MAGISTERARBEIT

Titel der Magisterarbeit

User Interfaces for the Organisation and Representation of Unstructured Data

Verfasser

Bojan Milicevic

angestrebter akademischer Grad

Magister der Sozial- und Wirtschaftswissenschaften (Mag.rer.soc.oec)

Wien, October 9, 2008

Sudienkennzahl lt. Studienblatt:	A 066 926
Sudienrichtung lt. Studienblatt:	Wirtschaftsinformatik
Betreuerun/Betreuer:	Univ.Prof.Dr. Wolfgang Klas
In Zusammenarbeit mit:	Mag. Bernhard Schandl

Abstract

The semantic approach to data storage, management and extraction is a promising concept that has constantly been gaining importance during the past years. However, the visualization of semantic data systems as well as the functionalities of their interfaces still contain some obstacles for both, the interaction designer and the interacting user. In order to accommodate to those obstacles, the interaction designer has to come up with an intuitive and efficient user interface design, introducing inexperienced users to data representation and extraction in a semantic environment. The intention of this work is to provide an overview over the disciplines of Human Computer Interaction and Interaction Design, with the intention to mediate the essential human-based and technical processes influencing the design of interaction. Methods and techniques for the design process will be presented, as well as elements and components of visual design and interaction, with a focus on the representation of semantic content. Finally, a web-based graphical user interface for SemDAV repositories will be introduced, combining the gained insights with the ongoing development in web design.

Contents

Chapter 1

Introduction

1.1 Motivation

These days more than ever, digital data is an essential element of our everyday lives. The mass of personal as well as professional data is constantly rising, leaving us with some unresolved issues, especially concerning the storage, management and aimed retrieval of such data masses. By taking a glance over nowadays IT-landscapes, it soon becomes obvious that in spite of numerous innovations in the fields of data storage and management that occured over the past decade, still the same hierarchical data-systems that we had thirty years ago prevail. Nevertheless, the semantic approach to data management is a quite promising concept that has manifested over the past years. The goal and purpose of semantic data-systems is to provide the user with descriptive knowledge instead of just providing him with attributes of some specific data-objects. Instead of a hierarchical structure serving as the most basic formation of data, the data-objects should be stored by using multidimensional relations to each other, even at the most abstract level of storage. Although the semantic approach could rightly be regarded as old news, there are still very few systems and applications that are really applying this concept, most of them still keeping a hierarchical formation as their skeletal structure.
However, there are some issues that need to be considered when seriously allowing for the assertion of strictly semantic data-systems. First of all, this vital change in the way that knowledge is stored and accessed will probably afford some accommodations in the ways the average user refers to the stored data. In order to simplify those accommodations as much as possible, it is crucial for the UI design to attempt to provide a familiar environment and at the same time to introduce the user to the new ways and possibilities of data management

and extraction. Another risk of semantic data-systems is the loss of data due to insufficient or inadequate data visualization. Thus the UI should aim to provide adequate visualization and data-browsing techniques in order to close possible gaps in the data structure.

1.2 Contributions

The first objective of this work is to give an introduction and a generalized overview over the disciplines of Human Computer Interaction and Interaction Design, with the intention to mediate the essential human-based and technical processes influencing the design of interaction. By analyzing interaction related human-factors as well as the methods and the basic elements of interaction design, this work aims to create a set of requirements and guidelines ensuring an intuitive and user-friendly graphical user interface. Also, the most common and prevalent methods and components for the visualization of unstructured semantic content will be introduced.
The second objective will be the testing and the analysis of some well chosen case studies, concentrating on innovative design, intuitive visualization and effective data representation. The goal is to develop a list of well working concepts, and others that may be rather inappropriate.
Finally, a design and implementation of a web user interface for SemDAV repositories will be presented. SemDAV is a protocol based on HTTP, and its purpose is to provide methods for an exchange of content and associated semantic metadata. The user interface should be intuitive and efficient, combining the influences and insights that emerged throughout the work with up to date methods and techniques of web design.

Chapter 2

HCI and Human Factors

2.1 Introduction

This chapter aims to mediate an overview over the discipline of Human Computer Interaction, with an emphasis on the human related aspects that are influencing the communication between the user and the machine. The main focus lies on the human diversity and on the attributes by which this diversity is determined, as well as on some basic introduction to the aspects of human perception and information processing. These aspects are independent from a specific type of the human-machine interaction and therefore they always remain the same as they are related to the human nature, representing the core and the starting point of any kind of human-computer related interaction.

2.2 Human Diversity and Characteristics

The majority of human interfaces are designed for a "typical" user, waiving most of the other user groups not matching the norm. However, the diversity of human characteristics, abilities and backgrounds has a crucial impact on the design of interactive systems. Not only are there different physical and cognitive abilities, but also cultural and international diversity, personality differences and different kinds of human disabilities that should be considered. Accommodating those deviations should be the goal for every designer.

2.2.1 Cultural and International Diversity

A major part of human diversity is simply a result of the different backgrounds people have. Across the globe we will find a huge amount of different languages, different cultural predispositions and different racial or ethnic backgrounds. Obviously all these factors will have an impact on the way people use and look at interfaces. For instance, people who are used to reading Chinese, Japanese or Arabic letters will scan a screen in a different order as people used to reading Latin descending languages. Also, depending on the culture, people may have different views concerning the layout and the graphical design of a user interface [Sch98].
"*The computer codes, interfaces, etc. - are culturally neutral [tools], ones that allow transparent communication between all cultures"* [ES98]. Especially web design should be culturally neutral and transparent, which is rarely the case. "*Communication - whether it is mass mediated, interpersonal or nonverbal - is inseparable from culture: each shapes and is shaped by the other"* [ZF03]. Although more and more similarities in content and design can be found through the World Wide Web, which is suggesting a cross-cultural tendency in the design process, some characteristics like the choice of language or the frequent usage of animations, are still culture bound. Here are some examples for cultural dependant content [Sch98]:

- Characters, numerals, special characters, and diacriticals
- Left-to-right, versus right-to-left versus vertical input reading
- Date and time formats
- Numeric and currency formats
- Weights and measures
- Telephone numbers and addresses
- Names and titles (Mr., Ms., Mme., M., Dr.)
- Social-security, national identification, and passport numbers
- Capitalization and punctuation
- Sorting sequences
- Icons, buttons, colours
- Pluralisation, grammar, spelling

- Etiquette, policies, tone, formality, metaphors

2.2.2 Personality

Personalities, personal taste and preferences can be quite divergent. Different personalities have a significant impact on the way people react to interfaces. Some users may prefer shiny, visually attractive interfaces, where others have their focus on the functionality and the outline. There can be many different interaction styles when concerning the different design possibilities implying graphics, the pace of interaction, data representation and so on. Thus, personality differences should be considered during the design process, especially when designing for a specific community of users. A possible technique to identify basic preferences of people is the Myers-Briggs Type Indicator (MBTI), which is based on the personality types presented by Carl Jung [Sch98]:

- *Extroversion - Introversion* Extroverts primarily apply to the external world, they react to external stimulation and their perception is sensitive to the things happening around them. Introverts on the other hand are oriented toward their inner world, they prefer to work alone and they rely primarily on their own opinions and ideas.
- *Sensing - Intuition* A person may either tend to rely on the process of sensing or on the process of intuition. Sensing types prefer to confide on the inputs provided by their five senses which means they tend to rely on the observable data and facts. They are rather precise at work and they are good at applying already known skills. Intuitive types however like the challenge of solving new problems. They are using rather abstract information and meanings, which are often unconscious to the mind.
- *Thinking - Feeling* People with a preference for Feeling can relate to other people's emotions, they are empathic and are trying to solve problems in a harmonic way, concerning everyone involved. Thinking types tend to solve problems impersonally, they are unemotional, reasonable and logical.
- *Judgment - Perception* This index describes how a person is reacting to the outside world. Judging types are relying on the process of thinking and they are following well made plans in order to solve their tasks. On the other hand however, they can sometimes seem a little static or inflexible. Perceptive types prefer new and unplanned situations; they rely primarily on their senses and intuition when solving problems.

Table 2.1: The 4 Dichotomies by Carl Jung

Extraversion	**I**ntroversion
Sensing	I**N**tuition
Thinking	**F**eeling
Judging	**P**erceiving

Table 2.2: The 16 MBTI Types

ISTJ	**ISFJ**	**INFJ**	**INTJ**
ISTP	**ISFP**	**INFP**	**INTP**
ESTP	**ESFP**	**ENFP**	**ENTP**
ESTJ	**ESFJ**	**ENFJ**	**ENTJ**

"*The purpose of the Myers-Briggs Type Indicator® is to make the theory of psychological types described by C. G. Jung (1921/1971) understandable and useful in people's lives. The essence of the theory is that much seemingly random variation in behaviour is actually quite orderly and consistent, being due to basic differences in the way individuals prefer to use their perception and judgment*" [Mye98].
The MBTI is an instrument based on the four dichotomies by Carl Jung and is used to identify the basic preferences of a person. By allocation those dichotomies, the MBTI tries to assign one of the 16 personality types 2.2.2 to a person, which doesn't mean that a person strictly represents a certain type, but that a person prefers or tends to a certain type. For further reading see [Cap02] and [Cap07].

2.2.3 Abilities and Disabilites

There is nothing like two same peas in a pod. Thinking about the whole array of human physical and cognitive divergences, it is obvious that concerning the design process, there can be no "average" user. The human physical variation is huge, and so are the different measures in dynamic actions. For instance, a rather tall user will have a different view angle to the screen than a rather small one, also a colour blind user will respond differently to a colourful interface than a user with no vision disorder. However, although the human physical abilities and their effects certainly have some impact on the design process of software interfaces, it is a rather small one. Human physical diversity would more likely be a part of ergonomic

research or human factors [Sch98].

2.2.4 Cognitive and Perceptual Abilities

For interface designers it is crucial to understand the cognitive and perceptual abilities of the users. Here is a classification of cognitive processes as suggested by the journal Ergonomics Abstracts [Erg07]:

- Short-term memory
- Long-term memory and learning
- Problem solving
- Decision making
- Attention and set (scope of concern)
- Search and scanning
- Time perception

Also, they are offering a set of factors affecting perceptual and motor performance:

- Arousal and vigilance
- Fatigue
- Perceptual (mental) load
- Knowledge of results
- Monotony and boredom
- Sensory deprivation
- Sleep deprivation
- Anxiety and fear
- Isolation
- Aging
- Drugs and alcohol

- Circadian rhythms

The reason why the perceptual and cognitive aspects are so important is that basically every design concerning interaction is, or should be, a design focused around the future users. No matter the kind of design the interaction designer strives for, nor the nature, form or purpose of the designing application, in the end all interaction will break down to the perceptual and cognitive abilities of the interacting user. Of course the designer cannot imply the same cognitive predispositions for a greater number of users, however it is possible to make certain assumptions concerning the perceptual and cognitive abilities according to the communities and the characteristics of the target groups.

2.2.5 Age

In general, the average user that interfaces are designed for is a younger one. Those younger users however eventually grow old, and with the slowly arising seniority, people may suffer some physical or cognitive losses. According to [GNZ02], here is a list of characteristics of older users when compared to the younger ones:

- The individual variability of physical, sensory, and cognitive functionality of people increases with increasing age.
- The rate of decline in that functionality (that begins to occur at a surprising early age) can increase significantly as people move into the "older" category
- There are different and more widely appearing problems with cognition, e.g. dementia, memory dysfunction, the ability to learn new techniques.
- Many older users of computer systems can be affected by multiple disabilities. Such multiple minor (and sometimes major) impairments can interact, at a human computer interface level to produce a handicap that is greater than the effects of the individual impairments. Thus research into accessibility focused on single impairments may not always provide appropriate solutions.
- Older people may have significantly different needs and wants due to the stage of their lives they have reached.
- The environments in which older people live and work can significantly change their usable functionality - e.g. the need to use a walking frame, to avoid long periods of standing, or the need to wear warm gloves.

- On a more positive note, older people can have access to a much wider experiences and knowledge of the world than younger people, and a more mature approach to problem solving.

It is a fact that physical, cognitive and sensory abilities simply do not remain static over time, whether this loss is caused by an accident, illness or aging. Also, the diversity in those disabilities certainly gains a wider range with the progressing age. Concerning interface design, this can be a quite difficult problem to solve when aiming for a generic system that can be used by all age groups. As a possible approach to design for this dynamic diversity, Gregor, Newell and Zajicak propose the User Sensitive Inclusive Design (USID) methodology, which will be discussed later in this work [GNZ02].

2.3 Human Information Processing

Cognition, or the human information processing, has always been an important topic regarding HCI. The way we gain and mediate knowledge, the creation of new ideas, our understanding and reasoning, this are all important processes concerning the interaction between men and machines. In the 1970s an information processing model was developed by Lindsay and Norman, which describes humans as information processors. The model tried to explain the movement of information in the human mind, from its input (everything that is sensed) to its output. The basic idea was that this movement happens via a series of ordered processing stages [JP96]:

- Stage 1: The external information is being encoded into an internal representation.
- Stage 2: The new representation from the input is compared to already stored representations.
- Stage 3: A response for the new stimulus is being selected.
- Stage 4: The response is being executed.

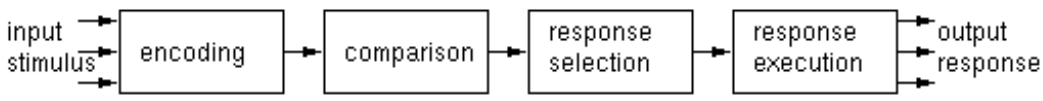


Figure 2.1: Human information processing stages [PHL77]

Later this model was extended [Bar88] with the processes of attention and memory, which have an effect on every of the four stages.

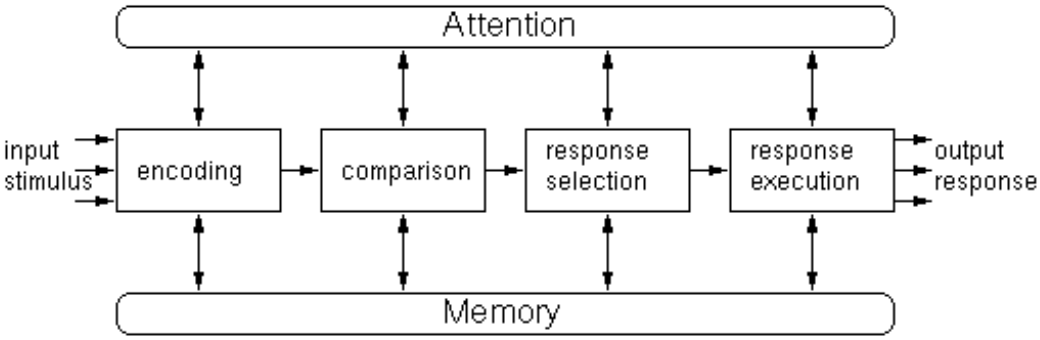


Figure 2.2: Extended stages [Bar88]

But there is also a quite different approach to human information processing. The multi-store model of memory [Atk68] divides the human perception into three separate memory stores [JP96]:

- **Sensory stores**: Sensory stores are short-term input buffers, where our senses store the received information before further processing takes place. However, only a small percentage of this information is actually being chosen for further processing, the rest is being overwritten.
- **Short-term memory store**: The short-term memory, or the working memory, is the area where the processing of information takes place. The capacity of this store is limited, and it can hold about seven information-chunks at the same time.
- **Permanent long-term memory store**: The information that is stored in the long-term memory is assumed to be permanent.

The information processing model was very influential in HCI, and helped to predict and conceptualize the user behaviour. One of the first user models, basing on the information processing model, was the model human processor (MHP). The MHP implies three interacting systems: the motor system, the perceptual system and the cognitive system.

Originating from the MHP, Card, Moran and Newell (1983) presented a family of models known as GOMS (Goals, Operations, Methods and Selection rules). In simple terms, GOMS

is a tool for describing a task and the user's ability and knowledge of how to perform that task by using the following criteria [Joh95]:

- Goals: What does the user want to accomplish and in which time frame? Goals are usually defined in a hierarchical manner.
- Operations: What actions does the software provide for the user in order to accomplish the desired task (buttons, menus, etc...)?
- Methods: Methods describe procedures for reaching a goal. They are usually a grouped together sequence of operations and sub-goals.
- Selection rules: If several methods are applicable for reaching a single goal, then selection rules are used in order to decide which method is to be applied.

The GOMS analysis is used to make quantitative as well as qualitative predictions concerning the usability of a system. For system designers, GOMS can evaluate rival design ideas at an early stage of design and suggest better design for critical system parts, which makes it a very powerful and effective tool.

2.4 Visual Perception

When considering human-computer interaction, visual perception is fundamental. Although the notion of an interface being always presented on a screen is slowly starting to change, nowadays the majority of human-computer interfaces are still based on visual representation an interaction. Therefore it is important to understand how our visual system works and interacts, considering all its abilities and disabilities. There are different theories attempting to explain the way we see, but roughly they can be summarized into two classes: the constructivist and the ecological approach. The assumption behind constructivism is that what we see is not a copy of the real world, but rather a model constructed through our representations, memories and expectations. Our visual system creates a model of our environment by transforming, enhancing or even discarding the visual input in order to provide us with a more stable a constant view of our environment. Hence, when we look at two-dimensional objects on a screen, what we perceive is a construct of our prior knowledge and of our expectations of what we should see [Cox05, JP96].

There are several principles that enable us to interpret objects as well organised patterns rather than single elements [JP96, ZTB06, Vis07]:

- Proximity: Objects that are very close to each other appear to be part of the same group or cluster. See Figure 2.3

Figure 2.3: Proximity

- Similarity: Objects that are similar in shape and colour will be seen as belonging to the same group. See Figure 2.4

Figure 2.4: Similarity

- Closure: Objects that are incomplete are completed by the mind, in order to see a figure we recognize. See Figure 2.5

Figure 2.5: Closure

- Continuity: We tend to see objects as part of a group that is defined by lines or curves rather than simply a set of dispersed objects. See Figure 2.6
- Symmetry: Symmetrical images are perceived as coherent figures. See Figure 2.7
- Common fate: Objects that are moving in the same direction are perceived as a single unit.

Figure 2.6: Continuity

Figure 2.7: Symmetry

These principles do not act independently but influence each other, creating a combination of all the grouping laws as the final picture. In contrast to the constructivist approach, the ecological or natural approach to perception implies that what we perceive is rather detected by our senses than constructed by our mind. The theory maintains that the visual input itself is sufficiently rich and meaningful, so that it can be perceived without further elaboration. Therefore the ecological approach is also known as direct perception. A fundamental concept of the ecological approach is affordance. The notion of affordance was coined by the perceptual psychologist James J. Gibson, but was introduced later on to the HCI community by Donald Norman (1988). According to Norman, the affordance of an object or system is the design aspect (i.e. a visual clue), which suggests the way that object or system should be used. The more the affordances of an object are obvious, the easier it is for the user to know how to interact with it [JP96, Soe07, VCvdV06].

2.5 Attention

Attention has a great significance in HCI and interaction design. The way we deploy, focus and divide our attention certainly has a considerable impact on the effectiveness of our interaction with a certain interface. A user interface should be able to drive the user's attention through the different tasks of the system, focusing his attention on the important information and trying to keep distraction as low as possible without preventing the user to perceive different or additional information.

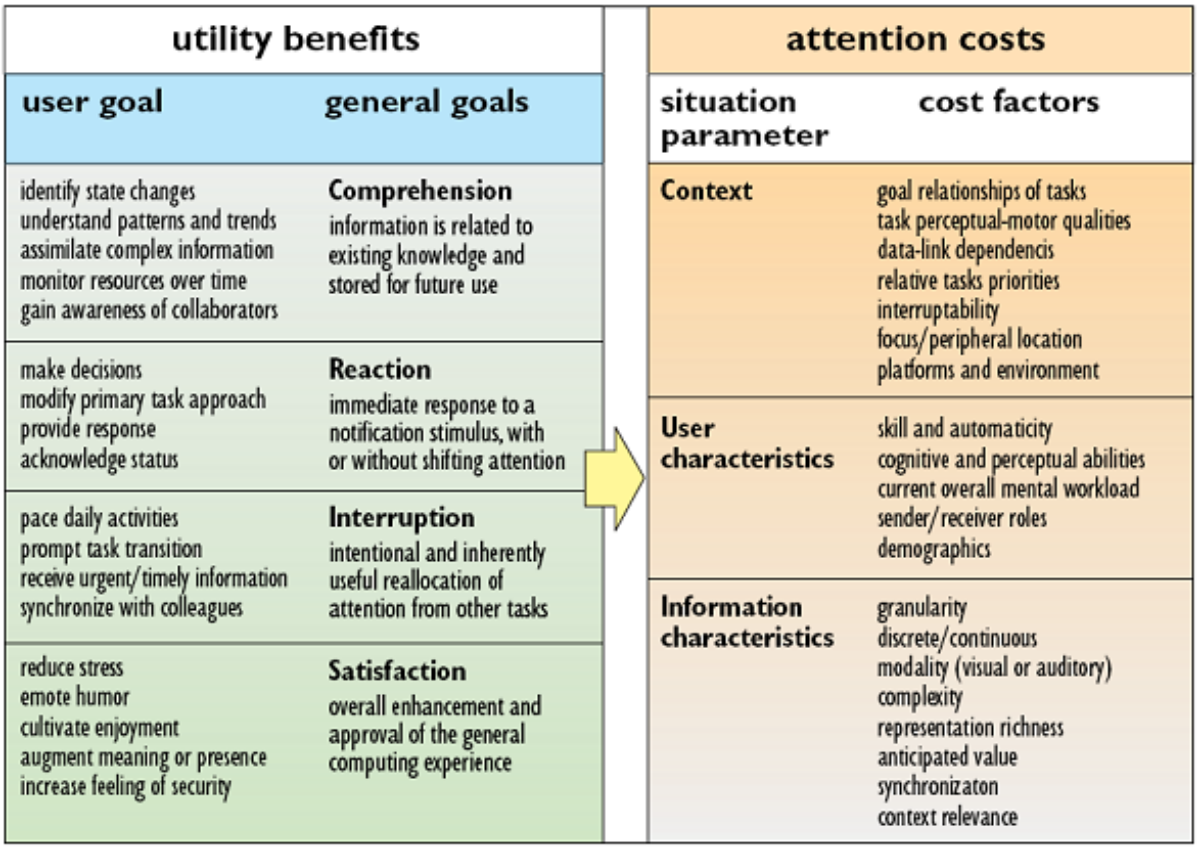


Figure 2.8: Utility Benefits - Attention Costs Tradeoff [MC03]

However, nowadays it is common for people to do multitasking rather than performing tasks in a serial manner. Although most people are quite flexible considering multitasking, they have to be able to switch between primary tasks, being the most important tasks at a certain time, and secondary tasks with a minimum of distraction. Hence, there is a need for effective but not interrupting notification systems, supporting the user to gain additional information without distracting him from the main task. However, in most cases it is not possible to notify the user about some valued information without diverting at least some of his attention, which leads to the following question: How much attention is the user ready to sacrifice for some specific information? In other words, the notification benefits for the user have to exceed the associated costs. "*The success of a notification system hinges on accurately supporting attention allocation between tasks, while simultaneously enabling utility through access to additional information* [MC03]". Figure 2.8 illustrates this attention-utility trade-off. On the left side we see user goals and the resulting sources of utility (comprehension, reaction, interruption, satisfaction), which can be thought of as critical parameters for system success. On the right side of the table we see factors causing the attention costs, determined by the

amount of attention removed from the primary task [MC03].

2.6 Memory

Just as perception and attention, memory is highly involved in our everyday life and activities. In fact, without memory we would not be able to do the simplest task because we simply wouldn't remember how to do it. But how exactly does the human memory work? Actually, the human memory is a very versatile instrument, allowing us to cope with the world around us by storing all the important sights, sounds, smells and feelings that we need to perform our everyday tasks. It is well known that some things are easier to memorize than others, like for instance it is easier to memorize a song verse rather than a lecture on statistics. The reason for this deviation is that the extent to which new input can be remembered depends on its meaningfulness for the person memorizing it. The more meaningful an item or a piece of information seems, the deeper the level at which it is processed by the observer and the better is his ability to remember it over time. Some of the factors determining meaningfulness would be the familiarity and the associated imagery of an item. For instance, a familiar word would be 'door' or 'work' due to their frequent use in everyday life. By comparison, 'sleep', 'eat' or 'bird' would also be high imagery words [JP96].

Obviously meaningfulness is an important topic in interaction design when considering its effects on the human memory. A possible conclusion could be suggesting the use of meaningfulness, which means the use of familiarity and imagery, throughout the appearing names, icons and forms of user interfaces. Unfortunately it is not as simple as that, for the use of everyday words and images could be more confusing than perspicuous when used out of context. However, in practice most cases are quite the opposite; designers are using rather abstract and sometimes even arbitrary terms for the command names of their interfaces. Such naming could prove quite effective for some user groups, whereas some other user communities would find the same terms rather ambiguous. For instance the terms 'cut', 'copy' and 'paste' are quite common and plain commands for some experienced computer user, but for a novice user they are probably rather elusive. So, a possible guideline for the selection of command names would be to consider the background and the contextual and cultural characteristics of the corresponding target group [JP96].

Another approach may suggest replacing the majority of the command names through meaningful icons. Just as with command names there are different factors affecting the meaning-

fulness of an icon: the context in which the icon is being used, the task for which it is being used and the form of the representation of the icon. The interpretation of an icon depends on the context it is used in. The more that context is general, the more the interpretation can be ambiguous. For instance, a symbol of a man's shoe shown in a restaurant will probably be interpreted as an indication for the restrooms, whereas the same symbol shown in a shopping mall could additionally also be interpreted as the way to the nearest shoe store. So, the more specific the context in which icons are being used, the more their meaning can be narrowed. Furthermore, the function or the task for which an icon is being used has significance for determining its meaningfulness. Not all types of function can adequately be represented by icons, especially tasks that require the user to specify or retrieve textual or verbal information. Nevertheless, icons can be a very effective representational form when dealing with other types of information retrieval tasks. Every type of task involving recognition of information rather than recall can successfully be represented by an icon, like for instance error or warning messages or the drawing tools of an image editing software. The understanding of an icons meaning highly depends on the way it is represented. The representational form of the icons can roughly be divided into three groups [JP96]:

- Concrete objects
- Abstract symbols
- A combination of both

Figure 2.9: a)Concrete Object b)Abstract Symbol c)Combination of both

Evidently the icons belonging to the third group, a combination of concrete objects and abstract symbols, are the most meaningful for the majority of users. Aside from the primary meaning provided by the concrete object, the additional use of abstract symbols (such as arrows, dots, lines) provides some dynamic information about the status of an icons and the actions that can be done with it. The representational form can also be categorized according to the mapping that is used to represent the underlying concept:

- **Resemblance icons** picture the underlying concept through an analogous image.

- **Exemplar icons** present a typical example of the representing concept.
- **Symbolic icons** are used to describe a concept that is at a higher level of abstraction than the image itself.
- **Arbitrary icons** have no relation to the underlying concept. The association has to be learned.

Figure 2.10: a)Resemblance b)Exemplar c)Symbolic d)Arbitrary

For further reading see [JP96].

2.7 Knowledge and Mental Models

Knowledge and mental models have great significance in HCI. By understanding what users know or believe to know about a system and its functions, whether those assumptions are correct or not, may help the system designer to predict likely errors, estimate the extent of the learning phase and the ease or severity with witch users perform system tasks. Our knowledge is stored in our memory and is assumed to be highly organized. There are three main types of knowledge representation:

- Analogical representations
- Propositional representations
- Distributed representations

Analogical representations are relatively concrete, stored images, like the image of a tea cup for instance, whereas propositional representations happen to be more abstract, language-like statements, like 'the tea-water is boiling'. Both representations are viewed as symbolic, which suggests that knowledge is stored in symbolic structures and that reasoning and cognition emerges from the syntax and semantics based manipulation of those structures. Distributed representations on the other hand are considered as sub-symbolic, which means they don't depend on any kind of rule-like structures. However there are some controversies

in cognitive science about the role of those three representation groups in cognitive processing. A popular theory implies that our knowledge is consisting of numerous schemata. Basically, a schema is a network of knowledge, based on previously made experience, allowing us to cope with commonplace events with a minimum of effort. The assumption behind the schema theory is that by repeating the same set of actions over a certain period of time, people develop a script for the prosecution of those actions. When confronted with a new but similar situation, this premade script allows us to cope with the situation in an appropriate way, by following familiar clues. Although this seems quite obvious, the resulting implication for interface design would be to aim toward a standardized interaction [JP96].

However, the schema-based theories have often been criticized as to inflexible, for people are capable of dealing with situation they have never encountered before, thus never have developed a script for. So another, alternative theoretical concept was developed: the mental models. By contrast to the static schema-based theories, the mental models are dynamically constructed, by using previously stored schemata. A mental model could be seen as an incomplete and rather instable representation of the external world in our head, consisting of our possible actions, their alternatives and the probable results. Donald Norman provided a well-known definition of mental models in the context of HCI: "*The model people have of themselves, others, the environment, and the things with which they interact. People form mental models through experience, training, and instruction*" [Nor88]. Basically, mental models can be divided into two groups: the structural and the functional models. The difference between the two, to put it simple, is the underlying view: structural models could be described as 'how-it-works' models, whereas functional model could be described as 'how-to-use it' models. In engineering, structural models are often used to describe the inner functions of devices, providing a model which enables the user to make predictions about the specified system, relating to any given sequence of actions. In our daily routine however, we seldom make use of structural models, considering that most of us happily use the TV, the cell phone and the microwave without even having a clue about the way it works. Instead we make use of functional models, which enable us to learn how to use new systems through accessing our previously gained experience and knowledge in the same domain. These models are context-dependant and they can't make any predictions about the system, but therefore they are easier to use [JP96].

In HCI mental models could be of great value. By making the assumption that people are using some kind of mental model when interacting with a system, which is not empirically proven, system designer should try to shape those models in order to provide the best pos-

sible usability.

2.8 The Way Interfaces Affect Users

Just like everything else we interact with, user interfaces are most likely to trigger some kind of emotional reaction or response from the user. These affective aspects of interaction must not be neglected, considering that the emotional state of the user, which is caused by interaction, certainly has a significant impact on his productivity and his acceptance toward the interface. But how exactly could an interface act affectively? One possible approach towards effectiveness could be realized by designing expressive interfaces and aiming to evoke a positive emotional reaction from the user, as well as giving him some reassuring feedback at the same time. There are different means by which such an expressive setting can be achieved, like for instance by using expressive graphical elements (i.e. dynamic icons or animations) or by embedding decent sounds, indicating actions and events (i.e. error messages). Moreover, the general "look and feel" and appearance of an interface has a vital effect on the user's emotional state. Therefore an agreeable interface will certainly find more acceptances by causing a pleasant surrounding and may even help to overlook some usability deficiencies. However, just as well interfaces can cause a pleasant emotional state they can also be the source of anger and frustration. User frustration is seldom caused deliberately and is rather caused by bad design or no design at all. Here are some of the most common frustration sources:

- **Gimmicks**: When the expectations of the user are not met, he should not be presented a gimmick instead of the desired content (i.e. the "men at work" sign for site contents that are under construction). Gimmicks rarely have soothing effects, quite the contrary.
- **Error messages**: Ideally, error messages should offer the user some possibilities of how to fix the error rather than just making a not very lucid error report. The use of cryptic language and long error messages is a major factor in user frustration.
- **Appearance**: As already mentioned, the overall appearance of an interface can affect its usability. Depending on the user target group, there are a lot of different design possibilities concerning the appearance, there are however some general guidelines that should be abided, like for instance not overloading the interface with text and graphics or avoiding flashing and distracting animations.

The best way of dealing with user frustration is by avoiding it. The design of an interface should be simple, but also elegant and perceptually silent. The designer should try to follow usability and graphic design principles as well as the ergonomic guidelines [Pic00, JP02].

2.9 Summary

In this chapter we outlined the human aspects in Human Computer Interaction and their possible effects on the usability of interfaces. Starting with human diversity, we saw that there is no single and perfect way for the design of interaction, but that the design has to be based on the expected target groups, considering their cultural and international background, their personality, their age and their abilities or possible disabilities. Furthermore we delivered an insight into the way human information processing works, as well as the related processes, like for instance the human perception, attention and memory. In this context, we introduced the concepts of meaningfulness, imagery and knowledge representations, and examined their meaning for the interaction design.

Chapter 3

Interaction Design

3.1 Introduction

This chapter deals with the concrete process of interaction design, with an emphasis on the design of visual user interfaces. Although the underlying discipline of interaction design itself is a rather vast and branched field, the following chapters will attempt to give an overview over the most common and basic elements, guidelines and methods that need to be considered for a successful design and implementation of user-based interaction.

3.2 Interaction Design Basics

There are many different approaches that interaction designers can choose from, but no matter what approach they finally decides to use, the basic underlying concepts will always stay the same. The basic resources and elements of interaction design as well as the principles guiding the design process will always find their appliance due to their rudimental character.

3.2.1 Elements of Interaction Design

No matter what kind of interaction a designer strives for, be it a digital interface (software), an analogue service (Coffee machine) or a combination of both (mobile phone), the basic design elements will remain the same for every kind of interaction. Although those elements are

quite conceptual, they can be very powerful tools for the designer. Here is a brief description of those conceptual interaction elements, according to Dan Saffer [Saf06].

Space "*All interactions take place in space*" [Saf06]. Interaction designers will always have to work with space, simply because of the fact that every movement or interaction happens in space. However, the word "space" has a wider definition in this context, and it includes not only the physical space we all live in, but also digital space like the representation of an application on a computer screen, or spaces with more unclear boundaries like for example the internet. But space is not only a scenery for interaction; designers can consciously make use of space during the design process, for instance by the spatial arrangement of the interactive interfaces. If we take a look at some interaction supporting stores, like for instance Ikea, we will notice that the spatial allocation is crucial when aiming to provide an effective interaction while preventing any bottlenecks. But space is also omnipresent in digital interfaces. Most designers use 3D visualization on a 2D space, by the use of perspectives. By the application of perspectives to a 2D visual interface, object can not only move along the X and Y coordinates, but also along the Z coordinate, adding some depth to the interface which often results in an usability boost.

Motion It should be pretty obvious that there can be no interaction without motion. Interaction is a special form of communication, and communication is all about movement; not only the physical movement, like for instance the articulating lips of a speaking person, but also the movement of the exchanging information between two actors. However, we could also take a look at interaction from another perspective, claiming that interaction, as the word itself suggests, is all about action. But even in this case it's all about motion. An action is triggered by movement, like for instance the pressing of a button on a keyboard. Then again most actions cause reactions, resulting in movement, as for instance an unfolding menu on a computer screen [Saf06].

Time "*All interactions take place over time*" [Saf06]. Since the appearance of Einstein, it is generally known that space and time are inseparably bound together. This means that every action or movement through space will take a certain amount of time to accomplish. This has a far-reaching scope considering interaction. Every keystroke, every digital information sent, every started action will take some time. Therefore it is needless to say that every interaction designer needs a good awareness of time. But it is not quite as simple as that. The duration

of actions and the time needed for certain tasks, combined with an interacting user, creates rhythm. Rhythm is essential in interaction, especially when considering animations. How fast or slow a menu unfolds, how quickly a folder opens, these are only examples of rhythm creation in an application and the designer controls this rhythm. Needless to say that bad rhythm or no unitary rhythm at all, certainly will have a negative impact on the usability.

Appearance Appearance is one of two essential elements determining affordance. As mentioned before, affordance is a term firstly introduced by James Gibson in 1966, and, it is a property of an object that subjectively suggests its purpose to the beholder or that rather provides some indication of how to interact with it. A button for instance has an affordance of pushing, whereas a box has an affordance of opening. But affordance can also convey other attributes, like for instance emotional content or qualifying attributes of some sort: is the object delicate or robust? Is it complex or simple? For designers, appearance has many modifiable attributes [Saf06]:

- Proportion
- Structure
- Size
- Shape
- Weight
- Color (hue, value, saturation)

In any case, every object or design has some sort of appearance, may it be deliberately determined by the designer or not.

Texture While texture may be seen as a part of appearance, it can certainly imply the same information as appearance, thus it can also convey affordance. Texture may also mediate emotion. An object with a soft, fleecy surface will certainly trigger a different response than an object consisting of cold, hard metal.

Sound Sound plays a rather small part in interaction design, but in some devices it may be an important one, especially when talking about devices with prevailing alert functionalities. Generally, the designer can adjust sounds by varying some of its three major components [Saf06]:

- **Pitch**: How high or low is the frequency of the sound?
- **Volume**: How loud is the sound?
- **Timbre or tone quality**: What type of sound is it? Sounds with the same pitch and the same volume can still seem very differently, as for instance when played by different instruments.

3.2.2 The Laws of Interaction Design

Interaction design, being a relatively new discipline, doesn't in fact have many regulatory rules or laws. However, there is a small amount of laws that have proved useful in the past. Here is a brief overview of those laws, but let it be said that their function in the design process should be of a guiding kind rather than dictating the work.

Moore's Law Moore's law is not really a guideline for interaction design, but it is a process that designers should be aware of. Moore's law is an observation made in 1965 by Gordon Moore, a co-founder of microchip maker Intel, and it implies that the number of transistors that can be placed on an integrated circuit is doubling approximately every two years. Amazingly, this trend has been constant for more than half a century, and is expected to continue for at least one or two future decades. Now, transistors and microchips aren't exactly the topic of interaction design, but it should be considered that a majority of the technology used by interaction design is based on microchips and transistors, and therefore can be expected to improve in performance and size very quickly.

Fitt's Law Fitts' law is a model published in 1954 by psychologist Paul Fitts. It is a simple formula generally describing the act of pointing and targeting, measuring the time it takes to move from a starting point to a desired target. In this context it doesn't matter if the pointing device is physical, like the human hand, or virtual, like a mouse pointer. Fitts' law states, that the time needed to acquire a target is a function of its size and the relative distance to

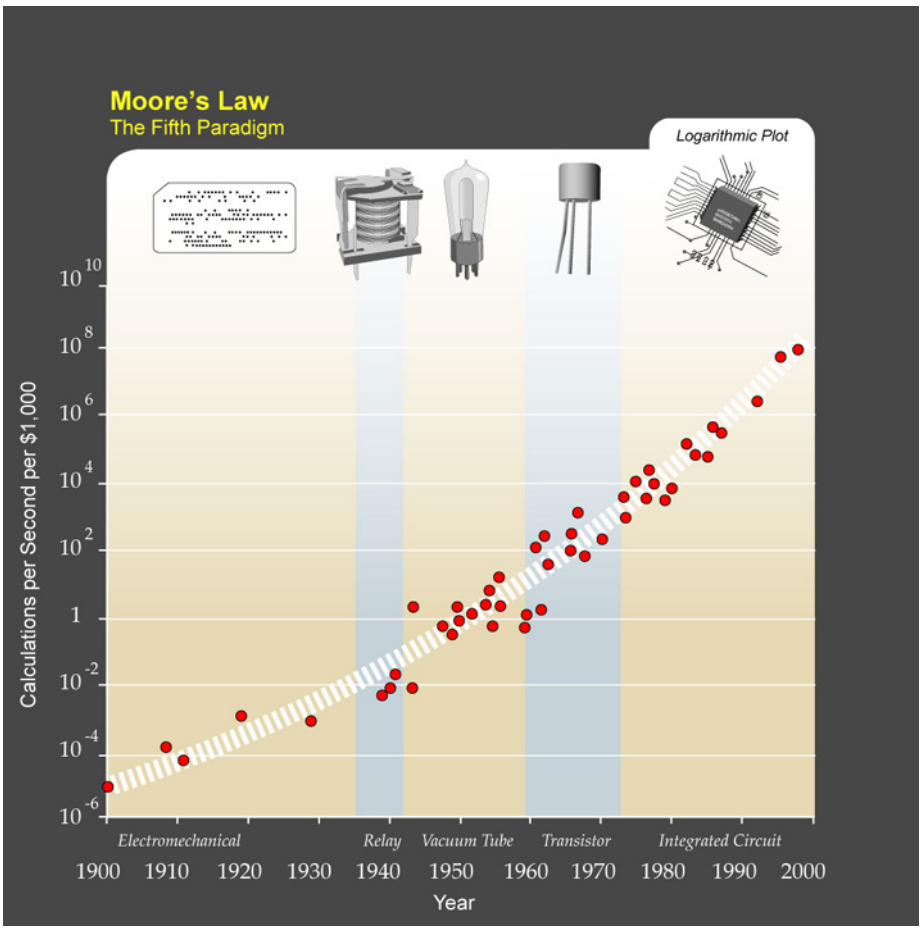


Figure 3.1: Moore's Law, The Fifth Paradigm [Kur08]

the starting point of the movement. In simple words, the larger and closer the target is, the faster it can be pointed to. Although this statement may at first seem pretty obvious, it is in fact the most ignored principle in interaction design. But what exactly are the implications of Fitts' law? Here are some possible derivations regarding the design of visual interfaces [Sta08, Tog07]:

- **Size**: Since size matters, all clickable objects should have a reasonable size. The bigger the target object is, the easier it is to manipulate. But obviously there will be some major issues if the size of a button takes a third of the whole screen. However, Fitts' law tells us that the targeting time is a product of size and distance, so the loss of usability caused by a smaller size of the target could be compensated by a decrease of its distance. This is especially applicable for slightly sophisticated interface elements, like menus or browsing components.

- **Positioning**: According to Fitts' law, the borders and especially the corners of the screen would be ideal places for the positioning of diverse interactive components, simply because of their pinning accuracy. No matter how fast or imprecisely you move the pointer towards the edge, you will never miss it.
- **Distance**: Since the distance of the target has a significant impact on the usability, popping up windows and context menus should be relatively near to the position of the pointer.

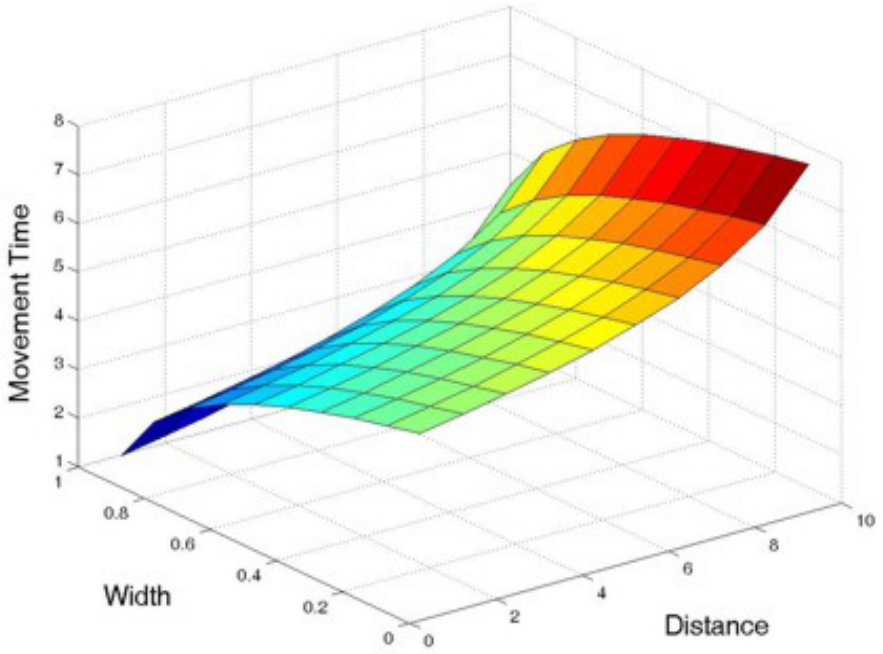


Figure 3.2: Fitts' Law [Sta08]

Hick's Law Hick's law states that the time it takes for a user to make a decision is determined by the number of possible choice he or she has. Also it says that when considering a group of possible choices, the human brain doesn't evaluate the choice one by one, but rather divides them into subcategories, eliminating about half of the choices in each step. This assumption certainly would have some interesting implications in interaction design and especially when designing extensive menus, it should be taken into consideration. According to Hick's law, when dealing with a large number of menu items, the best performance would be achieved by presenting as many items as possible in the same menu, avoiding further

submenus or any hierarchical breakdowns. Of course, this suggestion is only applicable to a certain degree.

The Magical Number Seven The "magical number seven (plus or minus two)" was first published in 1956 by psychologist George A. Miller. The theory claims, that the human mind achieves best results if trying to memorize information in chunks of seven items (plus or minus two). If attempting to memorize bigger chunks of information at once, the brain starts making errors. At some aspects, this theory obviously seems to be in contrast to Hick's law. However, it should be considered that there is a difference between memorization and visualization of content. For instance, when designing a hierarchical dropdown menu, the designer doesn't have to be strictly consistent with the number seven rule, simply because the content of the whole menu is visible at any time, and the user doesn't have to memorize it. However, when designing a navigation that spreads over multiple pages, the number seven rule should find its appliance.

Tesler's Law Tesler's law of the conservation of complexity was coined by the computer scientist and interaction designer Larry Tesler, and it claims that there is a certain degree of complexity that is inherent to every process. An attempt to simply a process beyond that level complexity would be futile. However, even if that basic complexity cannot be simplified any further, it can be split and shifted from one place to another, taking some of the complexity away from the user. A good interaction designer need to be aware of this principle, as it should be his goal to adopt as much complexity as possible by the product he creates [Saf06].

Poka-Yoke Principle The Poka-Yoke principle was created by the Japanese industrial engineer Shigeo Shingo in 1961, and roughly translated it means "avoiding inadvertent error". It also slightly reminds of the western adopt of Murphy's law stating that "Anything that can go wrong, will go wrong". Designers using the Poka-Yoke principle will try to prevent errors by forcing the user to interact correctly. There are many different ways such results can be achieved, like for instance by using visual signs and hints, or by making the correct procedure the only possible course of action. A good example for the Poka-Yoke principle would be the USB connector, which can only be plugged-in in a particular way, leaving the user no chance for errors [Saf06].

3.2.3 Characteristics Of Good Interaction Design

No matter the nature or the required functionalities of a given interface, there are certain characteristics that a good visual interface designer should always aim to achieve:

- **Lucidity**: A good GUI should be clear to the user, which means that the user shouldn't have any qualm about the way he is supposed to interact with the interface. In order to achieve the greatest possible degree of lucidity for a certain interface, the designer should stick to well proven and established procedures as well as to standardized representational forms. Buttons, icons or menu items should act consistently, enabling the user to intuitively interact with the interface without having to overthrow or relearn already internalized procedures and associations.
- **Intuitive handling**: The overall handling of the application should be as intuitive as possible for the average user. The designer should aim to stick to already pre-established interaction methods and techniques, or at least consider them when he tries to introduce the user to some new or variegated ways of interaction.
- **Simplicity**: It is crucial to keep the interface as simple as possible, providing the user with an overview over the frequently needed information and essential actions, but avoiding an overflow with unnecessary information. Unfortunately, the more simple an interface, the less functional it is, charging the designer to make balanced and thoughtful trade-offs between those two factors.
- **Consistency**: A consistent interface should provide unitary behavior throughout the whole application and should aim to build upon the premade knowledge of the user. When it comes to general consistency in relation to other applications and platforms, the degree of how strictly this consistency should be maintained varies throughout the different elements of the interface. The most basic interaction components, like for instance keyboard-shortcuts or standardized small visual structures likes icons, buttons or select boxes, should provide a high degree of consistency, unlike the overall look and feel of the application.
- **Redundancy**: The designer should aim to use as many cues as possible without making the interface too intrusive. A certain degree of redundancy may imply the desired effect of intuitive understanding of the provided functionalities and output messages. For instance an error message could have an error icon in front of it, in order to underline the alarming nature of the message, an additional error sound however could

easily be a source of frustration. Redundancy should also be considered when designing the interaction methods. Different users have different preferences and habits, therefore the designer should provide different parallel action courses for the same procedures. In order to save some data for instance, the designer could implement a "save" menu item in a contextual menu, an additional "save" icon in the upper toolbar and a complementary keyboard shortcut.

- **Feedback**: A good interface should prevent the user from building up a feel of uncertainty. The user should always be in the picture about the actual state of the application, which means he should be informed about the currently undergoing operations and processes and their estimated duration, as well as about the results and the outcome of the actions he performed. For instance, when saving some entered information, the user should be notified whether the information was successfully stored or not, and if not, then due to which reason or problem. The provided feedback however, should be kept strictly informative and simple; intrusive notifications very soon become the source of disturbance and frustration.

3.3 The Process of Interaction Design

Good interaction design is characterized by the philosophy of user-centered design, which means involving the future user through the whole design process, and by understanding the anticipated requirements. However those claims seem simpler than they actually are, for when trying to define future requirements, the designer will soon have to face some problematic questions like "What exactly is determining the requirements?" or "Who are the users, do they really know what they need and are they even able to envision the possibilities?". There are four fairly generic activities in interaction design, which can also be found in other design disciplines: Indentifying the needs and requirements, developing alternative designs considering those requirements, building interactive versions of the design and evaluating them. These activities are related to each other and they are intended to be repeated, in order to include the information gained through the design and evaluation process [JP02].

3.3.1 Design Emphases

There are several different approaches to interaction design that the designer can choose from, each focusing on a different aspect in the solution finding process. Most designers have at least one approach that suits them best and that they tend to use, however that specific approach may not always be the most appropriate one, as some situations or requirements may be more suitable for a different proceeding. Therefore, a good designer should be flexible and should try to use the best approach for any given situation, or even switch between multiple approaches within a single project. Here is a brief overview over the four most common design approaches in interaction design.

User-Centered Design The user-centered design is an approach that has been around for a long time, firstly popularized by Henry Dreyfuss in his 1955 book Designing for People. To put it simple, the underlying philosophy behind this concept is that an interface should be designed according the needs and preferences of the user, instead of adapting the user to the interface. Therefore, the main goal of the designer should be to help the user in reaching his own goal. In order to do so, it is necessary for the designer to involve the user in the design process, ideally at every single stage of the project. This kind of proceeding guarantees that the final interface will be conform to the user's wishes and needs, and should therefore feature high usability. The user-centered design is probably one of the most recognized approaches in interaction design, certainly because it has proven effective in the past, but also because it has been around for quite a time. However, this concept certainly has some flaws, as for instance long term user goals may become blurry over time, due to the lack of clear defined limitations of the interface functions. Also when designing for large user groups, it can be quite difficult to gather usable information representing the whole user community [Saf06, JP02, MVSC01].

Activity-Centered Design As opposed to user-centered design, activity-centered design doesn't focus on the needs, preferences and skills of the users, but on the activities itself. An activity can be described as a set of actions and decisions with a purpose. It usually consists of sub-activities and tasks, which then again are composed of actions and operations. The purpose of the activity may not necessarily be a goal, as for instance the goal of a guitar player may be to successfully play a composition, whereas the purpose of the guitar is simply to create tone. Although activity-centered design also relies on user insights (not so heavily

as user-centered design), it has a much greater focus on the activity itself, trying to preserve the underlying "natural" sequence of actions. Understand the activity, and you will understand the interface designed for it. This approach suggests that the user should be adapted to the activity and not the other way around, which sometimes can be a risky endeavour. When fixating on single tasks and activities, designers may easily lose the overview of the problem as a whole [Saf06, Nor05].

System Design Analogously to the user- and activity-design focusing on the user or the activity, the system design focuses on the system as a whole. In this approach the system is in the centre of the design focus, however, the notation "system" doesn't have to represent a specific technology-based system in this context; it generally describes a set of entities that act upon each other. According to this definition, a system could also be an interacting group of people or a set of objects related to each other. System design is a very structured and holistic concept. When applying system design, the designer focuses on the whole context of use instead of concentrating on single aspects or sub-goals. It is the task of the designer to clearly outline the single components of the system (a goal, a sensor, a comparator and an actuator) and to focus on designing each one of them [Saf06].

Genius Design Contradicting the other three approaches, genius design [Saf06] solely relies on the experience and the abilities of the designer. Generally, genius design doesn't involve the user at all during the design process, or at least not until the testing phase of the system. There can be diverse reasons for the lack of user-involvement during the design phase, for instance due to some security issues that the system or the designer may underlie. The most common reasons however, are insufficient time, material or labour resources. The fact that the designer doesn't interact with the future users, doesn't mean that he doesn't try to accommodate for their requirements and needs. However, in order to successfully empathize with the user, the designer has to be experienced, or the outcome may be rather doubtful.

3.3.2 Identifying the Needs and Requirements

"*A requirement is a statement about an intended product that specifies what it should do or how it should perform*" [JP02]. Through the process of requirement analysis, the designer aims to define the requirements and to make them as specific and clear as possible. On the

face of it, this may seem as a simple task, but it is in fact a rather complicated problem, for the customers may not have communicated all their requirements yet or they may not even know them at all. Even if there is an initial set of requirements, they are often not detailed enough for the development to begin. Requirement analysis should therefore be an iterative process, consisting of data-gathering and interpretation activities while considering and understanding the user's needs. There are many different types of requirements, but in general they all can be classified into two higher-level classes: The functional requirements, describing what the system should be able to do, and the non-functional requirements, describing the constraints that the system underlies. However, it may seem superficial to refer to all requirements that are not functional as "non-functional" requirements, so in order to show the relatively large variety of requirements, here is a more categorized classification [JP02]:

- **Functional requirements** describe what the system or product should do. For example a functional requirement for a web-browser could be to support multiple tab-browsing.
- **Data requirements** capture the characteristics, amount and size of the required data. In general those characteristics consist of the type, volatility, persistence and accuracy of the data.
- **Environmental requirements** refer to the environment and the conditions in which the system will probably operate. There are several aspects of the environment that have to be considered when defining the requirements. The first aspect is the physical environment, describing the physical surroundings and conditions in which the system and the users will operate (i.e. how much noise or lighting is to be expected). The second aspect is the social environment, stating social aspects of the system like coordination and collaboration. The third aspect is the organizational environment, describing the organizational resources and infrastructure. Finally, the last aspect is the technical environment, describing the technical features and limitations.
- **User requirements** refer to the characteristics, skills and experience of the intended user group. Furthermore, these requirements specify if the system will be used by a "typical user", or if it is intended for a wider user range, implying the need for multiple user profiles.
- **Usability requirements** capture the usability goals of the system and the associated measures (i.e. effectiveness, efficiency, safety, utility, learnability and memorability).

3.3.3 Data Gathering and Analysis

Data gathering is the most common and important method for the determination of requirements and its purpose is to gather sufficient, appropriate and relevant data. In most cases there is a set of premade initial requirements, which however often need to be specified, enhanced or confirmed by the means of data gathering. There is a small set of relatively simple and flexible techniques for data gathering, which, thanks to those attributes, can be extended, varied and combined in different ways. Table 3.1 gives a short overview over the most common of those techniques.
Choosing the most appropriate data-gathering technique for a specific project will depend on several different factors. For instance, the kind of desired information will probably depend on how much progress the project has already made in the cycle of design and evaluation. If the project is still at its first lines, then a questionnaire would probably be an inappropriate choice due to the lack of specific questions that need to be answered, so maybe an interview would be a good alternative. Also the choice will depend on the resources available, considering that some data-gathering techniques, like the naturalistic observation, are much more resources consuming than others. Then again some techniques may provide much more detailed and significant findings than other, so generally a trade-off has to be made between the needed resources and the significance of the results. Finally, also the knowledge needed by the analyst in order to perform the desired task has to be considered. As soon as the first round of data-gathering is completed, the interpretation and analysis of the data should be initiated. It is essential to keep the interval between those two processes as small as possible, for a quick interpretation assures that the experiences made by the participants are still fresh. At first, an initial interpretation is being made, in most cases by using a predefined template, containing general information and descriptions about the resulting requirement (i.e. source, description, history, customer satisfaction and dissatisfaction, dependencies and conflicts...). Then a more focused analysis of the data will follow, using different techniques and notations, which in most cases consist of different forms of diagrams. Functional requirements for instance are traditionally represented by work-flow charts, data-flow diagrams or state charts whereas data requirements can be analysed with entity-relationship diagrams. In the end, the whole process of requirement analysis iterates a certain number of times, making the description more extensive and relevant in each turn [JP02].

Table 3.1: Data Gathering techniques [JP02]

Technique	Good for	Kind of data	Advantages	Disadvantages
Questionnaires	Answering specific questions	Quantitative and qualitative data	Can reach many people with low resource	The design is crucial. Response rate may be low. Responses may not be what you want
Interviews	Exploring issues	Some quantitative but mostly qualitative data	Interviewer can guide interviewee if necessary. Encourages contact between developers and users	Time consuming. Artificial environment may intimidate interviewee
Focus groups and workshops	Collecting multiple viewpoints	Some quantitative but mostly qualitative data	Highlights areas of consensus and conflict. Encourages contact between developers and users	Possibility of dominant characters
Naturalistic observation	Understan-ding context of user activity	Qualitative	Observing actual work gives insights that other techniques can't give	Very time consuming. Huge amounts of data
Studying documentation	Learning about procedures, regulations and standards	Quantitative	No time commitment from users required	Day-to-day working will differ from documented procedures

3.3.4 Personas, Scenarios and Wireframes

Personas are a mean to represent the people who are somehow involved with a certain product or service, respectively in the case of visual interface design, the future users or the actors that are supposed to interact with the system. A persona should aim to picture a specific set of people, characterized by the common attributes of the people represented. First and foremost, those attributes should refer to the user behavior and to all the characteristics that affect that behavior, like the user's needs, desires, goals and motivations. For instance, when trying to define the personas for an online cinema ticket reservation, one could classify them according to their behavior in the following manner: the user always prefers to use online reservation systems over telephone based reservations system; the user only uses online reservations systems when he has no other choice; the user is never using online

reservation systems. The overall amount of personas should lie somewhere between 1 and 7 for any given project, as a higher amount would make it hard for the designer to remember and to distinguish the single personas. Also it should prove hard to design for such a large and diverse amount of user behaviors and preferences.

Scenarios are actually fictional, word-based prototypes, describing the way the interface is going to be used. Due to their low resource expenses, scenarios can be used to conduct a first quick grasp of the prevailing requirements and the needed sequences. Unlike real prototypes, scenarios can be written in only a few minutes, however their value is not to be despised, especially not in the early stages of the design. This is also where the personas find their appliance, putting them into the context of the application. A single scenario can and should be run through using different personas, as such a proceeding will certainly point out further needs that should to be implemented in the final release.

The procedure resulting from the appliance of the scenarios is the task analysis. The goal of task analysis is to identify all the needs and tasks that the final product needs to provide. Aside from the scenarios, those requirements can also be assembled from other recourses, like for instance from business requirements or from already existing similar products. Once the tasks have been assembled, task flows can be created, putting the tasks into logical orders and dependencies. Those task flows then again can be built into wireframes. A wireframe is a detailed specification and visual guide of a particular view or part of the application and it should provide placeholders for the content, the functionalities and the navigating elements of the interface. As the content will probably be pretty vague in this stage of the design, it can be represented by some dummy text, whereas the functional and navigational components should be represented by boxes with additional descriptions or references to some annotations outside the wireframe [Saf06].

3.3.5 Prototyping

The final step before the release of a product or the start of the testing phase is the creation of a prototype. Only a qualitative prototype has the ability to convey the final look and feel and the functionalities of the designing interface, which is crucial when aiming to provide a good understanding of the application to the client. The form of the prototype will surely depend on the designer's resources as well as on the nature and type of the interface that

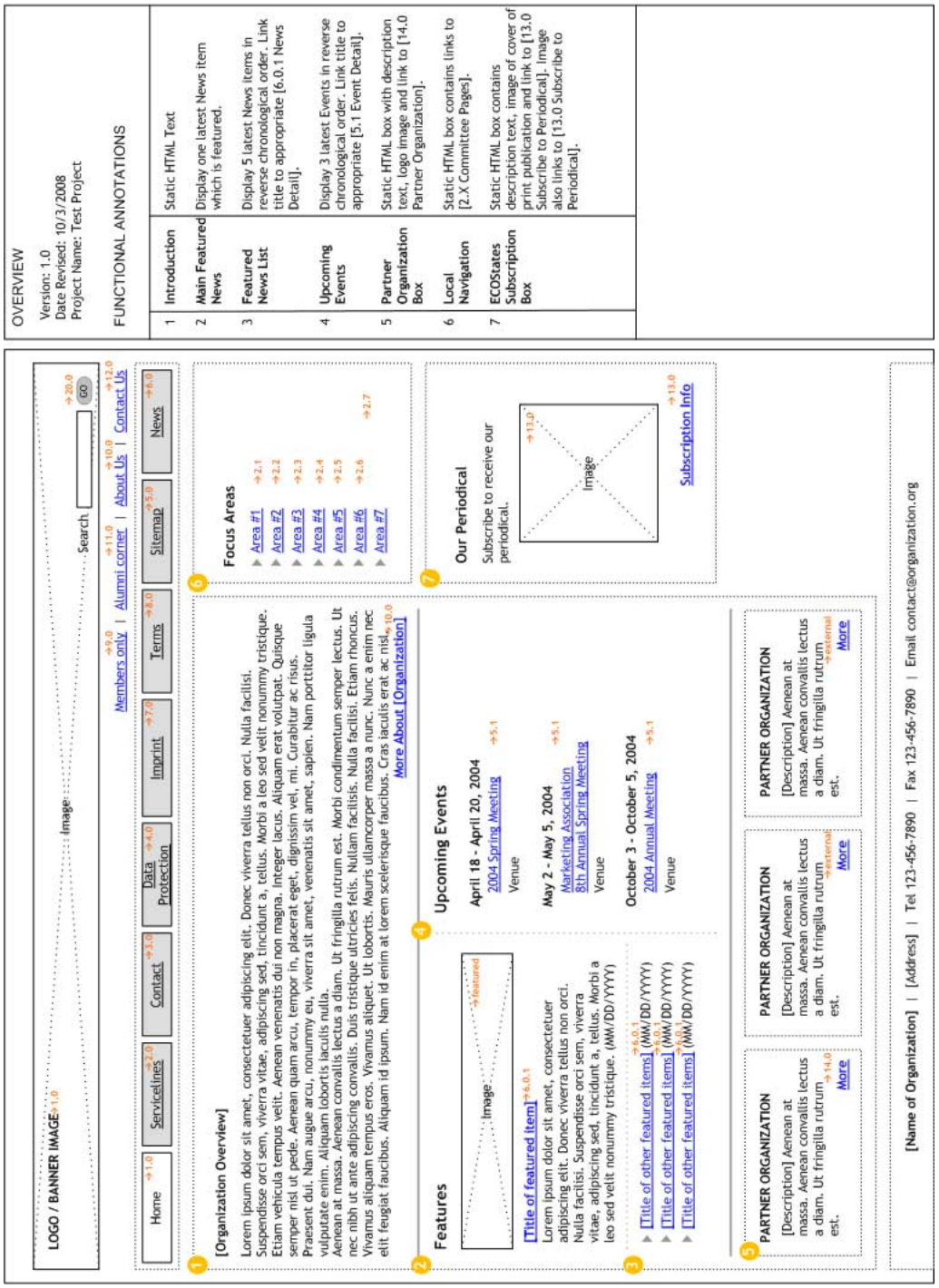


Figure 3.3: Wireframe example template [Str08]

he is implementing. There are different types of prototypes that the designer can work with; here are the two that will probably find the most appliances when it comes to the design of GUIs:

- **Paper prototypes**Apparently, paper prototypes are the fastest to create and they demand only a small amount of resources from the designer. Due to their static nature, paper designs can only represent one certain moment of the interaction. In order to picture a certain use case, the designer has to create multiple numbered pages, each page representing the following step of the interaction.
- **Digital prototypes**Digital prototypes can have many different forms, varying from static images, much like the paper prototypes, to complex pre-versions of the final interface with already implemented functionalities. In other word they may be rather horizontal or vertical. A horizontal prototype seems to have a large set of features as provided for in the final version, however only few of the underlying functionalities of those features may really be implemented. Such a prototype has the ability to convey a rather profound impression of the overall look and feel of the applications, though it often lacks to mediate the usability of the system due to the missing functionalities. A vertical prototype on the other hand provides only few of the intended features, however those features are nearly fully implemented.

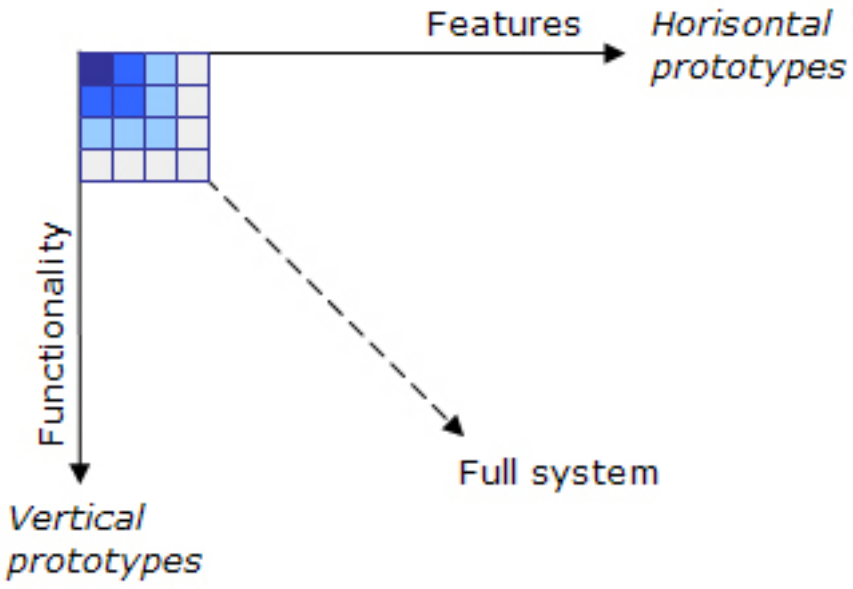


Figure 3.4: Horizontal and Vertical Prototypes [Soe08]

Prototypes can also be divided into three sub-sections, referring to their main purpose and implementation:

- **Explorative prototypes**The main purpose of explorative prototypes is to act as a communication platform between the users and the designer.
- **Experimental prototypes**The purpose of an experimental prototypes is to test whether the designed system is adequate and usable or not. This type of prototype is often used for a more precise specification of the requirements.
- **Evolutionary prototypes**These kinds of prototypes do not only act as prototypes, but they have a rather important role in the whole design and implementation process. In this context, the term evolutionary refers to an evolving system, a process which is achieved by the multiple versions of the prototype, each version succeeding the previous one.

These kinds of prototypes do not only act as prototypes, but they have a rather important role in the whole design and implementation process. In this context, the term evolutionary refers to an evolving system, a process which is achieved by the multiple versions of the prototype, each version succeeding the previous one [Saf06, Soe08].

3.4 Summary

In this chapter we gave an overview over some elementary concepts concerning the discipline of interaction design, starting from simple elements and aspects which have a considerable impact on the design of interaction, on to tools and methods regarding the process of design itself, like data analysis, prototyping or the creation of scenarios, personas and wireframes. Also we saw that there is no single and accurate course of action when it comes to the process of design, but rather a multiplicity of parallel courses, all depending on the future users, the needs and requirements and the emphasis of the designer.

Chapter 4

Trends for Web Applications and the Visualization of Semantically Rich Data

4.1 Introduction

In this chapter we will take a look at some trends concerning the state of the art and the future course of web applications, with an emphasis on Rich Internet Application features and AJAX functionalities. Since the development in this field is rather brisk, it is a quite interesting task to detect the ongoing improvements and especially to try to disclose what the future progression will bring. We will also take a look at the currently available platforms and frameworks concerning the development and deployment of Rich Internet Applications, and at their different application areas and features. In the second part of the chapter, we will aim to analyze methods and concepts for the visualization of semantically rich data. We will describe different graphical elements that can be used for the visualization and organization of such unstructured data, and how they can be applied and combined in a sensible way. In the end we will take a look at the concept of Linked Open Data and the connected visualization and browsing methods, as it is a new and interesting development in the web that could create new insights concerning the presentation of semantic data.

4.2 Trends for Future Web Applications

The introduction and further development of new web-based techniques as well as the rise of the Web 2.0 terminology that occurred during the last decade, both had a sensible impact

Figure 4.1: Diverse webtops (Atoolo, DesktopTwo, eyeOS, G.ho.st, YouOS)

on the web landscapes we see today. Especially the introduction and the derivative of the AJAX model and the consequential Rich Internet Application engines that emerged, made room for a new generation of web applications with a whole new range of functionalities and enhanced usability. This development has gone as far as that a constantly rising number of web applications start challenging their desktop related counterparts and sometimes even the desktop itself. In fact, in the meantime we can refer to a downright surplus of web-based desktop simulating systems, so called Webtops or WebOS, all similar in their functionalities, however, at least not for everyone, not very clear about their suggestive application. This trend, sometimes referred to as "Web Desktopification", as well as the likewise rising "Desktop Webification" on the other side of the spectrum, are a result of the same course towards shared and anywhere accessible information. Both sides are leveraged by big representatives, Microsoft on the desktop side and Google on the web side, and are currently rivaling each other over the future architecture of web supporting applications. Although the overall development seems to be heading towards native and enhanced web applications, a lot of people question the purpose and the effectiveness of such applications, or at least the form of their implementation: Just because something is possible doesn't necessarily implicate

Screenshot taken from http://mail.google.com

Figure 4.2: Google Mail recipient suggest

that it is needed.

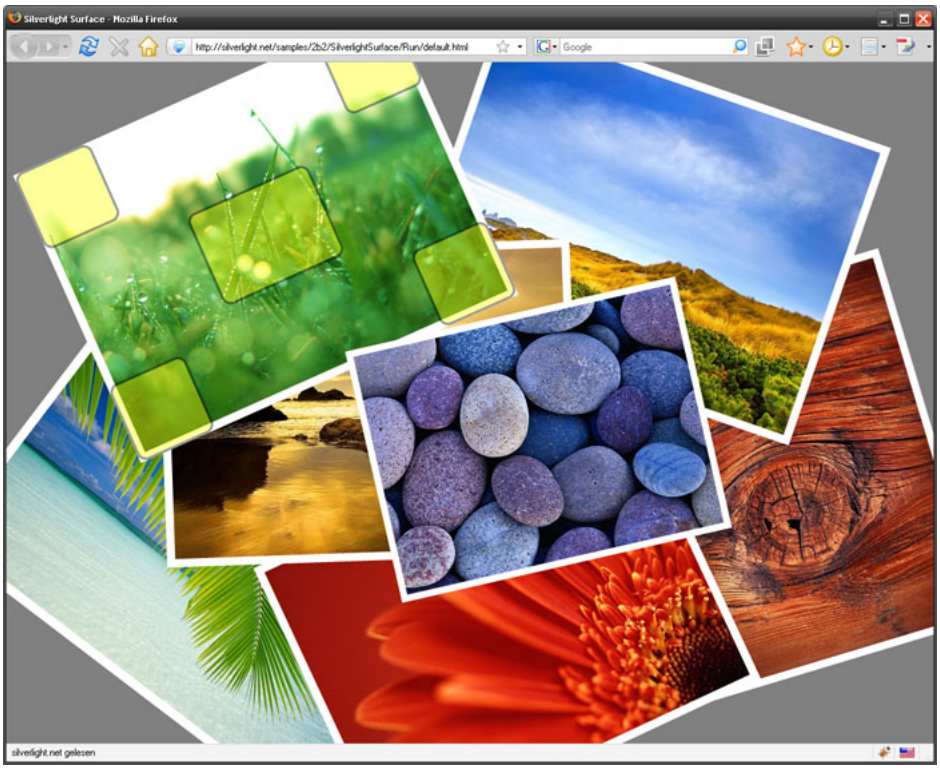
So, how exactly should a modern, intuitive and primarily suggestive web-application look like? It is obvious that yesterday's nice-to-haves are quickly becoming today's must-haves, and in the meantime no serious web-application will launch without having some additional interface features and usability improvements. However, the question is what features and design approaches are really improving the interface and making a useful contribution to the usability and the handling of the application, and what features are rather an eye-catch with no significant impact on the usability at all. If we take the Webtops for instance, is it really necessary to have movable desktop-like icons floating in the background of the page or a Windows like taskbar, with the single purpose of conveying a desktop-like feeling to the user, or is it rather an adopted gimmick that is out of place? Of course, such an approach certainly has some advantages, as for instance it triggers a sense of recognition and intuition regarding the application handling, as it is tying up to already established standards, however from a technical and long-term oriented point of view, it is probably the wrong course of action. One should consider that the browser as such is a limited container, not designed for the handling of very large and overlaying graphical content, or at least not yet. Nevertheless, there is a large amount of functions and graphical handling elements, all basing on the AJAX functionalities, which are very useful and which can certainly improve the usability of a web-application, taking it's handling to a whole new level. Drag and drop functions, blending effects or the possibility of dynamic and partial content updates, are all, if deployed correctly, enhancing the application handling and are on the best way of becoming tomorrow's standards. If we take a look at Google Mail's search suggest for the input of recipient addresses for instance (see Figure 4.2), we will see a good and sensible appliance of the AJAX tech-

nology.
All in all, it is hard to say what course exactly we will be heading in the future considering the development and enhancement of web-applications, but it is quite likely that today's best representatives of the art will lead the way. However, not only the application developers will define the future course, but also the next generation of browsers, as they are defining and limiting the scope of possibilities for every developer.

4.3 Rich Internet Appliation Platforms and Frameworks

Considering the development of Rich Internet Applications, nowadays there is a broad and versatile range of frameworks and platforms that the developer can resort to. However, it's the most basic approach to RIAs, which is the simple use of Javascript and XMLHttpRequest, the one that is gaining more and more advertence. Hence, various Javascript-based frameworks have appeared, facilitating the integration of AJAX functionalities as well as RIA-effects, as they provide a range of pre-implemented functionalities and user-interface components, which allow an application development on a higher level. JQuery for instance is a Javascript library with a focus on AJAX interactions, event handling and animation, providing all the needed tools for a successful RIA implementation. Whereas JQuery is a rather all rounded library, there are also more specific platforms that also provide AJAX and animation functionalities, as for instance the JSF-based ICEFaces, which allows RIA creation and deployment in pure Java.

Although Javascript-based AJAX frameworks are on the rise, Microsoft's Silverlight and Adobe's Flash are still the old bulls in the business. Silverlight is Microsoft's actual developing platform for Rich Internet Applications, with the purpose to compete with similar platforms from other providers, especially with Adobe's Flash, Flex and AIR. The deployment of Silverlight applications occurs over the web-browser; however the representation of the content on the client side is in need of a plug-in, which is an approach that was already widespread by Adobe's Flash player. Basically, Silverlight's presentation framework is designed to work with XAML (Extensible Application Markup Language), an XML-based markup language developed for the Windows Presentation Framework, and offers scripting possibilities over Javascript. Those functionalities enable Silverlight to embed animations, text effects, video content and other features into the application, as well as to interact with the DOM (Document Object Model) of the surrounding page. Furthermore, the release 2.0 of the Silverlight framework, which is currently available in a beta version, comes with several

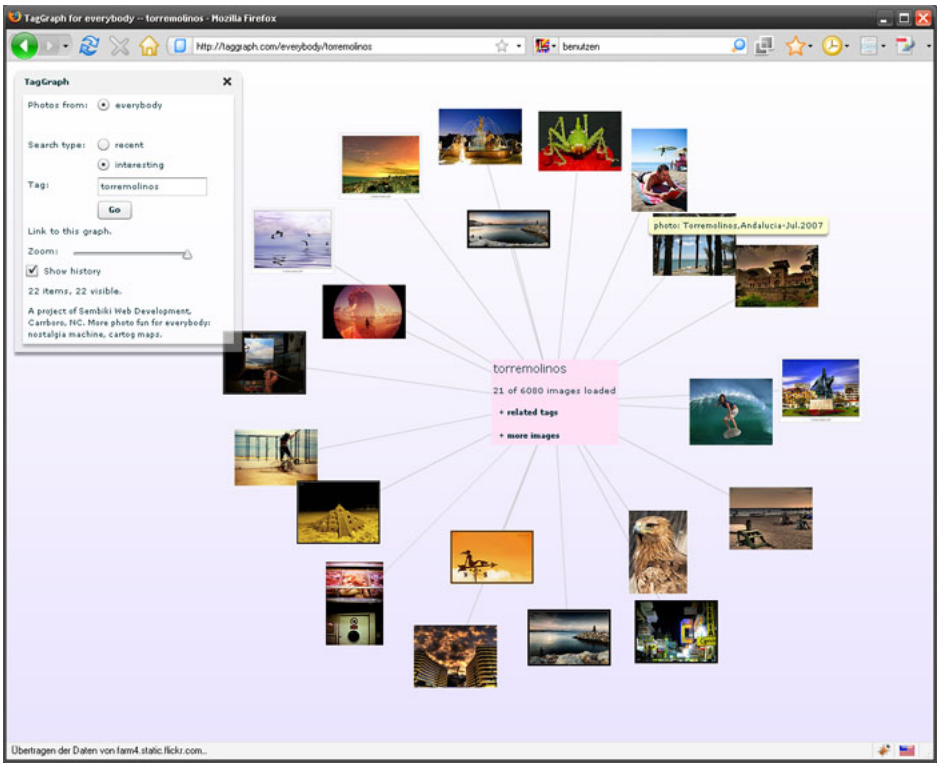


Screenshot taken from http://silverlight.net/samples/2b2/SilverlightSurface/Run/default.html

Figure 4.3: A Replication of Microsoft's Surface UI with Silverlight

new features, as it provides an integration of the .Net framework. This will give developers the possibility to make use of any kind of .Net programming language, as for instance C++, C#, IronRuby or VB.NET.

Adobe on the other hand actually provides two different approaches to the development of Rich Internet Applications, both having the Flash player as their core and needed plug-in on the client side. The first and rather classical approach is based upon the Flash authoring software, providing the developer with Flash proprietary concepts and animation metaphors like Frames, Timelines and diverse effects. The second approach is based upon Flex, which is a collection of technologies with the purpose of adapting the Flash metaphors to a programming model, thus making the implementation more appropriate to developers who are not used to the Flash authoring software. Adobe's Flex makes use of MXML, an XML-based markup language for the declaration of the graphic user interfaces, and employs ActionScript for the interactivity.



Screenshot taken from http://taggraph.com

Figure 4.4: A Flickr Tag Explorer Implemented with Flex

4.4 Ontology-based Information Visualization

The visualization of ontology-based content may require a closer look than general content visualization, as it implicates some specific requirements, navigational features and visualization techniques. Due to the potentially complex nature of semantic relationships, the ontology visualization often requires more powerful means of content representation, as normal visualization and browsing techniques often lack the necessary depth and complexity in order to effectively visualize and browse the semantic content. The only way that such visualization complexity can be achieved is by the use of graphical elements, graphs in most of the cases. However, when developing a graph or a graph related view for an ontological content system, the developer should keep in mind that it is not advisable to display the graph as a whole, as such a representation would most probably cause issues in the performance and a loss of lucidity.

There is set of possible techniques that can be used In order to reduce the section of the graph needed to be shown for a sensible content representation. One possible approach

is to use filtering techniques, reducing the shown content to a specific aggregation which is defined by user-entered queries. Such simple filtering methods are mostly based on the element-attributes, which sometimes may not be enough for the desired content depth. Therefore, many developers refer to more sophisticated approaches, like for instance the clustering of content, which is often regarded as the most efficient way to visualize large graphs. Clustering is the process of grouping information according to their content, but also to their context, to more generalized entities, reducing the complexity of the graph and the amount of displayed elements, which implicates a more lucid way of content representation due to the user-controlled determination of the level of detail.
Irrespective of the abstraction level of the graph, it is crucial to provide effective and usable tools for the navigation and manipulation of the content shown. Zoom and pan may be relatively old navigational techniques but they are still rather indispensable. Also it is advisable to support the overall navigation by some complementary views, like for instance an outline or a tree representation referring to the currently displayed graph section. Here is an overview over the common elements and tools used for the visualization of semantic content.

4.4.1 Indented Lists, Trees and Graphs

Indented lists and tree-like elements are the most simple and common tools when it comes to the visualization of ontology. The taxonomy is represented by expandable and retractable nodes, containing further subtrees, likewise to any tree-based folder or file system like for instance the windows explorer. In case of a top-down layout, the child nodes are placed under their parents and indented to the right, whereas a left-right layout positions the child nodes to the right, with a slight displacement to the bottom. In case of multiple inheritances the child nodes are placed under both parents. However, anyone who has tried to browse ontologies with a simple tree view will probably come to the conclusion, that trees aren't the ideal visualization method for semantic content, or at least not without complementary visualization. One of the problems is the representation of the node properties and attributes, as when displayed within the tree they need to be clearly separated from the child nodes of a given parent. A simple mean to achieve this separation would be to add an additional view, which is detached from the tree and used for the representation of the properties and attributes of the currently selected node. When displayed within the tree, the separation could be achieved by the visual detachment of the nodes and their containing attributes from the rest of the tree. Such a representation however would not be a simple text-based tree anymore, but rather a tree-based graph, which certainly will cause more effort during the implementation. Graphs are

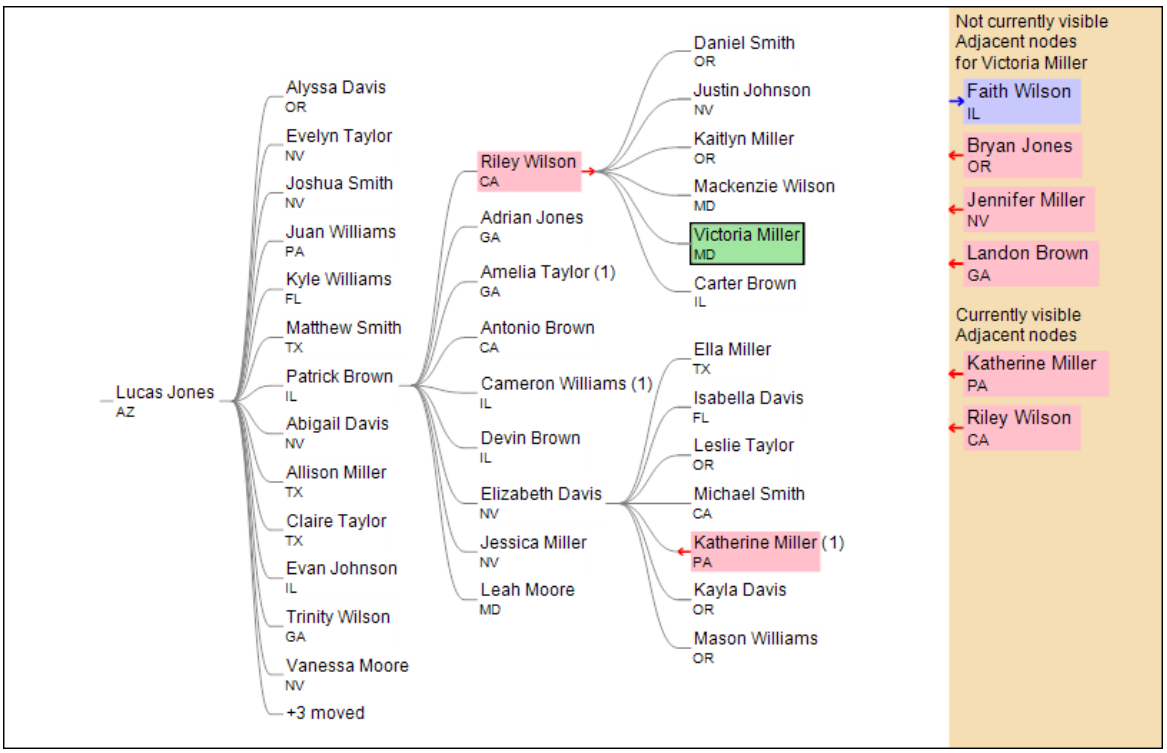


Figure 4.5: TreePlus, a Tree-based Graph Visualization [Tre08]

the natural derivatives of trees, and they allow a more lucid visualization of ontology-based content due to the possibility of a more customized subdivision of content and the display of multiple inheritance relationships. While graphs certainly are a quite effective mean for ontology visualization, as mentioned before, they tend to claim much of the disposable space rather quickly, which is always a serious problem in interaction design. In most cases it will be impossible to display the whole graph at once, therefore different techniques find their appliance, aiming to restrict the information shown to the currently desired level of abstraction [KHL+07].

As a good example of a tree-based graph, we could take a look at Treeplus [LPP+06]. Treeplus is a visualization graph based on a tree-style layout with enhanced interaction and visualization techniques (see Figure 4.5). Treeplus consists of a tree-like browser in the center of the view, and of an additional frame on the right, used for the display of the adjacent nodes of the currently selected node in the tree. Those nodes are grouped into the visible nodes, which would be the nodes that are currently visible in the tree, and in the not visible nodes. Also they have a blue or a red arrow attached, symbolizing whether the relationship to the node is of an incoming or outgoing nature. When a node in the tree is double-clicked, the adjacent nodes move from in right frame into the center, and are integrated into the graph.

For further details on Treeplus see [LPP+06].

4.4.2 Cluster Maps

Cluster maps could be regarded as an extension to the standard graphs concerning the navigation through different levels of abstraction. In a cluster map, the abstraction is achieved by the union of different entities to a common, superior entity, according to shared content or context of the contained information. This superior entity then acts as a node for the contained entities on a more abstract level of representation. The cluster map is not only a good visualization technique for semantic content, but also a very powerful and clear navigational tool, allowing the user not only to zoom into the desired cluster, but also to define and customize the means by which the clusters are built.

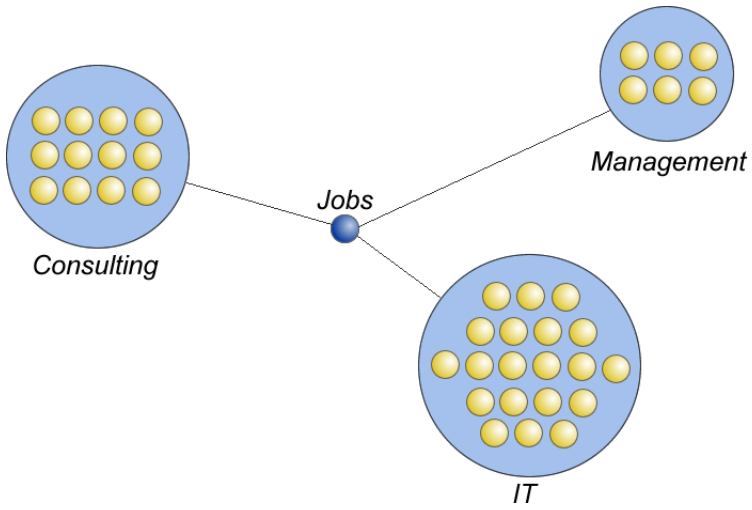


Figure 4.6: Example of Cluster Map

Figure 4.6 shows a simple cluster map for the results of a fictive job search. The view displays all the jobs that have been returned for a given query, divided into sub-clusters according to a more precise specification concerning their branches (IT, Consulting, Management). Figure 4.7 shows the same map, only this time on a more concrete level, as it shows the results contained in the IT cluster. As we can see, here again the results are divided into further sub-clusters, again according to more specified branches. However, this cluster map could be implemented in a way that allows the user to change the restrictions by which the sub-clusters are built during the navigation of the map. As a result, the sub-clusters in the

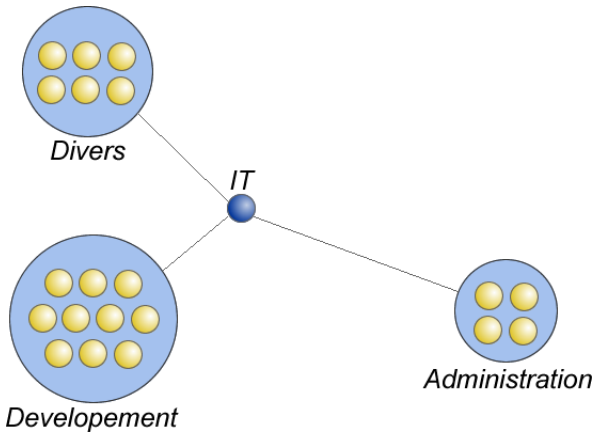


Figure 4.7: Cluster Map organized by Branches

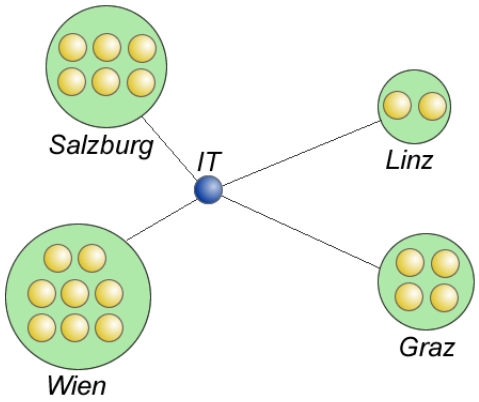


Figure 4.8: Cluster Map organized by Region

IT-cluster could be built according to the locations of the offered jobs as shown in Figure 4.8. It's obvious that cluster maps are very effective visualizing and navigational tools, and that they comply quite well to the high requirements and demands of ontological visualization. On the other hand, the developer should take in consideration the high expenses during the implementation phase, not only concerning the algorithms needed for the cluster generation, but also the relatively high effort needed for the realization of the graphical components and navigational behavior of the map [VG02].

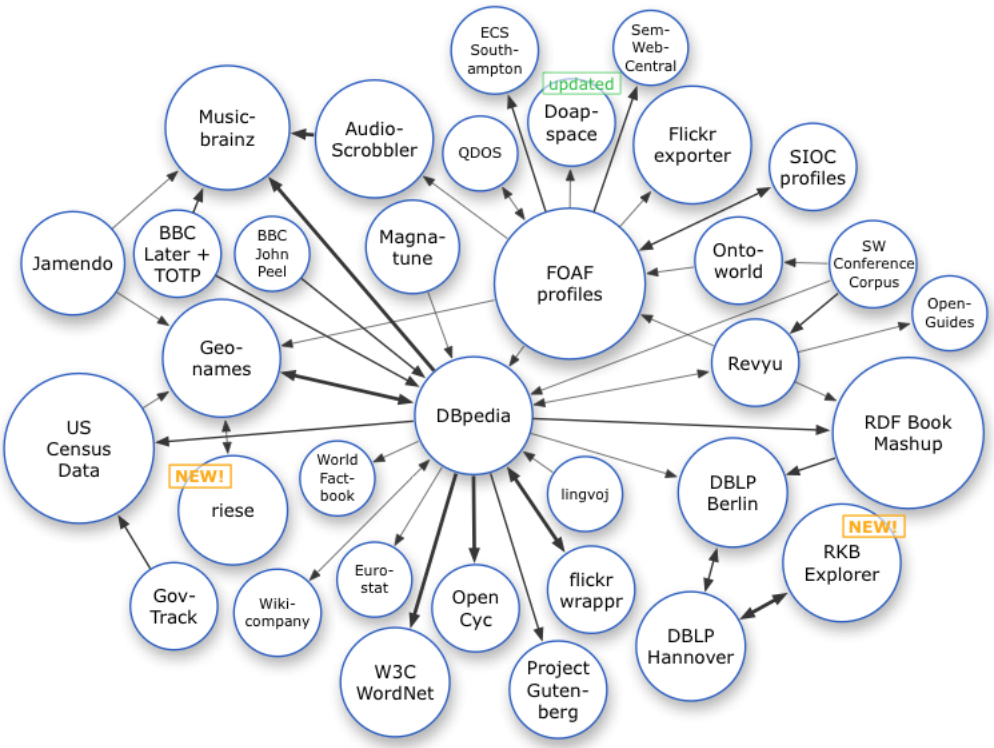


Figure 4.9: Datasets that are interlinked with DBPedia (Image by Richard Cyganiak, Licensed under a CC By SA 3.0 License)

4.5 Linked Open Data

The term "Linked Data" refers to a method of connecting and sharing data on the web by using referenceable URIs. This data landscape, just like the standard hypertext web, consists of documents situated on the web; however the relationships here are more arbitrary and are not defined by hyperlinks but are rather described by RDF. According to Berners-Lee [Ber07] there are 4 rules that need to be followed in order to provide a sensible and interconnected web growth:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information.
- Include links to other URIs so that they can discover more things.

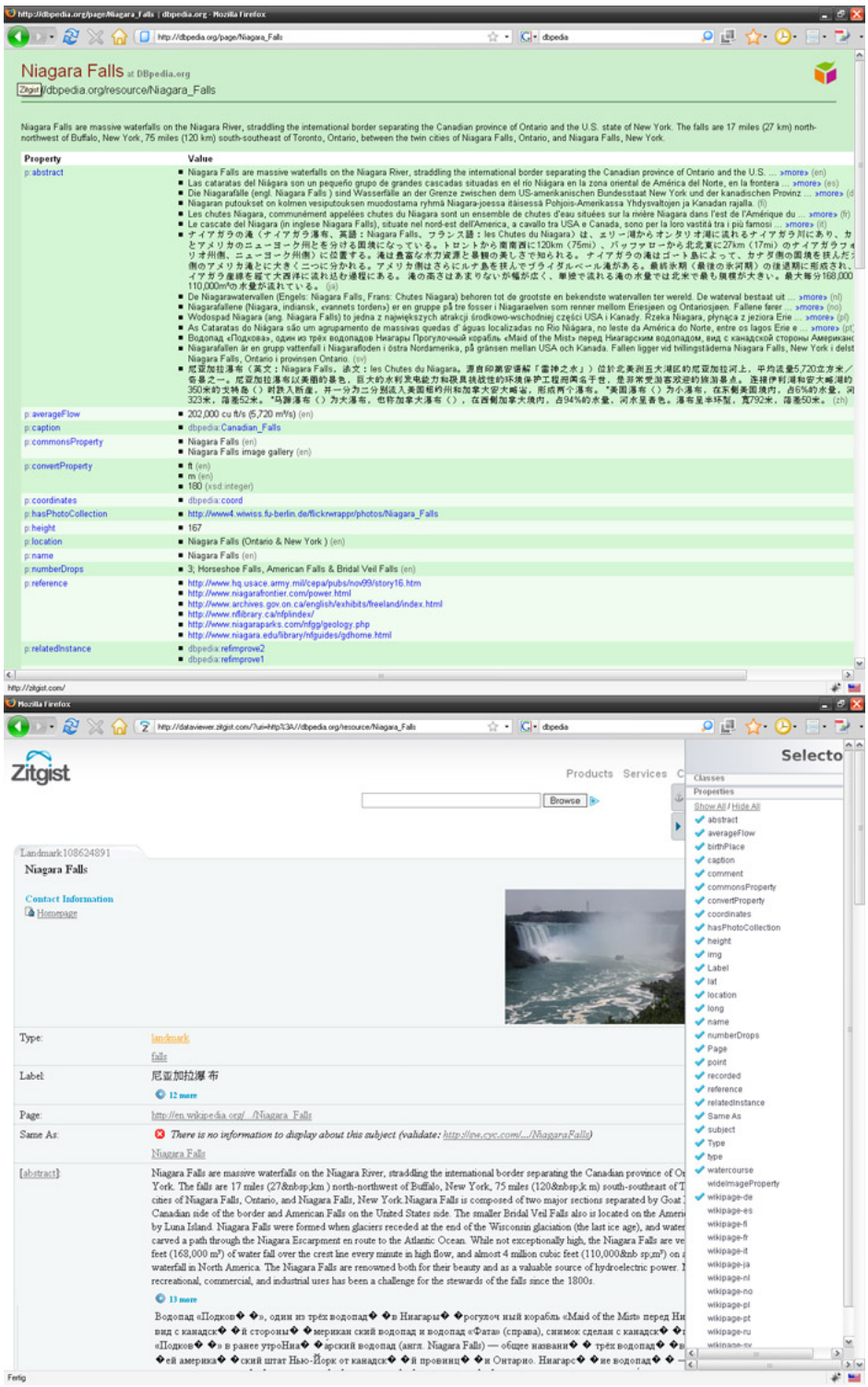


Figure 4.10: DBPedia URI viewed in a web browser and in Zitgist

A good example for a Linked Data realization is DBpedia. DBpedia is a community which aims to extract structured information from Wikipedia and to make this information accessible over the web. Furthermore, it allows to link diverse dataset from the web to the Wikipedia data, and to perform complex queries against the derived datasets. The extraction of structured data from Wikipedia occurs over the Wikipedia database dumps that are published on a regular basis, which are mapped onto RDF and added some additional information according the content of the articles. There are several ways to access those created datasets over the web [SA07]:

- Linked Data: Publishing of RDF data over URIs and the HTTP protocol.
- SPARQL Endpoint: Accessing the data over the SPARQL protocol, this is an RDF query language.
- RDF Dumps: Serializations of the datasets which are available for download.

In the meantime there is also a growing number of interfaces specially designed for the browsing of RDF data on the web. Most of them are directly available over the web, or directly accessible over the OpenLink Data Explorer Extension, which is an Add-on for the Mozilla Firefox browser. All these Linked Open Data browsers have in common that they have to cope with a very high amount of interconnected data and with the difficulties of presenting those data loads to the user in a sensitive and pragmatic way. Figure 1111 shows the Zitgist browser as an example compared to a simple browser view of a DBpedia URI. As we can see, Zitgist offers some tools for the filtering of the obtained data or more precisely it gives the possibility the blend out undesirable data, providing the user with a better outline of the content.
All things considered, the LOD browsers steel seem to be in the stage of development, especially when we take a look at the range of provided browsing functionalities or their effective appliance. However, one should mention that those browser can only perform accurately if the underlying data is suggestively linked, which is of course the desired state, but which will probably take some time to achieve.

4.6 Summary

In this chapter we analyzed the ongoing trends concerning the development of Rich Internet Applications. We took a look at the different courses that this development is pursuing,

and tried to convey which of these courses are pointing in the right direction and which could possibly emerge in dead ends. Furthermore, we gave an overview over the available frameworks and platforms that can be used for the development and deployment of those applications, ranging from rather generic Javascript-based frameworks, which can easily be embedded into already existent developing environments, to rather enclosed systems like Microsoft's Silverlight or Adobe's Flex. In the second part we tried to present an overview over the elements, methods and concepts that can be used for the visualization and organization of semantically rich data. We discussed some rather rudimental graphical elements, like lists, trees, and simple graphs, and how they can be combined together and enhanced in order to provide sensible navigational features and a well arranged visual presentation of the data. Finally we gave some insights into the concept of Linked Open Data and at the respective browsing interfaces, as this is a field that could contribute some interesting insights in the future concerning the organization unstructured data.

Chapter 5

Case Studies

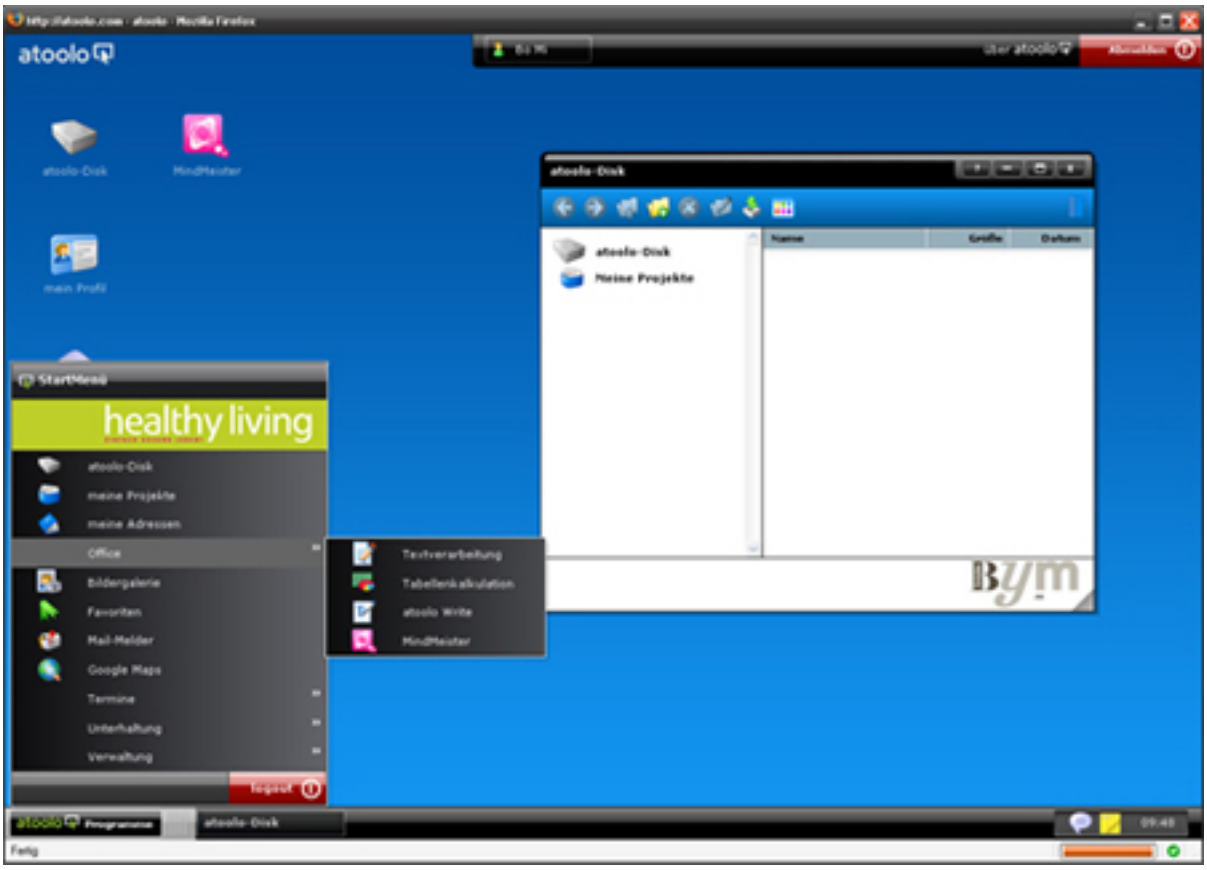
5.1 Introduction

In this chapter we will take a look at some case studies with an emphasis on the design of some ongoing web user interfaces as well as on several visualization techniques for semantically enriched data. The cases were chosen according to their relevance and innovations, either concerning the design and the implementation of web interface functionalities or the visualization and organization methods for unstructured, semantic data. This chapter aims not only to give an overview over those methods and functionalities, but also strives to derive expedient techniques and design approaches that could be used later on in the implementation phase of the SemDAV web user-interface.

5.2 Atoolo

Atoolo is one of numerous web-based virtual desktops that emerged over the past years, due to a growing orientation towards the internet concerning application development and data storage. This trend was also encouraged by the development of some new approaches to web-programming, facilitating the implementation and deployment of rich internet applications. Especially the growing prevalence of Asynchronous Javascript and XML (Ajax) involved a boom of various internet applications with augmented functionalities and improved user interfaces.
As the term "virtual desktop" may suggest, Atoolo is trying to mimic the functionalities, the usability and the overall look-and-feel of well-known desktop systems like Windows or Mac OS

X. After the login screen the user is confronted with a familiar, desktop-like environment, featuring desktop icons, Windows reminding task- and start-bars and an upper toolbar showing the actual user properties.



Screenshot taken from http://www.atoolo.com

Figure 5.1: Atoolo windows and menu

Considering the user-interface handling, Atoolo sticks with well tried and familiar desktop procedures like window-based content visualization, right-clicking context menus and it even provides some basic drag and drop functionalities here and there. Complying with the Windows desktop, the taskbar at the bottom of the screen appears as the main navigational reference point, being always visible and providing an overview and a quick access to every opened window. The whole application is embedded in a simplified browser window, with all toolbars and menus blended out, which intensifies the overall desktop-feeling. The real desktop is located at the background of the application. It contains some basic icons with a customizable arrangement (drag and drop) however it is not completely customizable. Al-

though the user has the possibility to change some appearance attributes, like for instance the background image, he or she cannot just place any desirable object on the desktop. Despite those restrictions, all in all Atoolo successfully manages to convey a desktop-like experience.

5.2.1 Features and Layout

Drag and Drop

The drag and drop functionalities of Atoolo are quite rudimentary and are mostly used for the rearrangement of the desktop icons. Also it is possible to drag some of the applications situated in the menu to the desktop, creating shortcuts for a frequently use. Unfortunately, this is more or less the whole magnitude of the drag and drop functionalities. When considering that Atoolo is designed for data storage, it would seem more useful to implement the drag and drop functionalities for the integrated file manager, which would certainly result in an overall usability boost.

Context Menus

Almost every object in Atoolo has a right-click context menu. For the desktop icons those menus seem quite rudimentary at first, containing only three different options ("open", "delete", "rename"), however, considering that Atoolo is only a virtual desktop, those options prove more than adequate. In the Atoolo-Disk, the virtual file-system explorer, the uploaded files even have a more sophisticated menu, offering some features like a preview or a send over email option, always according to the selected file type. However, not every object in Atoolo has an assigned context menu, like for instance the taskbar or some random window objects, involving that a right mouse click on those objects will show the default context menu of the browser which affects the overall homogeneity of the system. Also it raises the question whether or not the default browser context menu should be overridden at all, as it contains some crucial functions.

Window Handling

Atoolo attempts to mimic the window handling of real desktop system, implying free window positioning as well as minimizing, maximizing and resizing functions. The integration of the z-axis certainly results in a usability enhancement, especially when considering the simultaneous representation of different applications. However, when talking about browser

integrated window handling, one should always consider the consequential performance issues. Browsers nowadays still aren't conceptualized for the merge of motion with dense graphical contents (due to the lack of memory for instance), resulting in a performance loss. Therefore, when dragging a browser integrated window, it is not unusual for the user to experience some unpleasant performance problems. However, Atoolo tries quite successfully to compensate those issues, by blending out the window content during the moving action. The windows still don't move as fluid as on the desktop, but then again we are talking here about a virtual desktop integrated in a browser, which should make a little performance loss quite acceptable.

5.2.2 Conclusion

Atoolo is a good example of a browser-based virtual desktop, with solid and useful functions and a regard on the performance. However, the question here should be whether or not an internet application should mimic the desktop in the first place. The functionalities that Atoolo provides are certainly useful, this is beyond all question, but is it really necessary to completely reproduce the desktop? An internet application could never achieve the same performance as a real desktop system, simply because the lack of resources, so is it really sensible to take the real desktop as the guiding model? The above-mentioned functions are certainly useful and they are enhancing the overall usability of an internet application, but it is not necessary to copy the whole desktop just in order to find a useful application for those functions.

5.3 Mindraider

Mindraider is a Semantic Web outliner and is a free software project und the leading of Martin Dvorak. Its purpose is to organize the cognitive base of the user, but also to manage the appending of web and local resources, aiming a quicker navigation and a more effective knowledge representation. In other words, Mindraider aims to help the user to correlate information related to each other. In order to do so, Mindraider introduces some predefined termini and constructs for the information management and retrieval:

- **Concepts** Concepts are building blocks of the user's cognitive base that can be represented by Mindraider. A concept can contain tags, web links and any kind of attach-

ments; also it can be assigned to a certain category, which serves as a sort of labeling, due a special color that every category is assigned to. Concepts are arranged in a strictly hierarchical way, which means that every concept can have multiple children but only one parent.

- **Notebooks** Concepts are organized to Notebooks. These are not arranged hierarchically, however they can be merged to folders.
- **Folders** Folders enable the user to organize Notebooks into thematical domains.

5.3.1 Features and Layout

Mindraider is presented in a multiple panel design, which can be switched to different panel compositions according to some predefined templates. On the top of the application, a relatively broad toolbar is situated, featuring some navigational and organizational tools, like for instance backward and forward buttons, a search field and a selectable design assortment. The area beneath the toolbar is vertically divided into three frames. The top left frames acts as a navigational frame and is subdivided into three areas:

- **Notebook labels** This panel shows the different labels (which are actually the previously mentioned folders). In the upper part of the panel is a filter search field, allowing quick label search and display. The notebooks of the selected label are shown in the notebook panel beneath.
- **Notebooks** This panel contains a simple tree for the browsing of notebooks. The notebooks displayed here depend on the currently selected label. Also there are buttons for the creation, the removal and the editing of notebooks.
- **Recent Concepts** This panel acts as a sort of browsing history, showing all the concepts that have recently been opened.

The middle frame is the actual content representation area of Mindraider, which is horizontally separated into two different panels. The upper panel consists of a tree representation of the currently selected notebook, showing the hierarchical structure of the enclosed concepts. Also a brief content preview of the concepts is shown, featuring the first line of content along with the creation date and the color of the assigned category. In the upper part of the panel is a toolbar with some managing tools regarding the displayed content, allowing the user to create, delete and relocate the concepts. The lower panel actually represents the same content

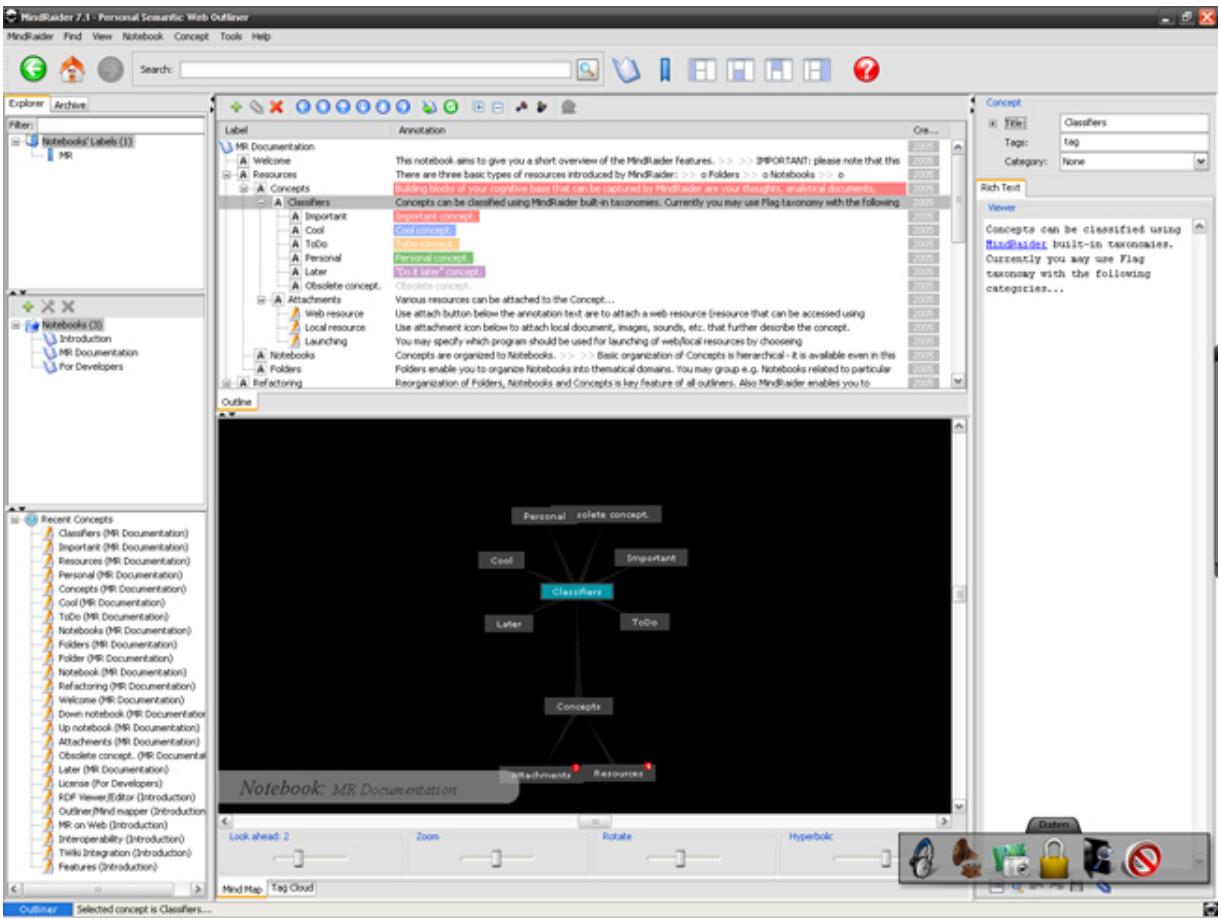


Figure 5.2: Screenshot of the Mindraider Application

as the upper one, however the content is shown in a maneuverable mind map instead of a hierarchical tree. The center of the map shows the currently selected concepts, surrounded by all the concepts that are in close relation to it, like the childs or parents for instance. However, the display of the map is quite customizable, featuring tools for the rotation and zoom of the map as well as the possibility to increase or decrease the displayed relations depth. Also this panel contains a second tab, which displays a tag cloud of all the concepts contained in the current notebook. Both panels, the tree representation as well as the mind map, are used for the navigation through the concepts of a certain notebook, and they each react on the selections of the other.

The top right frame is used for the management of concepts, allowing the user to add attachments, tags, categories and some random text to the selected concept.

5.3.2 Conclusion

All in all, Mindraider is a pretty easy-to-use application, even for inexperienced users. By using well-known and established procedures and representation methods, like for instance the left to right navigational concretion, which meanwhile is intuitive to almost every user, Mindraider ensures that even users, which are rather unfamiliar with such sorts of knowledge management systems, are able to cope pretty quickly with the handling and management of the data. Furthermore, the user integration is amplified by the breakdown of the main data representation panel into a classical, hierarchical tree and a mind map, providing a familiar reference to the user and enabling him to intuitively adapt to the new data visualization. However, such a high level of user integration is only possible due to the rather hierarchical nature of Mindraiders data structure. Although Mindraider provides efficient data visualization features, the basic concept of the stored data is hierarchical, which, considering the basic principle of data integration, doesn't require a rethinking from the user.

5.4 Personal Brain

Personalbrain is an ontology-based knowledge-management software, developed by The-Brain Technologies LP. As the name may suggest, Personalbrain refrains from a hierarchical data representation and management, and focuses on a more "humanized" knowledge representation instead. In order to do so, it uses a new concept of data representation as the most vital element of its knowledge administration, a several data format referred to as "thought". A thought can be related to other thoughts and it can hold any kind of digital information, like for instance attachments, shortcuts, websites or personal notes. Although this may be a nice concept, it is not the concept-in-itself that poses a problem for most developers, but rather its visualization. However, the visualization and the user interface presented by Personalbrain are certainly some of its most outstanding features. The whole navigation and most of the functions, as well as the creation and the import of new content are accessible through a graphical mind map that Personalbrain uses as its main interaction tool. It is this interactive mind map that marks Personalbrain, especially through its intuitive and easy-to-use design, and it certainly facilitates the introduction of the new data representation system to the user.

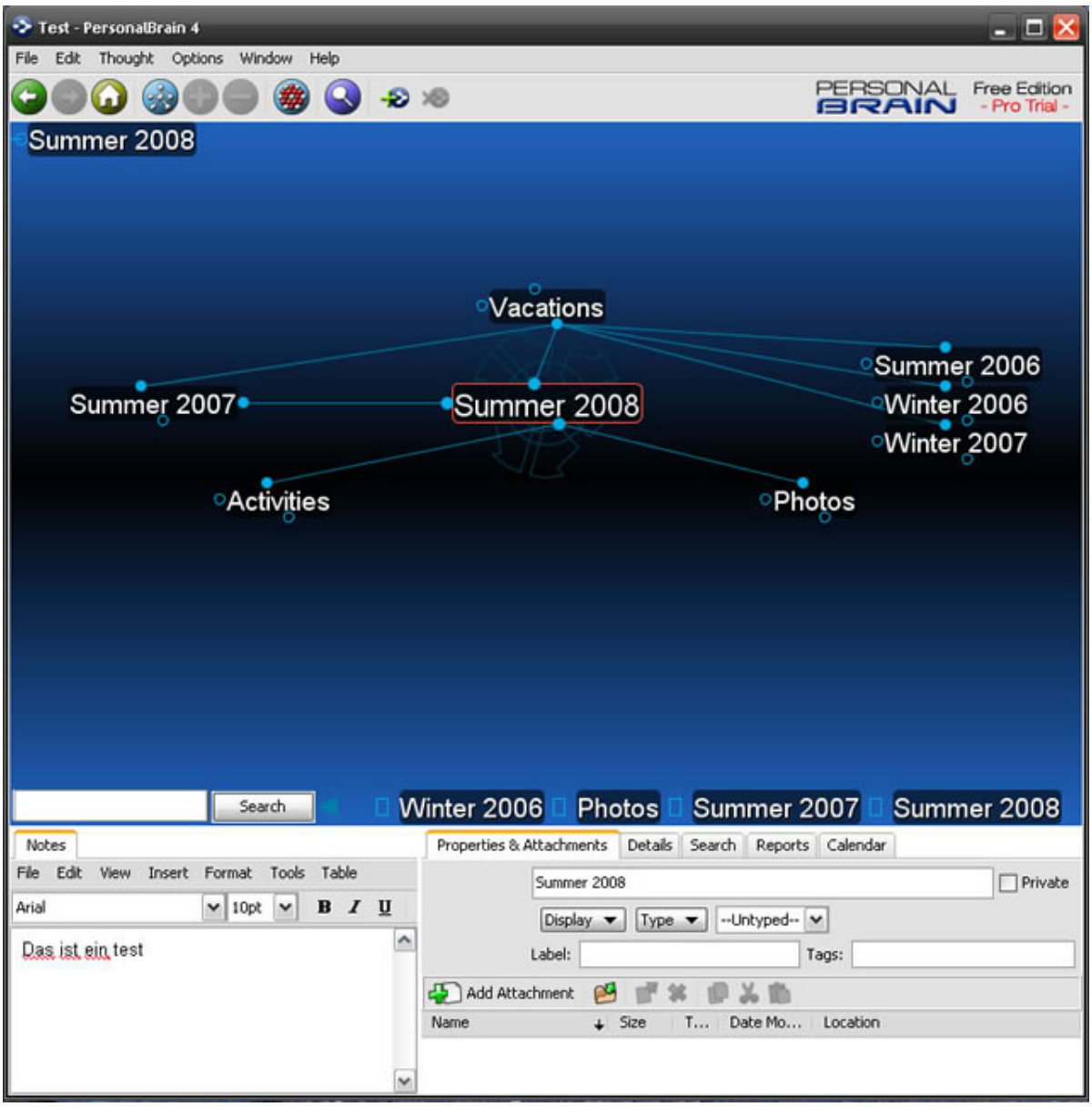


Figure 5.3: Screenshot of the Personalbrain Mindmap

5.4.1 Features and Layout

Interactive mind map (Plex)

The Plex (an interactive mind map) is the center of the application, and it is used for navigation and for the representation of content, as well as for the creation of new content and relationships. The core of the mind map is the "active thought", which is simply the currently selected thought. The active thought appoints the map that is shown to the user, aligning

all the directly associated relationships and thoughts. All these related thoughts are graphically situated in certain zones, defined by the type of relationship those thoughts have to the selected one:

- **Parent thoughts** Every thought that has at least one child thought can be defined as a parent thought, and every thought can have multiple parents. The parents of the active thought are situated in the upper area of the map.
- **Child thoughts** Of course, a child thought can only be a thought that is assigned to at least one parent. Child thoughts are situated in the lower area of the map, right beneath the active thought. Multiple child thoughts of the same parent are referred to as siblings, as they are belonging to the same genus, and they are shown in the right area of the screen.
- **Jump thoughts** Jump thoughts are used to represent any other type of relationship to a certain thought, and they are intended to be a sort of quick shortcut to thoughts that are not necessarily in a close relationship to the active one. They are situated in the left area of the screen.

In order to create new relationship, Personalbrain makes use of so-called gates, which are little circles situated on the top, left and bottom side of the active thought. Those gates can be dragged, either to an already existing thought, which creates a relationship to this thought according to the type of gate that has been dragged, or to an empty space, which will create a new thought. Also it is possible to drag files from the desktop to the map, adding them as an attachment to an existing thought or creating a new one referencing to the file. This simple drag and drop system combined with a neat application of graphics and animations makes it extremely easy for an inexperienced user to quickly cope with the probably unfamiliar interaction style.
Also there is the possibility of a "pin" creation. A pin is a shortcut to an existing thought somewhere in the Plex. A pin can be created on any given thought, simply by right-clicking it and by choosing the "create pin" option from the context menu. All pins are aligned horizontally in the upper part of the screen, allowing a quick access to the linked thoughts without having to navigate the Plex. As an addition to the navigational features, there is also a list with all the recently passed thoughts, which is located at the bottom part of the screen. This passed thought list is serving as a history and is showing the navigational path that has been taken through the Plex.

Tab tools

Personalbrain provides a series of tools situated in a tabbed-pane in the lower frame of the screen. Those tools refer to the currently active thought and they are accessible at all times:

- **Notes** The notes tool is a text editor allowing the user to quickly add and retrieve any desired notes to the selected thought, along with some extra features like for instance the timestamping of the entries.
- **Properties and Attachments** This tab allows the user to change the properties of the thought, like for instance the thought type, the tags, the label and the displayed image. Also it provides an interface for the adding of new attachments, either over a file system or simply by drag and drop.
- **Details** The detail pane provides some basic information of the active thought, like the overall size of the thought, the modification and creation dates and the internal id.
- **Search** The search tab displays the most recent search that has been performed. The results are summed up into two different views, starting with the thought search results, showing the thoughts that matched the search criteria, followed by the content search results bellow. Considering the colors as well the overall layout, the presentation of the results is almost identical to Google's, which will certainly evoke some recognition by the majority of the users. There is also the possibility of an advanced search, allowing the user to extend, restrain or filter the actual search.
- **Reports** The report tab show all the thoughts of a particular brain, providing filtering options according to the thought types, dates, attachments and names of the thoughts.
- **Calendar** The calendar enables the user to set up reminders for a particular thought.

5.4.2 Conclusion

All things considered, Personalbrain is a quite innovative and also intuitive knowledge management system. It consistently forbears from conventional hierarchical data storage systems and fully integrates an ontological and more human-related knowledge representation. However, although this relatively new approach to knowledge management may require some rethinking from the user, the overall learning process is simplified by the successful use and integration of functional but also easy understandable mechanisms in the user interface design. Especially the interactive mind map is highly intuitive, allowing the inexperienced user

to quickly adapt to the new form of information representation and to successfully interact with the system after only a short familiarization phase.

5.5 Grokker

Grokker is an information management solution developed by Groxis inc., and it aims to improve the organization, exploration and share of digital information. It provides some interesting features, which lift Grokker out from common known information search and visualizing engines, like for instance the federated content integration, a dynamic clustering and a quite unique visualization and exploration of large results sets. Grokker's clustering engine provides federated content representation of multiple and basically different data sources, which represents quite a challenge when it comes to build a set of intelligent and user relevant clusters. In order to do so, Grokker utilizes data augmentation techniques for an intelligent analysis of disparate data sources and for a useful clustering of the content. Also, Grokker aims to provide accurate labels that convey the content of each cluster, by analyzing not only the indexing of the results but also their complete body. As a result, the clusters enable the user to easily identify the desired entry points into large sets of results [Gro08b, Gro08a].

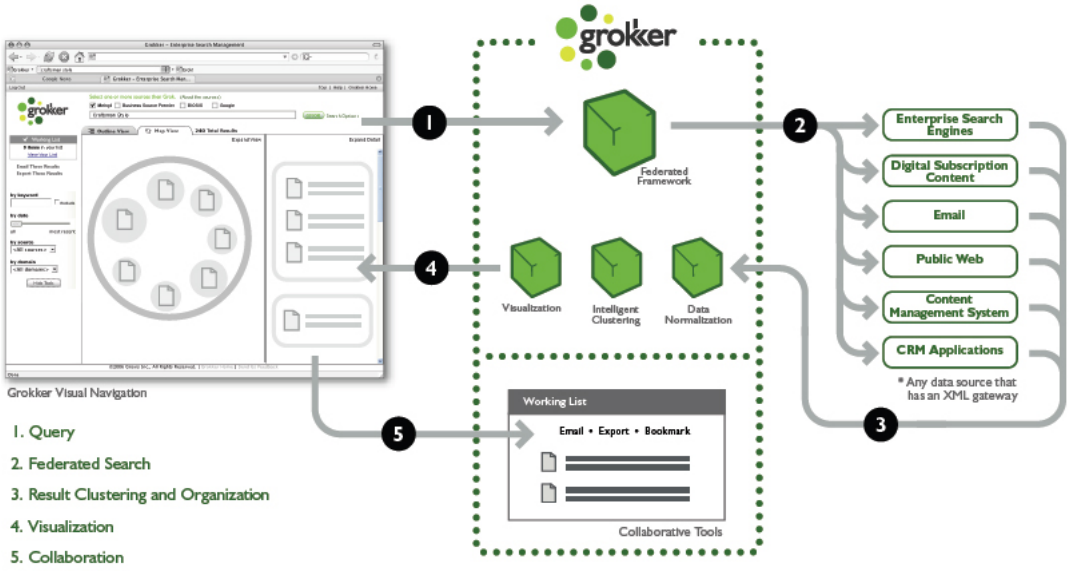
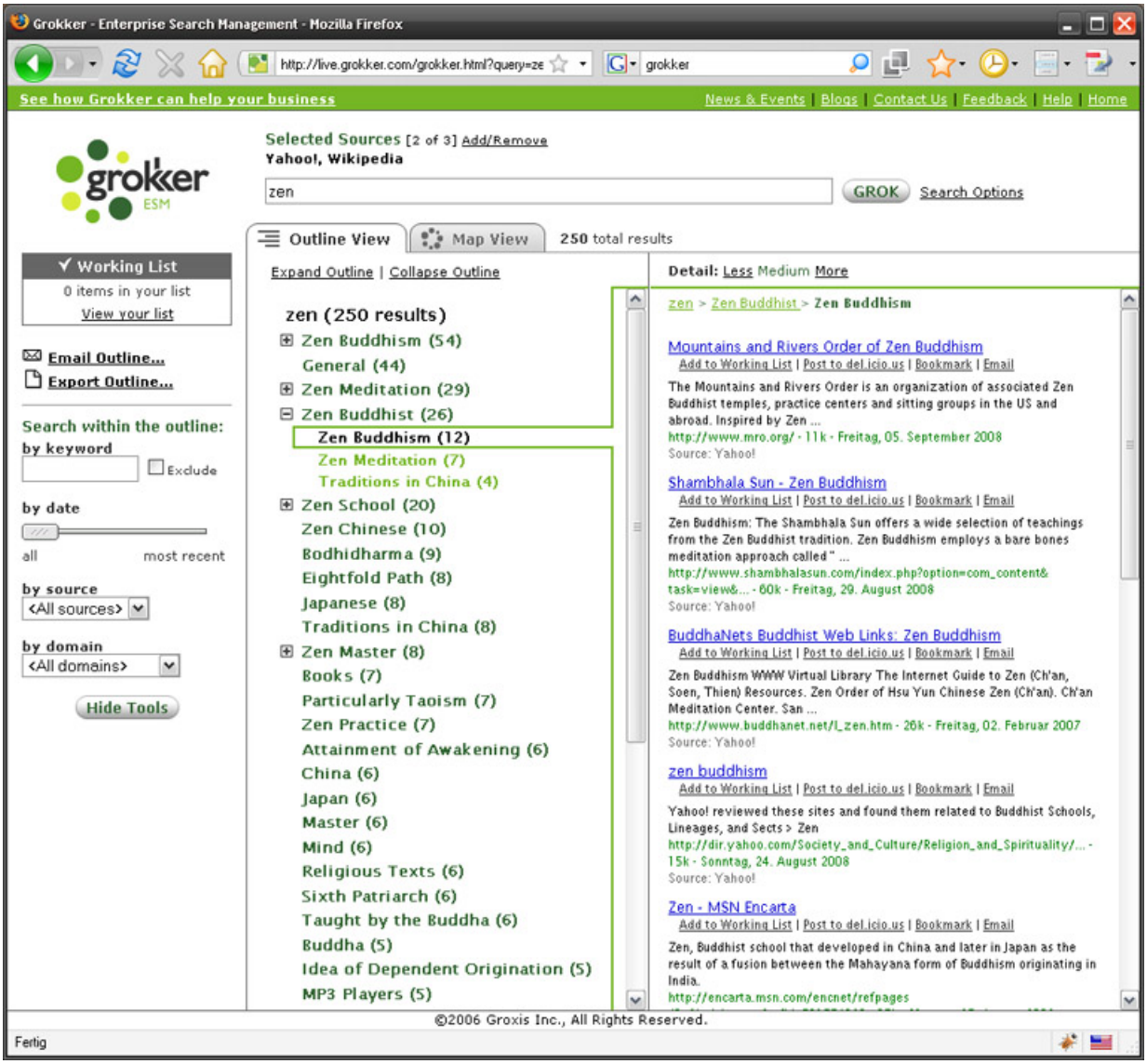


Figure 5.4: Grokker Architecture Overview [Gro08b]

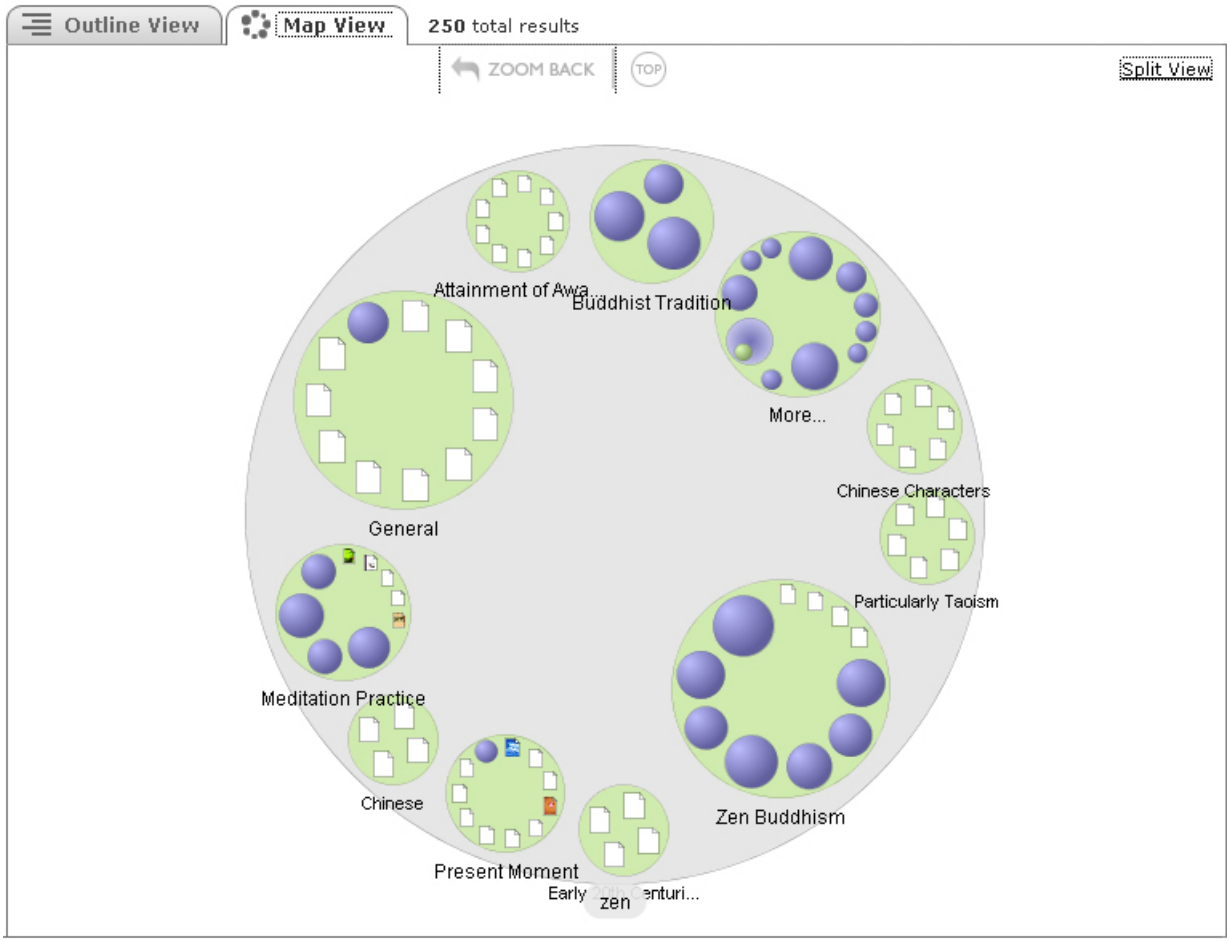
5.5.1 Features and Layout

Grokker's layout is vertically divided into three separated panels, with an upper panel situated beyond them. The upper panel is used for the input of the search queries, and it basically consists of a search field, a search button and the possibility to change some search options like for instance the addition and removal of data sources or the customization of the wait time that the user is willing to wait for Grokker to assemble its search results.



Screenshot taken from http://live.grokker.com/grokker.html

Figure 5.5: Grokker Layout



Screenshot taken from http://live.grokker.com/grokker.html

Figure 5.6: Grokker Map view

The middle and the top right panel of the screen are actually belonging together and their purpose is to visualize Grokker's search results. The middle frame shows a hierarchical outline view of the search matches, conflated to higher-level clusters, titled with sensible labels derived from their content. Those clusters can be subdivided into further sub-clusters, which are accessible in a tree-like manner by pressing the expand-button situated on the left side of the label. The outline view however, can be switched to a quite unusual map-view, which is used to graphically visualize the generated clusters. The map actually consists of a big circle, containing further small circles circularly arranged along the border, which on their side also act as a container for sub-circles arranged in the same manner. Every circle represents a cluster, which on the one hand is defined by its label and on the other hand by its diameter which depends on the size of the results contained by this certain cluster. By clicking into one

of the clusters, the application zooms into the cluster's content, showing a more detailed view of its sub-clusters. The right frame displays the actual search results, aggregated according to the currently selected cluster. Considering the format and the color of the search results, the overall appearance is obviously very similar to the style presented by Google. There are some additional features however, like for instance a short line with some practical links, giving the user the possibility to quickly bookmark, email or add the desired result to a working list.

On the top left side of the screen, Grokker provides a tool panel with some interesting features. An additional text field allows the user to perform a search within the generated outline, by including or excluding the entered query. Also there are two select boxes, which can be used to filter the outline by the underlying sources or the contained domains. The most interesting tool however is the horizontally adjustable date controller, which filters the outline results according to their date, allowing the user to customize the result set by selecting only the most recent results for instance. This panel also contains the working list, allowing a quick access to the associated search results.

5.5.2 Conclusion

It is quite evident that Grokker introduces some relatively innovative concepts concerning the user interaction design. It is In particular the rather unique and intuitive mind map, that Grokker is characterized by, that offers the user a new and well arranged way of exploring the desired content. Of course such a content representation is only possible due the way that Grokker aggregates the search results and summarizes them to predefined clusters. This content processing may also implicate some negative effects, as for instance users that are not used to such ways of content representation may feel forced to browse the results by means that are predetermined by the application. However, Grokker at the same time provides very common means of content representation, as shown by the right frame, which displays the result in a manner very similar to that of Google's result pages, ensuring user recognition and providing means for an easier adaptation to the new ways of content browsing.

5.6 Summary

In this chapter we analyzed some interesting cases concerning the user interface design for the visualization and representation of semantically enriched data. We ascertained some promising concepts and approaches of visualization of unstructured, semantic content, as well as means and techniques for its manipulation and organization. We saw that there are not one, but several possible courses of design, all of them having their advantages and disadvantages. The insights that we gained in this chapter will be used later on in the design phase of a web user-interface for SemDAV repositories, making use of the best practices that emerged throughout the case studies.

Chapter 6

Design and Implementation of a Web User Interface for SemDAV Repositories

6.1 Introduction

The purpose of the following chapter is to mediate an insight into the design of the web user interface for SemDAV repositories and to give an overview over its basic architecture and implementation. All the inferences made throughout the previous chapters should intensify in this attempt, by applying the resulting methodologies and avoiding the assumed sources of error, of course under the abidance of the available recourses.

6.2 What is SemDAV?

The SemDAV project, which is carried out by the University of Vienna and the Research Studio Digital Memory Engineering, aims to enhance the storage of unstructured data through semantic enrichment. The SemDAV protocol is based on the HTTP protocol, with the intension to extend protocol with semantic features. Its purpose is to enrich the transferred binary content and attributes with semantic metadata in order to provide means of transaction for semantically enriched content while guaranteeing a high level of compatibility at the same time. The protocol addresses SemDAV repositories, which store the content and the associated semantic metadata. The purpose of the SemDAV web user interface is to provide means to browse and search those repositories as well as to visualize the returned unstructured data. [Sch06, SK06]

6.3 Architecture

The SemDAV web-client is implemented with the help of a java application framework. This framework is based on a single java servlet, which is deployed on a tomcat server and which acts as a dispatcher for the request/response communication with the system. The framework offers a Swing-like environment concerning the development of applications, for which it creates a XML/HTML output which is sent to the browser. The requests coming from the browser are intercepted by a javascript engine with implemented Ajax and Rich Internet Application functionalities, which also is responsible for the integration of the returned responses into the page. The SemDAV client provides all functionalities for the user interaction, whereas the actual content related functionalities are accessed through a high level API provided by SemDAV. This API interacts with the SemDAV repository, which can be situated on the same or some remote server, in which case the communication is taking place over the HTTP protocol. Here is once more a brief overview over the major parts of the SemDAV web client architecture:

- **Webbrowser** This is the front end of the client and it consists of the generated HTML and CSS. This is also the place where the instance of the javascript AJAX and RIA engine is executed. This engine is responsible for the Rich Internet Application effects, as for instance the drag and drop functionality, and for the AJAX communication with the server.
- **Web Server** The web server consists of an Apache Tomcat servlet and JSP container. The main servlet of the application framework is deployed here, and it acts as a dispatcher for the incoming requests.
- **Application Framework** The application framework is where the actual client is situated. The main part of the framework is the core API, which provides all the classes and functionalities needed for the implementation of an AJAX-based web interface. The SemDAV web-client is an implementation of this underlying framework.
- **SemDAV API** This is the API that provides the functionalities for the interaction with the SemDAV repositories.

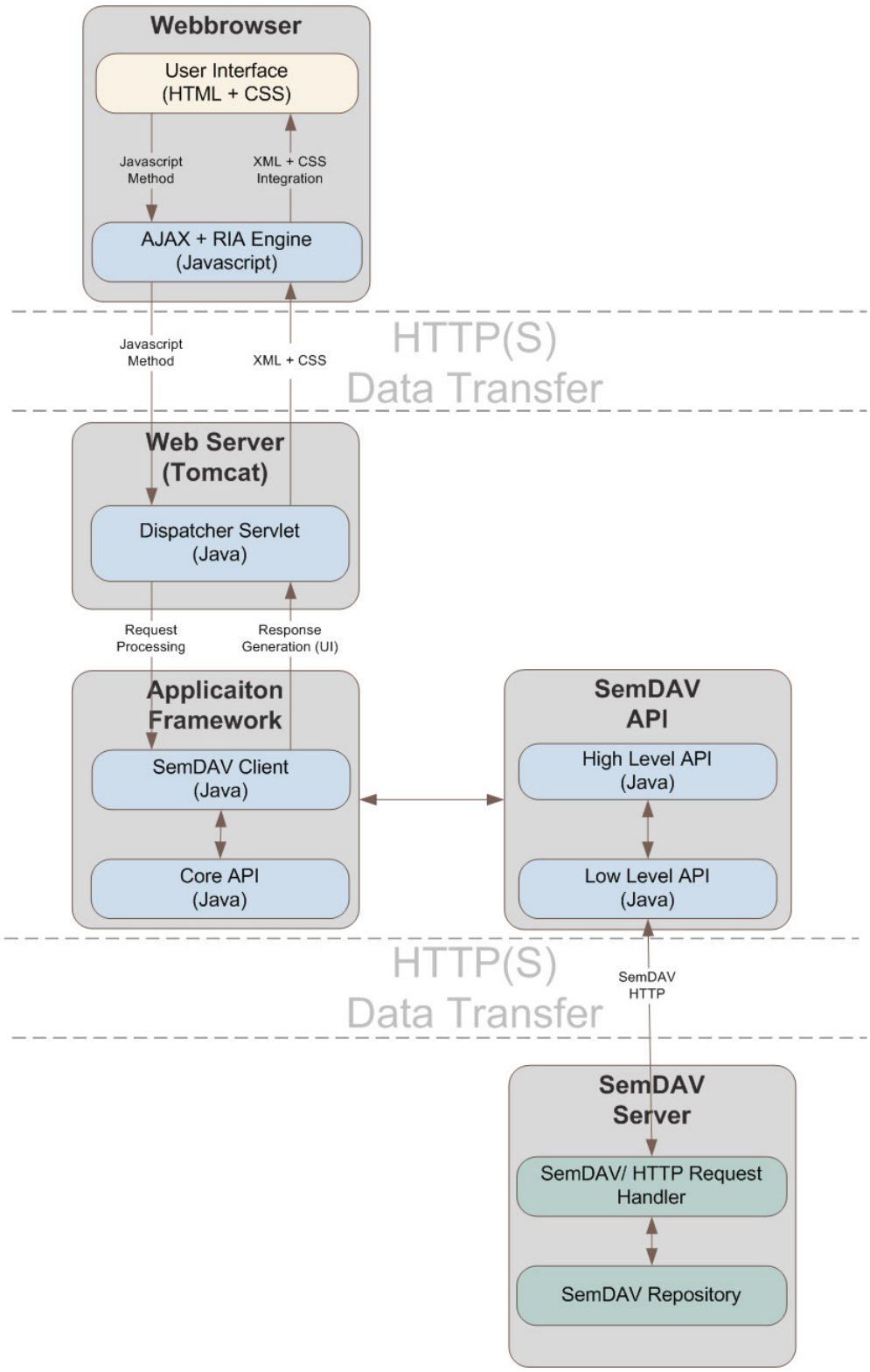


Figure 6.1: The Architecture of the SemDAV Web-client

6.4 Framework

6.4.1 AJAX And RIA Engine

In order to provide high usability and a high scale of navigational and functional features, the SemDAV web-client is making use of the Asynchronous Javascript and XML (AJAX) technology. The term AJAX stands for the functionalities provided by the XMLHttpRequest API, which has actually been around for quite some time, firstly introduced with Microsoft's Internet Explorer 5.0 and the integration of the ActiveX-Object. However, it was not before 2005 that this technology made its final breakthrough, most of all due to the appearance of Google's armada of web applications, like Google Mail, Google Maps or Google Finance, which were all demonstrating an Ajax based implementation.
The XMLHttpRequest provides means to dynamically retrieve data from a webserver via the HTTP protocol, without the necessity of reloading the actual site of the browser. Of course, the communication is still based on a request/response procedure, however, due to the XMLHttpRequest object, this procedure can now be performed with the use of javascript, transferring the communication in the background and allowing the exchange of data without the notice of the user. Furthermore, the XMLHttpRequest object processes the requests in an asynchronous manner, allowing the script to perform further tasks while waiting for a given response.
The SemDAV web-client provides a core Ajax engine, which is used for the communication with the dispatcher servlet and the Java framework beyond. Every dom-element in the displayed page has a unique id, which at the same time is its action-id. This action-id together with a certain dom-event (like for instance onClick or onMouseOver) indentifies a unique action, which is sent to the servlet via the XMLHttpRequest, which passes on the specified action to the Java framework, where it is processed. After that, according to the optional changes in the GUI that have taken place due the action processing, a XML response is constructed and sent back to the Ajax engine, where the response is integrated into the actual page. This response usually consists of the elements that have to be added, updated or removed from the page.
The integration of the Ajax functionalities obviously provides a whole range of benefits to the application. Most of all, the user can still interact with the page while a request is being processed and in some cases he doesn't even notice that there is a processing going on. Furthermore, due the reduced amount of the data transferred, as Ajax only updates the elements that have changed on the page, the overall transaction duration with the server is

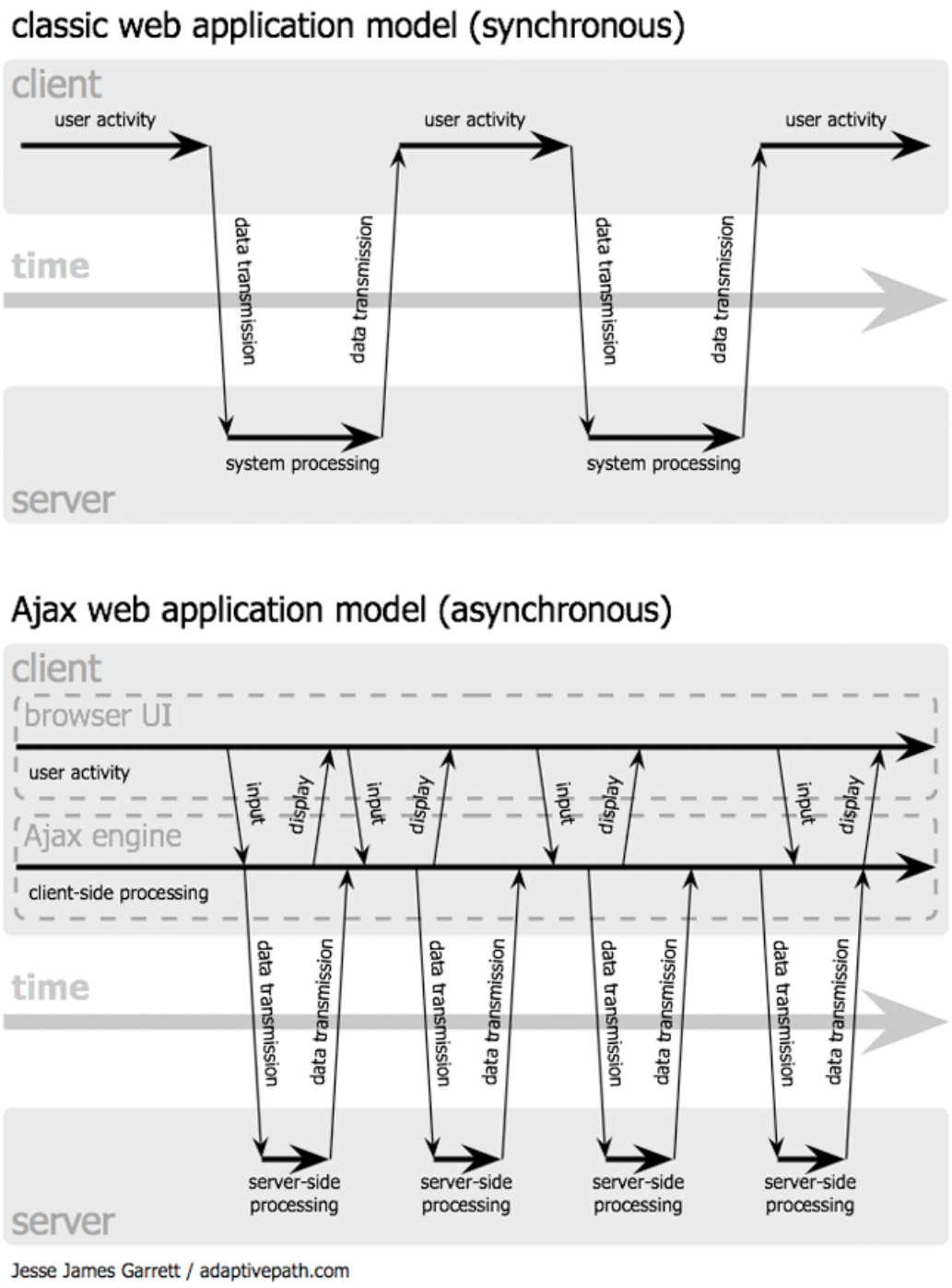


Figure 6.2: Synchronous and Asynchronous Web Interaction [Gar08]

distinctly reduced. Those attributes certainly highly contribute to the usability of the system, as the user often has the impression that the requests are processed instantly, which is partially resulting from the increased speed of the transactions or rather the decreased amount

of transferred data, but which is also consequential to the "invisible" nature of the transaction. Also, due to those characteristics, Ajax offers the possibilities to integrate a whole new range of functionalities that for a long time have been a privilege of the desktop applications. Some of those functionalities have also been implemented by the SemDAV web-client, and they are referred to as the Rich Internet Application (RIA) engine. Here is a short overview of these functions:

- **Frameset** As the RIA engine is entirely embedded in javascript, it is not possible, or at least quite costly, to execute a certain function over multiple pages. The reason therefore is that every page has its own instance and reference of javascript. Hence, a classical browser frameset would bear some serious problems, especially when it comes to some functions as for instance drag and drop. That's why the RIA engine provides an own implementation of a frameset, which is entirely based on javascript and div html elements.
- **Window Handling** Additional browser windows cause the same problems as a browser frameset, and additionally they are experienced as distracting by a large number of users. Here again, the RIA engine provides an own implementation of window-like element, that completely rely on javascript. Those elements are supposed to act as windows on the desktop: The can be moved, resized, closed or minimized.
- **Drag and Drop** Basically, the drag and drop function of the RIA engine pretty much mimics the standard drag and drop functionalities of the desktop systems. Every element can be defined as a dragable object, or as an object receiver. When a mouseDown action occurs on the dragable object, it can be dragged all over the page, including the implemented frameset and window objects. When the mouse is released, the system searches for a defined receiver underneath the release point. If a receiver is found, the system will perform the actions attached to it (these actions are all defined and performed in the java framework).
- **Context Menus** The implementation of context menus isn't such a big deal and it doesn't need any Ajax functionalities in order to work. However, an Ajax based implementation can provide some advantages, as for instance the context menus don't have to be embedded into the page during the buildup process, which can improve the overall buildup speed (think for instance of a large tree, where every node has its own context menu). Furthermore, the menus can be created dynamically, which allows them to react and change according to a certain state in the application.

6.4.2 Java Framework

The SemDAV web-client is embedded into a java application framework, which provides all the necessary functionalities for the implementation of rich internet applications. The implemented applications are transformed into an xml/html output (by using jdom) and sent to the servlet for visualization. The servlet acts as dispatcher, handling all the request coming from the client, and sending back the generated responses. Every request reaching the servlet is a result of a triggered action, which is uniquely indentified by its action-id and event type. This action is forwarded to the holding application, where the correspondent action-event listener is performed. After the processing of the action-event, the application returns an xml/html response to the servlet, which consists of all the elements that have altered, or have been added or removed due to the performed action. Here is an overview over the most important two classes:

- **Application** This class represents an application. It contains all the necessary settings needed for the interactions with the server (the general html skeleton, with the needed javascript functions and styles), and from the client side, it could be regarded as the displayed page in the browser. The application acts as the most high level container for any kind of components, and is keeping track of all the registered actions as well as of the components that might have changed during the action processing.
- **StandardElement** This class represents any kind of element that can be added to an element container, as for instance the application. At the low level, those elements are a representation of standard HTML elements, like for instance textfields, divs or tables. However, those elements can be merged to more sophisticated components, giving the possibility to create custom and easy reusable components. Every element can also act as a container for further elements, as does for instance the or class, by implementing the IElementContainer interface.

For a better understanding, Figure 6.3 shows the code for a simple HelloWorld application. As mentioned before, actions are performed by the use of action-listeners that can be added to any kind of element and which are identified by the id of the element and a certain dom-event. During the action processing, the altered elements are automatically added to the output, and refreshed in the client view after the action has been successfully performed. Figure 6.4 is an example for simple action handling.

```
public class HelloWorld extends Application{

    public HelloWorld(){
        super();
    }

    public void init(){
        Div container = new Div();
        container.setCssProperty("margin-left", "50px");
        container.setCssProperty("margin-top", "50px");
        addElement(container);

        Text text = new Text("Hello world!");
        text.setCssProperty("font-size", "18px");
        container.addElement(text);
    }
}
```

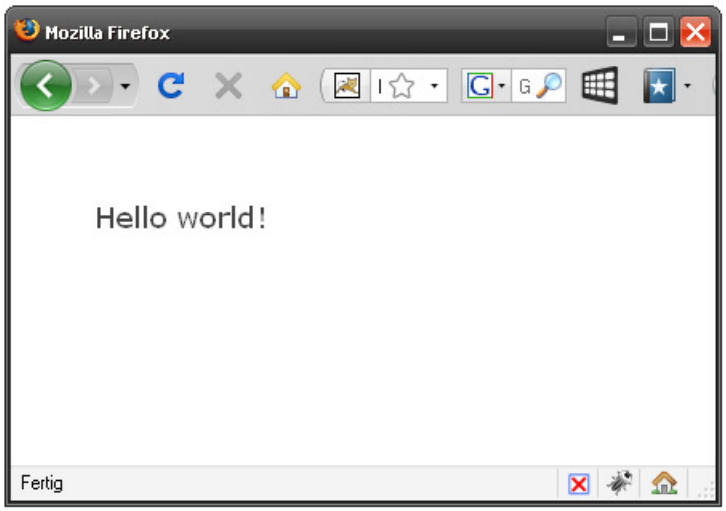


Figure 6.3: Hello World Code Sample

```
public void init(){
    Text text = new Text("Click me");
    text.setCssProperty("font-size", "18px");
    addElement(text);

    text.addActionListener(BrowserEvent.ONCLICK, new IActionListener() {

        public void doAction(BrowserEvent event) {
            Text response = new Text("Show me");
            addElement(response);
        }));
}
```

Figure 6.4: Example of Action Handling

Figure 6.5: Basic API Structure

6.5 Use Cases

The SemDAV web-client has to provide functionalities for the browsing, the presentation and for the search and filtering of ontology-based data, that is stored in the SemDAV repositories. The user needs the possibility to extract the desired data, preferably in a simple and intuitive manner. In order to accommodate those needs, the web-client aims to provide easy to use functions, which are based on the data structures and the ontology of the SemDAV reposi-

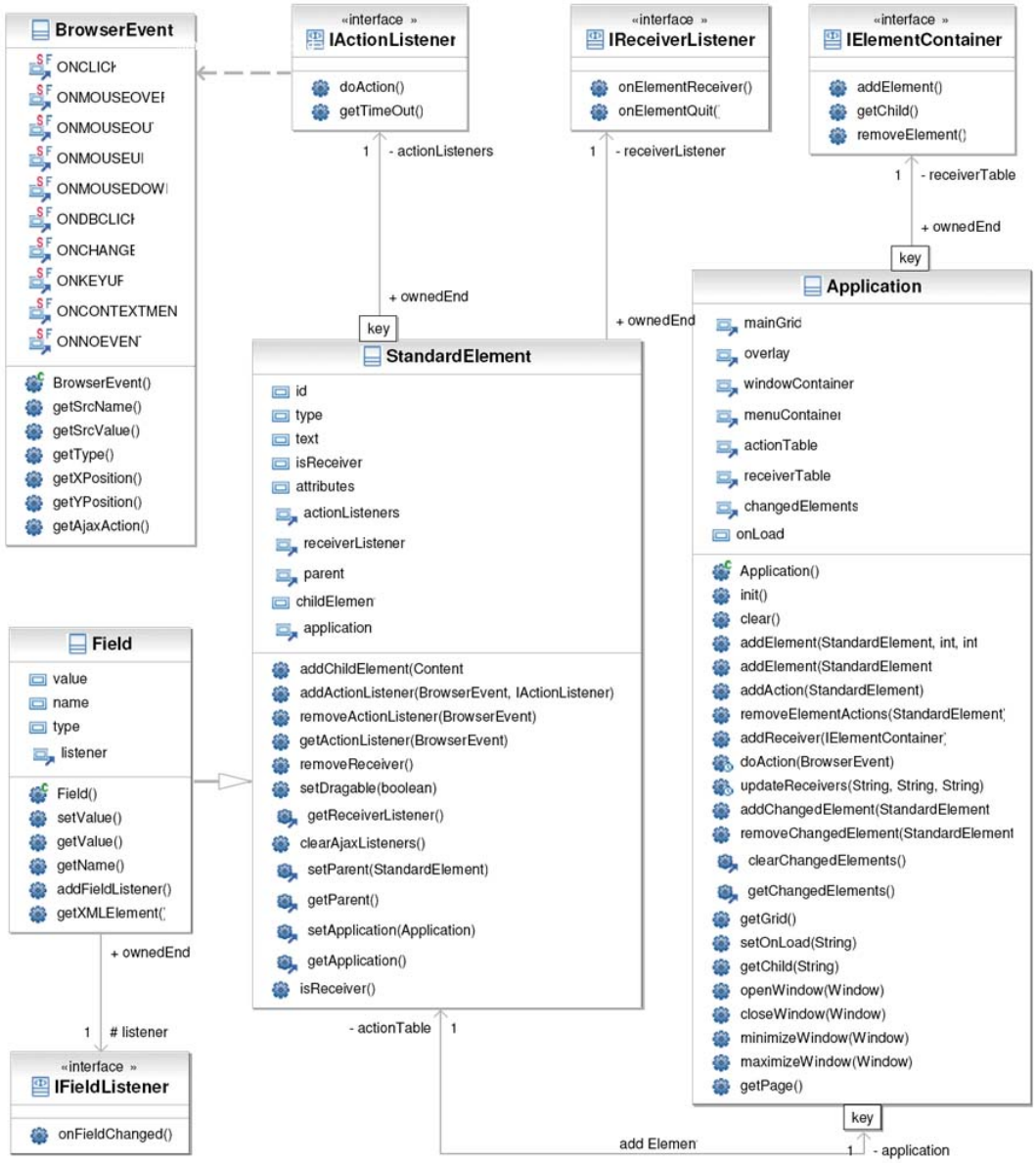


Figure 6.6: Event Handling Framework

tory.

The browsing and the data-selection of the SemDAV web-client are completely driven by a filtering component that is situated on the top of the page. The user creates a data representation by adapting the filter properties to the attributes of the desired data. The filter adapting is done by the adding and the removal of filter entries, which can represent any kind of data-type provided by SemDAV that are used for data integration, as for instance a tag, a spect or some attributes (for a detailed explanation of the SemDAV data-types see [SK06]).

After the filter selection, the system presents an overview of the siles that matched the filter properties to the user, and gives him the possibility to browse those siles and open them for a more detailed view of the contained data.

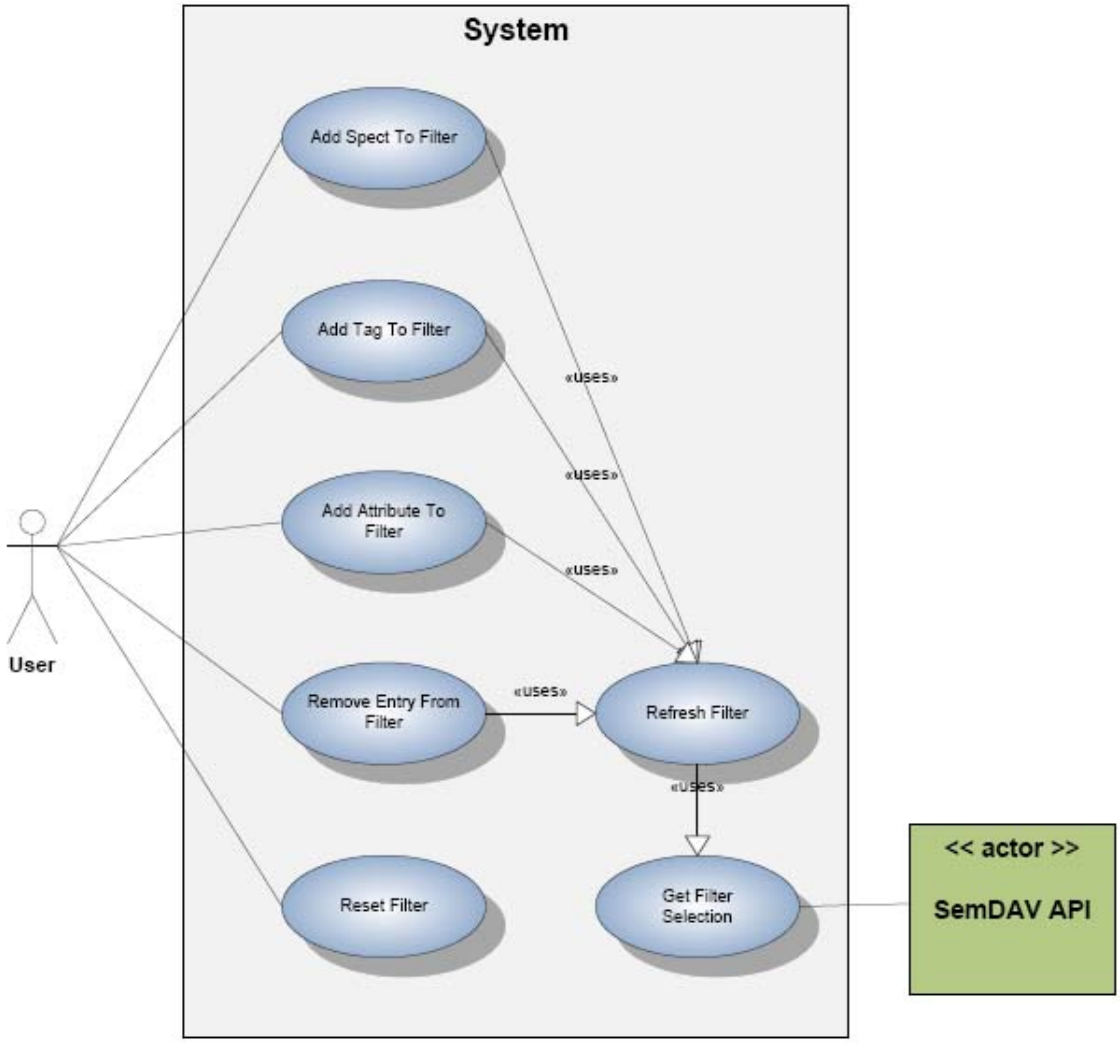


Figure 6.7: SemDAV Web-client Browsing Use-case

6.6 Basic Course of Design

The general l look and feel of the application should as far as possible resemble well-known data-browsing interfaces, as for instance the windows explorer, in order to invoke appliance recognition from the user. This mainly affects the basic layout of the interface, which should

therefore consist of a frame-divided view, situating the a navigational frame to the left, a tool-based frame to the top and a content oriented frame to the right, which also should occupy most of the available space. Concerning the interaction of the elements and the actions accessible to the user, those should as well emulate already established interaction procedures as much as possible, given the possibilities and restrictions provided by the browser. Here the new features and possibilities provide by an AJAX-based implementation will come into operation, providing the application with the necessary means for a desktop like interface implementation.
Aside from the standard toolbar or menu based interactive elements, the design of the interface aims toward an intuitive and facile form of interactions, which will be realized by the sensible appliance of context menus and drag and drop functionalities. Most items of the interface will be dragable, giving the user the possibility to intuitionally assemble the desired elements in order to browse the designated content. The context menus are intended to provide additional assistance for the user in order to perform the desired action, as every interactive element will provide its own context menu functionalities, acting as an additionally possibility for the user to achieve the desired result.
The actual content representation will be achieved by an icon based sile view, similar to the tile view of the windows explorer. Every sile will have an assigned icon depending on its content, which means the icons will vary according to the type of the allocated content, providing the user with a quick overview over the browsable results. The meaning of the icon will be amplified by some additional information concerning the sile, as for instance the name and the creation date. The actual content will be presented in separate view, which will be displayed in a movable window.

6.7 Layout

The overall look and feel of the SemDAV web-client sticks to rather familiar and common content-browsing application structures. The interface is divided into multiple frames, placing the navigational and browsing part on left, whereas the content related part is situated on the right, occupying most of the available space of the interface and making it the focused area of the application. On the top side of the application, a toolbar panel is situated, providing quick access to the most crucial functions. On the bottom side of the application lies the pocket view, which is a sort of deposition tray for desired siles. In order to present the content and detailed information of the siles, the SemDAV web-client works with integrated windows,

which can be moved, resized, maximized and minimized into the pocket if desired.

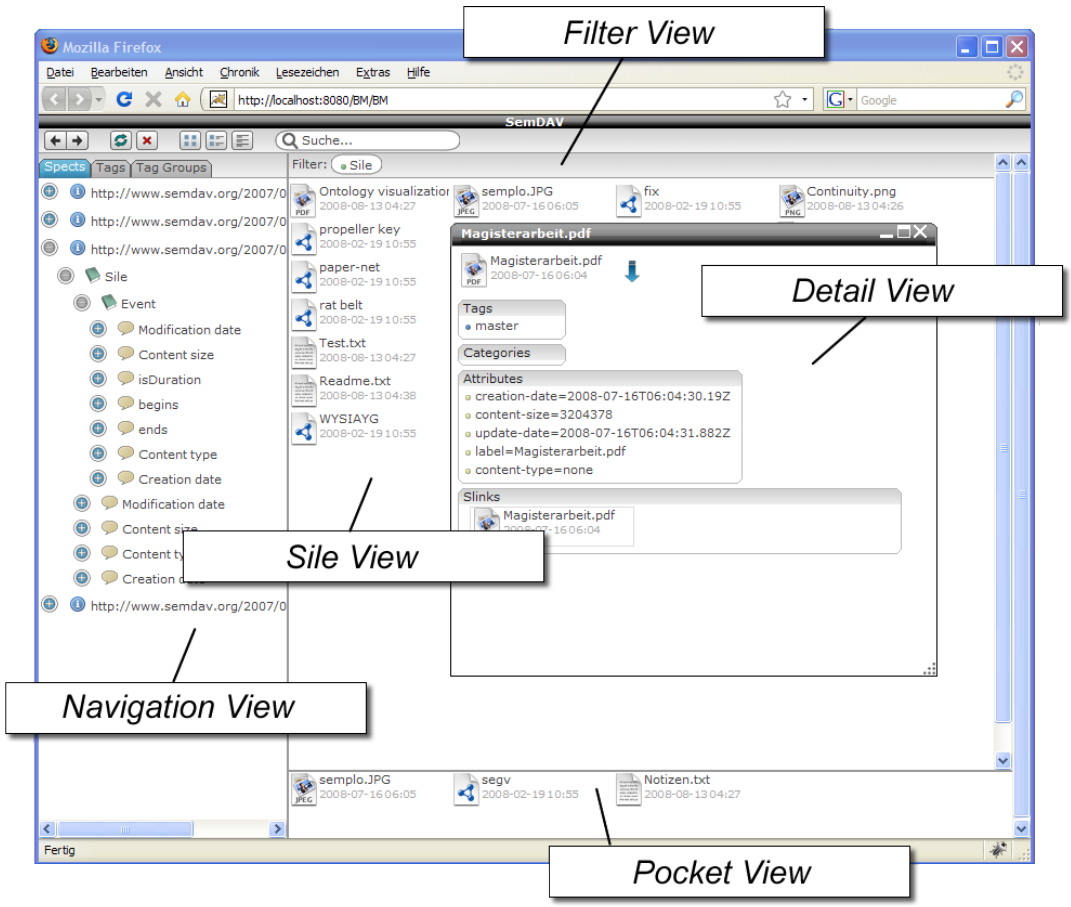


Figure 6.8: Layout of the SemDAV Web-client

6.7.1 Filter View

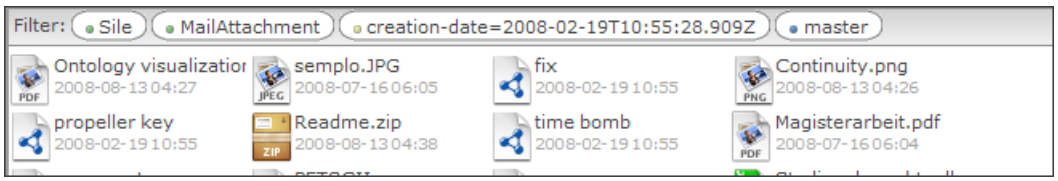


Figure 6.9: Filter View of the SemDAV Web-client

This view shows the currently made restrictions or the selections for the shown data. The filter can contain categories, tags or attribute values. Those can be added from any occurrence in the application, by a simple drag and drop action or over a context menu. Also those entries

can be removed from the filter by again using a drag and drop or a right-click functionality. The filter has also some additional functions, which are accessible through the toolbar above, like for instance the refresh and reset buttons, or the back and forward buttons (the filter keeps track of the last three selection changes). An actual filter entry consists of a label and a colored dot icon, which are used to identify the underlying element of the entry that is used for the filter restriction. As already mentioned, this element can be a category, which is identified by a green dot icon and the name of the category, a tag, identified by a blue dot icon and the name of the tag, or an attribute, identified by a yellow dot icon and by the attribute name and the according value. This specific representation of the different elements is not only used in the filter, but in the whole application in order to guarantee a high degree of recognition and avoid confusion.

6.7.2 Sile View

The sile view is the actual main view of the application, which is why it occupies the majority of the interface area. It displays all the siles that match the currently made selection in the filter. The overall look and feel of the sile view is related to the layout of standard file system browsers (as for instance the windows explorer) and therefore it offers different arrangement possibilities for the siles, ranging from a simple list layout to a tile view. The user has the possibility to drag desired siles from this view into the pocket situated right beneath. For more detailed information about the siles, the user can click on the desired sile in order to open a window with a representation of the sile's relations and content. The actual sile is represented by an icon, a label consisting of the name of the sile and some additional information, like for instance the creation date. The icon represents the type of sile, which on one hand can be a sile with no real file attached, in which case the sile is supplied with a general icon, or it can be a sile with a file attached to it, in which case the icon varies according to the data type of the file.

6.7.3 Detail View

The detail view, as the name suggests, shows all the detailed information for a certain sile, which means it shows all the assigned relations to categories, tags, attributes and slinks. The categories, tags and attributes are displayed in form of list-holding boxes, which can expand or collapse for a better overview. Every element can be dragged from the boxes to the filter

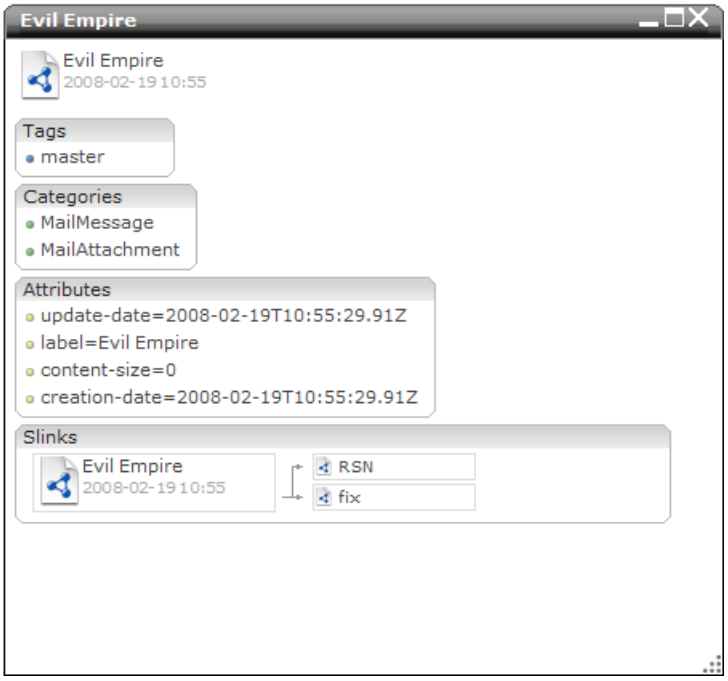


Figure 6.10: Detail View of the SemDAV Web-client

for additional content restrictions. The slinks are displayed in a sort of horizontal tree, on the left the incoming relations and on the right the outgoing ones. This tree can also be used as a navigational tools, as every slink can be clicked, causing the detail view to change to the selected slink. The detailed view as a whole is displayed in a resizable window, which also can be minimized into the pocket tray.

6.7.4 Pocket View

The pocket view has basically the same structure and functions as the sile view and the only difference between the two is that the pocket view doesn't react to filter changes. Its main purpose is to provide a deposition tray for the siles that the user wants to keep visible and independent from the filter for further uses.

6.7.5 Navigation View

The navigation view provides means to browse through all the elements that can be added to the filter. It consists of tho different views, which can be switched by using the tabs provided

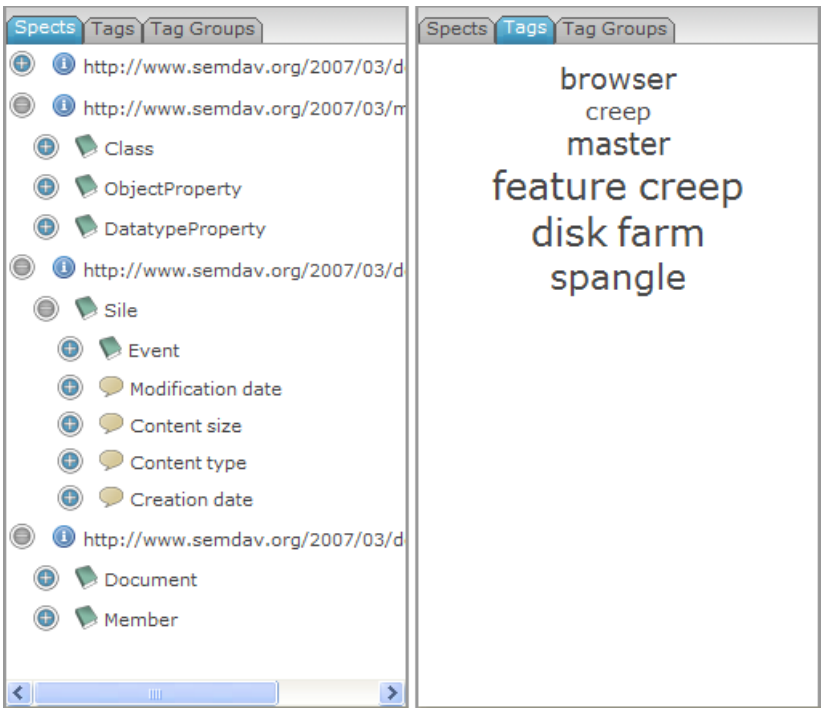


Figure 6.11: Navigation View of the SemDAV Web-client

at the top of the panel. The first tab is focused on the spects, which means here the user can find a list of all the available spects and he can access their containing elements by expanding the corresponding node. The second tab focuses on the tags and presents tag cloud of the available tags, where the entries vary in size according to their relative occurrences in the system. Every element of the navigation view can be added to filter by using drag and drop functionalities.

6.8 Summary

In this chapter we presented and documented the implementation of a web-client for the browsing of SemDAV repositories. We gave an overview over the fundamental architecture of the interface, starting with HTML-based front-end and following the transaction over the web server and the application framework, finishing with the SemDAV API and the communication with the SemDAV repositories. We introduced the underlying java and javascript framework, giving an insight into the main functionalities and classes, as well as into their structure and organization. In the end we took a look at the actual design of the interface, presenting the individual views as well as their functionalities and interaction between each other.

Chapter 7

Summary, Conclusions and Future Work

7.1 Summary

This work aimed to provide a knowledge base and a general overview over the disciplines of HCI and interaction design, with an emphasis on web-based visualization and the representation of ontology-based, unstructured data. Although the subjects of HCI and interaction design may seem rather theoretical, they provide important insights and guidelines concerning every form of interface design, and should therefore be taken into consideration by every designer. Furthermore, this thesis provides some common and established techniques for the actual process of interaction design, picturing a possible basic wireframe for an actual realization and implementation of any form of design.

The second part of the thesis deals with more practical and specific issues. It presents several case studies, consisting of well-chosen projects and applications, with an emphasis on general web design, the user interface functionalities and the representation of unstructured data. At last, the design and the implementation of a web user interface for SemDAV repositories is introduced, combining the insights made throughout the work with a practical realization.

7.2 Conclusion

There have been major changes in the way that people implement and refer to web-interfaces, especially since the appearance of the quickly spreading web 2.0 applications like for in-

stance Google Mail. As the web applications becomes more and more sophisticated due to the newly discovered technical possibilities (the introduction of AJAX), the designers tend to remedy the present flaws of web interfaces by mimicking the functions and the look and feel of desktop applications. However, whether web interfaces should reproduce desktop applications or not, is at this time a rather controversial topic. Although the web undoubtfully profits from the new range of functional possibilities, it is debatable whether those possibilities should be used for the reproducing of a system design that was never meant to be embedded into a web browser, or if the designer should be on the lookout for new faces of interface design.

The visualization of unstructured, semantic data is still a relatively young discipline, which makes it difficult to follow a single, predefined course of design. As the case studies showed, at this time there is no standardized procedure when it comes to the visualization of such content, which makes it difficult to find a highly usable solution due to the lack of experience in the field. However, at the same time this state makes the actual development of the ontology based visualization quite an interesting topic, as many different design approaches emerge almost simultaneously, all with their specific advantages and flaws.

7.3 Future Work

Due to the consistent growth of data that is in need of administration and organization, it is very supposable that the development of semantically enriched storage systems will significantly gain in importance in future development. Therefore it is to be assumed that sooner or later standardized procedures will emerge, not only considering the storage of such data, but also their querying, interaction and visualization. At the present time however, there seem to be several parallel development courses, each of them with their own advantages and focuses, and it is not possible to predict which one will prevail in the end.

Considering web applications and the overall movement toward linked and everywhere accessible data, it is obvious that the development steers towards native and enriched web applications. Web 2.0 and AJAX-based interfaces will definitely play a decisive role in the future of application landscapes, not only regarding the web but also the interaction with desktop systems. It is also very assumable that sooner or later the barrier between web

and desktop based applications will fall, making room for a new generation of platform independent and everywhere accessible applications. However, AJAX-based systems are still experiencing some teething problems, and although a high number of supporting frameworks and more powerful browsers emerge, there is probably still a long way to go until a complete assertion will occur.

List of Figures

Ich habe mich bemüht, sämtliche Inhaber der Bildrechte ausfindig zu machen und ihre Zustimmung zur Verwendung der Bilder in dieser Arbeit eingeholt. Sollte dennoch eine Urheberrechtsverletzung bekannt werden, ersuche ich um Meldung bei mir.

I gave my best to track down all the owners of the image copyrights and to get their approval for the usage of the images in this work. Nevertheless, should a copyright infringement emerge, please contact me.

List of Tables

Chapter 8

Bibliography

[Atk68] Shriffrin Atkinson. *Human memory: a proposed system and its control processes*. Academic Press, London, 1968.

[Bar88] Paul J. Barber. *Applied Cognitive Psychology: An Information Processing Approach*. Routledge, 1988.

[Ber07] Design issues-linked data. Website, May 2007. Available online at http://www.capt.org; visited on January 9th 2008.

[Cap02] L. F. Capretz. Implications of mbti in software engineering education. *SIGCSE Bull.*, 34(4):134–137, 2002.

[Cap07] Center for applications of psychological type. Website, November 2007. Available online at http://www.capt.org; visited on Novermber 10th 2007.

[Cox05] David Cox. A pragmatic hci approach: engagement by reinforcing perception with functional dsesign and programming. In *ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, pages 39–43, New York, NY, USA, 2005. ACM.

[Erg07] Ergonomics abstracts, November 2007. Available online at http://www.informaworld.com/openurl?genre=abstract_database&issn=1464-5084; visited on November 10th 2007.

[ES98] Charles Ess and Faye Sudweeks. Computer-mediated communication or culturally-mediated computing? *Electronic Journal of Communication/Revue Electronique de Communication*, 8, 1998.

[Flo84] Christiane Floyd. *A Systematic Look at Prototyping*, pages pp. 1–17. Springer Verlag, 1984.

[Gar08] Jesse James Garrett. Ajax: A new approach to web applications. Website, January 2008. Available online at http://www.adaptivepath.com/ideas/essays/archives/000385.php; visited on January 21th 2008.

[GNZ02] Peter Gregor, Alan F. Newell, and Mary Zajicek. Designing for dynamic diversity: interfaces for older people. In *Assets '02: Proceedings of the fifth international ACM conference on Assistive technologies*, pages 151–156, New York, NY, USA, 2002. ACM.

[Gro08a] Clustering and classification in grokker esm, 2008. Available online at http://www.groxis.com/service/grokker/resources.html; visited on September 19th 2008.

[Gro08b] Grokker master data sheet, 2008. Available online at http://www.groxis.com/service/grokker/resources.html; visited on September 19th 2008.

[Joh95] Bonnie John. Why goms? *interactions*, 2(4):80–89, 1995.

[JP96] David Benyon Simon Holland Tom Carey Jennifer Preece, Helen Sharp. *Human-Computer Interaction*. Addison-Wesley Longman, inc, 1996.

[JP02] Helen Sharp Jennifer Preece, Yvonne Rogers. *Interaction Design*. John Wiley and Sons, inc, 2002.

[KHL+07] Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia Giannopoulou. Ontology visualization methods—a survey. *ACM Comput. Surv.*, 39(4):10, 2007.

[Kur08] Ray Kurzweil. Moore's law, the fifth paradigm. Website, January 2008. Available online at http://www.kurzweilai.net; visited on January 8th 2008.

[LPP+06] Bongshin Lee, Cynthia S. Parr, Catherine Plaisant, Benjamin B. Bederson, Vladislav D. Veksler, Wayne D. Gray, and Christopher Kotfila. Treeplus: Interactive exploration of networks with enhanced tree layouts. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1414–1426, 2006.

[Mac92] I. S. MacKenize. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, pages 91–139, 1992.

[MC03] D. Scott McCrickard and C. M. Chewar. Attuning notification design to user goals and attention costs. *Commun. ACM*, 46(3):67–72, 2003.

[MVSC01] Ji-Ye Mao, Karel Vredenburg, Paul W. Smith, and Tom Carey. User-centered design methods in practice: a survey of the state of the art. In *CASCON '01: Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research*, page 12. IBM Press, 2001.

[Mye98] Quenk Hammer Myers, McCaulley. *MBTI Manual: A guide to the development and use of the Myers Briggs type indicator*. Consulting Psychologists Press, 3 edition, 1998.

[Nor88] Donald A. Norman. *The Psychology Of Everyday Things*. BASIC BOOKS; Reprint edition, 1988.

[Nor05] Donald A. Norman. Human-centered design considered harmful. *interactions*, 12(4):14–19, 2005.

[PHL77] Donald A. Norman Peter H. Lindsay. *Human Information Processing: Introduction to Psychology*. Academic Press Inc.,U.S., 1977.

[Pic00] Rosalind W. Picard. Perceptual user interfaces: affective perception. *Commun. ACM*, 43(3):50–51, 2000.

[SA07] J. Lehmann G. Kobilarov R. Cyganiak Z. Ives S. Auer, C. Bizer. Dbpedia: A nucleus for a web of open data. 2007.

[Saf06] Dan Saffer. *Designing for Interaction: Creating Smart Applications and Clever Devices (VOICES)*. Peachpit Press, Berkeley, CA, USA, 2006.

[Sch98] Ben Schneiderman. *Designing the User Interface*. Addison-Wesley Longman, inc, 3 edition, 1998.

[Sch06] Bernhard Schandl. Semdav: A file exchange protocol for the semantic desktop. In *2nd Semantic Desktop and Social Semantic Collaboration Workshop at the ISWC 2006*, Athens, GA, USA, 11 2006.

[SK06] Bernhard Schandl and Ross King. The semdav project: metadata management for unstructured content. In *CAMA '06: Proceedings of the 1st international workshop on Contextualized attention metadata: collecting, managing and exploiting of rich usage information*, pages 27–32, New York, NY, USA, 2006. ACM.

[Soe07] Mads Soegaard. Affordances. Website, 2007. This is an electronic document. Date of publication: November 6, 2007. Date retrieved: November 9, 2007. Date last modified: November 8, 2007. Available online at http://www.interaction-design.org/encyclopedia/affordances.html.

[Soe08] Mads Soegaard. Prototyping. Website, 2008. This is an electronic document. Date of publication: May 22, 2008. Date retrieved: May 24, 2008. Date last modified: May 23, 2008. Available online at http://www.interaction-design.org/encyclopedia/prototyping.html.

[Sta08] Tom Stafford. Size and selection times: Fitts's law. Website, January 2008. Available online at http://www.mindhacks.com/blog/2005/01/size_and_selection_t.html; visited on January 8th 2008.

[Str08] Strange systems: Using wireframes (revised). Website, January 2008. Available online at http://www.heise.de/tp/deutsch/inhalt/te/2860/1.html; visited on January 8th 2008.

[Tog07] Bruce Tognazzini. First principles of interaction design. Website, November 2007. Available online at http://www.asktog.com/basics/firstPrinciples.html; visited on Novermber 10th 2007.

[Tre08] Treeplus. Website, 2008. Available online at http://www.cs.umd.edu/hcil/treeplus/; visited on October 5th 2008.

[VCvdV06] Dhaval Vyas, Cristina M. Chisalita, and Gerrit C. van der Veer. Affordance in interaction. In *ECCE '06: Proceedings of the 13th Eurpoean conference on Cognitive ergonomics*, pages 92–99, New York, NY, USA, 2006. ACM.

[VG02] Chaomei Chen Vladimir Geroimenko. *Visualizing the Semantic Web. XML-based Internet and Information Visualization.* Springer Berlin, 3 edition, 2002.

[Vis07] Visual communication. Website, 2007. Available online at http://www.mikegreen.info/portfolio/websites/visCom/default.htm; visited on Novermber 10th 2007.

[ZF03] Murphy Zhao, Massey and Fang. Cultural dimensions of website design and content. *Prometheus, Vol. 21, No. 1, 2003*, 2003.

[ZTB06] Nicolas Zlatoff, Bruno Tellez, and Atilla Baskurt. Region-based perceptual grouping: a cooperative approach based on dempster-shafer theory. volume 6064, page 60640R. SPIE, 2006.

Appendix A

Zusammenfassung

Der semantische Ansatz bezüglich der Speicherung, Verwaltung und Extraktion von Daten ist ein vielversprechendes Konzept, das über die letzten Jahre konsequent an Bedeutung gewonnen hat. Die Visualisierung solcher semantsicher Datensysteme weist allerdings immer noch einge Hindernisse auf, sowohl für den Interface-designer als auch für den interagierenden Benutzer. Diese Hindernisse zu überwinden ist die Aufgabe des Interaktiondesigners, der durch den Einsatz von intuitiven und effizienten Design-komponenten dem unerfahrenen Benutzer den Einstieg in eine, auf semantischen Daten basierenden, Interaktionsumbegung erleichtern soll. Der Zweck dieser Arbeit ist es einen Überblick zu verschaffen über die Disziplinen des Human Computer Interaction sowie des Interaction Designs, mit der Absicht die grundlegenden menschlichen und technischen Prozesse zu vermitteln, die einen Einfluss auf das Entwerfen von Interaktion haben. Es werden Methoden und Techniken für den eigentlichen Design-prozess vorgestellt, ebenso wie Elemente und Komponenten die eine Rolle in visuellem Interface-design spielen, mit einem schwerpunkt auf die Darstellung semantischer Inhalte. Letztendlich wird ein web-basierendes, graphisches User-interface für SemDAV Repositories vorgestellt, dessen Entwurf auf den gewonnen Erkenntnissen beruht und dessen Implementierung sich nach den aktuellen Entwicklungen im Webdesign richtet.

Appendix B

Lebenslauf

Persönliche Daten

Bojan Milicevic
Pappenheimgasse 2/20, 1200 Wien

Telefon:	+43650/6441464
E-Mail:	bojan.milicevic@gmail.com
Geburtstag:	04.07.1981
Geburtsort:	Sarajewo, Bosnien u. Herzegowina
Familienstand:	ledig
Staatsbürgerschaft:	Österreich

Ausbildung

- **2000**
  Matura am Bundesrealgymnasium, 1210 Wien, Franklinstrasse 21
- **2000-2005**
  Bakk. Studium Wirtschaftsinformatik an der Universität Wien
- **2005-laufend**
  Mag. Studium Wirtschaftsinformatik an der Universität Wien

Beruflicher Werdegang

- **2003 - 2004**: Visionsnetwork GesmbH
  Software-Entwickler
- **2005 - laufend**: INNOFUTURE Software GmbH
  Teamleitung Entwicklung