-174.

[Cooper 2008] Cooper, W., 'The interactive television user experience so far'. Proceeding of the 1st international conference on Designing interactive user experiences for TV and video. UXTV '08, ACM, New York, 2008; pp. 133-142.

[Damiani et al. 2009] Damiani, S. & Deregibus, E. & Andreone, L., 'Driver-vehicle interfaces and interaction: where are they going?', European Transport Research Review, Volume 1, Number 2, Springer, 2009; pp 87-96.

[Darnell 2008] Darnell, M.J, 'Making Digital TV Easier For Less-Technically-Inclined People', Proceedings of the 1st international conference on Designing interactive user experiences for TV and video. ACM, 2008; pp. 27-30.

[Denning 2005] Denning, P.J., 'The locality principle', Communications of the ACM, Volume 48, Number 7, ACM, New York, 2005; pp. 19-24.

[Dix et al. 1998] Dix, A. & Finlay, J. & Abowd, G. & Beale, R., 'Human-Computer Interaction (second ed.).' Prentice Hall, New York, 1998.

[Epelde et al. 2011] Epelde, G. & Carrasco, E. & Zimmermann, G. & Alexandersson, J. & Neßelrath, R. & Dubielzig, M., 'Universal Remote Console-based next-generation accessible television', Universal Access in the Information Society, Springer, 2011; pp. 1-15.

[Freeman et al. 1994] Freeman, W.T. & Weissman, C., 'Television control by hand gestures', Proceedings of the International Workshop on Automatic Face and Gesture Recognition, 1994; pp. 179-183.

[Gawlinski 2003] Gawlinski, M., 'Interactive television production', Focal Press, Oxford, 2003.

[Graham 2004] Graham, Paul; Hackers & Painters; O‘Reilly 2004.

[Greenberg 2008] Greenberg, S. & Buxton, B., 'Usability evaluation considered harmful (some of the time)', CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems, ACM, New York, 2008; pp. 111-120.

[Guzdial et al. 1994] Guzdial, M. & Santos, P.J. & Badre, A. & Hudson, S.E. & Gray, M.H.., 'Analyzing and visualizing log files: A computational science of usability', Georgia Institutue of Technology, 1994.

[Ivory et al. 2001] Ivory, M. & Hearst, M., 'The state of the art in automating usability evaluation of user interfaces', ACM Computing Surveys (CSUR), Volume 33, Number 4, ACM, New York, 2001; pp. 470-516.

[Itten 1961] Itten, J., 'The art of color', Reinhold Pub. Corp., New York, 1961.

[Jensen 2008] Jensen, J. 'Interactive Television - A Brief Media History', Changing Television Environments: Proceedings of the sixth european conference on Interactive Television, Springer, 2008; pp. 1-10.

[Kern 2008] Kern, T.A., 'Entwicklung haptischer Geräte: Ein Einstieg für Ingenieure', Springer, 2008.

[Kim et al. 2004] Kim, S.H. & Ok, J. & Kang, H.J. & Kim, M.C. & Kim, M., 'An interaction and product design of gesture based TV remote control', CHI'04 extended abstracts on Human factors in computing systems, ACM, 2004; pp. 1548-1548.

[Krug 2005] Krug, S., 'Don't Make Me Think: A Common Sense Approach to the Web', New Riders Press, California, 2005.

[Kunert 2009] Kunert, T., 'User-centered Interaction Design Patterns for Interactive Digital Television Applications', Human-Computer Interaction Series. Springer Publishing Company, London, 2009.

[Lo et al. 2006] Lo, S.H. & Lin, C.C. & Chen, M.S., 'Controlling digital TV set-top box with mobile devices via an IP network'. IEEE Transactions on Consumer Electronics, Volume 52, Number 3, IEEE, 2006; pp. 52-59.

[Lu 2005] Lu, K.Y., 'Interaction design principles for interactive television', 2005.

[Lund 1997] Lund, A.M., 'Expert ratings of usability maxims', Ergonomics in Design: The Quarterly of Human Factors Applications, Volume 5, Number 3, 1997; pp. 15-20.

[Mager 2009] Mager, B., 'Service Design. Reihe Design studieren', UTB, Stuttgart, 2009.

[Moggridge 2006] Moggridge, B., 'Designing Interactions', MIT Press, Cambridge, 2006.

[Negroponte 1995] Negroponte, N., 'Being digital', Alfred A. Knopf Publishers, 1995.

[Nielsen 1993] Nielsen, J., 'Usability Engineering', Academic Press, Boston, 1993.

[Nielsen 1994] Nielsen, J., 'Usability inspection methods', Conference companion on Human factors in computing systems, ACM, New York, 1994; pp. 413-414.

[Norman 1988] Norman, A.D., 'Psychology of Everyday Things', 1988.

[Pohl et al. 2010] Pohl, M. & Wiltner, S. & Miksch, S., 'Exploring Information Visualization - Describing Different Interaction Patterns', BELIV'10.

[Rampersad 2010] Rampersad, H.K., 'Total Quality Management; an executive guide to continuous improvement', Springer, 2010.

[Roibás et al. 2005] Roibás, A. & Sala, R. & Ahmad, S. & Radman, M., 'Beyond the remote control: Going the extra mile to enhance iTV access via mobile devices & humanizing navigation experience for those with special needs', Proceedings of the 3rd European Conference on Interactive Television, 2005.

[Rubin et al. 2008] Rubin, J. & Chisnell, D., 'Handbook of usability testing: how to plan, design and conduct effective tests', John Wiley & Sons, 2008.

[Ruhrmann et al. 1997] Ruhrmann, G. & Nieland, J.U., 'Interaktives Fernsehen: Entwicklung, Dimensionen, Fragen, Thesen', Westdeutscher Verlag, Opladen, 1997.

[Shostack 1993] Shostack, G.L, 'How to design a service', European Journal of Marketing, Volume 16, Number 1, Emerald Group Publishing, 1993; pp. 49-63.

[Schwartz 2004] Schwarz, B., 'The paradox of choice', HarperCollins, New York, 2004.

[Shneiderman 1986] Shneiderman, B., 'Designing the user interface: strategies for effective human-computer interaction', Addison-Wesley Longman Publishing Co., Boston, 1986.

[Teixeira et al. 2009] Teixeira, C.A.C. & Melo, E.L. & Cattelan, R.G. & Pimentel, M.G.C., 'User-media interaction with interactive TV', Proceedings of the 2009 ACM symposium on Applied Computing, ACM, 2009; pp. 1829-1833.

[Tufte et al. 1983] Tufte, E.R. & Graves-Morris, P., 'The visual display of quantitative information', Graphics press Cheshire, 1983.

[Valdman 2001] Valdman, J., 'Log file analysis' Technical Report DCSE/ TR-2001-04, University of West Bohemia, 2001.

[Ware 2004] Ware, C., 'Information visualization: perception for design', Morgan Kaufmann, 2004.

[Wertheimer 1938] Wertheimer, M., 'Laws of Organization in Perceptual Forms ', A source book of Gestalt psychology, 1938; pp. 71-88.

[Wittenburg et al. 2006] Wittenburg, K. & Lanning, T. & Schwenke, D. & Shubin, H. & Vetro, A., 'The prospects for unrestricted speech input for TV content search', Proceedings of the working conference on Advanced visual interfaces, ACM, New York, 2006; pp. 352-359.

[ISO 9241] ISO 9241-410:2008, 'Ergonomic requirements for office work with visual display terminals - Design criteria for physical input devices', International Organization for Standardization, Geneva, 2008.

[ISO 13407] ISO 13407:1999, 'Human-centered design processes for interactive systems', International Organization for Standardization, Geneva, 2008.