MAD

Deliverable D16.1
Content retrieval & browsing interfaces - General Design Rules & The UI's in PrestoSpace

DOCUMENT IDENTIFIER	PS_WP16_ORF_D16.1_Interfaces_v1.5
DATE	20/11/2008
ABSTRACT	General Rules for the Designing of User-Interfaces, with an attempt to identify a common denominator in the surveyed Rule-compilations. Appliance of those rules on the Content & Browsing-Interfaces of PrestoSpace, with an extension of the review with other PrestoSpace-user-interfaces and an appendix with some examples of other MAM, CMS and content-related user-interfaces
KEYWORDS	User-Interface, Design-Rules, Retrieval, Browsing
WORKPACKAGE / TASK	WP16
AUTHOR, COMPANY	Christoph Bauer (ORF)
NATURE	RE : Report
DISSEMINATION	PP : Programme Participants
RELATED ITEMS	D18.2
INTERNAL REVIEWERS	Giorgio Dimino, RAI

DOCUMENT HISTORY

	Reason of change	
	First Draft	
0.2	Golden Rules – first version	Confidential
1.0	Golden Rules – RC, UI's	
1.1	UI's-Reviews added	
1.2	Appendix A added, minor changes	
1.3	Appendix B added, minor changes	
1.4	Screenshots added, Overview	
1.5	Finalised	

1 Executive Summary

This deliverable is a report containing the result of a survey on “Golden Rules” for designing User-Interface, focused on the elaboration of a “common denominator” in these different rule-compilations and the appliance of this summary for reviewing the content-retrieval and browsing Interfaces designed and created in PrestoSpace. In an enhancement other PrestoSpace user-interfaces are reviewed by using the same set of rules, and in an additional appendix the common use and implementation of the rules for designing User-Interfaces is explored by giving a short overview on the design used for user-interfaces from different MAM, CMS and related systems, both professional and common.

2 Overview

Since the very beginning of industrialisation and the first human-machine-interfaces like the controls of the first steam-engines, science started to find ways to design those interfaces in a way to support the main task of the operator and his machine. A very good example for the progress in this field is vehicle construction, where the main rules for the “interface” (steering wheel, pedals, dashboard and alike) has been set up in the very early years after the inventions of the first automobiles, which are still valid and in use.

The first IT-User-Interfaces has been taken directly from already existing interfaces (keyboard, 80-digit-screens, etc.), introducing very early the rule of “Familiarity” and “Real-World-Metaphors” in the course of development. Science and industry first forced some real valuable, concerted actions in standardising those design-rules (Screen-layouts in the 70’s and 80’s), but with the upcoming of GUI’s (like those from Apple and Atari), the industry more and more got caught in the common game of staking off the claim, so for quite a long time the developments got stuck in patents-struggles and setting up worse alternatives.

Since the mid-90’s the defacto market-dominance of Microsoft again brought again a kind of standardisation of User-Interfaces in the common IT-world; especially the “Office”-products and the Windows-OS’s contributed to this “pseudo-standardisation” in User-Interfaces. Funny enough, that with the introduction of “Office 2007” Microsoft started to leave this track and introduced a new user-interface, confusing hundreds of thousand common IT-users…

While in the “Office”- and SOHO-environment most developers and software-engineers followed the “Microsoft-tracks”, in the professional IT-business this trend was much lower, caused by the very often special tasks, the user-interfaces had to fulfil. So at least that area was always good for the introduction of fresh and new ideas in the domain of user-interfaces.

For the software-engineer and the designer of User-interfaces sets of rules had been put up from the very beginning of designing user-interfaces; in this report some of those written in the last two decades are united and subsumed to a “common denominator” of UI-rules.

The second part of this report now takes these subsumed rules and tests PrestoSpace-UI's on compliance with these. Annexes on a few non-PrestoSpace-CMS/MAM-User-Interfaces, a short glossary and a table of figures complete the report.

3 “Golden Rules”

3.1 ISO-Standards

3.1.1 ISO 9241-10 to 17 (incl. replacements)

Since the current ISO-standards on usability-aspects of software, including the for this paper relevant parts on user-interfaces, are currently rebuild part by part (e.g. 9241-110 replacing 9241-10) the author decided to only go into the general propositions and contents of the relevant parts in this ISO-standard.

3.1.1.1 ISO 9241-110 (replacing 9241-10) Dialogue principles

This part of ISO 9241 presents a set of usability heuristics that applies to the interaction of people and information systems (the heuristics are based on an earlier German standard). The standard refers to this interaction as a “dialogue” and describes seven “dialogue principles”. These general principles span the specific dialogue techniques that are discussed in parts 13-17 of ISO 9241. The seven principles are: suitability for the task (the dialogue should be suitable for the user’s task and skill level); self-descriptiveness (the dialogue should make it clear what the user should do next); controllability (the user should be able to control the pace and sequence of the interaction); conformity with user expectations (it should be consistent); error tolerance (the dialogue should be forgiving); suitability for individualisation (the dialogue should be able to be customised to suit the user); and suitability for learning (the dialogue should support learning). The standard describes applications and examples of the dialogue principles. It has one annex (a short bibliography). This part of ISO 9241 used to be known as ISO 9241 part 10, but has now been renumbered under ISO's revision and restructuring programme.

3.1.1.2 ISO 9241-11 – Guidance on Usability

This part of ISO 9241 introduces the concept of usability but does not make specific recommendations in terms of product attributes. Instead it defines usability as the:

“Extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.”

One of the benefits of this approach is that it helps design teams plan for usability as part of the development lifecycle, by setting and measuring usability goals for the product. The standard describes how it can be applied to: specify and measure the usability of products; specify and evaluate usability during design; and specify and measure a work system in use. The standard includes 5 annexes: an example of how to specify the context of use; examples of usability measures; an example of a usability requirements specification; relationship to other international standards; and a bibliography.

3.1.1.3 ISO 9241-12 – Presentation of Information

This part of ISO 9241 contains recommendations on how to present visual information on screens so that users can easily perform “perceptual tasks” (such as searching for information on the screen). The recommendations are based on seven guiding principles: clarity (information should be conveyed quickly and accurately); discriminability (information should be able to be distinguished accurately); conciseness (provide only the information necessary to complete the task); consistency (present the same information in the same way throughout the application); detectability (direct the user’s attention to the information required); legibility (information should be easy to read); and comprehensibility (the meaning should be clearly understandable). The recommendations are provided in three main areas: organisation of information; graphical objects; and coding techniques. There is no discussion of icon design in this standard. The standard has two annexes. The first is a sample procedure for assessing applicability and adherence and includes a detailed design checklist spanning 13 pages. This compliance procedure is based on the notion of “conditional compliance”: the assessor first determines which recommendations are relevant and then determines whether or not they have been adhered to. The reason for this approach is that users, tasks and technological solutions vary and it is therefore not possible to give blanket recommendations that apply to all systems that present visual information. The conditional compliance route is ISO’s acknowledgement of this variability and complexity. The second annex is a bibliography.

3.1.1.4 ISO 9241-13 – User Guidance

This part of ISO 9241 contains recommendations on user guidance. The recommendations cover general advice; prompts; feedback; status information; error management; and on-line help. This standard does not cover documentation (either on-line or paper) or on-line tutorials. The standard contains two annexes. The first is a sample procedure for assessing applicability and adherence and includes a six-page checklist (using a conditional compliance route, see the description of ISO 9241-12). The second annex is a bibliography.

3.1.1.5 ISO 9241-14 – Menu Dialogues

This part of ISO 9241 provides recommendations for the design of systems that use menus (such as pop-up, pull-down and text-based menus). The standard begins by reviewing the application areas where menus are most useful (for example, when use of the system is infrequent and the user does not know what options are available). The recommendations cover: menu structure (such as logical categories, grouping options and ordering items); menu navigation (including titles and access time); option selection and execution (including selection methods, use of the keyboard and voice activation); menu presentation (including placement and use of icons). The standard contains three annexes. The first is a sample procedure for assessing applicability and adherence and includes a ten-page checklist (using a conditional compliance route, see the description of ISO 9241-12). The second annex provides three scenarios of how to apply the standard (from the perspective of the user interface designer, the procurer and the evaluator). The third annex is a detailed bibliography.

3.1.1.6 ISO 9241-15 – Command Dialogues

This part of ISO 9241 provides recommendations for systems that use command line interfaces (such as DOS and UNIX). With a command line interface, the user works with the system by typing in commands that meet certain syntactic rules. The standard begins by describing the appropriate application areas for these interfaces (for example, applications that people use frequently, and that require speed and flexibility). The recommendations cover: structure and syntax (for example, macros and command arguments); command representation (for example, command names and abbreviations); input and output considerations (for example, command reuse and editing); and feedback and help (for example, command processing and error feedback).

The standard contains two annexes. The first is a sample procedure for assessing applicability and adherence and includes a five-page checklist (using a conditional compliance route, see the description of ISO 9241-12). The second annex is a bibliography.

3.1.1.7 ISO 9241-16 – Direct Manipulation Dialogues

This part of ISO 9241 provides recommendations for systems that use direct manipulation (such as Windows and Macintosh). With direct manipulation, the user acts directly on the objects on the screen (for example by dragging a document icon and dropping it on an application to open it). The standard points out that this is not the same as a graphical user interface, which may or may not implement direct manipulation. The standard begins by describing the appropriate application areas for these interfaces (for example, the system can simulate real-world task objects, their properties and operations). The recommendations cover: general information (metaphors; the appearance of objects used in direct manipulation; feedback; and input devices); manipulation of objects (general considerations; pointing and selecting; dragging; sizing of objects; and rotating); direct manipulation of text objects (pointing and selecting; and sizing of text); direct manipulation of windows (general considerations; pointing and selecting; and sizing of windows); and direct manipulation of control icons (pointing and selecting). The standard contains one annex and a bibliography. The annex is a sample procedure for assessing applicability and adherence and includes a three-page checklist.

3.1.1.8 ISO 9241-17 – Form Filling Dialogues

This part of ISO 9241 provides recommendations for systems that use form filling interfaces (now commonly seen on the web). With form filling (or “fill in the blanks”) interfaces, users see a display of related fields and enter data where required. The standard begins by describing the appropriate application areas for these interfaces (e.g., when users have experience with paper forms but limited experience with computers). The recommendations cover form filling structure, input considerations, feedback and navigation. Form filling structure covers: general; layout; and fields and labels. Input considerations covers: general; alphanumeric text entry; choice entries; control; and field validation. Feedback covers: echoing; cursor and pointer position; field errors; transmission acknowledgement; and database changes. Navigation covers: initial cursor position; movement between fields; return to initial field; tabbing; scrolling; and

form selection. The standard contains two annexes; the first is a sample procedure for assessing applicability and adherence and includes a nine-page checklist; the second annex is a bibliography and contains a table cross-referencing the recommendations in the standard to the academic sources.

3.1.1.9 ISO 9241 – Future Parts to replace also previous described parts

Only excerpts to address the relevant parts:

ISO 9241-111	Presentation principles	Planned to partially revise and replace ISO 9241-12
ISO 9241-112	Multimedia principles	Planned to revise and replace ISO 14915-1
ISO 9241-113	GUI and controls principles	Planned
ISO 9241-121	User guidance	Planned to revise and replace ISO 9241-13:2003
ISO 9241-122	Media selection and combination	Planned to revise and replace ISO 14915-3
ISO 9241-123	Navigation	Planned to partially revise and replace ISO14915-2
ISO 9241-124	User guidance	Planned to revise and replace ISO9241-13
ISO 9241-129	Individualization	Planned
ISO 9241-130	Selection and combination of dialogue techniques	Planned to incorporate and replace ISO 9241-1:1997/Amd 1:2001
ISO 9241-131	Menu dialogues	Planned to replace ISO9241-14
ISO 9241-132	Command dialogues	Planned to replace ISO9241-15
ISO 9241-133	Direct manipulation dialogues	Planned to replace ISO9241-16
ISO 9241-134	Form filling dialogues	Planned to replace ISO9241-17
ISO 9241-135	Natural language dialogues	Planned
ISO 9241-141	Controlling groups of information (including windows)	Planned to partially replace ISO9241-12
ISO 9241-142	Lists	Planned
ISO 9241-143	Media controls	Planned to revise and replace ISO 14915-2
ISO 9241-151	Guidance on World Wide Web software user interfaces	Under preparation
ISO 9241-152	Interpersonal communication	Planned
ISO 9241-153	Virtual reality	Planned
ISO 9241-171	Guidance on software accessibility	Under preparation

ISO 9241-200	Introduction to human-system interaction processes	Planned
ISO 9241-210	Human-centred design of interactive systems	Planned to revise and replace ISO13407
ISO 9241-220	Human-centred lifecycle processes	Planned to revise and replace ISO/PAS 18152
ISO 9241-230	Human-centred design methods	Planned to revise and replace ISO/TR 16982
ISO 9241-241	Templates for evaluation	Planned

3.1.2ISO/TS 16071

This ISO Standard addresses the accessibility of interactive systems. It addresses a wide range of solutions, including office applications, web pages and multimedia. It provides guidance on the design of accessible (work, home, education) software and covers issues associated with designing accessible software for people with the widest range of visual, hearing, motor and cognitive abilities, including those who are elderly and temporarily disabled.

The scope of this ISO standard deals very little with the scope of the User-Interfaces; only the User-Interfaces of the Documentation Platform are tangent to the topic.

3.1.3ISO 14915

“Software ergonomics for multimedia user interfaces” This Standard gives requirements and recommendations for the ergonomic design of multimedia applications mainly intended for professional and vocational activities such as work or learning.

Part 1 is a general introduction to the standard; part 2 provides recommendations for navigation structures and aids, media controls, basic controls, media control guidelines for dynamic media and controls and navigation involving multiple media; part 3 part provides general guidelines for media selection and combination, media selection for information types, media combination and integration and directing users' attention;

finally part 4 is intended to cover computer based training, computer supported co-operative work, kiosk systems, on-line help and testing and evaluation.

ISO 14915 was published in 2000 and was meant to be a “multimedia-addendum” for ISO 9241; various parts of ISO 14915 are currently planned to be revised and replaced by new parts in ISO 9241 (see 3.1.1.9.)

3.1.4 Conclusion

The fast developments and new technical potentialities achieved in all relevant parts covered by the ISO standards 9241, 14915, 16071, a.m.m. initiated a general revision and rebuilding of the mentioned standards by the ISO organization. Especially the new ISO 9241-110, introduced during the development-phase of the PrestoSpace User-Interfaces was the pivotal question for the author to use these standards “only” as a kind of norm to scrutinize the following “Golden Rules” on User-Interfaces from different sources and to evaluate the “Common Denominator” (Par.4).

3.2 The Golden Rules of User Interface Design

3.2.1 Introduction

"The Golden Rules of User Interface Design" has been published by Theo Mandel in 1997; in his publication "The Elements of User Interface Design"[i] he uses the introduction of 3 "Golden Rules" to focus on the very important parts in designing User Interfaces:

- Place users in control of the interface
- Reduce users' memory load
- Make the user interface consistent.

For Mandel User interface design principles are not just relevant to today's graphical user interfaces. He points out, that they have been around for quite some time., pointing out to Hansen, who proposed the first (and perhaps the shortest) list of design principles in his paper, "User Engineering Principles for Interactive Systems."(1971)[ii]. Hansen's principles were:

- Know the user
- Minimize memorization
- Optimize operations
- Engineer for errors.

He also refers to "The Human Factor" by Rubenstein and Hersch (1984)[iii]; this classic book on human-computer interaction presents 93 (!) design principles, ranging from "1. Designers make myths; users make conceptual models." to "93. Videotape real users." and points out, that also all major software OS vendors have published their (own) design guidelines and reference materials.

3.2.2 Mandels "Golden Rules"

3.2.2.1 Golden Rule #1: Place Users in Control

The first set of principles addresses placing users in control of the interface. A simple analogy is whether to let people drive a car or force them to take a train. In a car, users control their own direction, navigation, and final destination. One problem is that drivers need a certain amount of skill and knowledge before they are able to successfully drive a car. Drivers also need to know where they are going! A train forces users to become passengers rather than drivers. People used to driving their own car to their destination may not enjoy the train ride, where they can't control the schedule or the path a train will take to reach the destination. However, novice or casual users may enjoy the train if they don't know exactly where they are going and they don't mind relying on the train to guide and direct them on their journey. The ultimate decision to drive a car or take the train should be the user's, not someone else's. Users also deserve the right to change their mind and take the car one day and the train the next.

The principles that allow users to be in control:

1. Use modes judiciously (modeless)
2. Allow users to use either the keyboard or mouse (flexible)
3. Allow users to change focus (interruptible)
4. Display descriptive messages and text (helpful)
5. Provide immediate and reversible actions, and feedback (forgiving)
6. Provide meaningful paths and exits (navigable)
7. Accommodate users with different skill levels (accessible)
8. Make the user interface transparent (facilitative)
9. Allow users to customize the interface (preferences)
10. Allow users to directly manipulate interface objects (interactive)

3.2.2.2 Golden Rule #2: Reduce Users' Memory Load

Based on what we know about how people store and remember information, the power of the computer interface should help users from having to remember information while using the computer. We aren't good at remembering things, so programs should be designed with this in mind.

Design principles for this rule:

1. Relieve short-term memory (remember)
2. Rely on recognition, not recall (recognition)
3. Provide visual cues (inform)
4. Provide defaults, undo, and redo (forgiving)
5. Provide interface shortcuts (frequency)
6. Promote an object-action syntax (intuitive)
7. Use real-world metaphors (transfer)
8. User progressive disclosure (context)
9. Promote visual clarity (organize)

3.2.2.3 Golden Rule #3: Make the Interface Consistent

User interface consistency is a key aspect of usable interfaces. It's also a major area of debate. However, just like all principles, consistency might be a lower priority than other factors, so don't follow consistency principles and guidelines if they don't make sense in your environment. One of the major benefits of consistency is that users can transfer their knowledge and learning to a new program if it is consistent with other programs they already use. This is the "brass ring" for computer trainers and educators—train users how to do something once, then they can apply that learning in new situations that are consistent with their mental model of how computers work.

The design principles that make up user interface consistency:

1. Sustain the context of users' tasks (continuity)
2. Maintain consistency within and across products (experience)
3. Keep interaction results the same (expectations)
4. Provide aesthetic appeal and integrity (attitude)
5. Encourage exploration (predictable)

3.2.3Conclusion

Mandel's credo can be summarized in an apt quotation by Albert Einstein: "Make it simple, but not simpler". The User should be taken serious, he knows best what he needs, so the Interfaces should support, not instruct.

One important statement by Mandel should not be forgotten: Principles are not meant to be follow blindly, rather they are meant as guiding lights for sensible interface design!

3.3 A Summary of Principles for User-Interface Design

3.3.1 Introduction

This article on the main principles for User-Interface design has been published by David Joiner[iv] in 1998; Joiner, a Web-programmer and Designer from the very beginning of the World Wide Web, built up a compilation of fundamental principles for designing user interfaces, pointing out, that most of these principles can be applied to either command-line or graphical environments. In the following paragraph all of the 14 principles are introduced and explicated:

3.3.2 Principles

3.3.2.1 The principle of user profiling

“Know who your user is”

Before the question "How do we make our user-interfaces better" can be answered, another question has to bee answer before: Better for whom? A design that is better for a technically skilled user might not be better for a non-technical businessman or an artist.

One way around this problem is to create user models. Tognazzini’s “Tog On Interface”[v] has an excellent chapter on brainstorming towards creating "profiles" of possible users. The result of this process is a detailed description of one or more "average" users, with specific details such as:

- What are the user's goals?
- What are the user's skills and experience?
- What are the user's needs?

Then one may proceed to answer the question: How do we leverage the user's strengths and create an interface that helps them achieve their goals?

In the case of a large general-purpose piece of software such as an operating system, there may be many different kinds of potential users. In this case it may be more useful

to come up with a list of user dichotomies, such as "skilled vs. unskilled", "young vs. old", etc., or some other means of specifying a continuum or collection of user types.

Another way of answering this question is to talk to some real users. Direct contact between end-users and developers has often radically transformed the development process.

3.3.2.2 The principle of metaphor

“Borrow behaviours from systems familiar to your users”

Frequently a complex software system can be understood more easily if the user interface is depicted in a way that resembles some commonplace system. The ubiquitous "Desktop metaphor" is an overused and trite example. Another is the tape deck metaphor seen on many audio and video player programs. In addition to the standard transport controls (play, rewind, etc.), the tape deck metaphor can be extended in ways that are quite natural, with functions such as time-counters and cueing buttons. This concept of "extendibility" is what distinguishes a powerful metaphor from a weak one.

There are several factors to consider when using a metaphor:

- Once a metaphor is chosen, it should be spread widely throughout the interface, rather than used once at a specific point. Even better would be to use the same metaphor spread over several applications (the tape transport controls described above is a good example).
- There's no reason why an application cannot incorporate several different metaphors, as long as they don't clash. Music sequencers, for example, often incorporate both "tape transport" and "sheet music" metaphors.
- Metaphor isn't always necessary. In many cases the natural function of the software itself is easier to comprehend than any real-world analog of it. Don't strain a metaphor in adapting it to the program's real function. Nor should the meaning of a particular program feature be strained in order to adapt it to a metaphor.
- Incorporating a metaphor is not without certain risks. In particular, whenever physical objects are represented in a computer system, not only the beneficial functions of those objects are inherited but also the detrimental aspects.

- Be aware that some metaphors don't cross cultural boundaries well. For example, Americans would instantly recognize the common U.S. Mailbox (with a rounded top, a flat bottom, and a little red flag on the side), but there are no mailboxes of this style in Europe.

3.3.2.3 The principle of feature exposure

“Let the user see clearly what functions are available”

Software developers tend to have little difficulty keeping large, complex mental models in their heads. But not everyone prefers to "live in their heads" -- instead, they prefer to concentrate on analyzing the sensory details of the environment, rather than spending large amounts of time refining and perfecting abstract models. Both type of personality (labelled "Intuitive" and "Sensable" in the Myers-Briggs[vi] personality classification) can be equally intelligent, but focus on different aspects of life. It is to be noted that according to some psychological studies "Sensables" outnumber "Intuitives" in the general population by about three to one.

“Intuitives” prefer user interfaces that utilize the power of abstract models (command lines, scripts, plug-ins, macros, etc.) “Sensables” prefer user interfaces that utilize their perceptual abilities (in other words, they like interfaces where the features are "up front" and "in their face". Toolbars and dialog boxes are an example of interfaces that are pleasing to this personality type).

This doesn't mean that you have to make everything a GUI. What it does mean, for both GUI and command line programs, is that the features of the program need to be easily exposed so that a quick visual scan can determine what the program actually does. In some cases, such as a toolbar, the program features are exposed by default. In other cases, such as a printer configuration dialog, the exposure of the underlying printer state (i.e. the buttons and controls which depict the conceptual printing model) are contained in a dialog box which is brought up by a user action (a feature which is itself exposed in a menu).

Of course, there may be cases where you don't wish to expose a feature right away, because you don't want to overwhelm the beginning user with too much detail. In this case, it is best to structure the application like the layers of an onion, where peeling away each layer of skin reveals a layer beneath. There are various levels of "hiding": Here's a partial list of them in order from most exposed to least exposed:

- Toolbar (completely exposed)
- Menu item (exposed by trivial user gesture)
- Submenu item (exposed by somewhat more involved user gesture)
- Dialog box (exposed by explicit user command)
- Secondary dialog box (invoked by button in first dialog box)
- "Advanced user mode" controls -- exposed when user selects "advanced" option
- Scripted functions

The above notwithstanding, in no case should the primary interface of the application be a reflection of the true complexity of the underlying implementation. Instead, both the interface and the implementation should strive to match a simplified conceptual model (in other words, the design) of what the application does. For example, when an error occurs, the explanation of the error should be phrased in a way that relates to the current user-centred activity, and not in terms of the low-level fault that caused there error.

3.3.2.4 The principle of coherence

“The behaviour of the program should be internally and externally consistent”

There's been some argument over whether interfaces should strive to be "intuitive", or whether an intuitive interface is even possible. However, it is certainly arguable that an interface should be coherent, in other words logical, consistent, and easily followed. ("Coherent" literally means "stick together", and that's exactly what the parts of an interface design should do.)

Internal consistency means that the program's behaviours make "sense" with respect to other parts of the program. For example, if one attribute of an object (e.g. colour) is modifiable using a pop-up menu, then it is to be expected that other attributes of the object would also be editable in a similar fashion. One should strive towards the principle of "least surprise".

External consistency means that the program is consistent with the environment in which it runs. This includes consistency with both the operating system and the typical suite of applications that run within that operating system. One of the most widely recognized forms of external coherence is compliance with user-interface standards. There are

many others, however, such as the use of standardized scripting languages, plug-in architectures or configuration methods.

3.3.2.5 The principle of state visualization

“Changes in behaviour should be reflected in the appearance of the program”

Each change in the behaviour of the program should be accompanied by a corresponding change in the appearance of the interface. One of the big criticisms of "modes" in interfaces is that many of the classic "bad example" programs have modes that are visually indistinguishable from one another.

Similarly, when a program changes its appearance, it should be in response to a behaviour change; A program that changes its appearance for no apparent reason will quickly teach the user not to depend on appearances for clues as to the program's state.

One of the most important kinds of state is the current selection, in other words the object or set of objects that will be affected by the next command. It is important that this internal state be visualized in a way that is consistent, clear, and unambiguous. For example, one common mistake seen in a number of multi-document applications is to forget to "dim" the selection when the window goes out of focus. The result of this is that a user, looking at several windows at once, each with a similar-looking selection, may be confused as to exactly which selection will be affected when they hit the "delete" key. This is especially true if the user has been focusing on the selection highlight, and not on the window frame, and consequently has failed to notice which window is the active one. (Selection rules are one of those areas that are covered poorly by most UI style guidelines, which tend to concentrate on "widgets", although the Mac®[vii] and Amiga®[viii] guidelines each have a chapter on this topic.)

3.3.2.6 The principle of shortcuts

“Provide both concrete and abstract ways of getting a task done”

Once a user has become experienced with an application, she will start to build a mental model of that application. She will be able to predict with high accuracy what the results of any particular user gesture will be in any given context. At this point, the program's attempts to make things "easy" by breaking up complex actions into simple steps may seem cumbersome. Additionally, as this mental model grows, there will be less and less

need to look at the "in your face" exposure of the application's feature set. Instead, pre-memorized "shortcuts" should be available to allow rapid access to more powerful functions.

There are various levels of shortcuts, each one more abstract than its predecessor. For example, in the *emacs* editor[ix] commands can be invoked directly by name, by menu bar, by a modified keystroke combination, or by a single keystroke. Each of these is more "accelerated" than its predecessor.

There can also be alternate methods of invoking commands that are designed to increase power rather than to accelerate speed. A "recordable macro" facility is one of these, as is a regular-expression search and replace. The important thing about these more powerful (and more abstract) methods is that they should not be the most exposed methods of accomplishing the task. This is why emacs has the non-regexp version of search assigned to the easy-to-remember "C-s" key.

3.3.2.7 The principle of focus

“Some aspects of the UI attract attention more than others do”

The human eye is a highly non-linear device. For example, it possesses edge-detection hardware, which is why we see Mach bands whenever two closely matched areas of colour come into contact. It also has motion-detection hardware. As a consequence, our eyes are drawn to animated areas of the display more readily than static areas. Changes to these areas will be noticed readily.

The mouse cursor is probably the most intensely observed object on the screen -- it's not only a moving object, but mouse users quickly acquire the habit of tracking it with their eyes in order to navigate. This is why global state changes are often signalled by changes to the appearance of the cursor, such as the well-known "hourglass cursor". It's nearly impossible to miss.

The text cursor is another example of a highly eye-attractive object. Changing its appearance can signal a number of different and useful state changes.

3.3.2.8 The principle of grammar

“A user interface is a kind of language -- know what the rules are”

Many of the operations within a user interface require both a subject (an object to be operated upon), and a verb (an operation to perform on the object). This naturally suggests that actions in the user interface form a kind of grammar. The grammatical metaphor can be extended quite a bit, and there are elements of some programs that can be clearly identified as adverbs, adjectives and such.

The two most common grammars are known as "Action→Object" and "Object→Action". In Action→Object, the operation (or tool) is selected first. When a subsequent object is chosen, the tool immediately operates upon the object. The selection of the tool persists from one operation to the next, so that many objects can be operated on one by one without having to re-select the tool. Action→Object is also known as "modality", because the tool selection is a "mode" which changes the operation of the program. An example of this style is a paint program -- a tool such as a paintbrush or eraser is selected, which can then make many brush strokes before a new tool is selected.

In the Object→Action case, the object is selected first and persists from one operation to the next. Individual actions are then chosen which operate on the currently selected object or objects. This is the method seen in most word processors -- first a range of text is selected, and then a text style such as bold, italic, or a font change can be selected. Object→Action has been called "non-modal" because all behaviours that can be applied to the object are always available. One powerful type of Object→Action is called "direct manipulation", where the object itself is a kind of tool -- an example is dragging the object to a new position or resizing it.

3.3.2.9 The principle of help

“Understand the different kinds of help a user needs”

In Laurel’s “The Art of Human Computer Interface Design”[x] it states that are five basic types of help, corresponding to the five basic questions that users ask:

1. Goal-oriented: "What kinds of things can I do with this program?"
2. Descriptive: "What is this? What does this do?"
3. Procedural: "How do I do this?"
4. Interpretive: "Why did this happen?"
5. Navigational: "Where am I?"

The essay goes on to describe in detail the different strategies for answering these questions, and shows how each of these questions requires a different sort of help interface in order for the user to be able to adequately phrase the question to the application.

For example, "about boxes" are one way of addressing the needs of question of type 1. Questions of type 2 can be answered with a standard "help browser", "tool tips" or other kinds of context-sensitive help. A help browser can also be useful in responding to questions of the third type, but these can sometimes be more efficiently addressed using "cue cards", interactive "guides", or "wizards" which guide the user through the process step-by-step. The fourth type has not been well addressed in current applications, although well-written error messages can help. The fifth type can be answered by proper overall interface design, or by creating an application "roadmap". None of the solutions listed in this paragraph are final or ideal; they are simply the ones in common use by many applications today.

3.3.2.10The principle of safety

“Let the user develop confidence by providing a safety net”

Ted Nelson once said "Using a DOS-PC is like juggling with straight razors. Using a Mac is like shaving with a bowling pin."[xi]

Each human mind has an "envelope of risk", that is to say a minimum and maximum range of risk-levels which they find comfortable. A person who finds herself in a situation that is too risky for her comfort will generally take steps to reduce that risk. Conversely, when a person's life becomes too safe -- in other words, when the risk level drops below the minimum threshold of the risk envelope -- she will often engage in actions that increase their level of risk.

This comfort envelope varies for different people and in different situations. In the case of computer interfaces, a level of risk that is comfortable for a novice user might make a "power-user" feel uncomfortably swaddled in safety.

It's important for new users that they feel safe. They don't trust themselves or their skills to do the right thing. Many novice users think poorly not only of their technical skills, but of their intellectual capabilities in general (witness the popularity of the "...for Dummies" series of tutorial books.) In many cases these fears are groundless, but they need to be

addressed. Novice users need to be assured that they will be protected from their own lack of skill. A program with no safety net will make this type of user feel uncomfortable or frustrated to the point that they may cease using the program. The "Are you sure?" dialog box and multi-level undo features are vital for this type of user.

At the same time, an expert user must be able to use the program as a virtuoso. She must not be hampered by guard rails or helmet laws. However, expert users are also smart enough to turn off the safety checks -- if the application allows it. This is why "safety level" is one of the more important application configuration options.

3.3.2.11The principle of context

“Limit user activity to one well-defined context unless there's a good reason not to”

Each user action takes place within a given context -- the current document, the current selection, the current dialog box. A set of operations that is valid in one context may not be valid in another. Even within a single document, there may be multiple levels -- for example, in a structured drawing application, selecting a text object (which can be moved or resized) is generally considered a different state from selecting an individual character within that text object.

It's usually a good idea to avoid mixing these levels. For example, imagine an application that allows users to select a range of text characters within a document, and also allows them to select one or more whole documents (the latter being a distinct concept from selecting all of the characters in a document). In such a case, it's probably best if the program disallows selecting both characters and documents in the same selection. One unobtrusive way to do this is to "dim" the selection that is not applicable in the current context. In the example above, if the user had a range of text selected, and then selected a document, the range of selected characters could become dim, indicating that the selection was not currently pertinent. The exact solution chosen will of course depend on the nature of the application and the relationship between the contexts.

Another thing to keep in mind is the relationship between contexts. For example, it is often the case that the user is working in a particular task-space, when suddenly a dialog box will pop up asking the user for confirmation of an action. This sudden shift of context may leave the user wondering how the new context relates to the old. This confusion is exacerbated by the terseness of writing style that is common amongst application writers. Rather than the "Are you sure?" confirmation mentioned earlier, something like

"There are two documents unsaved. Do you want to quit anyway?" would help to keep the user anchored in their current context.

3.3.2.12The principle of aesthetics

“Create a program of beauty”

It's not necessary that each program be a visual work of art. But it's important that it not be ugly. There are a number of simple principles of graphical design that can easily be learned, the most basic of which was coined by artist and science fiction writer William Rotsler[xii]: "Never do anything that looks to someone else like a mistake."

An interface example can be seen in the placement of buttons -- imagine five buttons, each with five different labels that are almost the same size. Because the buttons are packed using an automated-layout algorithm, each button is almost but not exactly the same size. As a result, though the author has placed much care into his layout, it looks carelessly done. A solution would be to have the packing algorithm know that buttons that are almost the same size look better if they are exactly the same size -- in other words, to encode some of the rules of graphical design into the layout algorithm. Similar arguments hold for manual widget layout.

Another area of aesthetics to consider is the temporal dimension. Users don't like using programs that feel sluggish or slow. There are many tricks that can be used to make a slow program "feel" snappy, such as the use of off-screen bitmaps for rendering, which can then be blitted forward in a single operation.

3.3.2.13The principle of user testing

“Recruit help in spotting the inevitable defects in your design”

In many cases a good software designer can spot fundamental defects in a user interface. However, there are many kinds of defects which are not so easy to spot, and in fact an experienced software designer is often less capable of spotting them than the average person. In other cases, a bug can only be detected while watching someone else use the program.

User-interface testing, that is, the testing of user-interfaces using actual end-users, has been shown to be an extraordinarily effective technique for discovering design defects.

However, there are specific techniques that can be used to maximize the effectiveness of end-user testing. These can be summarized in the following steps:

- Set up the observation. Design realistic tasks for the users, and then recruit end-users that have the same experience level as users of your product (Avoid recruiting users who are familiar with your product however).
- Describe to the user the purpose of the observation. Let them know that you're testing the product, not them, and that they can quit at any time. Make sure that they understand if anything bad happens, it's not their fault, and that it's helping you to find problems.
- Talk about and demonstrate the equipment in the room.
- Explain how to "think aloud". Ask them to verbalize what they are thinking about as they use the product, and let them know you'll remind them to do so if they forget.
- Explain that you will not provide help.
- Describe the tasks and introduce the product.
- Ask if there are any questions before you start; then begin the observation.
- Conclude the observation. Tell them what you found out and answer any of their questions.
- Use the results.

User testing can occur at any time during the project, however, it's often more efficient to build a mock-up or prototype of the application and test that before building the real program. It's much easier to deal with a design defect before it's implemented than after. Tognazzini[5] suggests that you need no more than three people per design iteration -- any more than that and you are just confirming problems already found.

3.3.2.14The principle of humility

“Listen to what ordinary people have to say”

Some of the most valuable insights can be gained by simply watching other people attempt to use your program. Others can come from listening to their opinions about the product. Of course, you don't have to do exactly everything they say. It's important to

realize that each of you, user and developer, has only part of the picture. The ideal is to take a lot of user opinions, plus your insights as a developer and reduce them into an elegant and seamless whole -- a design which, though it may not satisfy everyone, will satisfy the greatest needs of the greatest number of people.

One must be true to one's vision. A product built entirely from customer feedback is doomed to mediocrity, because what users want most are the features that they cannot anticipate.

But a single designer's intuition about what is good and bad in an application is insufficient. Program creators are a small, and not terribly representative, subset of the general computing population.

Some things designers should keep in mind about their users:

- Most people have a biased idea as to the what the "average" person is like. This is because most of our interpersonal relationships are in some way self-selected. It's a rare person whose daily life brings them into contact with other people from a full range of personality types and backgrounds. As a result, we tend to think that others think "mostly like we do." Designers are no exception.
- Most people have some sort of core competency, and can be expected to perform well within that domain.
- The skill of using a computer (also known as "computer literacy") is actually much harder than it appears.
- The lack of "computer literacy" is not an indication of a lack of basic intelligence. While native intelligence does contribute to one's ability to use a computer effectively, there are other factors which seem to be just as significant, such as a love of exploring complex systems, and an attitude of playful experimentation. Much of the fluency with computer interfaces derives from play -- and those who have dedicated themselves to "serious" tasks such as running a business, curing disease, or helping victims of tragedy may lack the time or patience to be able to devote effort to it.
- A high proportion of programmers are introverts, compared to the general population. This doesn't mean that they don't like people, but rather that there are specific social situations that make them uncomfortable. Many of them lack social

skills, and retreat into the world of logic and programming as an escape; As a result, they are not experienced people-watchers.

The best way to avoid misconceptions about users is to spend some time with them, especially while they are actually using a computer. Do this long enough, and eventually you will get a "feel" for how the average non-technical person thinks. This will increase your ability to spot defects, although it will never make it 100%, and will never be a substitute for user-testing.

3.3.3Conclusion

Although Joiner’s paper and summary lacks an academic approach, his findings and results, although coming sometimes “right from the guts”, are easy to follow and plausible. By putting the user and his needs in the centre of his reflections, he ends up with similar results such as Mandel, although starting from an different approach. Joiner’s principles specify and go more into detail than Mandels rules, but no mayor discrepancies can be found, so Joiners approach can be used to refine Mandels statements.

3.4 Eight Golden Rules of Dialog Design

3.4.1 Introduction

Ben Shneiderman, one of the pioneers in the development of user-interfaces, proposed this collection of rules[xiii], which is deduced heuristically from experience and applicable in most interactive systems after properly refined, extended and interpreted. The eight rules are introduced in the following paragraph; the descriptions have been shortened, as some approaches are quite similar or even congruent to the previous ones.

3.4.2 Rules

3.4.2.1 Strive for consistency.

Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent commands should be employed throughout.

3.4.2.2 Enable frequent users to use shortcuts

As the frequency of use increases, so do the user's desires to reduce the number of interactions and to increase the pace of interaction. Abbreviations, function keys, hidden commands, and macro facilities are very helpful to an expert user.

3.4.2.3 Offer informative feedback.

For every operator action, there should be some system feedback. For frequent and minor actions, the response can be modest, while for infrequent and major actions, the response should be more substantial.

3.4.2.4 Design dialog to yield closure

Sequences of actions should be organized into groups with a beginning, middle, and end. The informative feedback at the completion of a group of actions gives the

operators the satisfaction of accomplishment, a sense of relief, the signal to drop contingency plans and options from their minds, and an indication that the way is clear to prepare for the next group of actions.

3.4.2.5 Offer simple error handling

As much as possible, design the system so the user cannot make a serious error. If an error is made, the system should be able to detect the error and offer simple, comprehensible mechanisms for handling the error.

3.4.2.6 Permit easy reversal of actions

This feature relieves anxiety, since the user knows that errors can be undone; it thus encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data entry, or a complete group of actions.

3.4.2.7 Support internal locus of control

Experienced operators strongly desire the sense that they are in charge of the system and that the system responds to their actions. Design the system to make users the initiators of actions rather than the responders.

3.4.2.8 Reduce short-term memory load

The limitation of human information processing in short-term memory requires that displays be kept simple, multiple page displays be consolidated, window-motion frequency be reduced, and sufficient training time be allotted for codes, mnemonics, and sequences of actions.

3.4.3Conclusion

From an academically or research point of view Shneiderman’s book and his “Golden Rules” are maybe more important or valuable than Joiner’s “Principles”, as the results came out of long lasting and proper research instead of simple and selective “observations”. But matching the results and investigating the methods, both are quite similar, using the same approach and ending up with comparable findings.

3.5 General Principles of User Interface Design

3.5.1 Introduction

In the frequently cited book “Principles and Guidelines in Software User Interface Design”[xiv], Deborah J. Mayhew has proposed a set of design goals, the general characteristics that any interface should have. These principles cover almost all aspects of user interface design. The results of her research are once again similar to the previous paragraphs, so again only a short description is provided for each principle.

3.5.2 Elements

3.5.2.1 User compatibility

Perhaps the most fundamental principle, from which all other derive, is to know the user. Designer should be familiar with cognitive psychology, or an understanding of the general strengths and weaknesses of the human mind so as to make the design acceptable to most users, not only a few of them.

3.5.2.2 Product compatibility

Often the intended user of a new system is already a user of other systems, who has already invested a great deal of time and/or money in learning the existing systems. The across-product compatibility is important since it allows the user to adapt to a new system quickly.

3.5.2.3 Task compatibility

The structure and flow of a system should match and support the task that is being carried out. The user doesn't need to navigate back and forth between applications in order to complete a task.

3.5.2.4 Work flow compatibility

A system should be organized to facilitate transitions between tasks. For example, a windowed, multitasking system can support the nature of this type of user's work more effectively.

3.5.2.5 Consistency

Consistency refers to similarities within a product, rather than across products. It allows people to reason by analogy and predict how to do things they have never done before.

3.5.2.6 Familiarity

Concepts, terminology, and spatial arrangements that the user is already familiar with can be incorporated into the interface.

3.5.2.7 Simplicity

Don't try to provide all the functionality that any user could possibly ever want or need. Instead, make the interface relatively simple.

3.5.2.8 Direct manipulation.

A direct manipulation interface is one in which users directly perform actions on visible objects. This is in contrast to interfaces in which users specify actions, parameters, and objects indirectly through language.

3.5.2.9 Control

User prefer to feel a sense of mastery and control over any tool at their disposal, and the computer is no exception. The designer should be sensitive to this and present a tool-like interface.

3.5.2.10WYSIWYG

Try to make what you see on the screen is what you get on printed output or stored files.

3.5.2.11Flexibility

Allow more user control and accommodates variations in user skill and preferences.

3.5.2.12Responsiveness

Give users feedback as soon as possible and let users be aware of the progress.

3.5.2.13Invisible technology

Hide the technology from users. Only present the functionality that users need to know.

3.5.2.14Robustness

A system should tolerate common and unavoidable human error. System crashes should be minimized, and simple to understand and execute recovery measures should be presented. A robust system encourages users to learn new features and thus increases productivity.

3.5.2.15Protection

People make errors, especially when they are working quickly or under pressure. Users should be protected against the catastrophic results of common human error. "Undo" or other recovery measures should also be provided.

3.5.2.16Ease of learning and ease of use

System should be both easy to learn for the novice and efficient and easy to use for the expert.

3.5.3Conclusion

Mayhew's principles are like Joiner's more refined in their conclusion and meet with very "practical" and real-life-situations; but also here in her collection only a few new approaches can be spotted, like principle no. 13 demanding an "invisible technology". But even that approach can be seen in Mandel's rule no.2, where he postulates real-life metaphors. An other well-known terminus is introduced for the first time in these collection of rules: WYSIWYG, but others like Familiarity and Robustness point out even more striking what good user-interface design should be aiming for.

3.6 Design Principles for Tomorrow

3.6.1 Introduction

The following collection of principles is not from an single author or scientist but from an major player on the IT-market: IBM has presented a collection of design principles[xv], which is derived from traditional design principles with extensions to address evolving aspects of future interfaces, based on their design experience in creating an object-oriented user interface. Like publications by other vendors and competitors[7] [8], IBM uses their experience to publish general guidelines on designing user-interfaces.

3.6.2 Elements

3.6.2.1 Simplicity: Don't compromise usability for function

Keep the interface simple and straightforward. Users don't benefit from function that is not easily accessible and usable. A poorly organized interface cluttered with many advanced functions distracts users from accomplishing their everyday tasks. Basic functions should be immediately apparent, while advanced functions may initially be less obvious to new users. Function should be included only if required based on task analysis. Therefore, keep the number of different objects and actions to a minimum while allowing users to accomplish their tasks.

3.6.2.2 Support: User is in control with proactive assistance

Allow users to be in control of the interface. Don't limit users by artificially restricting their choices to your notion of the "correct" sequence of steps needed to accomplish a task. For instance, provide users with alternative courses of action appropriate to their way of thinking.

Allow users to establish and maintain a working context, or frame of reference, from within which they can perform actions. The current state of the system, and possible user actions, should be obvious. Users should be able to leave their systems for a moment or

a day and find the system in the same familiar state when they return. This contextual framework contributes to the feeling of stability.

3.6.2.3 Familiarity: Build on users' prior knowledge

Allow users to build on prior knowledge, especially knowledge they have gained from experience in the real world. A small amount of knowledge, used consistently throughout an interface, can empower the user to accomplish a large number of tasks. Concepts and techniques can be learned once and then applied in a variety of situations. Users should not have to learn new things to perform familiar tasks. The use of concepts and techniques that users already understand from their real world experiences allows them to get started quickly and make progress immediately.

Avoid the tendency to employ consistency without understanding your users, their tasks, and shared experiences. When choosing a dimension within which to be consistent, seek to understand what the user expects and be consistent with those expectations. Providing a familiar experience is the ultimate use of consistency in which a truly intuitive interface will result.

3.6.2.4 Obviousness: Make objects and their controls visible and intuitive

Where practical, use real-world representations in the interface. Real-world representations and natural interactions (direct action) give the interface a familiar look and feel and can make it more intuitive to learn and use. Icons and windows were early attempts to draw on user experiences outside the computing domain. As we move toward real-world representations, reliance on such computer artefacts should decline.

The controls of the system should be clearly visible and their functions identifiable. Visual representations provide cues and reminders that help users understand roles, remember relationships, and recognize what the computer is doing. For example, the numbered buttons on the telephone object indicate that they can be used to dial a telephone number.

3.6.2.5 Encouragement: Make actions predictable and reversible

User actions should cause the results the user expects. In order to meet those expectations, the designer must understand the users' tasks, goals, and mental model.

Use terms and images that match users' task experience, and that help users understand the objects and their roles and relationships in accomplishing tasks.

Users should feel confident in exploring, knowing they can try an action, view the result, and undo the action if the result is unacceptable. Users feel more comfortable with interfaces in which their actions do not cause irreversible consequences.

3.6.2.6 Satisfaction: Create a feeling of progress and achievement

Allow the user to make uninterrupted progress and create a sense of accomplishment. Reflect the results of actions immediately; any delay intrudes on users' tasks and erodes confidence in the system. This allows users to assess whether the results were what was expected and allows them to take alternative action immediately. For example, when a user chooses a new font, the font of all applicable text, or of sample text, should change immediately. The user can then decide if the effect is what was desired, and if not, can change it before switching attention to something else.

3.6.2.7 Accessibility: Make all objects accessible at all times

Users should be able to use all of their objects in any sequence and at any time. Avoid the use of modes, those states of the interface in which normally available actions are no longer available, or in which an action causes different results than it normally does.

3.6.2.8 Safety: Keep the user out of trouble

Users should be protected from making errors. The burden of keeping the user out of trouble is on the designer. The interface should provide visual cues, reminders, lists of choices, and other aids, either automatically or on request, especially since humans are much better at recognition than recall. Contextual and hover help, as well as agents, can provide supplemental assistance. Simply stated, eliminate the opportunity for user error and confusion.

3.6.2.9 Versatility: Support alternate interaction techniques

Allow users to choose the method of interaction that is most appropriate to their situation. Interfaces that are flexible in this regard are able to accommodate a wide range of user skills, interactions, and usage environments.

Each interaction device is optimized for certain uses or users and may be more convenient in different situations. For example, a microphone used with voice-recognition software can be helpful for fast entry of text or in a hands-free environment. Pen input is helpful for people who sketch and mouse input works well for precise indication in selection. Alternate output formats such as computer-generated voice output for foreign language instruction are useful. No single method is best for every situation.

3.6.2.10Personalization: Allow users to customize

The interface should be tailor able to individual users' needs and desires. No two users are exactly alike. Users have varying backgrounds, interests, motivations, and levels of experience. Customization can help make an interface feel comfortable and familiar.

Personalizing a computer interface can also lead to higher productivity and user satisfaction. For example, allowing users to change default values can save them time and hassle when accessing frequently used functions.

3.6.2.11Affinity: Bring objects to life through good visual design

The goal of visual design in the user interface is to surface to the user in a cohesive manner all aspects of the design principles. Visual design should support the user model and communicate the function of that model without ambiguities. Visual design should not be the "icing on the cake" but an integral part of the design process. The final result should be an intuitive and familiar representation that is second nature to users.

3.6.3Conclusion

In IBM’s Design Principles for Tomorrow the same results or rules are used as in some older collections like Mayhew’s, Shneiderman or Joiner’s. The paper by “Big Blue” comes to the same solutions and principles like the previous described rule-collections. Although the various principles by IBM and their explanation seem sometimes like a well-done mix of already existing knowledge and findings, the durability of this collection of principles in the specialised press and literature seem to prove their validity.

3.7 Effective User Interface Design: The Four Rules

3.7.1 Introduction

Kevin Kruse[xvi] published this collection of 4 basic rules for user-interface design in 2000 as part of a training-seminar on web-design for e-learning purposes. Kruse, principal of RKA, a provider of technology-based training solution and author of several papers on Design and Technology-based Training, exemplifies by using paradigms from the learning and e-learning area his collection of rules. A very practical part of his paper is the concluding GUI evaluation checklist.

3.7.2 Elements

3.7.2.1 Help them remember.

Following are several design objectives to keep in mind to help learners navigate an information-packed site.

Chunking information and organizing menu structure

Using what we know about short- and long-term memory, we can apply the following strategies to maximize the effectiveness of a program's menu system. A menu should ideally have no more than seven items on it. If it does have more than seven, determine whether it can be split logically into a higher-level menu and a submenu. This helps learners remember which menus contain certain items.

The order or placement of menu items should match the structure of the tasks and subtasks. For example, a main menu for a sales training program might list the classic four-step sales call in chronological order:

- Lesson 1: Building Rapport with Customers
- Lesson 2: Presenting Product Information
- Lesson 3: Handling Objections

- Lesson 4: Closing the Deal.

In turn, when the learner selects Lesson 3, a submenu might be sequenced along the consecutive steps of Classifying Objections, Responding to Objections, and Confirming Satisfaction. If there is no sequence associated with menu items, place the most commonly used options at the top of the menu and least-used items on the bottom.

Submenus should have titles that reflect the selected option from the previous menu. For example, the submenu described above for "Handling Objections" should maintain those words as the submenu's title. The cleanest way to handle nesting menus is to use expanding menus where the learner never loses sight of the original menu structure.

Using mental models or visual metaphors

A mental model or visual metaphor is the internal picture we create to help us understand how things work. Though we're not conscious of our mental models, they help us to use computers effectively. Designers use visual metaphors to take advantage of what we already know when helping us understand something new.

A good example of a visual metaphor is the directory structure of a computer. While the computer actually stores files and data haphazardly on its hard drive, the visual metaphor presented to the user is that of file folders and a vertical ordering system. This metaphor gives an artificial but clear sense of order to the system. As users we imagine that our documents are being held in these little folders, and that there is some kind of depth to them.

Mental models and metaphors, however, are still subject to short-term memory restrictions. Most users begin to get lost when their model contains more than three layers or paths. Imagine a training program that has a main menu (the first level) from which students gain access to a specific lesson (the second level), and eventually click on a More Information hyperlink, which displays some additional text (the third level). At this point, most users will still have a clear understanding of where they are in the program, and how to retrace their steps, if necessary. But if they are again presented a link for more information, such as a case study from within the More Information section, they will begin to lose track of their location or the relationship of the onscreen content to the overall lesson.

Using multiple access points

A simple way to relieve the burden on users' memory is to provide multiple ways to locate and access the content. Common methods are described below.

- Main menu. The primary access point is always the program's main menu, which should be well organized and descriptive. Rather than using generic names, such as Lesson 1 and Lesson 2, use descriptive headings such as "1: Overview to Customer Service" and "2: Dealing with Difficult Customers."
- Index. An index of key topics or all learning objectives helps users find specific information. A well-indexed system will enhance any training program's subsequent use as a just-in-time support tool.
- Keyword search. The keyword search enables users to type a word and have the program scan the entire text for all occurrences. While a powerful feature, a keyword search only looks at onscreen text and cannot identify information presented as audio narration or in pictures.
- Site map or content map. A visual representation of the order of the topics in the program, or content outline, is called a site map. Typically, it displays the entire menu system graphically, extending down to individual learning objectives.

3.7.2.2 Put the user in control.

An effective interface puts users in control of the program, or at the very least, lets them feel like they're in control. By giving users control, you ease their anxieties, minimize their confusion, and create an environment that's conducive to learning. Descriptions of a number of time-tested ways of putting the user in the driver's seat via status or warning messages follow.

- Loading delays. If a computer is busy for longer than three seconds, the program should display what's called a status message. If the delay is due to bandwidth limitations, there is little you can do to forewarn the user. However, if you're using streaming media or Flash animations, you should anticipate the preloading delay and provide a warning message of some type. This message alleviates users'

concerns that the computer may have "frozen." Though the message in itself doesn't provide control to the user, the communication helps them to feel that they're still in control.

- Taking a test. Final exams are often timed and intentionally prevent learners from leaving the test module until they're finished with the test. This kind of program control is designed to keep users from looking for correct answers in the lessons. Before the test is started, a confirmation message should appear that advises the student, "You are about to begin the test. Once you start this test you will have to finish it in one sitting. You will not be able to take the test again. Are you sure you are ready to take this test now?" Action buttons should be clearly labelled Take Test or Return to Main Menu.
- Previous page. Perhaps the most obvious undo feature is the Previous page button or Back button in a linear tutorial. In addition to giving learners the control to move forward in a program, an effective interface also enables them to move back to a previous page.

3.7.2.3 Use consistent and logical designs.

Users can quickly learn a new visual metaphor or a new mental model, but they also quickly create expectations that the interface they see will be consistent. A program's interface is the door between the student and the instruction. In order to facilitate access and reduce confusion, consistency in interface appearance and behaviour are paramount.

Clear and logical screen layouts

An intuitive interface begins with the overall layout and design of the screen. Four principles of screen layout include the following:

- Place screen objects together in a logical order.
- Place buttons where the user's eye can easily find them.
- Give buttons clear symbols or labels.
- Group buttons together based on their function and frequency of use.

Web-based training programs often have navigation bars running vertically down the left side of the screen, or horizontally across the top of the screen. This is because scrolling text windows are a common user interface element on the Internet, making horizontal buttons on the bottom of the screen--which are standard on CD-ROM programs--impractical. Buttons should always be given clear labels, with both text and graphical indicators, if possible. Common button-naming rules include

- Use Menu to label the button that accesses the main menu. Don't use the ambiguous Main.
- Use Help to access navigational guidance. Don't use Hint or Panic.
- Use Exit to end the program. Don't use Quit, End, or Stop, which might refer to quitting the immediate exercise or lesson.
- Use Forward or Next and Back or Previous to designate page turning. Don't use Up or Down.
- Use complete screen counters, such as "1 of 30," not partial counters, such as "Page 5," that don't indicate how much longer the lesson will last.
- If your program runs on Windows, refer to the keyboard Enter key as Enter, not as Return.

Consistency in visual cues

Early seafaring explorers used celestial navigation to make their way across the high seas. Like the North Star, the buttons your students use to navigate should never change in their appearance or location on the screen. Button identification is a fundamental part of a mental model. Changing a button's location or appearance will cause users to think they're seeing a new button with new functions.

Menus that behave predictably

Menus are the key structures for organizing and accessing information and must be planned with great care. In addition to logical sequencing and having no more than three layers of menus (see Organizing Menu Structure above), the menu action must be consistent throughout the program. When a user clicks on a menu item, similar actions

must occur for each item on the menu. Main menu items that are clicked can lead to submenus, or the buttons can directly launch a lesson or simulation. But don't mix the two actions on the same menu. For example, if clicking "Module 1: The Cardiovascular System" launches a submenu, but clicking "Module 2: The Nervous System" launches a 30-screen linear tutorial, students can get confused. They may think "Oops, where is that submenu? Did I accidentally click something to launch this tutorial?" or "What's going on? Will I get to the submenu after this tutorial?"

3.7.2.4 Provide informative guidance and feedback.

Web-based training has significantly transformed training, replacing many traditional classroom sessions. But students of all ages are still students and perform best when given guidance and feedback. Just as in personal relations, politeness and courtesy should be extended in all technology-based training situations.

Page counters

Every linear tutorial should have an onscreen page counter that tells users which screen or page they're on and how many more are in the lesson. A simple message such as "Screen 5 of 25" clearly describes what's required to finish the lesson in the program and engenders learner confidence. With self-paced programs that can be taken at any time, this type of progress marker helps users answer such questions as, "I have a meeting in 15 minutes--can I finish this lesson or should I quit now?"

Some designers recommend using time estimates rather than page counters. For example, "Lesson 1: Overview (10 to 15 minutes)." However, estimating the time needed for self-paced training is difficult. Be aware that although a range is given, some learners may feel anxiety from the implied time limitation.

3.7.2.5 GUI evaluation checklist

When reviewing and evaluating the computer interface of your technology-based training program, you should be able to answer yes to the questions below. This checklist is suitable for CD-ROM and online learning programs.

- Do all buttons and icons have a consistent and unique appearance?
- Are visual cues, such as mouse cursor changes and rollover highlights, used on all buttons consistently?
- Are buttons labelled with text descriptions (or with rollover text)?
- Do buttons “gray out” or disappear when they are inactive?
- Do “nonbutton” graphics have design properties distinct from that of buttons?
- Are navigation buttons displayed in exactly the same screen position every time they appear?
- Are buttons grouped logically and located where the user is likely to be looking?
- Do users have one-click access to help, exit, and the main menu?
- Are users returned to where they left off after closing the help window and cancelling the exit screen?
- Does every menu have a title?
- Does every menu screen include an option to return to the previous or main menu?
- Are there fewer than three levels of menus?
- Do menus have seven or fewer items on them?
- Are items on menus descriptive rather than general?
- Are menu items listed in a sequential or logical order?
- Do menus indicate which items a learner has completed?
- Are confirmation messages used in such areas as student registration, exit, and final exams?
- Are there clear instructions associated with menus, questions, and other tasks?
- Are error messages written in plain language?
- Are status messages displayed during delays greater than four seconds?
- Are exclamation points and sound effects used sparingly?
- Is there a bookmarking feature that enables users to exit and resume later where they left off?
- Can users move backward and forward in linear tutorials?

- Are page or screen counters used to show progress in linear lessons?
- Is the visual metaphor consistent and intuitive in nonlinear simulations?
- Are all pop-up windows positioned on the screen so they don't cover up relevant information?
- Does text appear clearly and with normal margins and spacing?
- Do information input screens force all capital letters, and is input evaluation case insensitive?
- Can users interact with the program from either the keyboard or the mouse?
- Are text fonts used consistently?
- Are audio volume levels consistent?
- Do users have the option to replay video or audio narration?

3.7.3Conclusion

Kruse's paper has some similarities with Joiner's, offering a more practical than academic approach to the topic; nevertheless the findings and results are comparable to those in the more research-oriented and on the first glance more precise ones. But especially the real-life examples and the easy-to-use evaluation-checklist provide good knowledge and tools for the task of designing and/or evaluating user-interfaces.

3.8 Ergonomic Guidelines for User-Interface Design

3.8.1 Introduction

The following points are guidelines to good software interface design based on those contained in “Developing User Interfaces: Ensuring usability through product and process”[xvii]. Due to the fact, that these guidelines aren’t explicitly carried out as a list of rules, the author tried to form and group them in such a way. The guidelines themselves are now arranged in 11 thematically blocks like previous collections.

3.8.2 Guidelines

3.8.2.1 Consistency ("Principle of least astonishment")

- certain aspects of an interface should behave in consistent ways at all times for all screens
- terminology should be consistent between screens
- icons should be consistent between screens
- colours should be consistent between screens of similar function

3.8.2.2 Simplicity

- break complex tasks into simpler tasks
- break long sequences into separate steps
- keep tasks easy by using icons, words etc.
- use icons/objects that are familiar to the user

3.8.2.3 Human Memory Limitations

- organize information into a small number of “chunks”
- try to create short linear sequences of tasks

- don't flash important information onto the screen for brief time periods
- organize data fields to match user expectations, or to organize user input (e.g. auto-formatting phone numbers)
- provide cues/navigation aids for the user to know where they are in the software or at what stage they are in an operation
- provide reminders, or warnings as appropriate
- provide ongoing feedback on what is and/or just has happened
- let users recognize rather than recall information
- minimize working memory loads by limiting the length of sequences and quantity of information - avoid icon mania!

3.8.2.4 Cognitive Directness

- minimize mental transformations of information (e.g. using 'control+shift+esc+8' to indent a paragraph)
- use meaningful icons/letters
- use appropriate visual cues, such as direction arrows
- use 'real-world' metaphors whenever possible (e.g. desktop metaphor, folder metaphor, trash can metaphor etc.)

3.8.2.5 Feedback

- provide informative feedback at the appropriate points
- provide appropriate articulatory feedback - feedback that confirms the physical operation you just did (e.g. typed 'help' and 'help' appear on the screen). This includes all forms of feedback, such as auditory feedback (e.g. system beeps, mouse click, key clicks etc.)
- provide appropriate semantic feedback - feedback that confirms the intention of an action (e.g. highlighting an item being chosen from a list)

- provide appropriate status indicators to show the user the progress with a lengthy operation (e.g. the copy bar when copying files, an hour glass icon when a process is being executed etc.)

3.8.2.6 System messages

- provide user-centred wording in messages (e.g. "there was a problem in copying the file to your disk" rather than "execution error 159")
- avoid ambiguous messages (e.g. hit 'any' key to continue - there is no 'any' key and there's no need to hit a key, reword to say 'press the return key to continue)
- avoid using threatening or alarming messages (e.g. fatal error, run aborted, kill job, catastrophic error)
- use specific, constructive words in error messages (e.g. avoid general messages such as 'invalid entry' and use specifics such as 'please enter your name')
- make the system 'take the blame' for errors (e.g. "illegal command" versus "unrecognized command")

3.8.2.7 Anthropomorphization

- don't anthropomorphize (i.e. don't attribute human characteristics to objects) - avoid the "Have a nice day" messages from your computer

3.8.2.8 Modality

- use modes cautiously - a mode is an interface state where what the user does has different actions than in other states (e.g. changing the shape of the cursor can indicate whether the user is in an editing mode or a browsing mode)
- minimize pre-emptive modes, especially irreversible pre-emptive modes - a pre-emptive mode is one where the user must complete one task before proceeding to the next. In a pre-emptive mode other software functions are inaccessible (e.g. file save dialog boxes)
- make user actions easily reversible - use 'undo' commands, but use these sparingly
- allow escape routes from operations

3.8.2.9 Attention

- use attention grabbing techniques cautiously (e.g. avoid overusing 'blinks' on web pages, flashing messages, 'you have mail', bold colours etc.)
- don't use more than 4 different font sizes per screen
- use serif or sans serif fonts appropriately as the visual task situation demands
- don't use all uppercase letters - use and uppercase/lowercase mix
- don't overuse audio or video
- use colours appropriately and make use of expectations (e.g. don't have an OK button coloured red! use green for OK, yellow for 'caution, and red for 'danger' or 'stop')
- don't use more than 4 different colours on a screen
- don't use blue for text (hard to read), blue is a good background colour
- don't put red text on a blue background
- use high contrast colour combinations
- use colours consistently
- use only 2 levels of intensity on a single screen
- use underlining, bold, inverse video or other markers sparingly
- on text screens don't use more than 3 fonts on a single screen

3.8.2.10 Display issues

- maintain display inertia - make sure the screen changes little from one screen to the next within a functional task situation
- organize screen complexity
- eliminate unnecessary information
- use concise, unambiguous wording for instructions and messages
- use easy to recognize icons

- use a balanced screen layout - don't put too much information at the top of the screen - try to balance information in each screen quadrant
- use plenty of 'white space' around text blocks - use at least 50% white space for text screens
- group information logically
- structure the information rather than just presenting a narrative format (comprehension can be 40% faster for a structured format)

3.8.2.11 Individual differences

- accommodate individual differences in user experience (from the novice to the computer literate)
- accommodate user preferences by allowing some degree of customization of screen layout, appearance, icons etc.
- allow alternative forms for commands (e.g. key combinations through menu selections)

3.8.3 Conclusion

Focusing on rather detailed, but very practical and easy to follow basics of the designing of user-interfaces, this collection introduces a thematically new part to the set of rules and principles; by claiming the prevention of Anthropomorphisation, the author gives rejection to a too aggressive humanisation of the computer. But in total this collection provide no completely new or conflicting results or conclusions.

3.9 Ten Usability Heuristics

3.9.1 Introduction

The “Ten Usability Heuristics” are a collection of ten general principles for user interface design by Jacob Nielsen[xviii]. They are called "heuristics" because they are more in the nature of rules of thumb than specific usability guidelines. Nielsen, best known by his uncompromising critics on usability in the Web[xix] and nicknamed “king of usability” (by the “Internet Magazine”[xx]) or “World’s leading expert on user-friendly design” (by Stuttgarter Zeitung[xxi]), resumes the results and findings of over 20 years of researching usability-design in software and the Web in 10 easy to comprehend rules. Originally these heuristics have been developed in collaboration with Rolf Molich in 1990[xxii], but constantly revised and currently republished via the Web.

3.9.2 Elements

3.9.2.1 Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

3.9.2.2 Match between system and the real world

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

3.9.2.3 User control and freedom

Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

3.9.2.4 Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

3.9.2.5 Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

3.9.2.6 Recognition rather than recall

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

3.9.2.7 Flexibility and efficiency of use

Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

3.9.2.8 Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

3.9.2.9 Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

3.9.2.10 Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to

search, focused on the user's task, list concrete steps to be carried out, and not be too large.

3.9.3Conclusion

The author decided to use Nielsen’s heuristics as the backbone and starting-point for the mapping-attempt of the different rule- and principle-collections because of their accuracy, simplicity and great diffusiveness in research and designing circles.

3.10Golden Rules for BAD User Interfaces

3.10.1Introduction

This first part of the report ends with a complete different collection of rules, a collection of rules for BAD user interface design. As it is very often quite helpful to address a certain task or challenge from the opposite side and therefore gain a new sight of the problems, the author decided to incorporate this collection in the report to get a kind of “control group” or “advocatus diaboli” to control and monitor the results of the mapping and the testing of the PrestoSpace User Interfaces.

The following list of rules for bad user interfaces has been collected and published by Gerd Waloszek (SAP Product Design Center) in 2007 for the SAP Design Guild Website[xxiii]; in consequence to the topic he also starts bottom up…

3.10.2Rules

3.10.2.1Golden Rule Nr.15: Make it illogical

Reasoning: We all know that illogical bits and pieces in the user interface are no serious barriers for professional users, but at least a couple of beginners will get stuck...

Example: Label a button that will only prepare an operation so that users believe that it will already do the operation. Here is a real-word example: In many e-mail applications, the Forward button does not actually forward an e-mail but prepares it only for forwarding (because, for example, the recipient has still to be provided).

3.10.2.2Golden Rule Nr.14: Do not let users interrupt time-consuming and/or resource-hungry processes

Reasoning: Making processes that put the computer into agony more or less uninterruptible ensures that users take their mandatory coffee breaks.

Example: Start a backup or indexing process while users are not aware of it. Make this process hard to cancel, that is, let it ignore the users' mouse clicks and key presses.

3.10.2.3Golden Rule Nr.13: Leave out functionality that would make the users' life easier – Let them do it the hard (cumbersome) way

Reasoning: Additional functionality would provide users with too many choices and might confuse them.

Example: When users want to add items to a list, allow them to add items at the end of the list only and let them then move the items to the correct position. That is, do not offer additional functionality for inserting items at their target locations. To add some spice, introduce spurious errors that return items to the bottom when users have already moved them half-way up.

Example: Do not offer the option of selecting multiple items, for example, for moving or deleting items. The option of working on one single item suffices to let users achieve their goals – apart from that it may take a little bit longer...

Example: After inserting a set of new items (for example, by command, drag-and-drop or copy-and-paste) don't show them as selected. This would help users to recognize where in the list the items were sorted in. To detect the items that were just inserted will consume quite some time, besides the pure recall of which items were inserted.

3.10.2.4Golden Rule Nr.12: Destroy the work context after each system reaction

Reasoning: Destroying the work context allows users to reflect their work and ask themselves whether it really makes sense.

Example: Deselect selected screen elements after a system reaction.

Example: Move HTML pages or tables that have been scrolled down by the user to the top after a system reaction (e.g. a round trip); in the case of multiple pages (e.g. in hit lists or document list) return the user to the first page.

3.10.2.5Golden Rule Nr.11: Take great care in setting bad defaults: Contrary to the users' expectations, disastrous, annoying, useless, …

Reasoning: Bad defaults are a nice way to surprise users, be it immediately or – at best, unexpectedly – anytime.

Example: Set default options in Web forms so that users get unwanted newsletters or offers, have their addresses distributed, etc.

Example: Set the default option in dialog boxes on the most dangerous option, for example, on deleting a file or format the hard drive.

Example: In forms, set dates (or other data) on useless default values. For example, set the date for applying for a vacation on the current day.

3.10.2.6Golden Rule Nr.10: Spread the message of bad examples!

Reasoning: Examples are a perfect teaching method. But as we all know, bad examples are the best – they allure most.

Example: Just follow any of the other golden rules on this page, that's a perfect start.

Example: If you have to make presentations make sure that you include your bad examples in the presentations.

Note: Good examples are hard to find and typically criticized until nobody appreciates them anymore. Why waste time with unproductive discussions?

3.10.2.7Golden Rule Nr.9: Keep away from end users!

Reasoning: You are the expert and know what users need – because you know what you need. Why should they need something else?

Example: If you think that a certain functionality is not needed don't implement it – why should other people need it?

Example: Many end users have many opinions, you have one. That's far easier and faster to implement.

Note: Doing without site visits saves your company a lot of time and money.

3.10.2.8Golden Rule Nr.8: Make using your application a real challenge!

Reasoning: This teaches people to take more risks, which is important particularly in economically harder times.

Example: Do not offer an Undo function.

Example: Do not warn users if actions can have severe consequences.

Note: If you want to top this and make using your application like playing Russian roulette, change the names of important functions, such as “Save” and “Delete”, temporarily from time to time...

3.10.2.9 Golden Rule Nr.7: Make your application mouse-only – do not offer any keyboard shortcuts

Reason 1: This will make your application completely inaccessible to visually impaired users. Therefore, you can leave out all the other accessibility stuff as well. That will save you a lot of development time.

Reason 2: This will drive many experts crazy who used to accelerate their work with keyboard shortcuts. Now, they will have more empathy for beginners because they are thrown back to their speed.

3.10.2.10 Golden Rule Nr.6: Hide important and often-used functionality from the users’ view

Reasoning: This strategy stimulates users to explore your application and learn a lot about it.

Example: Place buttons for important functions off-screen so that users have to scroll in order to access them.

Example: Hide important functions in menus where users would never expect them.

3.10.2.11 Golden Rule Nr.5: Educate users in technical language

Reasoning: Lifelong learning is hip. As many of us spend a lot of their time at the computer, it's the ideal stage for learning. Moreover, sociologists bemoan that people's vocabulary is more and more reducing. Applications with a challenging vocabulary can go against this trend.

Example: Always send URLs as UTF-8 (requires restart) (advanced settings in MS Internet Explorer)

3.10.2.12Golden Rule Nr.4: Use abbreviations wherever possible, particularly where there would space enough for the complete term

Reasoning: Abbreviations make an application look more professional, particularly if you create abbreviations that are new or replace commonly used ones.

Example: Use abbreviations for field labels, column headings, button texts even if space restrictions do not require this.

Examples: Use "dat." instead of "date," "TolKy" instead of "Tolerance Key," "NxOb" instead of "Next Object," and many more...

3.10.2.13Golden Rule Nr.3: Make it slow!

Example: There are nearly unlimited possibilities of making software slow. For example, you can include long lasting checks or roundtrips after each user input. Or you can force users through long chains of dialog boxes.

3.10.2.14Golden Rule Nr.2: Don’t obey standards!

Example: Do not use standard screen elements for a given purpose, such as single selection (e.g. use checkboxes instead of radiobuttons because they look nicer).

Example: Do not place menu items into the categories and locations they typically belong to (e.g. place "Save" in the "Edit Menu").

3.10.2.15Golden Rule Nr.1: Keep the users busy doing unnecessary work!

Example: It's a "good" habit to let users enter data that the system already knows and could provide beforehand.

Example: Let users enter data into fields only to tell them afterwards that they cannot enter data there (e.g. an application lets you enter data on holidays.

3.10.3Conclusion

Although meant as an entertaining essay and not as a practical or even academic paper, those “bad”-rules for user-interfaced design opens the eyes of an evaluator to nuisances in software and interface-design, that are often overseen by professionals and trained persons. For example Rule Nr.4 and 5: in nearly every user-interface you stumble over an unknown abbreviation or terminus technicus…

Therefore these set of rules was also used to control, verify and refine the results achieved by the professional rule-sets by reviewing the PrestoSpace User Interfaces; sometimes they even helped to reveal previously hidden design-bugs and –flaws.

4 MAPPING / “Common Denominator”

4.1 Introduction

Using Nielsen’s Ten Usability Heuristics (3.9.) as starting point, the following list shows the results of an attempt to map all the previous introduced rules, principles and heuristics with the goal to find a kind of “common denominator”, that can be used to review the PrestoSpace user interfaces and their design.

4.2 Mapping List

Nielsen	Mandel	Joiner	Shneiderman	Mayhew	IBM	Kruse	Hix	SUMMARY
Visibility of system status	Reduce Users' Memory Load	The principle of state visualization	Offer informative feedback	Responsiveness	Satisfaction: Create a feeling of progress and achievement	Provide informative guidance and feedback	Feedback	Status Overview & Feedback
Match between system and the real world	Reduce Users' Memory Load	The principle of metaphor	Reduce short-term memory load	Familiarity WYSIWYG	Familiarity: Build on users' prior knowledge	Help them remember	Cognitive Directness	Familiarity & Real World Metaphors
User control and freedom	Place Users in Control		Support internal locus of control	Control	Accessibility: Make all objects accessible at all times Personalization: Allow users to customize	Put the user in control		The user is in control
Consistency and standards	Make the Interface Consistent	The principle of coherence	Strive for consistency	Consistency		Use consistent and logical designs	Consistency	Consistency
Error prevention	Make the Interface Consistent			Protection	Safety: Keep the user out of trouble	Use consistent and logical designs	Human Memory Limitations	Error prevention
Recognition rather than recall	Reduce Users' Memory Load	The principle of focus	Reduce short-term memory load			Help them remember	Attention	Reduction of User's memory Load
Flexibility and efficiency of use	Place Users in Control	The principle of context		Flexibility		Put the user in control		Flexibility
Aesthetic and minimalist design	Make the Interface Consistent	The principle of aesthetics		Simplicity	Simplicity: Don't compromise usability for function	Use consistent and logical designs	Simplicity	Simplicity and Aesthetics
Help users recognize, diagnose, and recover from errors	Reduce Users' Memory Load	The principle of safety	Offer simple error handling Permit easy reversal of action	Robustness	Encouragement: Make actions predictable and reversible	Provide informative guidance and feedback	Modality	Robustness & Safety by easy error handling
Help and documentation	Make the Interface Consistent	The principle of help			Support: User is in control with proactive assistance	Provide informative guidance and feedback	System messages	Help providing
		The principle of feature exposure						
		The principle of shortcuts	Enable frequent users to use shortcut					
		The principle of grammar						
		The principle of user testing						
		The principle of humility						
			Design dialog to yield closure					
		The principle of		User			Individual	User profiling

		user profiling		compatibility			differences	
				Product compatibility				
				Task compatibility				
				Work flow compatibility				
				Direct manipulation				
				Invisible technology				
				Ease of learning and ease of use				
					Obviousness: Make objects and their controls visible and intuitive			
					Versatility: Support alternate interaction techniques			
					Affinity: Bring objects to life through good visual design			
							Anthropomorphization	
							Display issues	

4.3 Conclusion / Results

Although some of the rules, principles and recommendations from the various collections aren't easy to be mapped directly, the general direction and conclusion of all models and incorporated rules and principles shows clearly an agreement on the following topics:

- The user is (to be put) in control: The user and his task have to be the constitutive elements in every step of a software and the incorporated user-interfaces. A highest possible flexibility should provide support even in uncommon situations and tasks for the user.

  →UIP1

- Consistency: If a user-interface should support and not constrict the user and his task, it has to be consistence and following standards.

  →UIP2

- Reduce users memory load: As computers have a much more "perfect" memory than humans, all short- and medium-time memory-load should be carried by the machine.

  →UIP3 This leads directly to:

- Familiarity & Real World Metaphors: The more the user feels like being "at home", the more efficient and positive-minded the user will work on his task. The keyword of an "Invisible technology" by Mayhem (3.5.) may fit in here as well.

  →UIP4

- Status Overview & Feedback: Provide precise and constant feedback on the status of the program/task, including not only subset-based information, but an complete overview as well.

  →UIP5

- Error Prevention: The user-Interface should prevent errors by the user by guiding him through error-prone and complicated passages; but nevertheless the best solution would be to prevent those passages completely.

→UIP6

- Robustness & Safety by easy error handling: Always provide to the user the comforting sureness of foolproof mechanisms to repair and recover errors.

  →UIP7

- Help and Documentation: Even the most intuitive user-interface incorporates parts, where for some users and certain tasks and situations proper documentation and help provided is necessary.

  →UIP8

- Simplicity and Aesthetics: Omit and avoid “unnecessary” and “confusing” functionalities, elements and information. Keep the user focused and don’t distract him by improper design and dispensable features.

  →UIP9

5 Interfaces for Content Retrieval and Browsing in PrestoSpace

5.1 Introduction

In the IST-project PrestoSpace several User-Interfaces for different purposes had to be developed and built. The technical developers tried, in close contact with the user-partners in the project-consortium, to meet all requirements and specifications made by the underlying workflows and tasks as well as incorporating all recommendations and proposals made by the advising user-partners.

In the following paragraphs of this report the current user-interfaces for content retrieval and browsing (Publication Platform and Annotation GAMP) developed under the above mentioned specifications will be checked and tested using the (during the last paragraph) aggregated principles.

5.2 Publication Platform / Content Retrieval

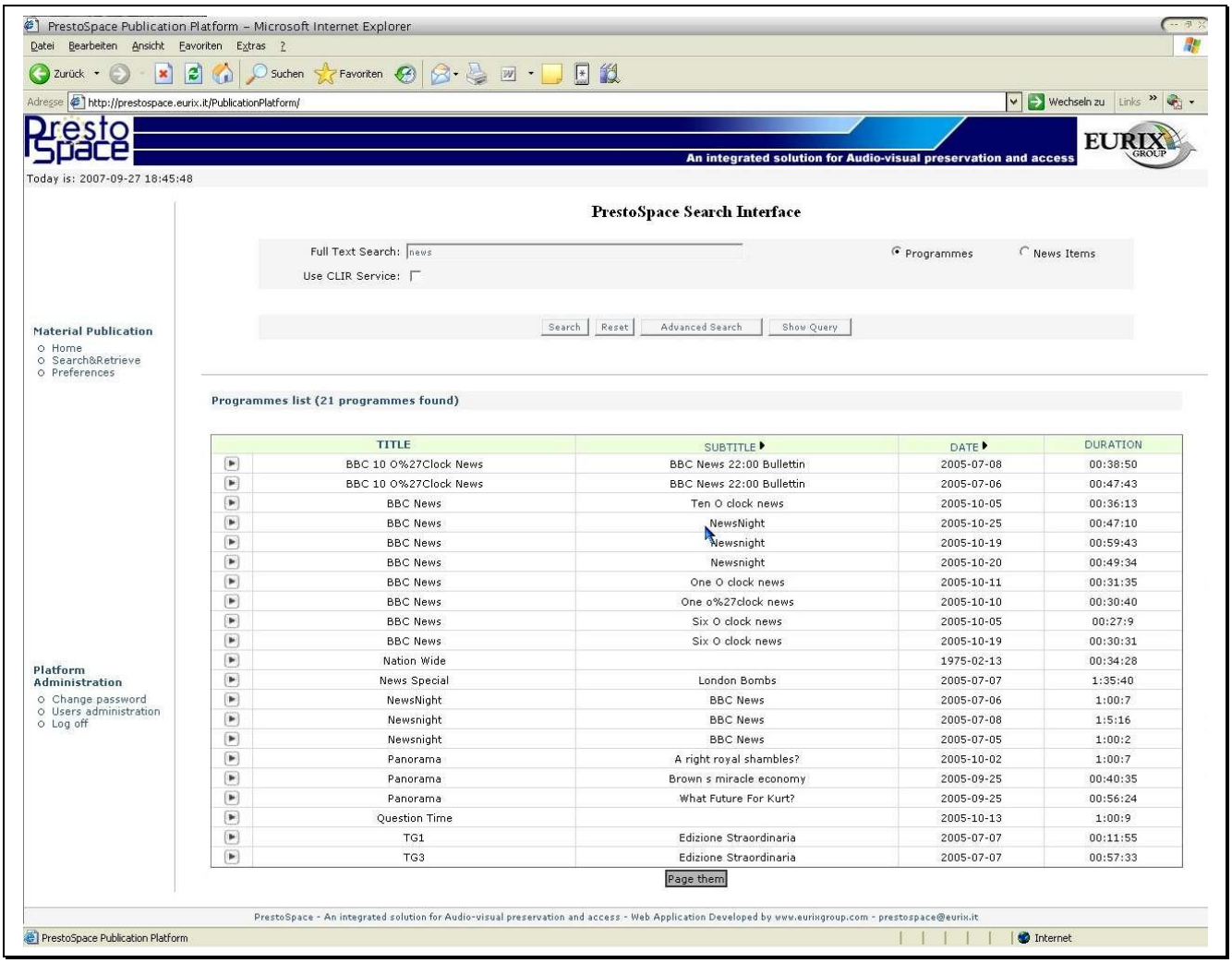


Figure 1 – Publication Platform (Search Interface)

This is the main Search-Interface for the Documentation-Platform (including a first result-list); the screenshot shows the “Simple Search” status of the interface.

5.2.1 Description

The main search-interface is the key-entry-point to all electronic data-collections; up to 50% of an overall acceptance of a CMS is directly deducible from the acceptance and quality of the search-interfaces.

The main search-interface of MAD's Documentation-Platform is intentional plain and simple, especially at the first call (figure 2); only the main controls and functionalities are presented, the focus is on the (full text) search field: the user is "invited" to start right off, without any "preparing actions" to be made, thus lowering the inhibition threshold dramatically.

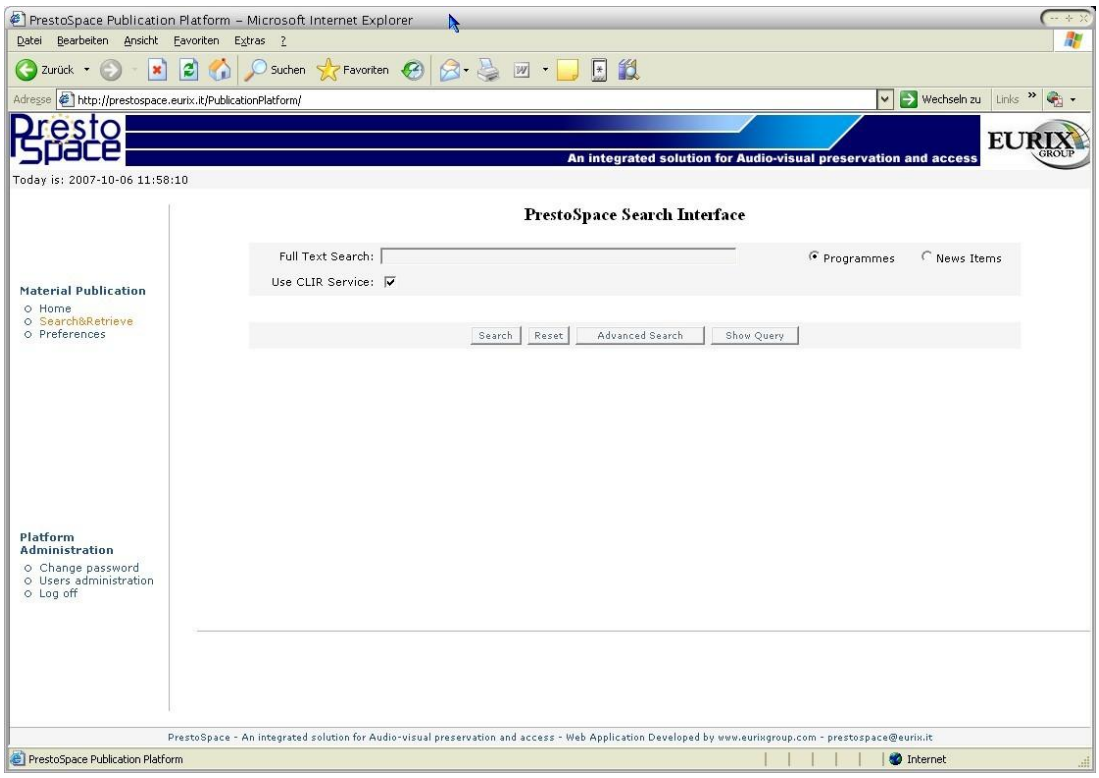


Figure 2 – Simple Search

On the left hand panel, two small lists of parameters lead the user to infrequently used functionalities (preferences and other administrational tasks), in parallel providing feedback on the current "whereabout" by highlighting the list-item showing the current page (see figure 2: "Search&Retrieve").

The use of Tick-boxes for selecting different data-parts and/or functionalities is also a convenient method to ease the use of the search-interface; but there is also a fact to

complain: the use of the abbreviation “CLIR” may be significant enough for professionals, but surely is a cause for confusion for the naïve user.

The four buttons under the search-field are articulative and coherent; using the “Advanced Search” one, the more experienced user can set up his own, task-oriented set of search-terms, including Boolean expressions (figure 3).

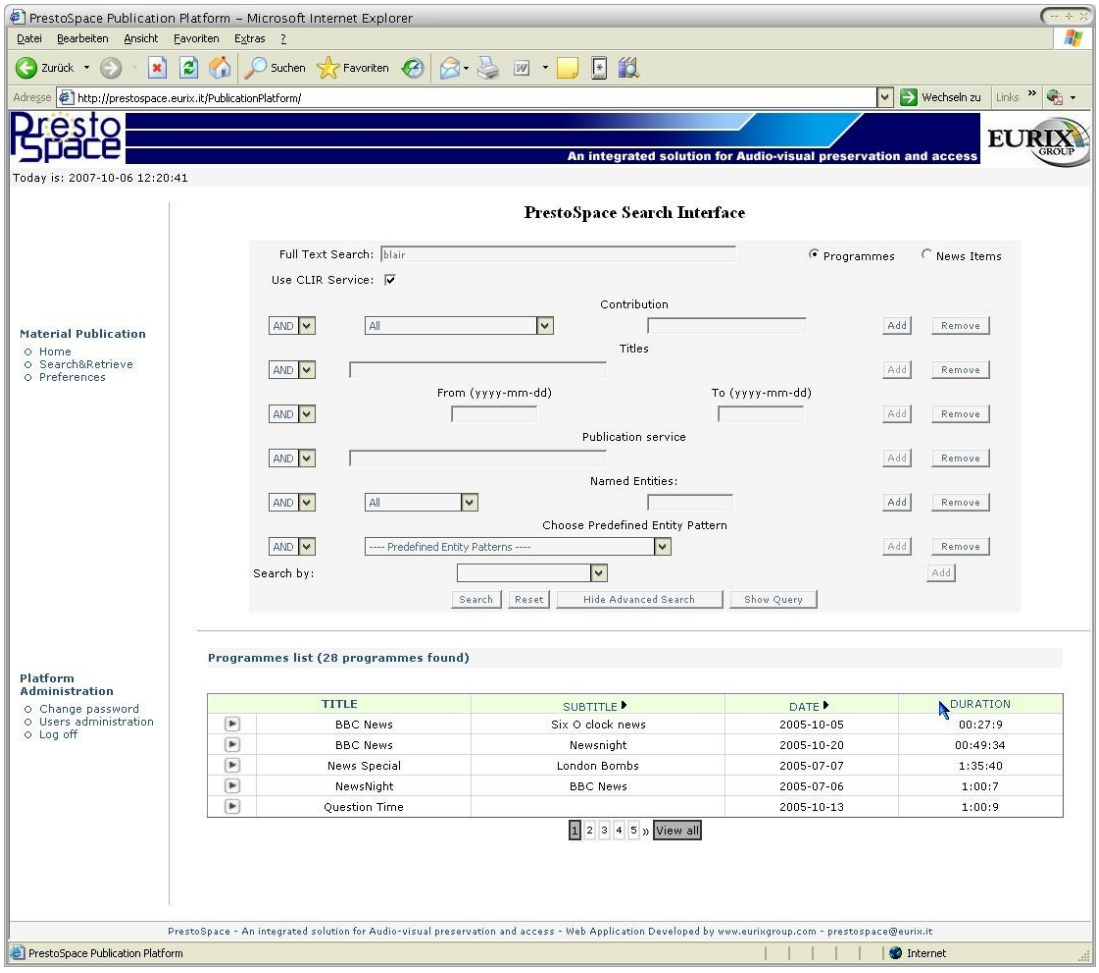


Figure 3 – Advanced Search

The list of results can be expanded either via the “preferences”, choosing there the number of results shown in the list, or can be expanded completely by pressing the “View all” button at the end of the list.

Via a simple “Click” on one of the items shown in the list, the corresponding result-subpage (single item) is shown; by choosing the “Play”-button on the very left of the list, the corresponding preview-item is shown.

For full documentation please refer to PrestoSpace-D18.2[xxiv].

5.2.2Checking of the User-Interface-Principles UIP

5.2.2.1 UIP1: fulfilled

The User is in control; he can choose either the “naïve user” approach as shown in figure 1, with a simple full text search, avoiding the confusion of the user by asking for too much input, the user can’t give (yet), or a fully customizable, advanced search for professional users as shown in figure 3. The whole layout is flexible to different screen-resolutions as well as to different user-needs: e.g. the user can choose, how many results per page he wants to get shown in the result-lists.

5.2.2.2 UIP2: fulfilled

The Interface follows the common standards in interface-design; the user finds all items on familiar positions and conditions.

5.2.2.3 UIP3: fulfilled

The User can use the “preferences” to build up “his” version of the interface, supporting “his” personal workflow and preferences. The interface provides feedback on the status and the position (the professional user can even recall the “technical” implementation of the query by using the “Show Query” button).

5.2.2.4 UIP4: fulfilled

As said in paragraph 5.2.2.2, the user finds everything on familiar positions and conditions, buttons and control-elements are common. The “Play”-button in the result-list can be seen as a real life metaphor.

5.2.2.5 UIP5: fulfilled

Most feedbacks are given, but the interface is not verbose or tedious. Also “technical” feedback is recallable for professional users.

5.2.2.6 UIP6: (nearly) fulfilled

By providing a simple and plain design, starting with a single search-field, most “error-provoking” miss-configurations by naïve users will be prevented; only the technical-term “CLIR” may lead to maloperations.

5.2.2.7 UIP7: fulfilled

The immediate disposability of the results, the easy and quick to handle controls and the possibility to do a quick “re-search” will help the user in situations of unexpected results. The possibility for the professional user to recall the “technical” query done by services like CLIR will support the user in “debugging” his query. By using Web-services the robustness of the system should be given under all common situations.

5.2.2.8 UIP8: only partly fulfilled

There is documentation available, but no online-version incorporated into the interface; especially support and help for more “special” controls like “CLIR”, the “composition” of the “Advanced Search”, etc. would be necessary for non-trained users.

5.2.2.9 UIP9: fulfilled

This interface is a good example for the reaching of the goal “Keep it simple and the user focused”; all elements and controls are strictly task-oriented, clear and simple. No gimmicks or other distractions are incorporated. The user is lead directly to the main focus of this interface, the searching.

5.3 EDOB-Viewer (Browsing-Interface)

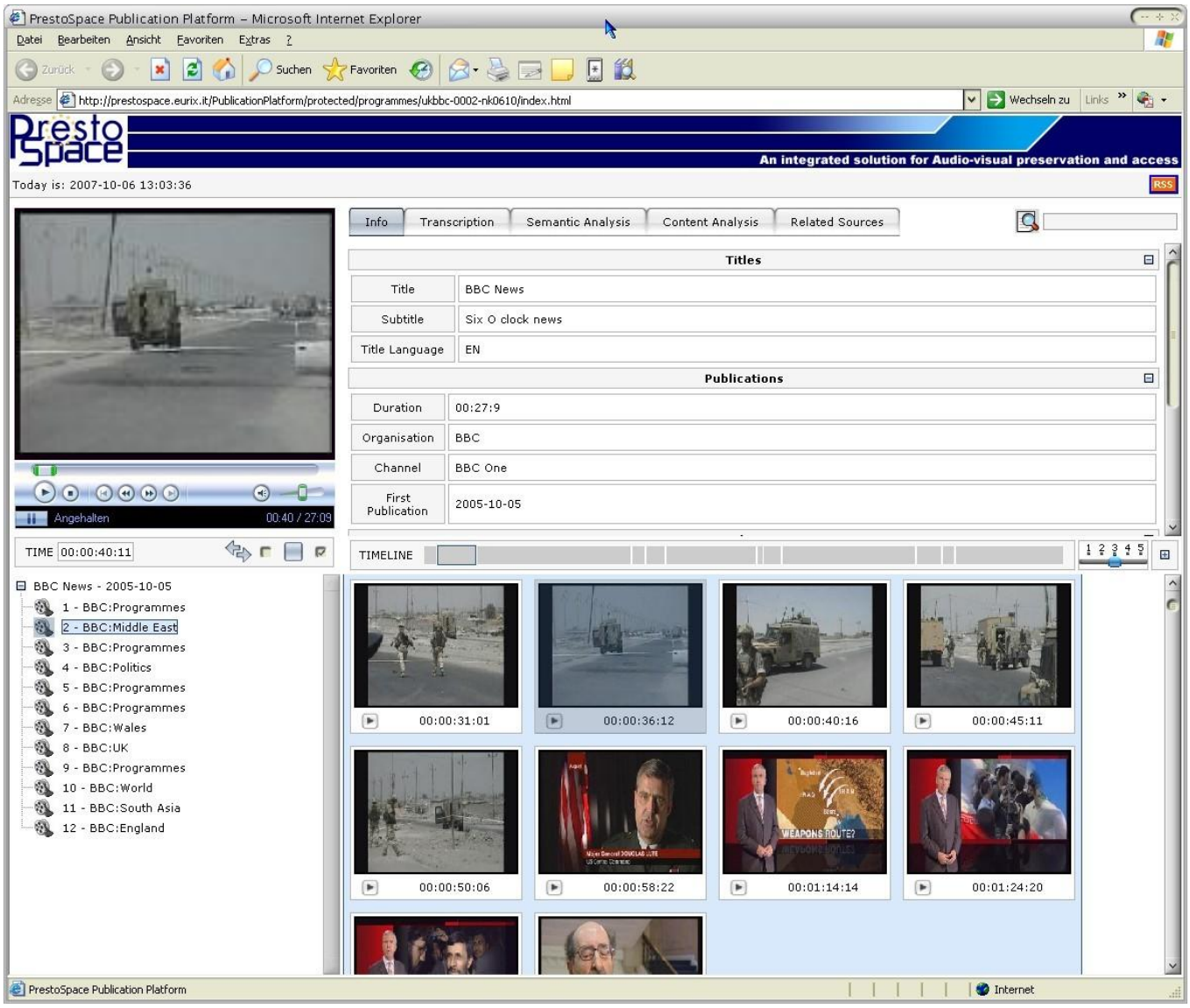


Figure 4 – EDOB-Viewer

This is the so-called “EDOB-Viewer” (EDOB = EDitorial OBject), the Single-Item-Interface for browsing purposes, showing all instances of metadata plus different representation-formats of the content-file (like preview-video, keyframes, etc.)

5.3.1 Description

The single-item-interface is the tool to give the user access to all information on and about the item represented. This includes no only textual information, but preview-video, keyframes, stripe-image, etc. as well.

The interface is build up on the presumption of a single-screen workstation, so the preview-video-screen had to incorporated, arranged in the upper left part of the screen. All main information (“Info”) can be found directly once the user called the page, arrange on the left space adjacent to the video-space. The main structure of the item (program, contribution, etc.) is shown via a expandable tree-structure-list underneath the preview-video. Underneath the main information part the keyframe-structure of the item is arrange. Between the upper part (video + main information) and the lower part (structure + keyframes) the time-information is situated, giving a simple TimeCode under the video-part and a timeline, representing the whole item, underneath the main-information part.

The video is shown via the “Microsoft Media Player”®, providing all controls incorporated there. To avoid structural problems, the video-part has fixed dimensions and size.

Figure 5 – Video Player

The program-structure-part is controlled similar to the tree-structured parts in OS’s like Windows, showing the sub-parts of an item.

Figure 6 – Program structure

The keyframe-part is highly customizable; the width can be changed as well as the size of the keyframes (and herewith the number of shown frames). Underneath every keyframes the exact TimeCode, showing where from the frame has been taken, is shown.

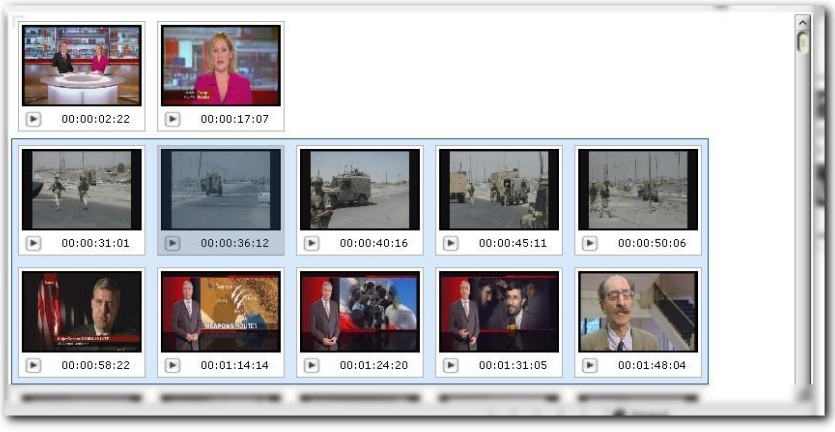


Figure 7 – Keyframe table

The most information-rich area is the main-information part; the different informations on the item, generated and coming from different (technical) sources, are chosen by tabs. The info-tab (default on entry) shows the (manual annotated or migrated)

"common" metadata (e.g. titles, durations, producer, etc.) (figure 8); the "Transcription"-tab shows the results from the automatic speech-to-text transcription (figure 9); the "Semantic Analysis" shows all results coming from that part of content-analysis (figure 10), while the "Content-Analysis"-tab leads to the results of more technical-oriented analysis (e.g. Stripe-Image, Camera-movement-detection, etc.) (figure 11); finally the "Related Sources"-tab shows links to sources, found during the analysis of the content, which have the same topics as the analysed item (figure 12).

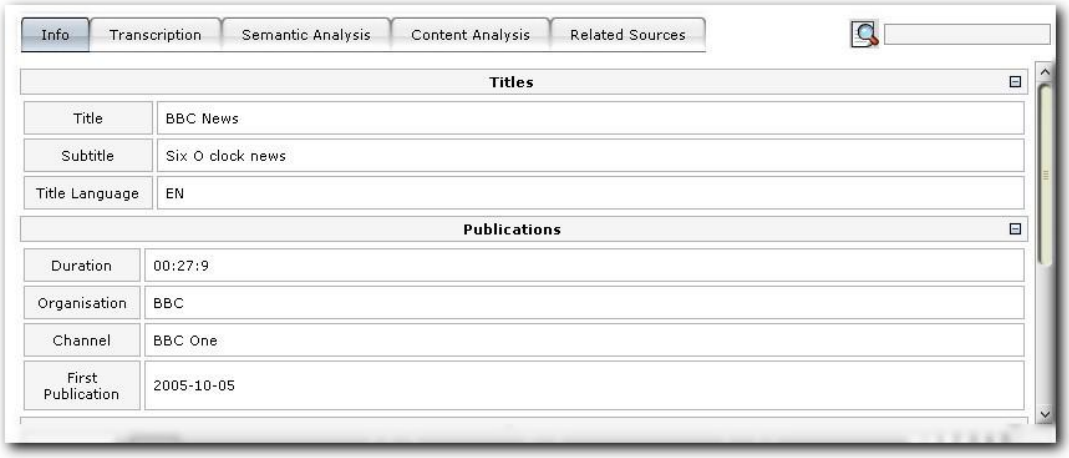


Figure 8 – Core Metadata

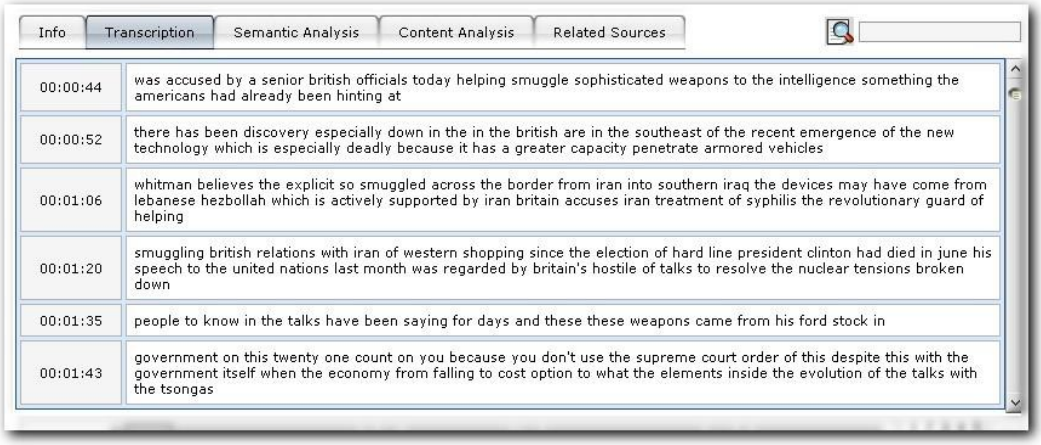


Figure 9 – Transcription (Speech to Text)

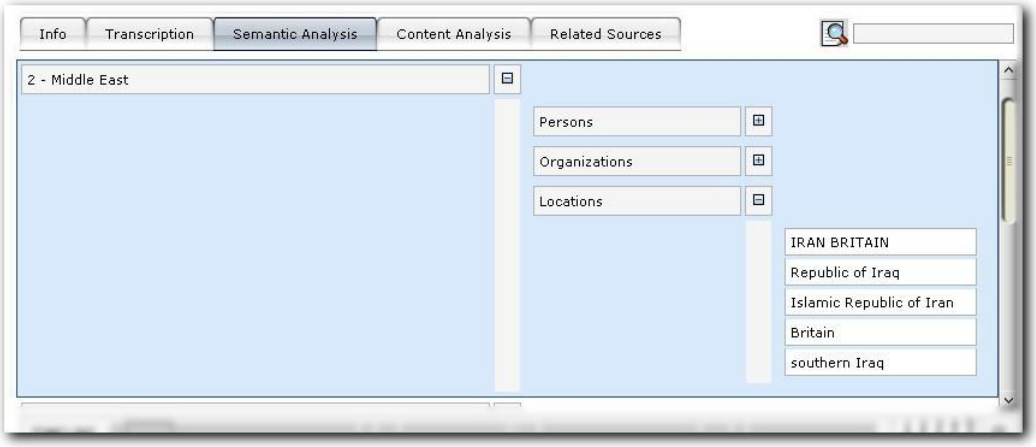


Figure 10 – Results from the Semantic Analysis

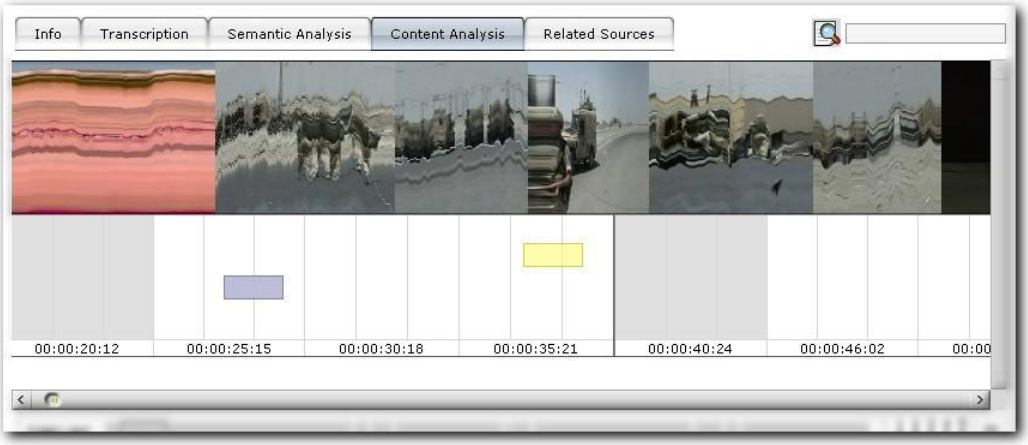


Figure 11 – Stripe Image and Camera Movement

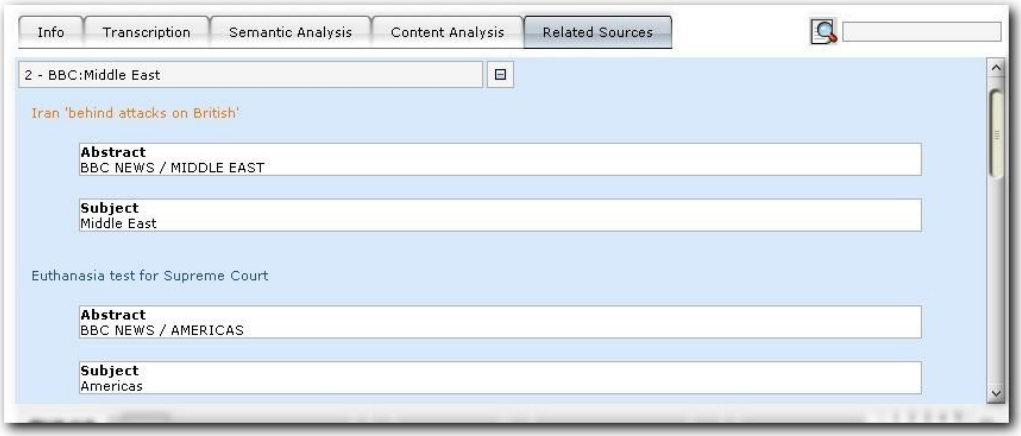


Figure 12 – Links to related Sources

All parts are interlinked; while replaying a video, all related parts are automatically tracked and the corresponding data is highlighted. From all parts navigation can be done and the video-replay started.

Via a simple search-field in the upper right corner of the interface certain information and sub-parts within the item can be found.

For more information please refer to D18.2[24].

5.3.2Checking of the User-Interface-Principles UIP

5.3.2.1 UIP1: fulfilled

The User is in permanent control; all necessary changes and preferences can be made every time. The interface provides the highest possible flexibility, but avoids critical changes (see 5.3.2.6).

5.3.2.2 UIP2: fulfilled

By using common design-standards and following “traditional” screen-partitioning, the interface shows high consistency. The common research-workflow (from “global” to “refined/special”) is strictly supported.

5.3.2.3 UIP3: (nearly) fulfilled

In that step an interface can provide only small support in reducing the users memory-load; by the consequent interlinking of the different content-representation-parts and metadata-parts the interface provides full backup in this part; only a highlighting of the used search-terms would be appreciated.

5.3.2.4 UIP4: fulfilled

Although the task of providing “invisible technology” is a real hard one in this kind of interface, the reduction of “heavy technological” to the minimum by “hiding” them in sub-parts (“Content Analysis”) avoids a possible confusion of the naïve user; the usage of “common” technology like the “Microsoft Media Player”® for presenting the preview-media helps and supports the user in providing well-known and familiar surroundings, avoiding long training and familiarisation.

5.3.2.5 UIP5: fulfilled

A selectable highlighting of the interrelated areas in the different representations- and metadata-parts during preview gives perfect feedback to the user; the bluish background is informative, but not distracting. (see figures 6,7,9,10,12).

5.3.2.6 UIP6: fulfilled

As mentioned in 5.3.2.1, a intentional, but small constraint in the flexibility of the interface is of immense importance for the error-prevention of this interface.

5.3.2.7 UIP7: fulfilled

Permanent feedback and the mentioned error-prevention lead to a robust behaviour of the interface, as the user is always well aware about what’s going on and where he is; maloperation is easily detected and fixed.

5.3.2.8 UIP8: only partly fulfilled

As some of the functionalities and controls are only comprehensible to professional and highly trained users, an online-help-functionality, that complements the available documentation, is highly missed.

5.3.2.9 UIP9: (nearly) fulfilled

The interface tries to follow the path taken with the search-interface; the mere amount of functionalities aggravates a complete success.

5.4 Manual Annotation GAMP (Browsing IF)

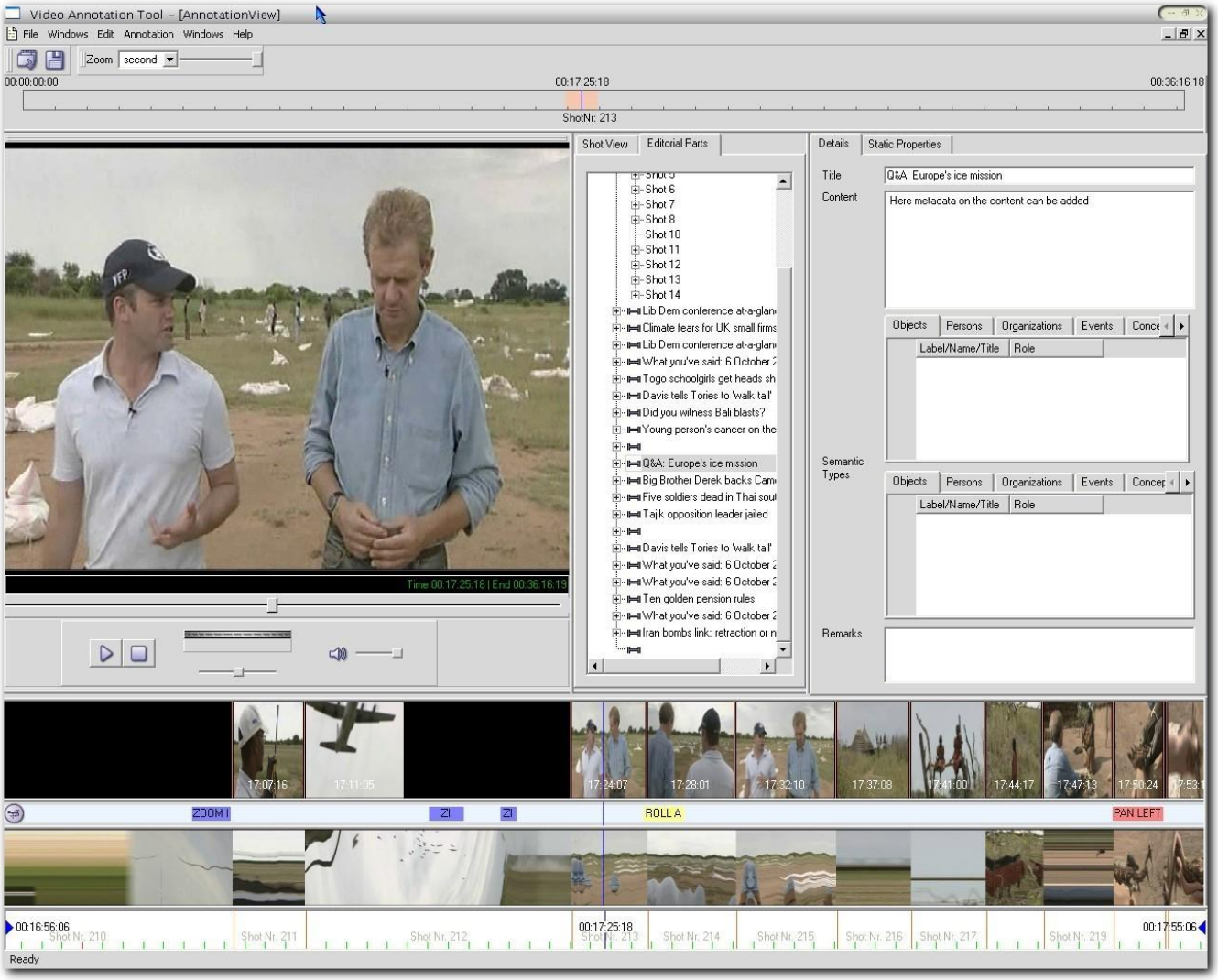


Figure 13 - Manual Annotation GAMP

5.4.1 Description

Although a lot of documentation and annotation, technical and content-related metadata are gained via the different automatic content analysis GAMPs and semantic tools or imported from legacy systems, there's always the need to add human aggregated data and manually annotate the items or parts within.

The highly customizable and flexible interface consists of the following areas:

5.4.1.1 The video-player

Figure 14 – AG-Videoplayer

The player is kept pretty simple, offering only controls for start/pause and stop, a slider for shuttle/jog function, mute and volume-control, a slider for quick navigation through the video and basic information on the current TimeCode and the End-TimeCode of the item.

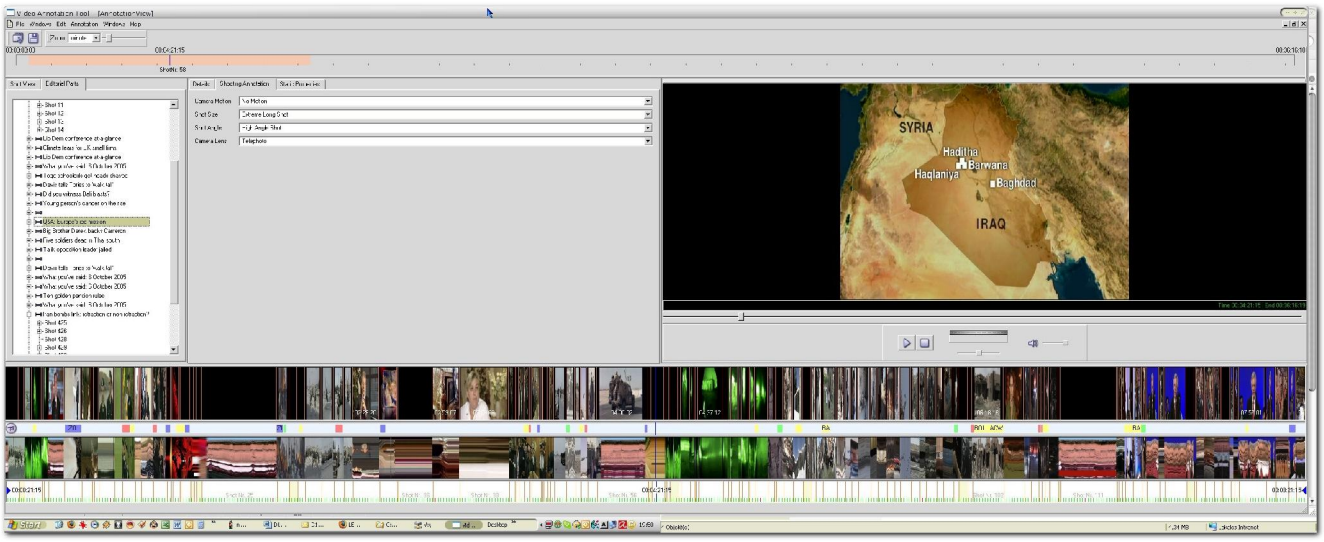


Figure 15 – Dual-screen flexibility

The video-player can be undocked and is can be snapped in wherever wanted; even a dual-screen environment is supported.

5.4.1.2 Program-structure – segmentation area

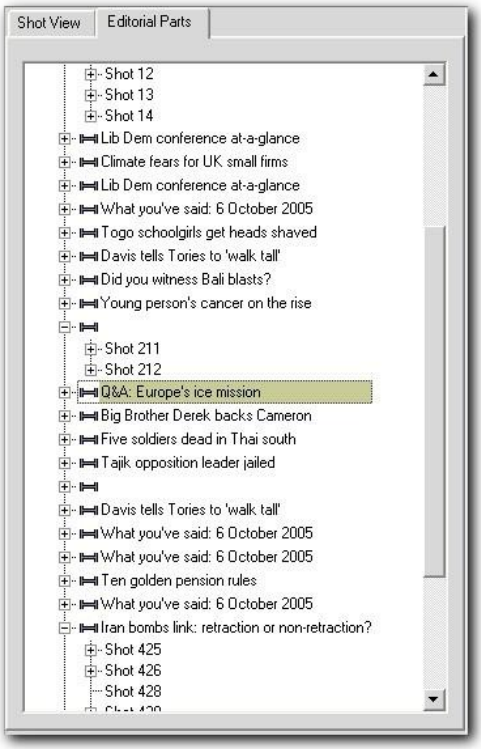


Figure 16 – Segmentation-tree-structure

In this area of the GAMP the structure of the program-item is shown as segmented by the automatic segmentation GAMP or as segmented by manual engagement. There

are two views: the “Shot-View” (figure 17) and the “Editorial Parts”-View (figure 16); in the Shot-View the automatic segmentation can be controlled, in the Editorial-Parts View the shots can be combined and assigned to editorial (sub)parts of the item. Each part and subpart can be edited, expanded, splitted, etc.

5.4.1.3 Metadata-Details Area

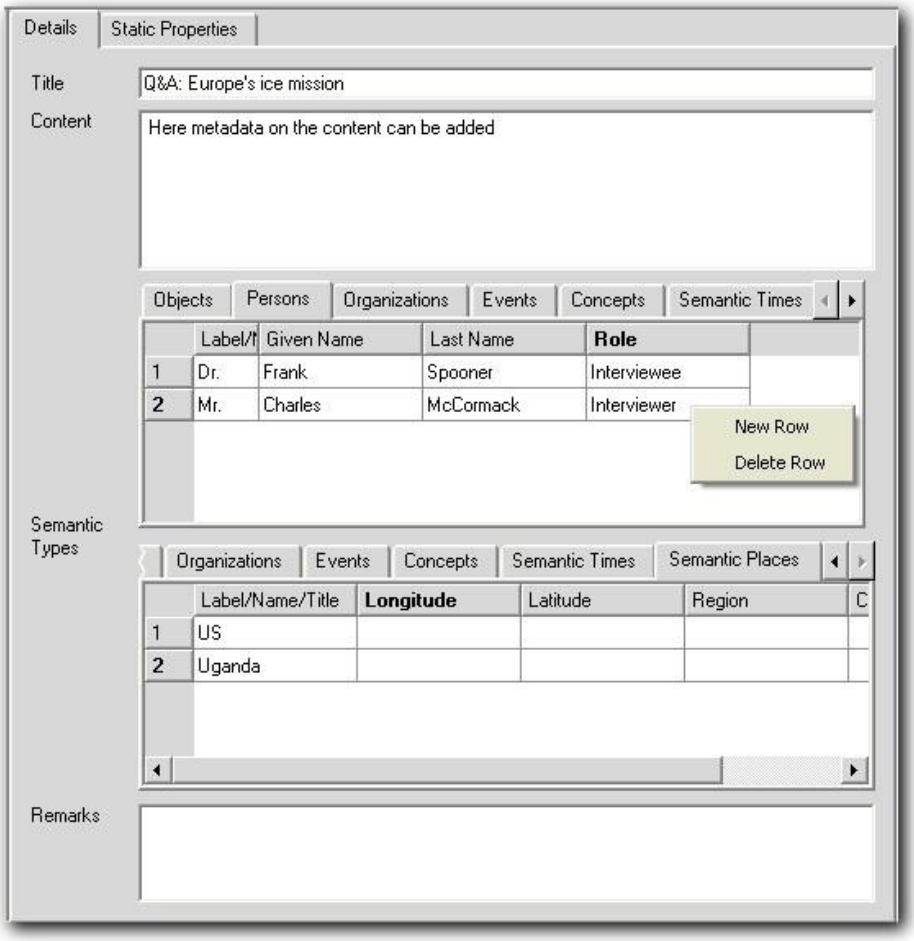


Figure 17 – Metadata Details

This is the main area for manual annotation; for each shot, keyframe, editorial part or subpart, descriptive metadata can be annotated. Some of the most important parts of the datamodel: Title, Content, Objects, Persons, Organizations, Events, Concepts, Semantic Times, Semantic Places, Shooting Annotation (Camera Motion, etc.).

5.4.1.4 Keyframe-view

Figure 18 – Keyframe-Line

The line of keyframes show representative frames from the entire video, each standing for a single shot; in the “Display Shot Border”-mode, thin orange lines show the shot-boundaries and each keyframe is jolted or black space is added to show the proportional rate of the shot. This is altered whenever the length of the shown program-part is changed by the user. In the lower part of the keyframes the corresponding TimeCode is shown as well.

5.4.1.5 Local Event Viewer

Figure 19 – Local Event Viewer

The Local Event Viewer shows the automatically detected camera-movements or lens-movements, grouping them by colour-code; an additional overview on the complete Local Events can be seen by activating the “CameraMotion Full Time”-mode.

5.4.1.6 Stripe Image View

Figure 20 – Stripe Image View

The Stripe-Image view of a video gives the professional user a perfect overview and good estimation on the action in a video-part without looking at the video itself; the Stripe-Image is generated out of single pixel-lines, taken from every single frame of the

video. With the use of the Stripe-Image a user can detect talking heads, pans, moving objects, and many more.

5.4.1.7 Time Line(s)

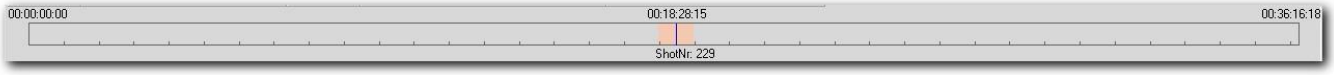


Figure 21 – Time-Line 1

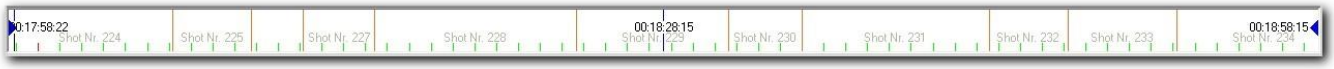


Figure 22 – Time-Line 2

The Time-lines allow to the user a direct-access to subparts of the video and give permanent feedback on the current position and the currently visible part of the whole program-item.

5.4.1.8 Other navigation-controls and tool-parts

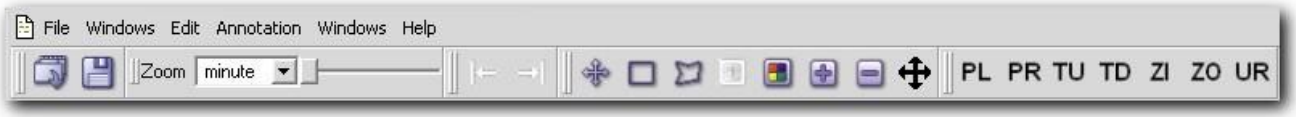


Figure 23 – Navigation and Tools

Beneath the common menu-part there are also a special Zoom-Tool, a Shot-per-Shot navigation tool, a paint-tool for adding graphics to the video and a panel for direct entering of certain camera-movements.

All parts described in the precedent paragraphs are freely customizable in ways of changing their position in the interface, switching them on and off and altering their width and/or height.

5.4.2Checking of the User-Interface-Principles UIP

5.4.2.1 UIP1: fulfilled

The flexibility of the program is extremely high, putting the user in total control how he organizes his “desk” and set. But this extreme flexibility can be a inhibition threshold for

the inexperienced user, as the training-effort for managing all the functionalities and possibilities of this tool is rather high.

5.4.2.2 UIP2: nearly fulfilled

Although a good consistency is mostly given throughout the whole interface, some slight puzzle may occur by the limitations given for resizing certain areas of the tool (e.g. video-area).

5.4.2.3 UIP3: fulfilled

The computer provides a clear and versatile view of the item to be annotated; the user is well supported in his task, all information available is made accessible by the tool.

5.4.2.4 UIP4: nearly fulfilled

The mere copiousness of tools and functionalities may be a problem to get familiar with this interface without further ado, but the use of common icons and control-items, and the flexibility to reduce the interface to the functionalities really needed to fulfil the task given, the user will pass this challenge as well.

5.4.2.5 UIP5: fulfilled

As the complete video and all it's segments can be easily tracked and the tool provides a customizable overview, permanent feedback and control is assured.

5.4.2.6 UIP6: partly fulfilled

Again the amount of functionalities bears the risk of errors; but the clear and concise preference-controls support the user in avoiding most of them. Unfortunately some resizing and rearranging-procedures may end up in a messy interface, which may lead to delays in the workflow.

5.4.2.7 UIP7: partly fulfilled

As said in the previous paragraph, the user can get lost in the attempt to rearrange his interface; the tool provides no "Undo"-functionalities to recover from such a situation and no function to store and recall certain settings. The main task of annotation is robust and safe.

5.4.2.8 UIP8: not fulfilled

Currently no online-help is available, only printed documentation; given the complexity of the tool, rich online-help functionality would be appreciated.

5.4.2.9 UIP9: fulfilled

Although the interface provides a bundle of functionalities, the user is nearly always kept focused by the different views and controls. The graphical design is smooth and professional, graphical gimmicks are avoided and the given controls and icons precise and simple.

6 Other User-Interfaces in PrestoSpace

6.1 Introduction

In PrestoSpace several User-Interfaces for different purposes had to be developed and built. The technical developers tried, in close contact with the user-partners in the project-consortium, to meet all requirements and specifications made by the underlying workflows and tasks as well as incorporating all recommendations and proposals made by the advising user-partners.

In the following paragraphs of this report some of the others, “non-content-retrieval & browsing” user-interfaces will be checked and tested using the aggregated principles for good user-interface design.

6.2 PrestoSpace WIKI

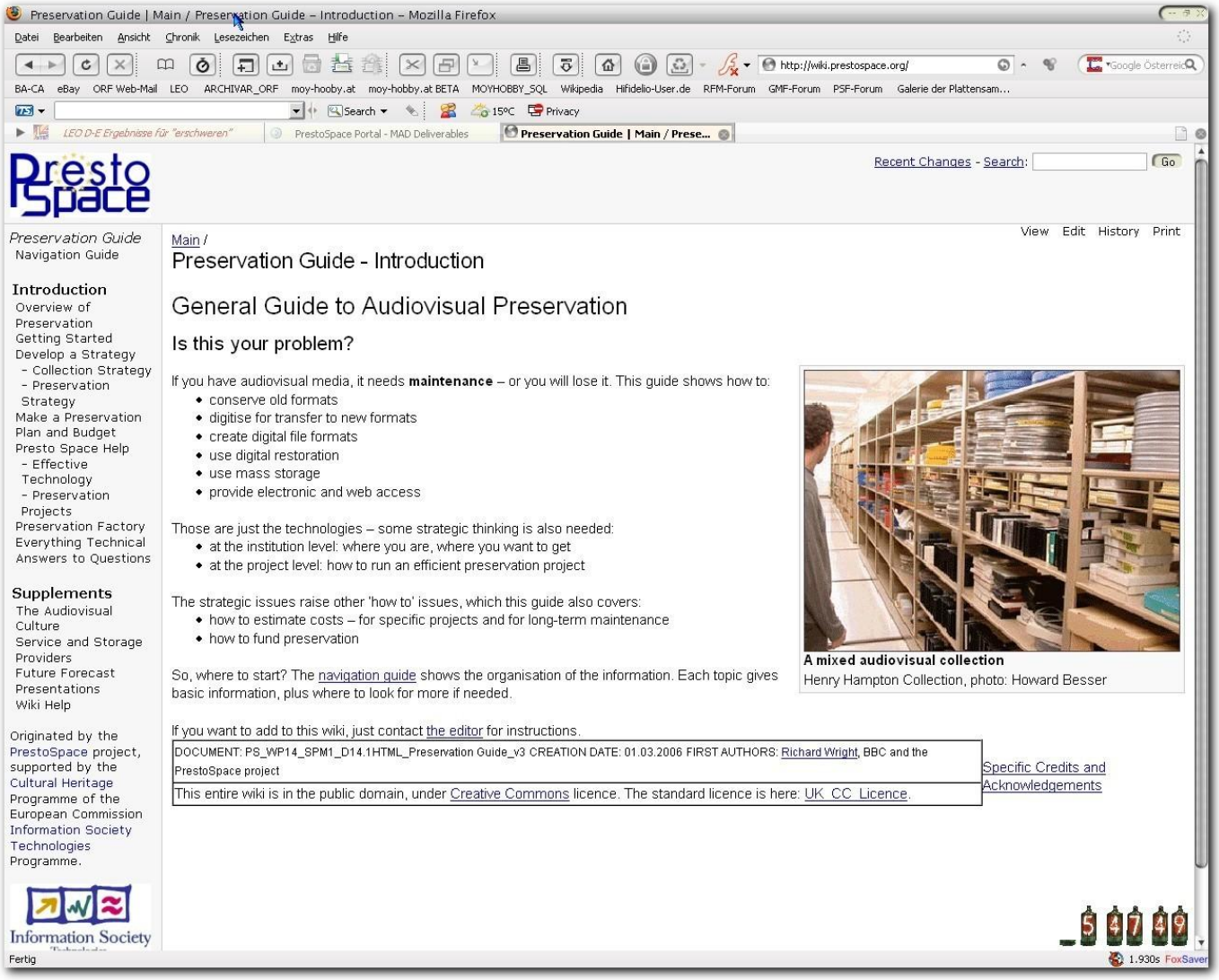


Figure 24 – Main page of PS-WIKI

This is the main page of the “PrestoSpace-WIKI”, a Preservation Guide to Audiovisual Preservation; since this guide is organized as a WIKI ("the simplest online database that could possibly work"[xxv]), a collaborative way of further development and enrichment of knowledge is assured.

6.2.1 Description

The PrestoSpace WIKI was as a first step a personal development by Richard Wright of BBC, done in the SAM-part of PrestoSpace. It was built up to provide everyone, seeking for information and knowledge about preservation and the technologies and procedures connected, a “one-stop-shop” of advice and recommendations.

The organisational form as a WIKI assures, that from each interested professional additional information can be added directly, providing fresh information for the user and a collaborative platform for the authors and specialists.

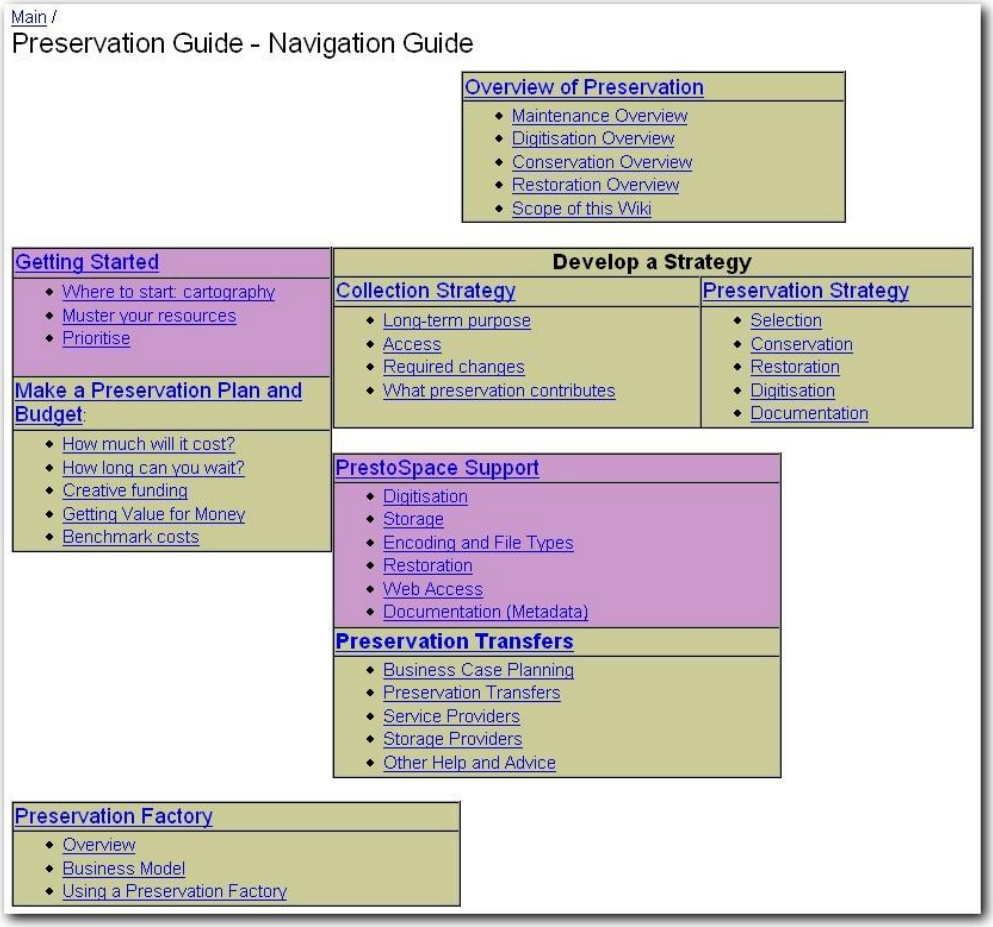


Figure 25 – Navigation Guide

A navigation guide and dozens of tutorials and instruction-subpages cover all parts of preservation-activities, from planning to implementation.

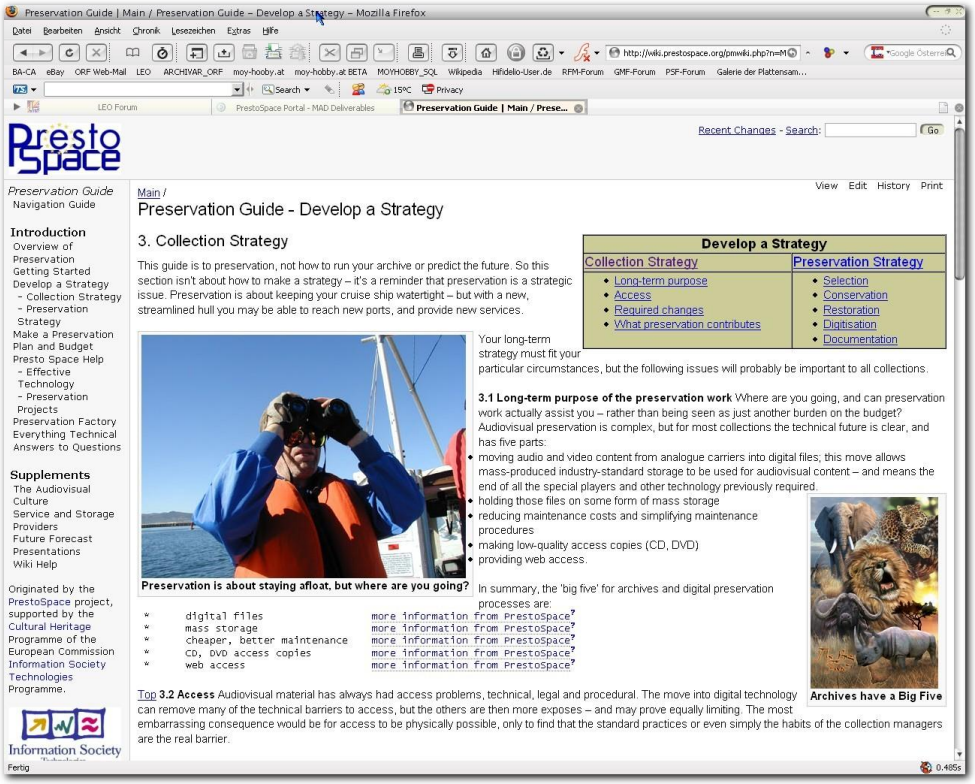


Figure 26 – Page on Collection Strategy

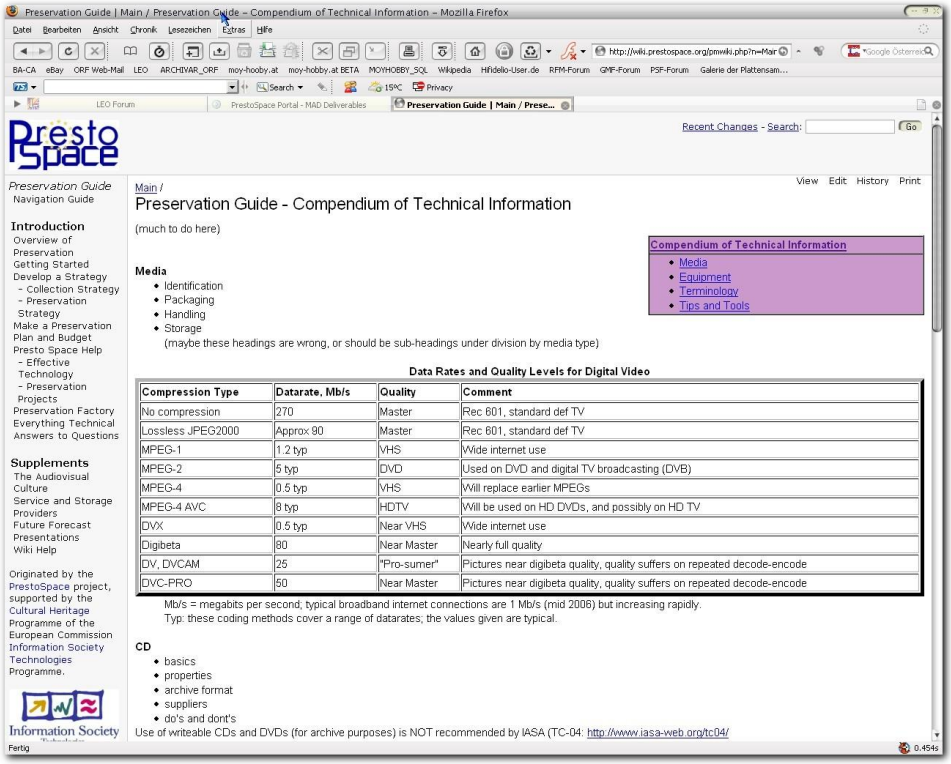


Figure 27 – Compendium of Technical Information

6.2.2Checking of the User-Interface-Principles UIP

6.2.2.1 UIP1: fulfilled

The complete navigation-design and all pages and tutorials are strictly designed to support the individual needs of each user, who is in permanent control and can reach every part of the site at any moment. Although the WIKI tries to guide the user and always proposes a certain step, the user's always free to negate that proposal and go on his own.

6.2.2.2 UIP2: fulfilled

The WIKI follows the common design standards for those collaborative information-sites, all the subparts are consistent and follow the same principles.

6.2.2.3 UIP3: fulfilled

No special needs to fulfil here, the standard functionalities of each web-browser-software provides the necessary tools like "history"-functionalities, etc.

6.2.2.4 UIP4: partly fulfilled

The "Look&Feel" is a little bit different to well-known WIKI's like Wikipedia; also no real world metaphors are in use (icons or alike); nevertheless the design is up to common standards and self-explanatory (see 5.5.2.2 and 5.5.2.9)

6.2.2.5 UIP5: nearly fulfilled

The user should be always aware where he is and on what part he's reading on; only a feedback on the current page given by the left-hand-sided menu is missing.

6.2.2.6 UIP6: fulfilled

Since the WIKI itself is concentrated on a straight-forward design, the error-problem should be reduced to those coming from the underlying technologies like browser-software and internet-connectivity.

6.2.2.7 UIP7: fulfilled

The only error caused by the user can be the switching to a wrong page by clicking a wrong link; re-do's and undo's are easy done via the controls of the web-browser-software used.

6.2.2.8 UIP8: fulfilled

Since the whole WIKI is "one big helpfile", addressing also the problems that may occur using the site, this principles can be seen as fulfilled as well.

6.2.2.9 UIP9: fulfilled

The functionalities of the WIKI are reduced to the absolute necessary controls for navigating through the site; the overall design may be seen as too simple and lacking of aesthetic refining, but it's keeping the user focused and concentrated on the given informations.

6.3 Preservation Cost Calculator

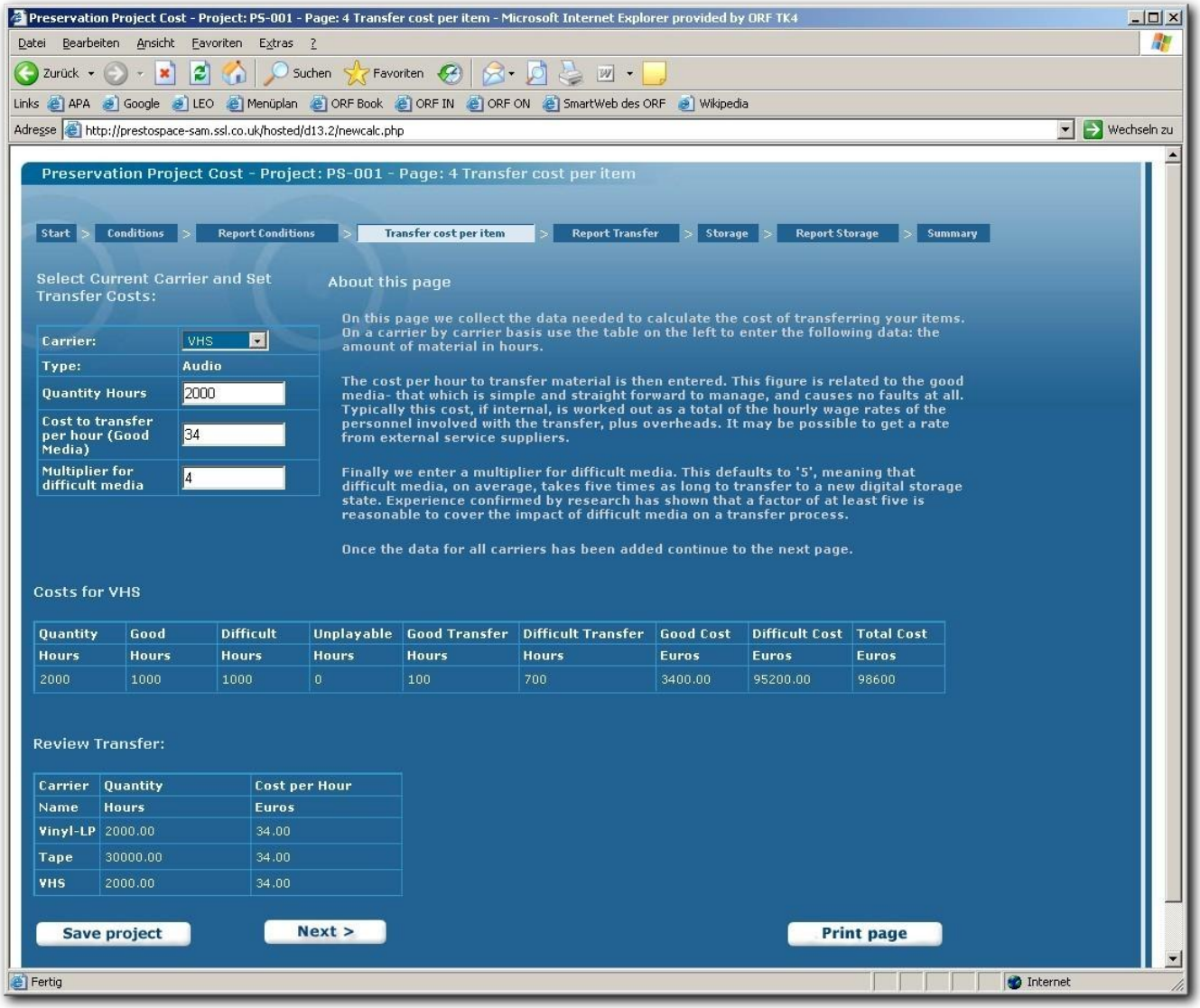


Figure 28 – Cost Calculator for Preservation projects

“Welcome to the Preservation project cost calculator. Over seven pages we will guide you through a simplified but useful analysis of your requirements for preservation and digital storage of your archive, including a cost breakdown.”[xxvi]

6.3.1 Description

As it has been found that one of the hardest tasks for those taking care of preservation is to calculate the presumably cost-range of such a project, this online-tool for a rough-calculation of preservation project costs has been developed.

Figure 29 – Process-line

On only 4 stages the user has to provide input to the calculator; each step is concluded by a report-page for controlling the inputs given.

At the Start-page the user has to provide information on the media to be transferred and can name his project; at the Condition-page he has to provide information on the overall condition of each media-type and format, divided in “good”, “difficult” and “unplayable”, each to be given in percent. On the next page the amounts for each carrier and the transfer-costs per item have to be inserted for each format and condition. The last input to be given is the kind of digital storage format the user wants his digitized media to be stored at and the corresponding storage costs.

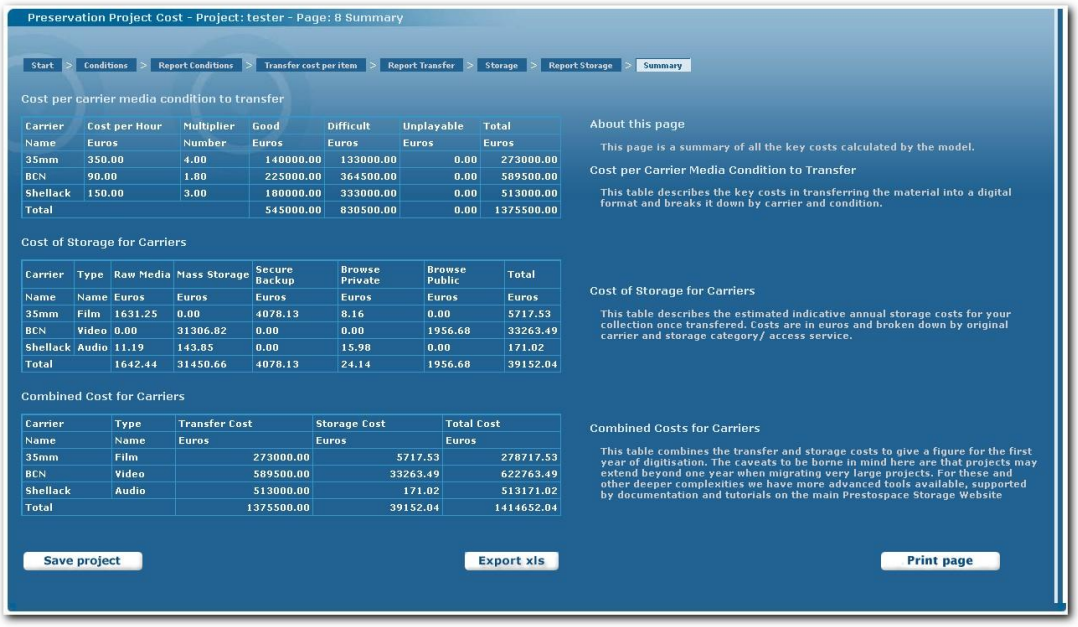


Figure 30 – Summary Page

On the last page the results of the calculation are presented to the user; he can change every step and every input at every stage of the process. The results can be stored, exported, printed and further processed.

	A	B	C	D	E	F	G	H
1	Cost per carrier media condition to transfer							
2	Carrier	Cost per Hour	Multiplier	Good	Difficult	Unplayable	Total	
3	Name	Euros	Number	Euros	Euros	Euros	Euros	
4	35mm	350,00	4,00	140000,00	133000,00	0,00	273000,00	
5	BCN	90,00	1,80	225000,00	364500,00	0,00	589500,00	
6	Shellack	150,00	3,00	180000,00	333000,00	0,00	513000,00	
7	Total			545000,00	830500,00	0,00	1375500,00	
8								
9	Cost of Storage for Carriers							
10	Carrier	Type	Raw Media	Mass Storage	Secure Backup	Browse Private	Browse Public	Total
11	Name	Name	Euros	Euros	Euros	Euros	Euros	Euros
12	35mm	Film	1631,25	0,00	4078,13	8,16	0,00	5717,53
13	BCN	Video	0,00	31306,82	0,00	0,00	1956,68	33263,49
14	Shellack	Audio	11,19	143,85	0,00	15,98	0,00	171,02
15	Total		1642,44	31450,66	4078,13	24,14	1956,68	39152,04
16								
17	Combined Cost for Carriers							
18	Carrier	Type	Transfer Cost	Storage Cost	Total			
19	Name	Name	Euros	Euros	Euros			
20	35mm	Film	273000,00	5717,53	278717,53			
21	BCN	Video	589500,00	33263,49	622763,49			
22	Shellack	Audio	513000,00	171,02	513171,02			
23	Total		1375500,00	39152,04	1414652,04			
24								

Figure 31 – MS-Excel-Export

```
<parameter name="page_4_good_cost" value="225000" />
<parameter name="page_4_difficult_cost" value="364500" />
<parameter name="page_4_total_cost" value="589500" />
<parameter name="page_4_review_quantity_0" value="500.00" />
<parameter name="page_4_review_cost_0" value="350.00" />
<parameter name="page_4_review_quantity_1" value="5000" />
<parameter name="page_4_review_cost_1" value="90" />
<parameter name="page_4_review_quantity_2" value="2000.00" />
<parameter name="page_4_review_cost_2" value="150.00" />
<parameter name="page_6_carrier_0_rawmedia" value="X" />
<parameter name="page_6_carrier_0_securebackup" value="X" />
<parameter name="page_6_carrier_0_browseprivate" value="X" />
<parameter name="page_6_carrier_0_rawmedia_quality_select" value="Uncompressed" />
<parameter name="page_6_carrier_0_massstorage_quality_select" value="Lossless Compression" />
<parameter name="page_6_carrier_0_securebackup_quality_select" value="Master" />
<parameter name="page_6_carrier_0_browseprivate_quality_select" value="Broadband" />
<parameter name="page_6_carrier_0_browsepublic_quality_select" value="Broadband" />
<parameter name="page_6_carrier_0_rawmedia_quality_text" value="400" />
<parameter name="page_6_carrier_0_massstorage_quality_text" value="160" />
<parameter name="page_6_carrier_0_securebackup_quality_text" value="100" />
<parameter name="page_6_carrier_0_browseprivate_quality_text" value="2" />
<parameter name="page_6_carrier_0_browsepublic_quality_text" value="2" />
<parameter name="page_6_carrier_1_massstorage" value="X" />
<parameter name="page_6_carrier_1_browsepublic" value="X" />
<parameter name="page_6_carrier_1_rawmedia_quality_select" value="Uncompressed" />
<parameter name="page_6_carrier_1_massstorage_quality_select" value="Lossless Compression" />
<parameter name="page_6_carrier_1_securebackup_quality_select" value="Master" />
<parameter name="page_6_carrier_1_browseprivate_quality_select" value="Broadband" />
<parameter name="page_6_carrier_1_browsepublic_quality_select" value="Broadband" />
<parameter name="page_6_carrier_1_rawmedia_quality_text" value="100" />
<parameter name="page_6_carrier_1_massstorage_quality_text" value="40" />
<parameter name="page_6_carrier_1_securebackup_quality_text" value="25" />
<parameter name="page_6_carrier_1_browseprivate_quality_text" value="0.5" />
<parameter name="page_6_carrier_1_browsepublic_quality_text" value="0.5" />
<parameter name="page_6_carrier_2_rawmedia" value="X" />
<parameter name="page_6_carrier_2_massstorage" value="X" />
<parameter name="page_6_carrier_2_browseprivate" value="X" />
<parameter name="page_6_carrier_rawmedia_cb" value="on" />
<parameter name="page_6_carrier_2_rawmedia_quality_select" value="Uncompressed" />
<parameter name="page_6_carrier_2_rawmedia_quality_text" value="0.7" />
<parameter name="page_6_carrier_massstorage_cb" value="on" />
<parameter name="page_6_carrier_2_massstorage_quality_select" value="Lossless Compression" />
<parameter name="page_6_carrier_2_massstorage_quality_text" value="0.3" />
<parameter name="page_6_carrier_2_securebackup_quality_select" value="Master" />
<parameter name="page_6_carrier_2_securebackup_quality_text" value="0.2" />
<parameter name="page_6_carrier_browseprivate_cb" value="on" />
<parameter name="page_6_carrier_2_browseprivate_quality_select" value="Broadband" />
<parameter name="page_6_carrier_2_browseprivate_quality_text" value="0.1" />
```

Figure 32 – XML-Export

6.3.2Checking of the User-Interface-Principles UIP

6.3.2.1 UIP1: fulfilled

Although the workflow-support of this tool is trying to guide the user along logical steps, he is free to do several inputs (like costs, etc.) via leaving the given route.

6.3.2.2 UIP2: fulfilled

The interface follows common design-standards, providing a strict “left-to right” and “top down” workflow; the interface is consistence, no workflow- or design-severance has been detected.

6.3.2.3 UIP3: fulfilled

The software automatically keeps all input given stored; the user can change from stage to stage without the loss of any information.

6.3.2.4 UIP4: nearly fulfilled

The “negative” design (bright writing on dark background) can be seen as violation of UIP4, as our real-life very seldom uses this approach; but more important is the correct use of common control-items like tick-boxes, drop-down-menus and using an “x” -icon for deleting items. 

6.3.2.5 UIP5: fulfilled

This UIP is perfectly fulfilled; the permanent overview and feedback given by the top-line of actions (figure 18) and the periodical provided control-pages gives the user the feeling of permanent control over his task.

6.3.2.6 UIP6: partly fulfilled

The tool is only error-proof to a user, who is a audiovisual professional; it’s possible to enter completely senseless figures. This boosts the flexibility and future-proofness of the tool, but may lead to usability-problems.

6.3.2.7 UIP7: fulfilled

Since every input and data given to the tool can be changed and altered at any stage or time, the calculator provides good safety and robustness.

6.3.2.8 UIP8: fulfilled

Every necessary step is well documented and precise instructions are given on every page of the tool.

6.3.2.9 UIP9: fulfilled

All functionalities and controls are strictly task-oriented; the tool is kept simple wherever possible. The user is perfectly kept focused on his task and the overall aesthetical impression of the tool is sufficient.

6.4 Storage Calculator

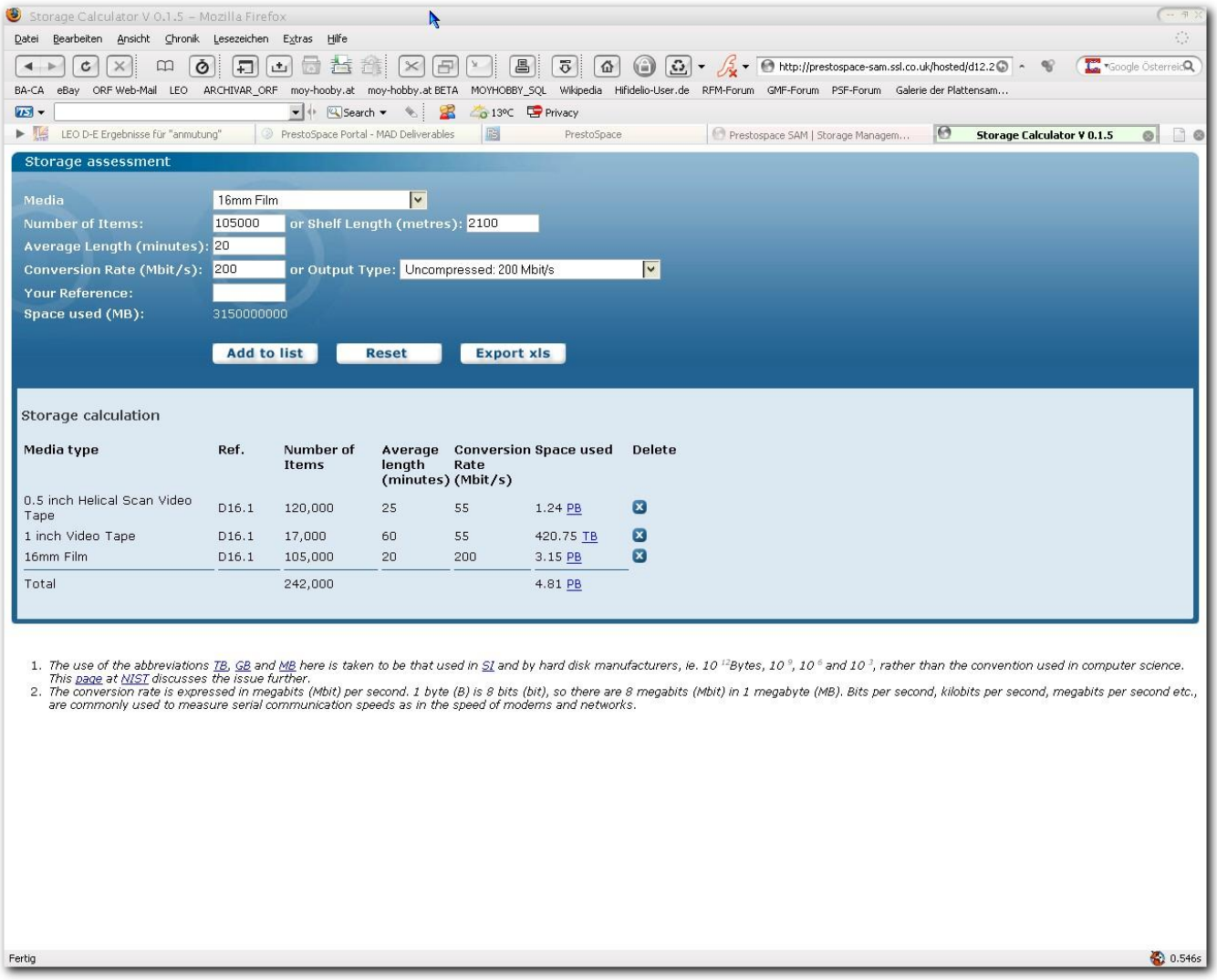


Figure 33 – Storage Calculator

The Storage Calculator is a very basic tool to calculate the amount of digital storage space needed for a given amount of analogue media to be digitized in a given quality.

6.4.1 Description

This basic online-tool is meant to help the user, who's going to plan a digitisation-project, to calculate the total amount of digital space he'll need to store all his digitized media.

An nearly unlimited amount of different analogue media can be entered in the tool, combined with the amount of items or "length of shelves" (as many companies don't know the exact amount of items they have stored) and an average length for every item. Additionally the conversion rate or the digital output-type is needed to let the tool calculate the total amount of space needed.

The different digital measurements are described and explained as well.

6.4.2 Checking of the User-Interface-Principles UIP

6.4.2.1 UIP1: nearly fulfilled

The user is control and the best possible flexibility is given; only the room to add special (analogue and digital) carriers is missing

6.4.2.2 UIP2: fulfilled

The interface is following common standards and provides good consistency.

6.4.2.3 UIP3: fulfilled

The software automatically keeps all input given stored; the user can change from stage to stage without the loss of any information. Only the possibility to store the results is missing, but export is possible.

6.4.2.4 UIP4: fulfilled

The “negative” design (bright writing on dark background) can be seen as violation of UIP4, as our real-life very seldom uses this approach; but more important is the correct use of common control-items like drop-down-menus and using an “x” -icon for deleting items. 

6.4.2.5 UIP5: fulfilled

Since the results are permanently updated and the tool is a “one-pager”, constant feedback is provided.

6.4.2.6 UIP6: partly fulfilled

The tool is only error-proof to a user, who is a audiovisual professionalist; it’s possible to enter completely senseless figures. This boosts the flexibility and future-proofness of the tool, but may lead to usability-problems.

6.4.2.7 UIP7: fulfilled

Since all input and data given to the tool can be changed and altered, the storage-calculator provides good safety and robustness.

6.4.2.8 UIP8: partly fulfilled

A online-documentation is missing; the tutorial and legend on the used abbreviations is helpful.

6.4.2.9 UIP9: fulfilled

The functionalities and controls are task-oriented; the tool is kept as simple as possible. The user is kept focused on his task and the overall aesthetical impression of the tool is again sufficient.

6.5 M.I.R. – Moving Image Retoucher

Figure 34 – M.I.R.

A simple, but powerful tool for manual retouching of moving images; the tool can be used to manually restore and repair visual defects in digitized film sources. The tool can be seen as part of the DIAMANT™-Digital Film Restoration Suite, but is meant to be used as stand-alone tool, while DIAMANT™ is a complete suite for automatic, semi-automatic and manual film-restoration.

6.5.1Description

The moving image retouching tool M.I.R. is a stand-alone software for Windows® and Mac® OS's; M.I.R. provides a solution for manual dust busting and any other kind of single image defects retouching issues by considering the specific characteristics of film and offering a powerful environment for application in post-houses and film production environment for restoration, wire-removal or any other retouching issue.

Using the "retouch-by-clone" methodology, the tool only applies existing information to fix moving images by cloning content from other regions and images. The built-in retouch monocle is a tool to support cloning by showing target and source in zoom.

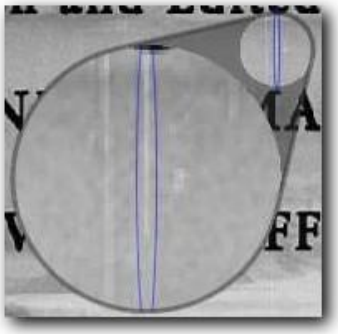


Figure 35 – M.I.R. Monocle

The software has been developed to work in a professional environment and does support a large amount of image file formats like DPX, CIN, TIFF, TGA, JPG and many more.

6.5.2Checking of the User-Interface-Principles UIP

6.5.2.1 UIP1: fulfilled

The tool is strictly focused on the task of manual retouching; the user is permanently in control of the workflow and task, given high flexibility to react to uncommon situations

6.5.2.2 UIP2: fulfilled

The interface follows the standards commonly used in the area of digital restoration and fits into the DIAMANT™-“family”; the interface is found to be consistent with the needs of the user in the underlying task.

6.5.2.3 UIP3: fulfilled

The user can keep focus on the main task, the tool keeps track of all necessary information and alike.

6.5.2.4 UIP4: fulfilled

Restoration professionals will be highly familiar with the “Look&Feel” of the interface; in this context the “negative” design (bright information on darker background) follows the real-life-metaphor, as the common editing-suites in the film-world are a rather dark environment. Other metaphors like the “monocle” (figure 24) are also pointing in the same direction.

6.5.2.5 UIP5: fulfilled

Tools and functionalities like the stripe-image-view (figure 25) provide permanent feedback and a proper overview.

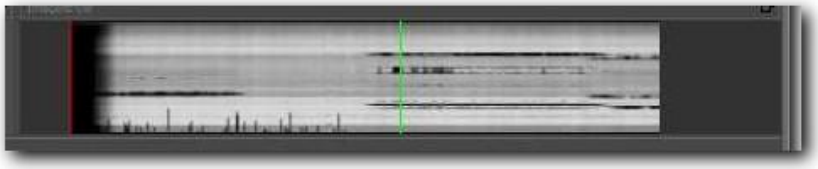


Figure 36 – Stripe Image

6.5.2.6 UIP6: fulfilled

Since the tool and it’s interface is meant to be used by professionalists and trained personnel, the existing error-prone and complicated passages of the tool are part of the functionality and therefore not retarding the compliance of this principle.

6.5.2.7 UIP7: fulfilled

The tool and interface provides robust and easy error-handling; all operations are non-destructive and undo-able.

6.5.2.8 UIP8: fulfilled

Two different manuals are available (Full[xxvii] and Light[xxviii]) plus a list[xxix] of the shortcuts used by the interface; a task-oriented online-help-functionality is provided as well.

6.5.2.9 UIP9: fulfilled

The whole interface has a very “professional” touch, combining simple, precise and unobtrusive controls and icons with a permanent focus on the main task.

6.6 RMT – Restoration Management Tool

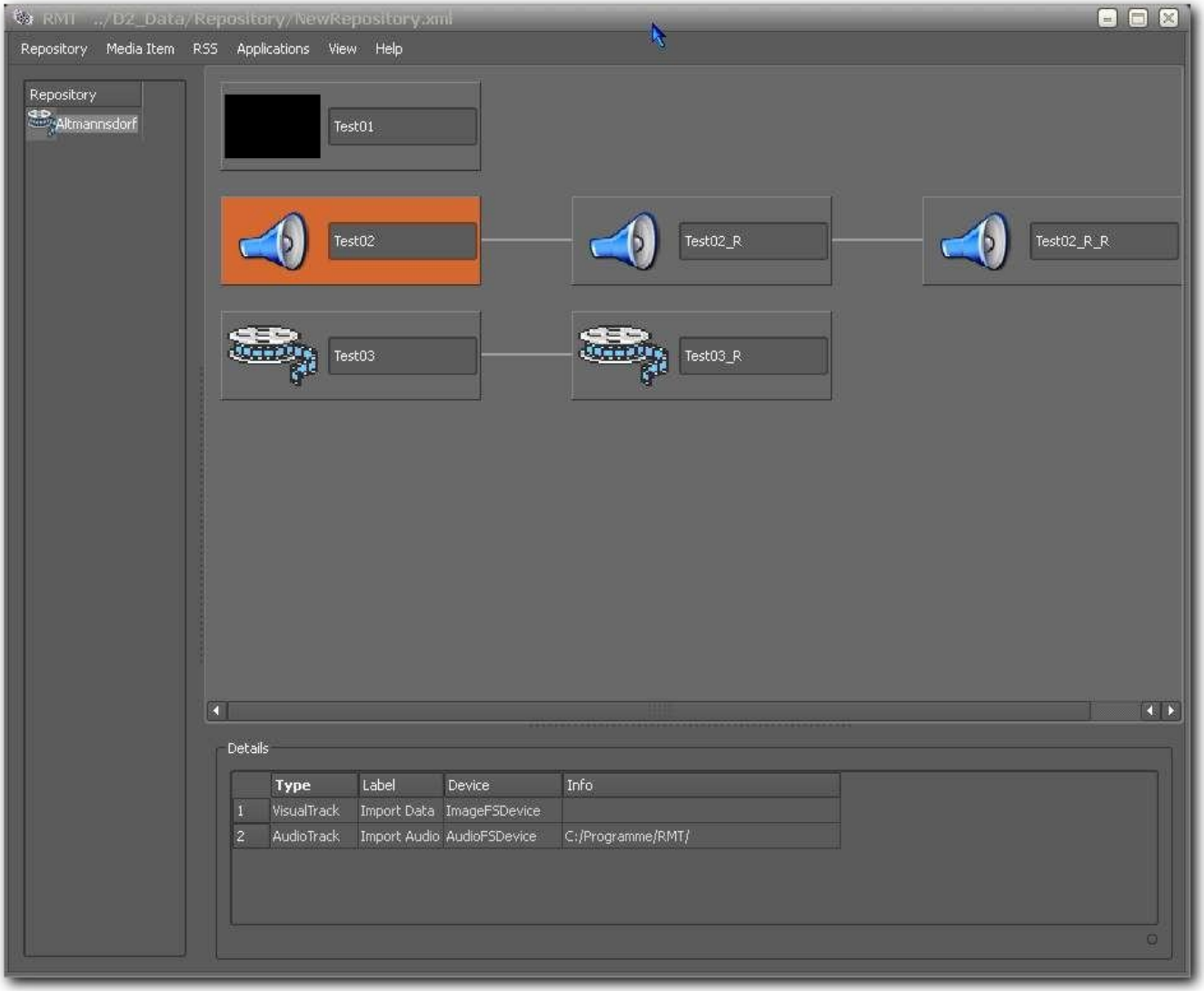


Figure 37 – Restoration Management Tool

The RMT – Restoration Management Tool provides an interface to control different, but depending restoration tasks;

6.6.1 Description

The Restoration Management Tool is considered as the main operator interface for all restoration tasks inside the PrestoSpace factory. It can be operated on any workstation that has access to the content repository. The RMT communicates, by means of a communication library based on GSOAP, with the various Restoration Subsystems (RSS) available in the PrestoSpace factory.[xxx]

Although communication is generic, the main focus in the development has been the integration of 3 types of RSS, namely DIAMANT™, BRAVA™ and AUDIOCUBE™. Those RSS are described in D10.1[xxxi], D10.2[xxxii] and D10.3[xxxiii].

6.6.2 Checking of the User-Interface-Principles UIP

6.6.2.1 UIP1: fulfilled

As this tool is constructed as a help to put the user in control of a complex and multilayer process, the completion of this UIP can be seen as necessary step to prove the usefulness of this tool.

6.6.2.2 UIP2: fulfilled

Guiding the user through the complex restoration tasks, the tool tries to provide as much consistency as possible, following common interface standards as far as possible.

6.6.2.3 UIP3: fulfilled

Via using common icons and metaphors, the tool fully supports the concept of the “familiar environment”; but is has to be kept in mind, that the RMT is a tool for professional use and therefore the threshold is rather low.

6.6.2.4 UIP4: fulfilled

The presentation of all activities in the Restoration-chain as a table of interlinked tasks, represented as boxes, helps to get the necessary comprehension on the operations.

6.6.2.5 UIP5: fulfilled

The main task of this tool is to provide a permanent overview and status-report on all restoration-processes within the linked RSS's; this is done in a very convincing way, so the UIP is to be regarded as fulfilled.

6.6.2.6 UIP6: partly fulfilled

As the tool is meant for professional-use only, the error-prevention is rather low on that level, but mostly sufficient.

6.6.2.7 UIP7: fulfilled

The RMT is pretty robust and provides the necessary error-handling.

6.6.2.8 UIP8: nearly fulfilled

The documentation is sufficient, only the Online-Help-functionality should be slightly improved; but this objection is to be seen in relevance to the professional background of the tool and therefore nearly negligible.

6.6.2.9 UIP9: fulfilled

The tool has been given the professional touch of some of the supported RSS, is straight forward and gives maximum workflow support.

6.7 PrestoSpace Website(s)

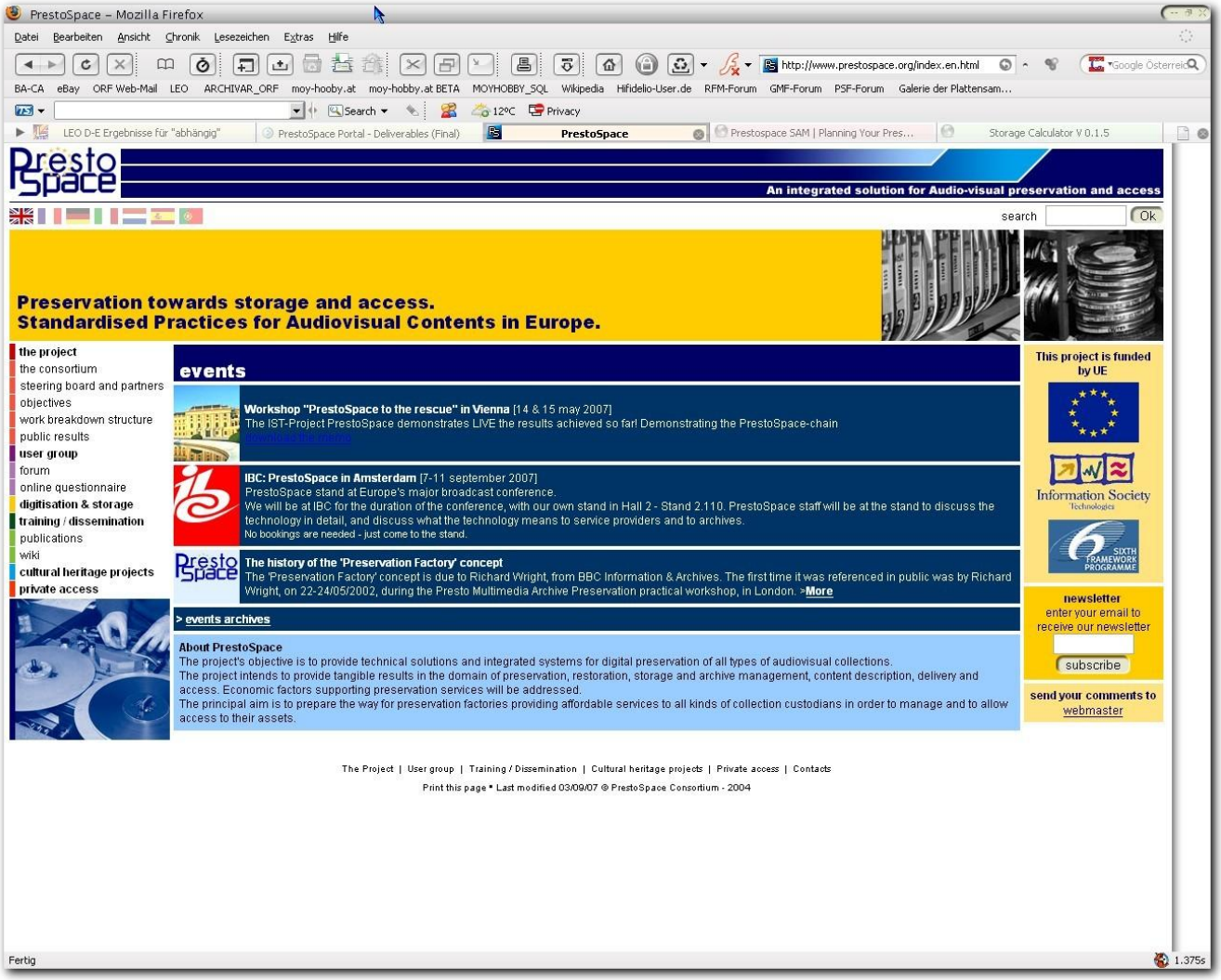


Figure 38 – PrestoSpace Main Site

The Project-Website(s) of PrestoSpace is the main information-source for interested parties and knowledge-pool for help-seeking archive-personnel and responsible units for preservation of audio-visual cultural heritage.

6.7.1 Description

The main Website of the IST-funded Integrated Project PrestoSpace is the entry-point to all public available information on and knowledge cumulated in the project.

Figure 39 – Website-languages

The Site is multilingual, providing information on the project in the main languages in Europe (English, French, German, Italian, Dutch, Spanish and partly Portuguese).

Figure 40 – Main Menu

In addition to giving information on the project, the site provides access to a thematically dedicated forum[xxxiv], the WIKI-pages (see 5.5), online-videos on training, tutorials on digitisation and storage, publications and information on other projects in the domain of cultural heritage.

6.7.2Checking of the User-Interface-Principles UIP

6.7.2.1 UIP1: nearly fulfilled

The main menu is providing full control to the user, giving him the room to reach all parts of the site from all subparts; links/menu-items leading to external sources and/or independent sub-sites should be labelled more clearly.

6.7.2.2 UIP2: partly fulfilled

The constant growth of the information and functionalities of the site lead to a partly inconsistent behaviour; independent sub-parts like the WIKI open up in the same window, while others like the forum force a new window to be opened. But on the sub-pages of the main site a high consistency has been diagnosed.

6.7.2.3 UIP3: fulfilled

The use of a colour-code helps the user to keep track of his movements through the site (see 5.10.2.5).

6.7.2.4 UIP4: fulfilled

The interface provides real-life-metaphors when feasible (printer-symbol) and familiarity by following the common web-standards.

6.7.2.5 UIP5: nearly fulfilled

The colour-code used in the main-menu (figure 29) and the corresponding sub-pages helps the user to get a permanent feedback on his position in the site; unfortunately this feedback is missed when certain independent sub-sites are entered.

6.7.2.6 UIP6: nearly fulfilled

Only by choosing a non-marked “outside”-link the user can be confused by leaving the common design and page-standard.

6.7.2.7 UIP7: nearly fulfilled

See 5.10.2.6; the misguided user can get lost at an independent site (Forum, WIKI, etc.), but can always return via a link by the PrestoSpace-logo.

6.7.2.8 UIP8: nearly fulfilled

No special help on the site is provided, but all parts are self-explanatory. Since the site follows the common web-rules and standards, the lack of a special help-file or functionality is negligible.

6.7.2.9 UIP9: fulfilled

All functionalities given are simple and following common standards; the user is mostly kept focused on the important parts (= information). The aesthetical value of the used colour-code is worth discussing, but nevertheless useful and convenient.

7 SUMMARY

Most of the User-Interfaces of Content-Retrieval and browsing as well as most of the other tested user-interfaces fulfil the main design-rules.

The main violations happen in UIP8 – Help & Documentation, but it can be seen as an implication of the fact, that the implementation of help-files and documentation-sources take place rather late in the development-phase of a tool, so that hopefully some of the gaps will be closed soon.

Other violations like in UIP6 – Error Prevention are based on the fact, that some of the interfaces are rather complex and proper training is a need. Nevertheless some error-prone situations in some interfaces should be taken in consideration to be eased or avoided.

8 ANNEX A

8.1 Non-PrestoSpace User-Interfaces

This additional part just provides some screenshots from non-PrestoSpace-User - Interfaces of Content – Management - Systems, Media – Asset -Management systems, and alike, with a focus on the Content-Retrieval- and Browsing-interfaces.

Obvious violations of the UIP's and examples of good user-interface design has been looked for and pointed at; the reader is invited to look at those accessible for the public (and other examples known by him) to proceed and use the UIP's for himself to gain personal experience.

Thus the significance and rating of the PrestoSpace-interfaces in comparison with those in use elsewhere in the domain should be shown.

8.1.1 FESAD – TV-CMS of the ARD (Germany)

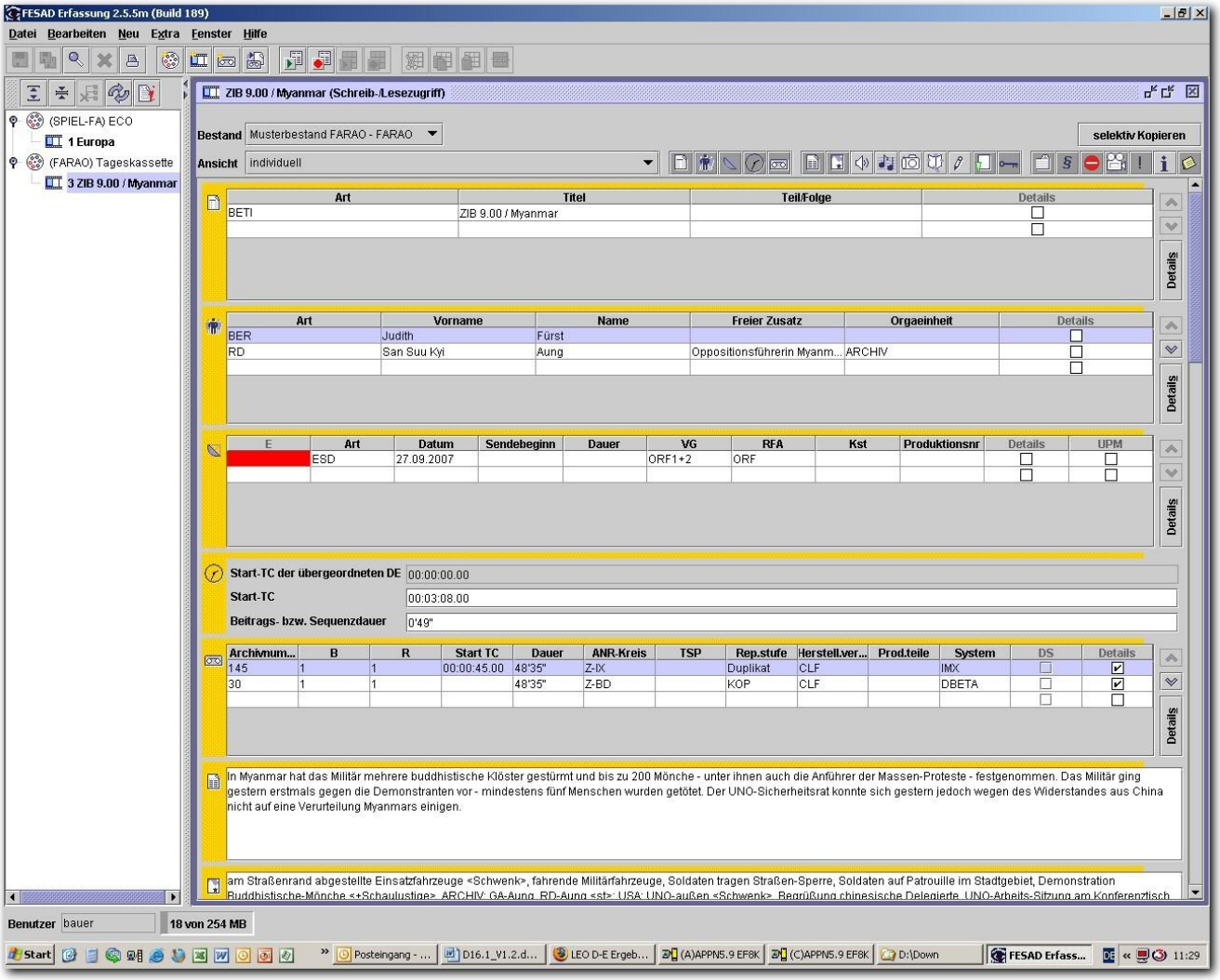


Figure 41 – FESAD Acquisition Platform

FESAD is a highly specialised system and aims only for the use of professional users and highly trained staff. The abundance of functionalities and controls, subpages and information-areas is controlled by a highly flexible and powerful customization-facility.

On of the biggest problems for FESAD is the violation of UIP5, as the non-trained user is easily lost in the mere flood of functionalities and windows.

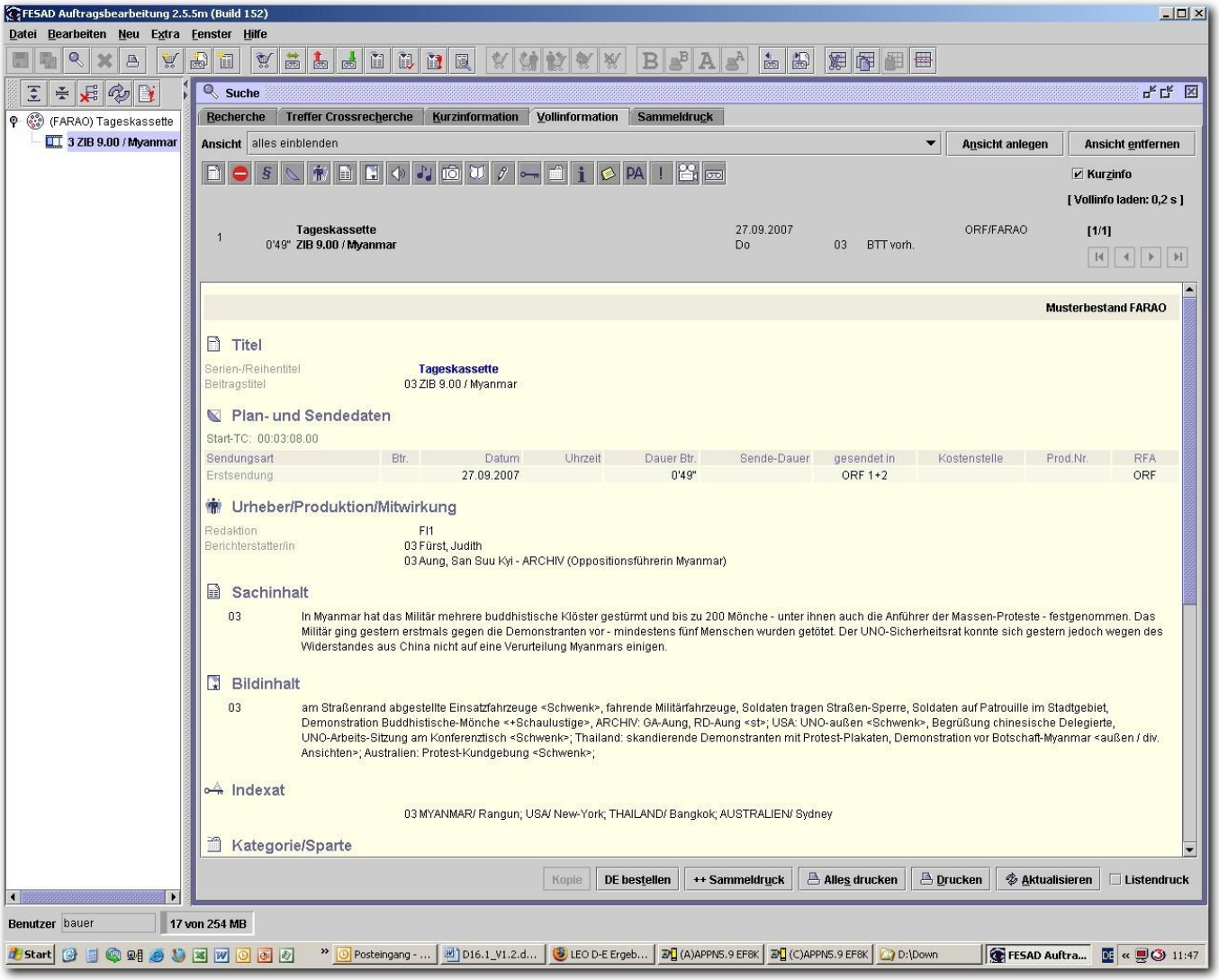


Figure 42 – FESAD "Full Info"

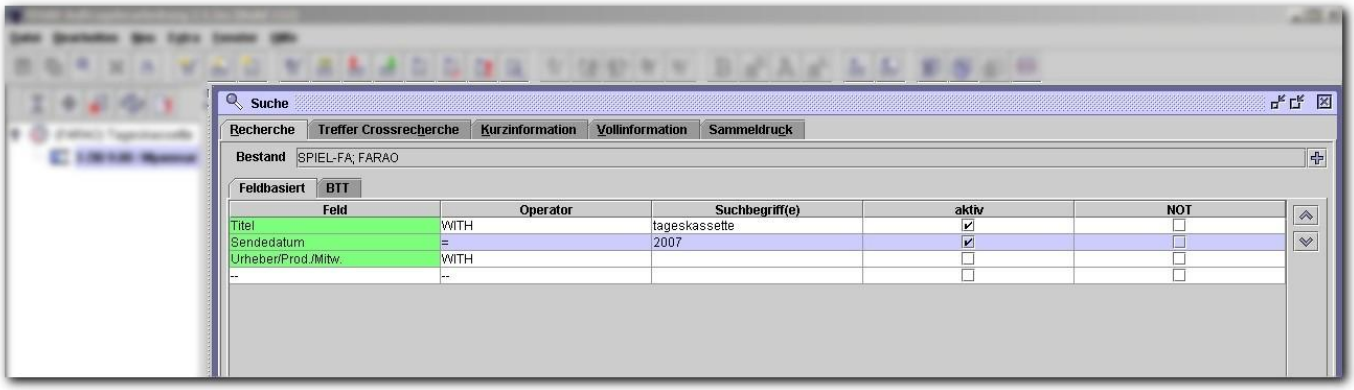


Figure 43 – FESAD Retrieval

The Retrieval-functionalities are also highly customizable and very flexible (UIP1, UIP4).

8.1.2mARCo – Retrieval-tool (Data-broker) of ORF (Austria)

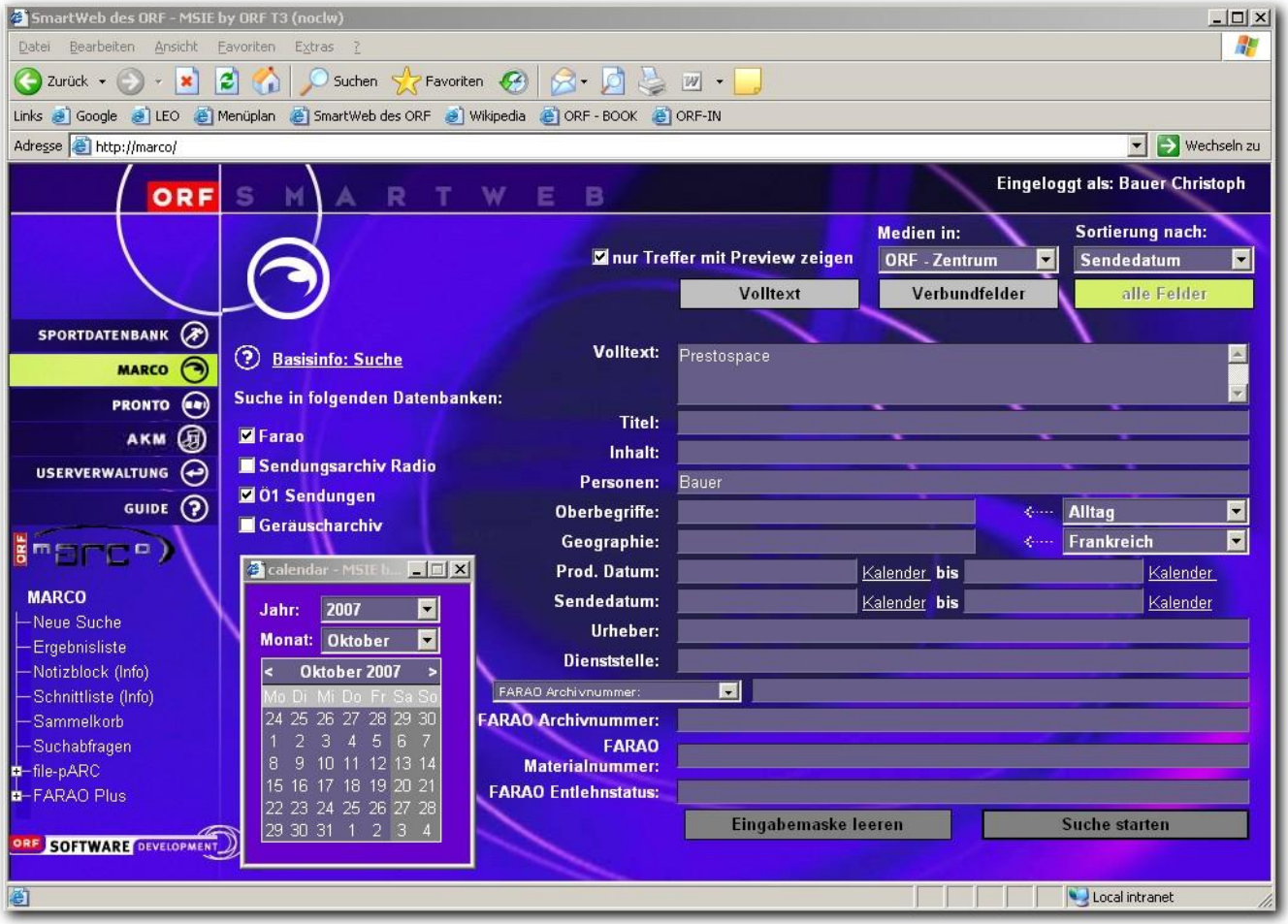
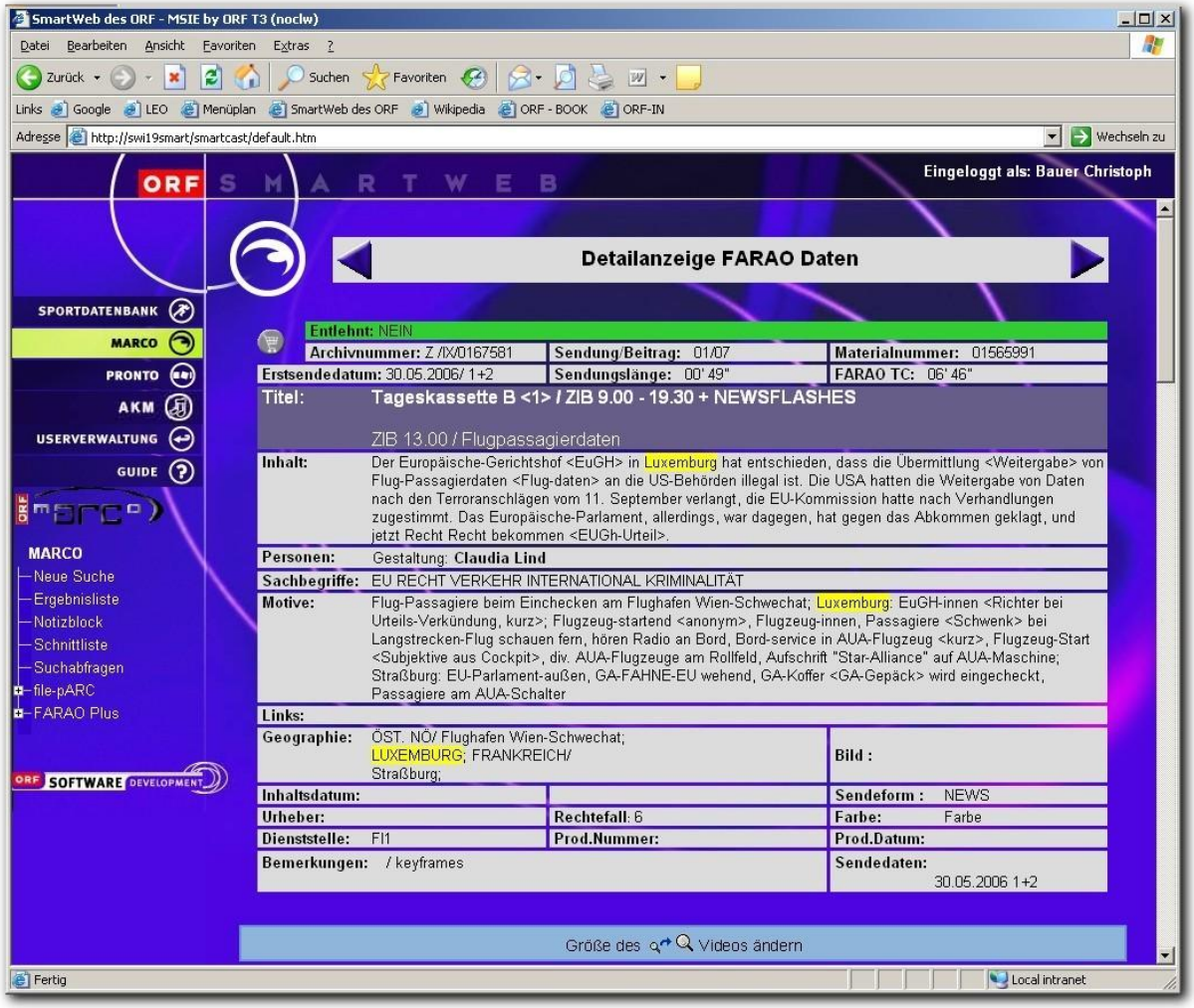
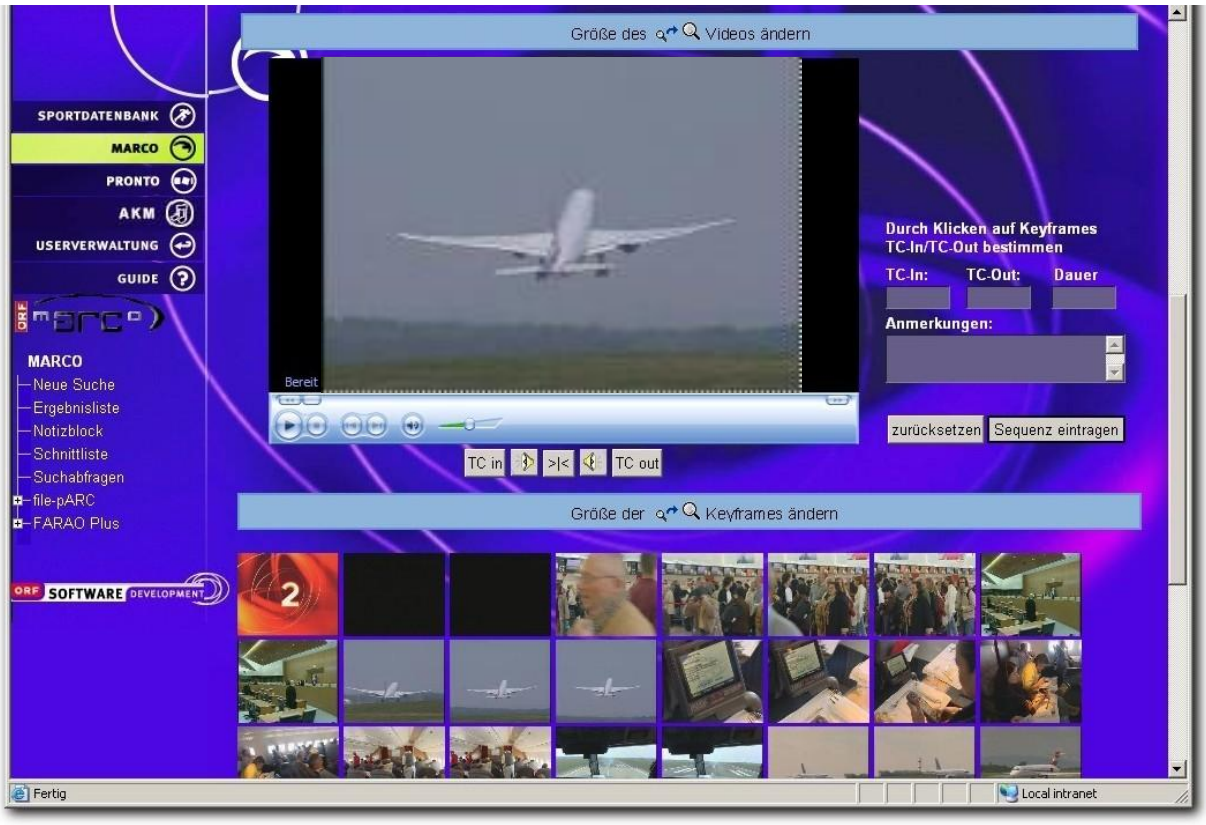


Figure 44 – mARCo - Retrieval

ORF's mARCo has been developed to make a number of different systems searchable via one interface, and to add multimedia-functionalities to the old, host-based TV-database FARAO.

Whilst FARAO is a tool for the archive-specialist, mARCo was meant to support the naïve user as well; therefore functionalities, not available in the old (underlying) system, has been introduced like: customizable search-interface, keyframes, browsing-video, highlighting, and many more.


SmartWeb des ORF - MSIE by ORF T3 (noclw)
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten
Links Google LEO Menüplan SmartWeb des ORF Wikipedia ORF - BOOK ORF-IN
Adresse http://swi19smart/smartcast/default.htm
Wechseln zu
ORF SMARTWEB
Eingeloggt als: Bauer Christoph
Detailanzeige FARAO Daten
SPORTDATENBANK
MARCO
PRONTO
AKM
USERVERWALTUNG
GUIDE
MARCO
Neue Suche
Ergebnisliste
Notizblock
Schnittliste
Suchabfragen
file-pARC
FARAO Plus
ORF SOFTWARE DEVELOPMENT
Entlehnt: NEIN
Archivnummer: Z /IX/0167581
Sendung/Beitrag: 01/07
Materialnummer: 01565991
Erstsendedatum: 30.05.2006/ 1+2
Sendungslänge: 00' 49"
FARAO TC: 06' 46"
Titel: Tageskassette B <1> / ZIB 9.00 - 19.30 + NEWSFLASHES
ZIB 13.00 / Flugpassagierdaten
Inhalt: Der Europäische-Gerichtshof <EuGH> in Luxemburg hat entschieden, dass die Übermittlung <Weitergabe> von Flug-Passagierdaten <Flug-daten> an die US-Behörden illegal ist. Die USA hatten die Weitergabe von Daten nach den Terroranschlägen vom 11. September verlangt, die EU-Kommission hatte nach Verhandlungen zugestimmt. Das Europäische-Parlament, allerdings, war dagegen, hat gegen das Abkommen geklagt, und jetzt Recht Recht bekommen <EUGh-Urteil>.
Personen: Gestaltung: Claudia Lind
Sachbegriffe: EU RECHT VERKEHR INTERNATIONAL KRIMINALITÄT
Motive: Flug-Passagiere beim Einchecken am Flughafen Wien-Schwechat; Luxemburg: EuGH-innen <Richter bei Urteils-Verkündung, kurz>; Flugzeug-startend <anonym>, Flugzeug-innen, Passagiere <Schwenk> bei Langstrecken-Flug schauen fern, hören Radio an Bord, Bord-service in AUA-Flugzeug <kurz>, Flugzeug-Start <Subjektive aus Cockpit>, div. AUA-Flugzeuge am Rollfeld, Aufschrift "Star-Alliance" auf AUA-Maschine; Straßburg: EU-Parlament-außen, GA-FAHNE-EU wehend, GA-Koffer <GA-Gepäck> wird eingecheckt, Passagiere am AUA-Schalter
Links:
Geographie: ÖST. NÖ/ Flughafen Wien-Schwechat; LUXEMBURG; FRANKREICH/ Straßburg;
Bild :
Inhaltsdatum:
Sendeform : NEWS
Urheber:
Rechtefall: 6
Farbe: Farbe
Dienststelle: FI1
Prod.Nummer:
Prod.Datum:
Bemerkungen: / keyframes
Sendedaten: 30.05.2006 1+2
Größe des Videos ändern
Fertig
Local intranet


Größe des Videos ändern
SPORTDATENBANK
MARCO
PRONTO
AKM
USERVERWALTUNG
GUIDE
MARCO
Neue Suche
Ergebnisliste
Notizblock
Schnittliste
Suchabfragen
file-pARC
FARAO Plus
ORF SOFTWARE DEVELOPMENT
Bereit
Durch Klicken auf Keyframes TC-In/TC-Out bestimmen
TC-In:
TC-Out:
Dauer
Anmerkungen:
zurücksetzen
Sequenz eintragen
TC in
>|<
TC out
Größe der Keyframes ändern
Fertig
Local intranet

Although most of the UIP's seem to be fulfilled by mARCo, the extensive use of background-graphics and dominant colour can be rated as violation or faulty implementation of UIP9.

On the other hand, more important UIP's for the professional workflow like UIP1-7 are very well implemented; only the help-functionalities should be developed further on.

8.1.3YouTube

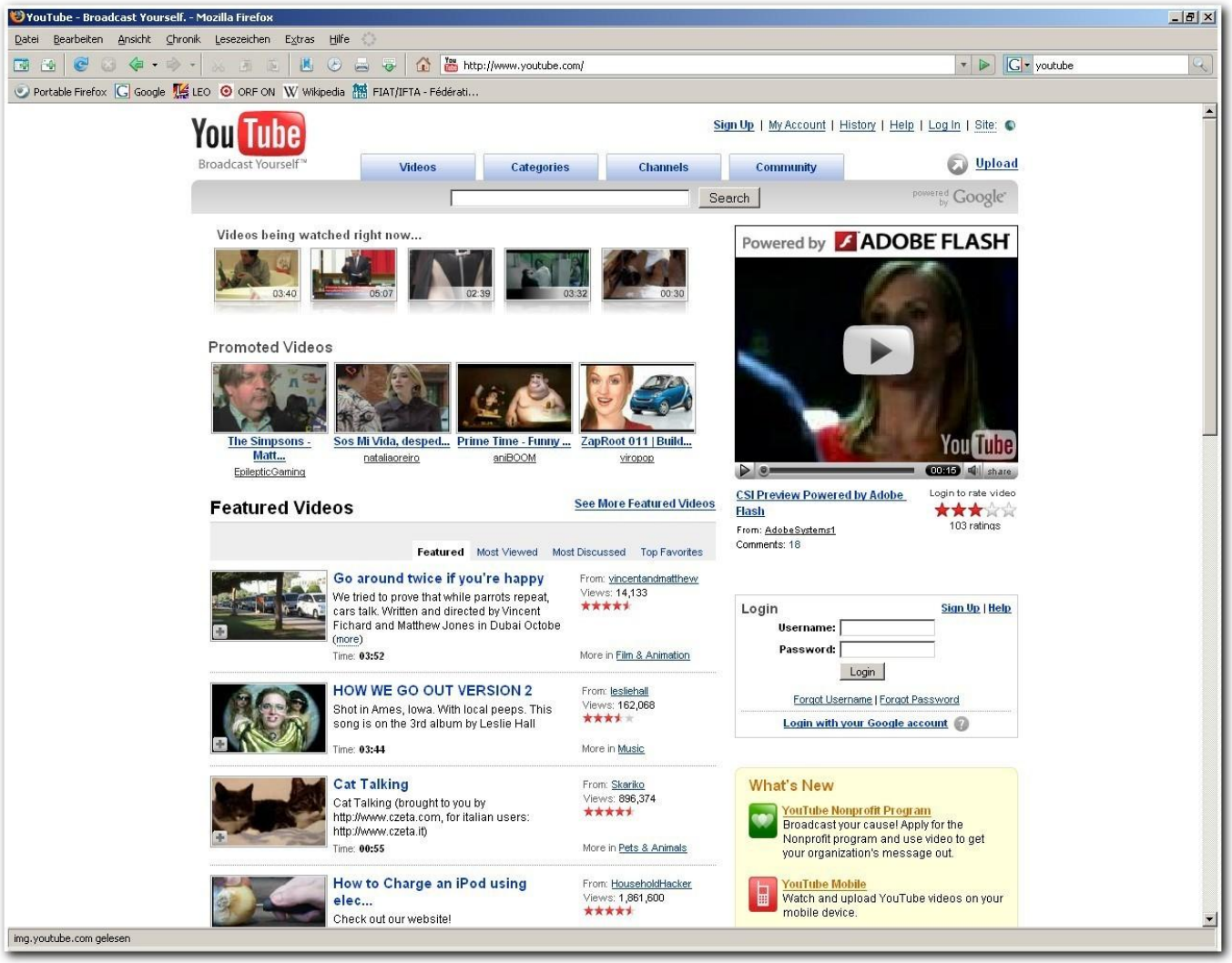


Figure 45 – YouTube / VideoChannel, Frontpage

Google®'s YouTube® is currently the most well known Internet-Video-Channel worldwide. The platform for both private and professional content-provider offers a maximum of "entertainment" and "coolness" in their design, but only a minimum of accessibility in terms of retrieve-functionalities. Nevertheless the minimalism in the current design of the User-Interface seems to support the main goal of the site: to produce traffic and attract users to visit the sponsor/advertisement-links. From the UIP-side of view most of them are fulfilled, but only in a rudimental way; the "all-inclusive"-approach retards the full implementation (e.g. UIP1).

Professional content-providers started to present their content on YouTube as well to attract users to visit the original sites of them (see the example of Beeld en Geluid).

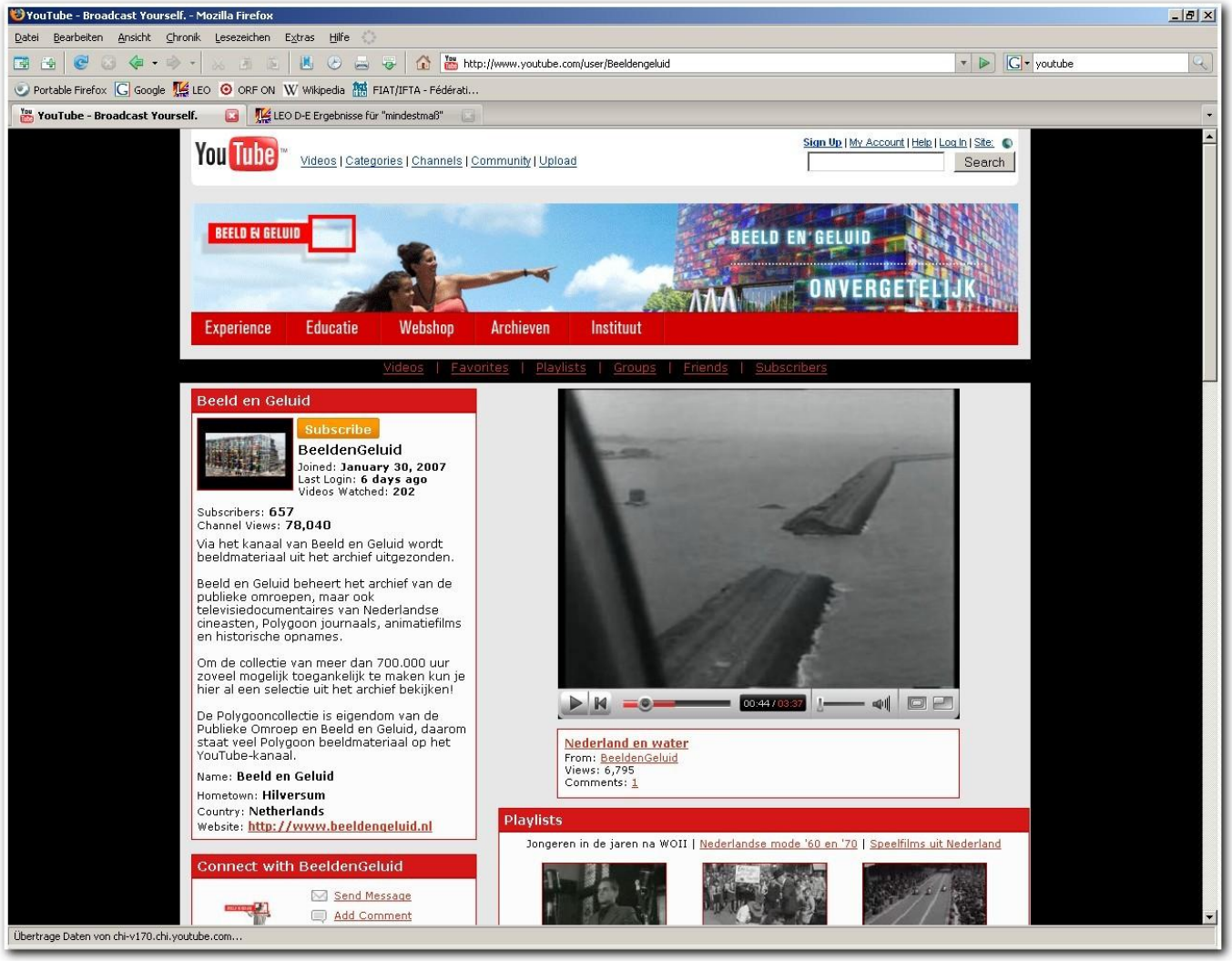


Figure 46 – YouTube-Channel of Beeld en Geluid (Netherlands)

8.1.4Archives@Risk

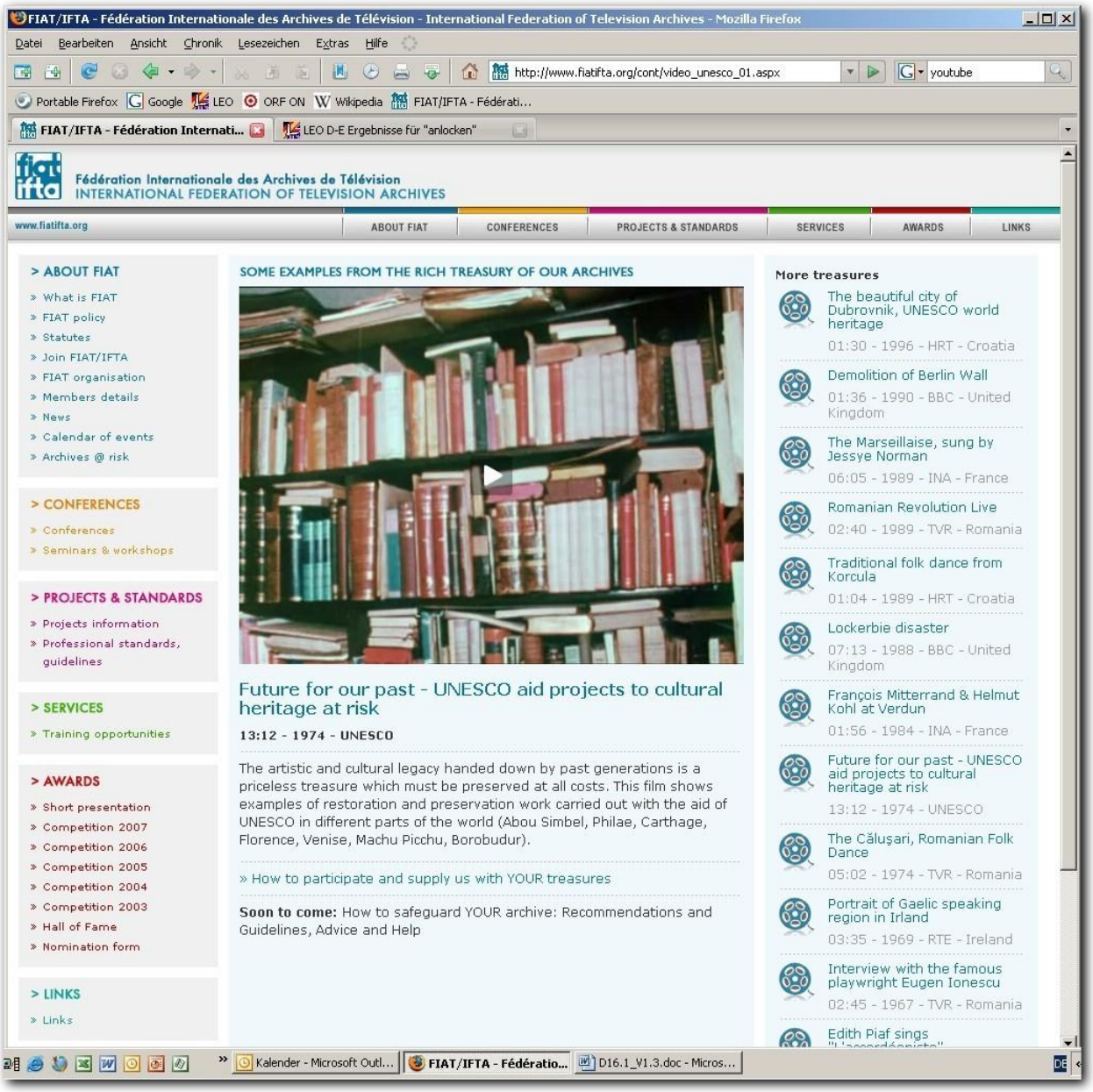


Figure 47 – Videopage of the first edition of A@R

The first edition of the videopage of the UNESCO-EBU-WBU-FIAT/IFTA-project Archives@Risk is another example of a very simple and basic UI-design; as the project will enhance this site in the upcoming years, it will be quite interesting to see how they'll implement further functionalities and how they'll proceed with the implementation of UI-rules, as most of the members of the project-board are associations of professional content-providers.

8.1.5Birth-of-TV

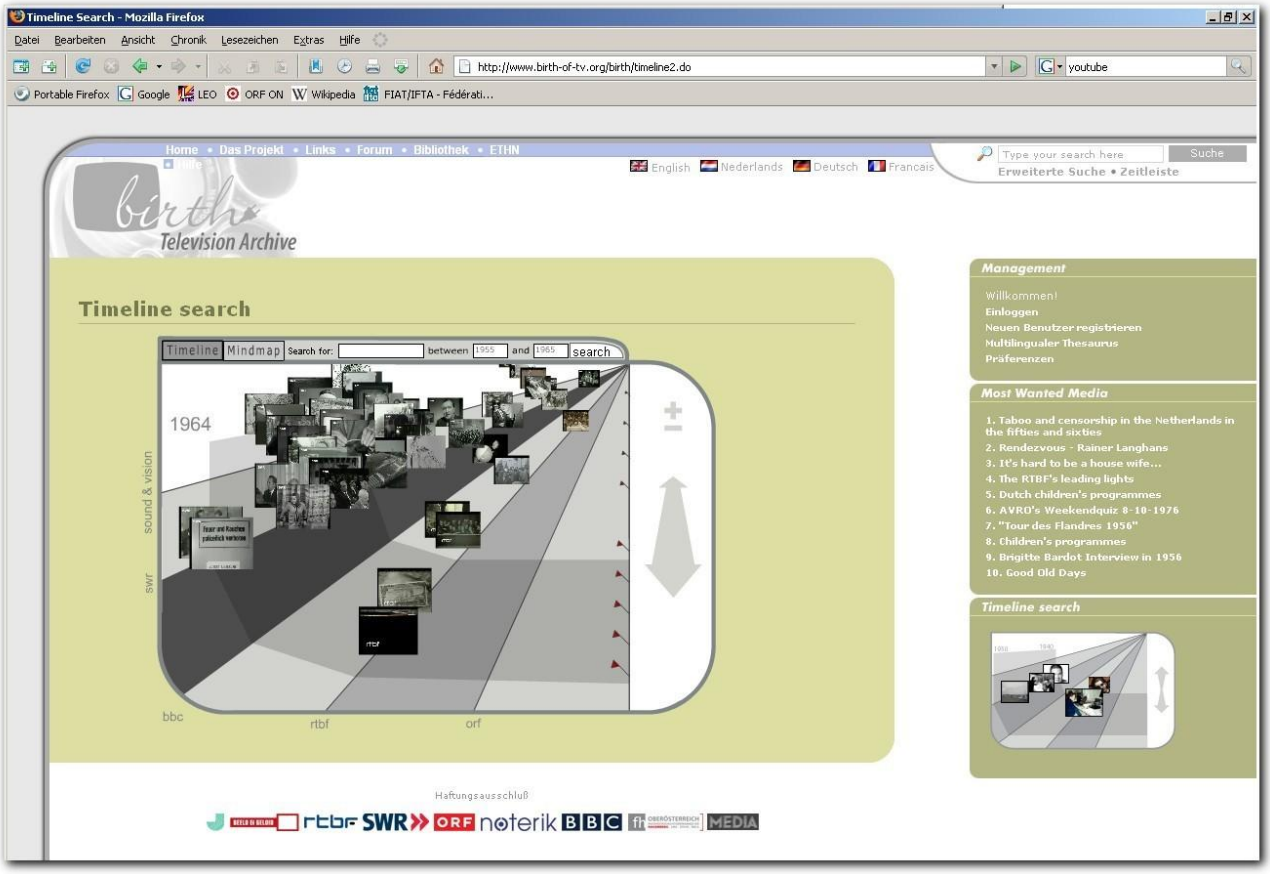


Figure 48 – Birth-of-TV / Timeline-Search

The Birth-of-TV-project (co-funded by MEDIAplus-program) shows some great examples of new and fresh attempts in UI-design; the “Timeline-Search” (see figure 48) gives the user the opportunity to “travel through time”, the items are represented via single keyframes. The “Advanced-Search” (figure 49) and the presentation of the single item (figure 50) are more up to common standards. The designers of BIRTH fulfil most of the UIP’s, only some implementation-lacks can be found for UIP3, 5 and 7.

As some members of the Birth-consortium continued their work in the “Video-Active”-project (eContent-Plus), it will be very interesting to see how the further development of the design-approach will be.

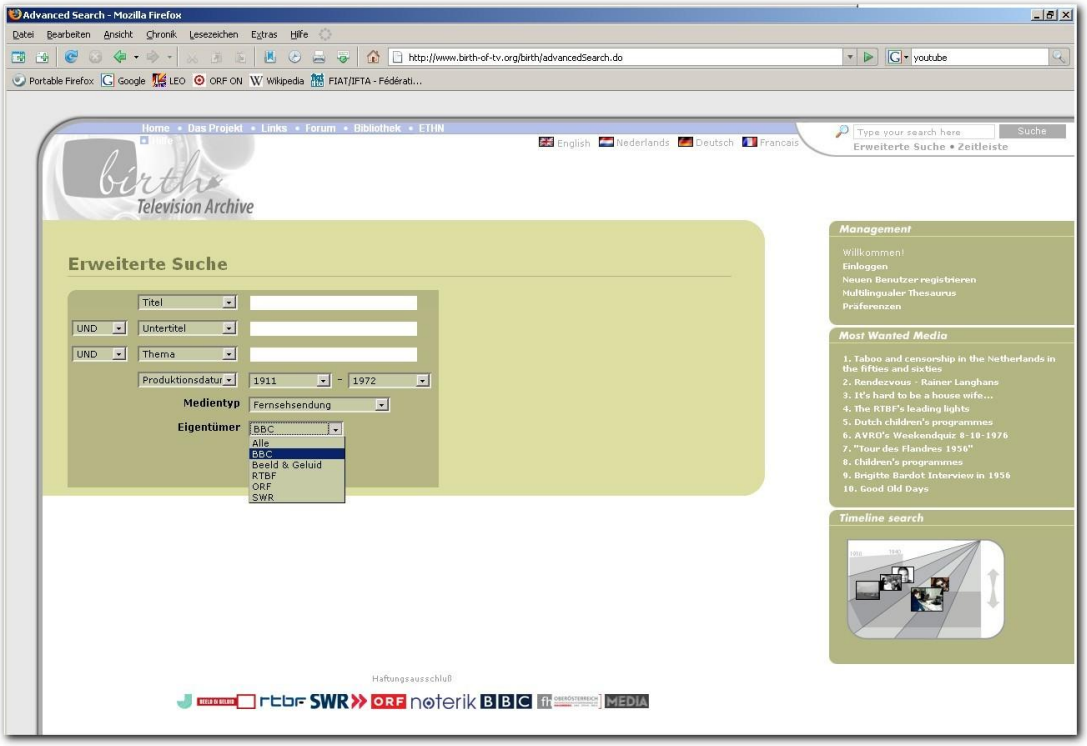


Figure 49 – Birth-of-TV / Advanced Search

Figure 50 – Birth-of-TV / Single Item

9 ANNEX B

9.1 Glossary

TERM	DESCRIPTION
AV	Audiovisual
Browsing	To scan, to casually look through in order to find items of interest, especially without knowledge of what to look for beforehand; in the audiovisual context: to navigate through AV-content, mostly via a representative of the original AV-content (LQ-video, keyframes, LQ-audio)
CMS	Content Management System
Content	Audiovisual program(s), productions or parts of productions, stored on audiovisual media or in electronic files.
EDOB	EDitorial OBject
GAMP	Generic
GUI	Graphical User Interface, see also UI
ISO	International Organization for Standardization, an international standard-setting body
Keyframes	Single frames drawn from videostreams (on a regular basis or based on amount of change of video-content, e.g. cut), representing the video via a lighttable.
Lighttable	All keyframes of a particular video
LQ	Low Quality
MAM	Media Asset Management
UI	User-Interface; aka Human-Machine-Interface; the user interface is the aggregate of means by which people (the users) interact with a particular machine, device, computer program or other (complex) tool (the system).
UIP	User Interface Principle - common denominator of the introduced User-Interface Rules and Recommendations
WIKI	A collaborative website which can be directly edited by anyone with access to it
WYSIWYG	“What You See Is What You Get” - used in computing to describe a system in which content during editing appears very similar to the final product

9.2 Table of figures

[i] Mandel, T. 1997. The Elements of User Interface Design. John Wiley & Sons: 5.1-5.28

[ii] Hansen, W. 1971. User Engineering Principles for Interactive Systems. AFIPS Conference Proceedings 39. AFIPS Press: 523-532.

[iii] Rubenstein, R. and H. Hersch. 1984. The Human Factor: Designing Computer Systems for People. Massachusetts: Digital Press.

[iv] Joiner, D. 1998. A Summary Of Principles For User-Interface Design. San Francisco Press

[v] Tognazzini, B. 1991. Tog On Interfaces. Addison-Wesely

[vi] Myers, I. Briggs, K. 1995. Gifts Differing : Understanding Personality Type. Davies-Black Publishing. --> Myers-Briggs type indicator

[vii] Apple Div. 1993. The Macintosh Human Interface Guidelines. Addison-Wesely

[viii] Commodore Amiga Div. 1991. The Amiga User Interface Style Guide. Addison-Wesely

[ix] Stallman, R. 1976ff. Open-Source Text-Editor for UNIX, GNU/Linux, MAC-OS and Windows-OS's

[x] Laurel, B. 1990. The Art Of Human Interface Design. Addison-Wesley Professional

[xi] Nelson, T. 1988. Computer Lib., Microchips

[xii] Rotsler, W. (July 3, 1926 - October 8, 1997)

[xiii] Shneiderman, B. 1995. Graphical User Interface Design And Evaluation. Addison-Wesely

[xiv] Mayhew, D.J. 1992. Principles and Guidelines in Software User Interface Design. Prentice Hall PTR

[xv] IBM. 1998. Design principles for tomorrow. http://www-03.ibm.com/easy/page/6

[xvi] Kruse, K. 2000. Effective User Interface Design: The Four Rules. ASTD

[xvii] Hix, D. & Hartson, H.R. 1993. Developing User Interfaces: Ensuring usability through product and process. Wiley, NY.

[xviii] Nielsen, J. 2005. Ten Usability Heuristics, NNG Fremont

[xix] Nielsen, J. & Tahir, M. 2001. Homepage Usability: 50 Websites Deconstructed. New Riders Publishing, Indianapolis

[xx] Internet Magazine. 2005

[xxi] Stuttgarter Zeitung. 2006. http://www.stuttgarter-zeitung.de/stz/page/detail.php/228123

[xxii] Nielsen, J & Molich, R. 1990. Heuristic evaluation of user interfaces. ACM CHI'90, Seattle. 249-348

[xxiii] http://www.sapdesignguild.org

[xxiv] D18.2 The Publication Platform. Deliverable No.18.1 in PrestoSpace

[xxv] Cunnigham, W. 1995. Original description on www.wiki.org

[xxvi] PrestoSpace. 2006. About the Preservation Project Cost Calculator. page 1

[xxvii] http://www.hs-art.com/download/mir/MIR_Manual.pdf

[xxviii] http://www.hs-art.com/download/mir/MIR_Manual_Light.pdf

[xxix] http://www.hs-art.com/download/mir/MIR_Shortcuts.pdf

[xxx] D8.2 Restoration Management Tool. PrestoSpace 2006

[xxxi] D10.1 RSS1 Visual Restoration Software Subsystem. PrestoSpace 2006

[xxxii] D10.2 RSS2 Visual Restoration Hardware Subsystem. PrestoSpace 2006

[xxxiii] D10.3 RSS3 Audio Restoration Software Subsystem. PrestoSpace 2006

[xxxiv] http://www.prestospace.org/user_group/forum/index.php