User Interfaces for Personal Knowledge Management with Semantic Technologies

Zur Erlangung des akademischen Grades eines Doktors der Wirtschaftswissenschaften (Dr. rer. pol.) von der Fakultät für Wirtschaftswissenschaften des Karlsruher Instituts für Technologie (KIT) genehmigte Dissertation von

Dipl.-Psych. Heiko Haller

Tag der mündlichen Prüfung:	29. Juli 2011
Referent:	Prof. Dr. rer. nat. Rudi Studer
Korreferent:	Prof. Dr. sc. nat. Hartmut Wandke

Karlsruhe, 2011

"Design is not just what it looks like and feels like. Design is how it works."

Steve Jobs

Abstract

This thesis describes iMapping and QuiKey, two novel user interface concepts for dealing with structured information. iMapping is a visual knowledge mapping technique that combines the advantages of several existing approaches and scales up to very large maps. QuiKey is a text-based tool to author, browse and query graph-structured knowledge bases in a step-by-step manner. It can be seen as an interactive semantic command-line that offers an alternative way to access the same structured information with very high interaction efficiency. Both tools are primarily intended for the domain of personal knowledge management, although they could also be applied more generally.

Based on an analysis of the strengths and weaknesses of established visual knowledge mapping techniques, a set of requirements is derived that an ideal visual knowledge mapping system for personal knowledge management should address. These requirements form the basis for the design of iMapping. iMapping combines the core advantages of the mind-mapping, concept mapping and spatial hypertext techniques, which are incompatible in their original form.

By taking a zooming and nesting approach, iMapping allows for deep hierarchical structures, which are crucial for dealing with large amounts of information items. Linking these items in various ways – both formal and informal – allows users to build knowledge models at just the level of formalization that is beneficial for their specific needs.

QuiKey combines auto-completion techniques with an interactive query construction paradigm to offer text search and fine-grained access to graph structured knowledge bases in an interaction efficient and error avoiding way.

This thesis describes the design and implementation of iMapping and QuiKey as two combined pieces of software. They have been implemented in an open source Java application that is based on semantic desktop technologies such as the Conceptual Data Structures framework in order to support the full range from informal note taking over more structured graphical representations up to semantically formal knowledge models. Combined, iMapping and QuiKey provide semantic functionalities without restricting the user's modeling freedom that he has in other tools.

Several evaluation studies have shown that the design goals have been met: In a comparative user study, all participants preferred iMapping to the market leading mind-mapping application. Interaction efforts for QuiKey in both time and interaction units were below comparable tools. An additional long-term user study has provided evidence, that users succeed in utilizing both tools in the intended ways and yielded good ratings for overall user experience.

Acknowledgements

My thanks go to the following people (in rough chronological order of their influence):

- Felix Kugel for seeding my visions
- Dr. Max Völkel for sharing and reflecting my visions and bringing me to Karlsruhe
- Prof. Dr. Rudi Studer for creating a wonderful research environment
- Dr. Andreas Abecker for being the most supportive boss anyone can whish for
- My colleagues from FZI, AIFB for making this research environment come alive
- The European Commission for partially funding this work through the research projects Nepomuk and Mature
- My collegues from the Nepomuk project, especially those from KTH for hosting me, building prototypes and having stimulating discussions
- The DenkWerkzeug community for advancing the topic of PKM in a refreshingly unacademic way
- Tanja Burkhardt for loving and supporting me
- Henning Sperr and Florian Simon for their long lasting commitment in actually implementing iMapping and QuiKey
- Dr. Valentin Zacharias for constructive feedback
- Prof. Dr. Klaus Müller, Dr. Reik Winkel and Dr. Denny Vrandečić for convincing me to carry on
- Prof. Dr. Hartmut Wandke with Michael Sengpiel and Knut Polkehn for discussions and advice
- All the participants of the user studies
- All users that submitted their iMaps
- Steve Gunn and Sunil Patel for creating this thesis template in LaTeX
- Markus Krötzsch for advice in the domain of logics
- Dr. Holger Leven, Dr. Olaf Grebner and Dr. Timon Schroeter for proof-reading and typographic advice
- Dr. Felix Schulze for providing perfect working conditions in his lovely appartment
- Again: Max, Andreas, Florian, Henning and Markus for being incredibly patient with me

Contents

All URLs referred to in this thesis have been checked in March 2011.

Chapter 1

Introduction

1.1 Motivations for Personal Knowledge Management

> "The most important contribution of management in the 20th century was to increase manual worker productivity fifty-fold. The most important contribution of management in the 21st century will be to increase knowledge worker productivity – hopefully by the same percentage. [...] The methods, however, are totally different from those that increased the productivity of manual workers."
>
> Peter F. Drucker (1999) [1]

How can knowledge worker productivity be increased?

Which perspective does the use of the concept *knowledge* imply?

Most of the numerous distinctions between the terms *information* and *knowledge* define knowledge as something cognitive and subjective that resides in people's minds as opposed to information as something more universal (Zins, 2007) that can be formalized, processed and stored by IT (*information* technology). [2] Consequently, personal information management (PIM) mainly deals with managing pre-existing information like documents, messages, contacts, personal tasks, events and the like (Jones and Teevan,

[1] Although often quoted like this, this is not the wording used in the actual article by Drucker (1999) but rather the rephrased online abstract of it. It is, however, more concise and does not alter the meaning of the original abstract. It can be found online at http://cmr.berkeley.edu/search/articleDetail.aspx?article=4872.

[2] For a comprehensive overview of the various distinctions between data, information, knowledge and wisdom see http://en.wikipedia.org/wiki/DIKW

2007). While such pre-existing information is external to the user's mind, personal *knowledge* management (PKM) takes the perspective of managing a user's internal knowledge like capturing his ideas and structuring his thoughts.

The seeming paradox can be resolved as follows: *We cannot manage knowledge itself, but we can manage knowledge cues.* Anything can act as a knowledge cue: a knot in a handkerchief, a symbol, a keyword, a scribbled note, a checklist or a mind-map. Anything that reminds a user of what it signifies. Of course, as Jones (2010) notes, any external knowledge cue can be seen as an *information* item. However, from a PKM perspective, it is less important if these items are intelligible by other people. What matters is, that the cue actually triggers or at least helps the reconstruction of the original thoughts in the user's mind.

It could be asked, when knowledge already resides in people's mind by definition, then why is PKM needed at all, since human long-term memory seems to be virtually unlimited (Anderson, 2005). While long-term memory seems to be effectively unlimited in capacity, there are still some significant limitations that burden the knowledge worker (of which only the following two shall be mentioned here):

- We "forget" things although they are still engraved in our memory – we simply cannot access them. When something is "remembered", it is in fact being reconstructed from inter-related fragments (Anderson, 2005). This is why in order to facilitate later recall, it is crucial to relate the learning matter to the learner's prior knowledge (Reigeluth, 1983).
- Short-term memory is very limited. In fact, a human mind cannot have more than around four to seven items consciously present at the same time (Miller, 1956; Cowan, 2001). When dealing with complex subjects, this is a problem.

To be able to process higher amounts of items and grasp complex topics, the mind uses techniques of chunking and abstraction (Anderson, 2005). Also, literature on complex problem solving (Dörner, 2003; Vester, 2002) identifies as a core difficulty to understand the interrelations and interactions between things.

These cognitive shortcomings can be partly relieved by the use of external knowledge media that have been given many names like "memex" (Allegedly either from *mem*ory *ex*tension or *mem*ory ind*ex*, Bush 1945), "augmentation to human intellect" (Engelbart,

1962), "cognitive tools" (Lajoie and Derry, 1993; Kommers et al., 1992) "denkwerkzeug" [3] or "extra-cortical organizers of thought" [4]. Many such cognitive tools already exist.

One unpleasant side effect however can be encountered especially with the more sophisticated ones, and it is the main reason why many people still prefer paper and pencil to support their thinking: Apart from helping the user in certain aspects of his knowledge work, the tools themselves also consume some of the precious limited cognitive capacities of their user. In fact, recent studies suggest that reducing such *cognitive overhead* (Conklin, 1987) must be of central concern to the PKM researcher.

A problem that comes with the use of such knowledge media is that of course any cognitive tool used to support a thinking process also has an influence on this process. Knowledge tools should therefore be scrutinized in respect to *how they shape* the knowledge processes they support.

One category of information tools that promises great improvements in knowledge work of many kinds consists of those tools based on semantic technologies. Automatic reasoning and data integration techniques promise to improve findability, interoperability and, in general, automated processing of information and knowledge items.

David Huynh's video [5] about *parallax* (a system discussed in Section 5.4) impressively demonstrates how semantically enriched content enables more convenient information retrieval experiences. Krötzsch et al. (2007) describe the presumed benefits of basing Wikipedia on semantic technology, and Haller (2010b) outlines a future scenario of how knowledge management and electronic science could look like, if semantic technologies should be more advanced and more widely used than today.

When semantically formalized knowledge structures are used, content is typically fine grained and highly structured. Such content structures are typically more complex than plain text or classical hypertext structures. Shipman and Marshall (1999a) warn that such semantic formality should be "considered harmful", because it often forces the user to make formal decisions when these are premature, inconvenient or simply not possible. Even with the relatively simple structures in classical hypermedia, with interlinked information objects on the granularity level of whole pages or documents, hypertext research has shown that users may get "lost in hyperspace" when browsing without additional navigational help (Edwards and Hardman, 1999).

[3] "Denkwerkzeug" is German for *think tool* and a community has formed around this concept: http://denkwerkzeug.org/

[4] The term "extra-cortical organizers of thought" is by several authors attributed to have been coined by Lev Vygotsky (Vygotsky and Cole, 1978). While this is plausible and Vygotsky actually writes about the role of tools for thought in this book, the actual phrase could not be found there.

[5] http://vimeo.com/1513562

Knowledge management systems in general, and especially those that rely on highly structured information and meta data being entered and maintained by the users, often fail because users do not make this additional effort if it does not yield an immediate benefit (Völkel, 2010). This may be one of the reasons why semantic technologies have not yet found widespread adoption despite the benefits they promise. For these technologies to find more widespread use, it is crucial that they are very easy to use and do not constrain users in their daily work. This immediate benefit is more likely to be experienced when users manage their own everyday knowledge resources like personal notes, files, bookmarks etc. e. g., in the setting of a semantic desktop environment (c. f. Sauermann et al., 2009). In the end, it should be up to the user to decide, how much effort he wants to put into formalizing his content.

In knowledge-intensive activities, it is even more crucial than otherwise, to unburden the user of all cognitive overhead in order to leave as much of the user's limited working memory to the actual task at hand. Cognitive overhead is that part of a user's cognitive load that is not directly related to the intended action, but rather to dealing with side-issues or the software as such.

All in all, without claiming completeness, this leads to the following list of roles or requirements for PKM systems (which is, like other parts of this chapter, also published by Haller, 2010a):

1. A PKM system should act as an aide-mémoire, supporting the reconstruction of its user's prior knowledge.
2. It should be able to represent the interconnections between knowledge items.
3. It should support the externalization of knowledge in an easy and flexible manner.
4. It should facilitate abstraction and clustering.
5. It should constrain its user's way of thinking either to the least possible extent or only in carefully considered ways.
6. In order to leave as much of the knowledge worker's cognitive capacity to the actual task at hand, cognitive tools specifically, like any software in general, should be diligently designed to avoid cognitive overhead.

1.2 Motivations for Visual Knowledge Tools

The human sense of orientation has developed over millions of years. It is highly optimized for orientation in a three dimensional world and on large plains and not for finding

a way through complex hypertexts or abstract formal structures. Using graphical environments for structuring externalized knowledge enables the users to use their highly efficient sense of spatial orientation on their *personal knowledge and information space*.

Allowing users to spatially arrange information items may enhance the link between their mental and external models because it enables the use of diagrammatic depictions whose obvious structure corresponds more closely to the structure of the content. This helps the user to intuitively grasp an overview of the subject matter. Unlike text, diagrammatic knowledge representations carry a structural analogy to the content they represent. In other words: A diagram's structure looks similar to the structure it represents. A flowchart e. g., *depicts the structure* of a process. A text does not – which is why it takes a longer way in the user's mind until it can be related to the user's mental model (Schnotz and Bannert, 2003).

Visual mapping techniques provide easy ways to intuitively structure fine-grained information objects. iMapping was developed as a new visual mapping approach that tries to unite the strengths of established mapping techniques. It combines these strengths with modern IT approaches like deep zooming and semantic technologies.

1.3 Overview

This thesis begins with an overview of related work that forms the basis of the concepts developed here (Chapter 2). Based on this existing work, particularly on the analysis of the strengths and weaknesses of established visual knowledge mapping techniques, a set of requirements for visual PKM tools is specified (Chapter 3).

The main contribution of this thesis is the design of iMapping, a novel visual knowledge mapping technique that is based on these requirements. iMapping combines the advantages of several existing visual techniques that are incompatible in their original form. Due to its nesting and zooming approach, it allows to deal with large amounts of interrelated information items. Chapter 4 illuminates the essential design decisions taken on the way to create both the iMapping *technique* as a novel approach to visual knowledge mapping and the current *implementation* of an iMapping tool, that implements the core concepts of the iMapping technique.

This visual approach is complemented by QuiKey, a light-weight interactive command line tool that provides text search and fine grained, map independent access to the same structured information. In order to provide high interaction efficiency and avoid errors, QuiKey combines advanced auto-completion techniques with a novel interactive query construction paradigm. Chapter 5 describes the motivation, design and implementation

of QuiKey, which is, again, both a novel interaction concept and a first implemented tool.

The resulting PKM application supports the full range from informal note taking over more structured graphical representations up to semantically formal knowledge models. It provides semantic functionalities without restricting the user's modeling freedom that he has in other tools without semantic technology. Chapter 6 elaborates on several evaluation studies carried out in order to prove the usefulness of the interaction concepts of iMapping and QuiKey combined in this novel PKM application.

Finally, Chapter 7 surveys the fulfillment of the requirements, summarizes results and contributions and gives an outlook on possible future work.

Chapter 2

Foundations

This chapter describes related work that iMapping is based on: It shortly credits pioneering work in the field of hypertext and PKM (Section 2.1) and categorizes the focus of this thesis (Section 2.2). It then presents the three basic visual knowledge mapping approaches (Section 2.3) and a set of design principles (Section 2.6), which form the basis of the requirements described in Chapter 3 and consequently of the iMapping technique itself. Section 2.4 gives a short overview over the history of zooming user interfaces and Section 2.5 introduces semantic desktops – both are technical bases of the iMapping tool. Other related work that has not influenced the design of iMapping but is comparable in some aspect is presented later, in Section 4.4.

2.1 Pioneering Work in Hypertext and PKM

Before the inception of the World Wide Web in 1990 by Tim Berners-Lee et al. (1999), hypertext systems were conceived much more from a personal perspective: In his famous article *As we may think*, Bush (1945) describes the *memex* as a mechanical machine based on microfilm technology, which should be capable of storing, interlinking and reproducing visual information. [1] Although the memex should also be capable of copying contents for the purpose of sharing, it was mainly a personal information storage device. The article also hypothesizes on how such technology might benefit human cognitive abilities:

> "Presumably man's spirit should be elevated if he can better review his shady past and analyze more completely and objectively his present problems".

The memex was never built as an actual device, and, as discussed by Lunn et al. (2010), its base technology had even been patented before by Daniel Goldberg (Buckland, 2006).

[1] For an animated description of the memex see `http://www.youtube.com/watch?v=c539cK58ees`.

However it was Bush's article that sparked a development that led to other groundbreaking work like the more detailed visions of Ted Nelson and Douglas Engelbart (1962).

Ted Nelson, who coined the terms *hypertext* and *hypermedia* in 1963 (Nelson, 1999), described the idea of hypertext in a way in which it is still barely found nowadays: Fine-grained content structures, which could be interlinked bidirectionally, i. e., every link could be seen and followed both directions. Nelson (1995) also coined and strongly advocated the concept of *transclusion*, where contents are included in different contexts by reference as opposed to being duplicated.[2] Early hypertext systems were Ted Nelson's Xanadu[3], Douglas Engelbart's NLS (standing for *oNLine System*, later dubbed *Augement*)[4]. The first widely adopted hypermedia system was Apple's HyperCard, created by Bill Atkinson (Goodman, 1998).

Much has been written about the origins of hypermedia systems and their influence on nowadays information environments, shared or personal. A broad overview is given by Davies (2011).

2.2 Knowledge Visualization

In order to clarify the scope of this thesis in the broad context of visualization as a topic, it will be shortly categorized within a framework proposed by Eppler and Burkhard (2005), detailed by Burkhard (2005). Firstly, *knowledge visualization* is distinguished from *information visualization*: Consistent with the above distinction between information and knowledge (see Section 1.1), information visualization is usually concerned with visually presenting large amounts of pre-existing structured data. There is a wealth of approaches and algorithms for information visualization corresponding to different data structures and properties – for an overviews are given by Andrews (2010); Bederson and Shneiderman (2003); Card et al. (1999); Spence (2007).

Information visualization however, is not the topic of this thesis. *Knowledge visualization*, in contrast, constitutes techniques that do not primarily visualize pre-existing structured data, but knowledge, which a priori resides in people's minds. A detailed distinction between knowledge and information visualization is made by Burkhard (2005). However, this definition of knowledge visualization needs to be slightly widened here: He writes "Knowledge Visualization examines the use of visual representations to improve the transfer and creation of knowledge *between at least two persons*" (p. 242, emphasis added). However, especially in the growing field of personal knowledge management,

[2] http://en.wikipedia.org/wiki/Transclusion
[3] http://xanadu.com/, http://en.wikipedia.org/wiki/Project_Xanadu
[4] http://www.dougengelbart.org/about/augment.html

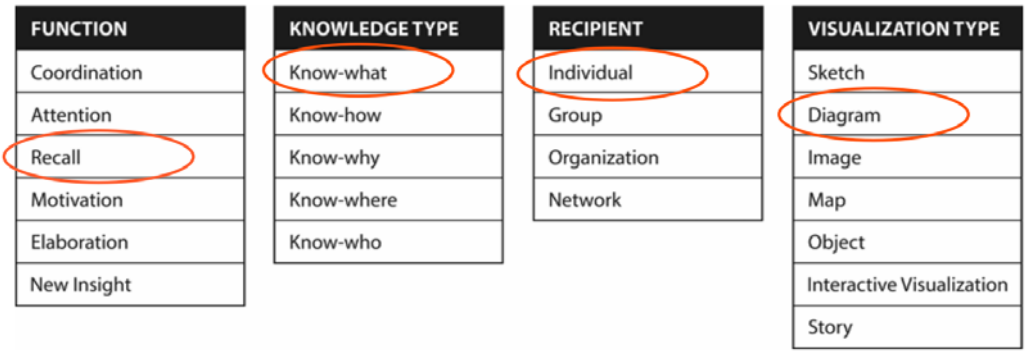

FIGURE 2.1: The Knowledge Visualization Framework of Burkhard (2005, p. 245). Marked are the categories that are primarily targeted in this thesis.

the intended *recipient* of a visual knowledge artifact is typically the author himself, who made a sketch to support his current cognitive process or who created a diagram to facilitate the recall of his own knowledge. As this appears to be a valid application of knowledge visualization techniques, it is proposed to drop this constraint. In this thesis, the personal use case is expressly included.

From the many different forms of knowledge visualization, the target here is the general use of diagrammatic knowledge maps, created by an individual modeling his knowledge in visual and more or less structured ways in the tradition of mind-mapping, concept mapping, knowledge mapping and the like (described below). As depicted in Figure 2.1, this focus can be categorized in the above mentioned Knowledge Visualization Framework as follows: The *function* (purpose) is primarily the structured externalization of knowledge in order to later *recall* or better re-construct the knowledge from the visual artifact. However, also the aspects of *elaboration* and *new insight* are touched by this thesis, as well as an important aspect that seems to be missing in the framework: serving as a thinking aid or cognitive substrate that unburdens the knowledge worker's working memory. The *knowledge type* used in such artifacts would typically be *know-what*, i. e. factual knowledge or keywords and concepts. As such knowledge maps can be rather flexible and generic (depending on the actual technique), they could also be used for other kinds of knowledge. Although the *recipient* of the artifact could be any number of people in principle, the use case addressed here is an *individual* user dealing with his knowledge artifacts, which are targeted either at himself or at others who share enough common context and prior knowledge to interpret the artifact. Many of the established techniques that are in this thesis referred to as *visual knowledge mapping* carry the word "map" in their name (e. g., mind-maps, concept maps . . .). However, in this framework, these techniques' *Visualization Type* is that of *diagrams*, where typically discrete abstract items are depicted with their interrelations. In the framework, the category *map* refers to more geographically oriented techniques, which are only touched marginally by this thesis.

This focus implies two fundamental characteristics:

1 Structures are typically not homogenous: Instead of being instances of a common schema, items are usually of varying or unspecified types and their interrelations are often also vague.
2 The actual items often do not carry all relevant information themselves but are rather used as knowledge cues in the sense of mnemonic devices: Because the actual knowledge in its full richness is only stored in the user's memory, and can not simply be externalized, it is in many cases sufficient to externally store and provide such cues to re-activate the original knowledge in the user's mind.

This thesis deals with requirements for diagrammatic knowledge mapping techniques as detailed above. It is about how to enable users to map and edit their own structured knowledge representations in a cognitively adequate way. Though it may be applied more generally, this thesis focuses on use cases, where a mapping tool is used to manage a potentially large and heterogeneous collection of information items, which can serve as a large personal knowledge repository.

2.3 Visual Knowledge Mapping Techniques

From the 1970s on, a number of visual mapping techniques has evolved, some of which have found wide-spread use and have proven their usefulness e. g., as learning aids in numerous studies. According to their basic topology, most of them can be related to the following three fundamentally different primary approaches:

2.3.1 Mind-Maps

The term "mind-map" is often wrongly used for various kinds of informal graphic knowledge artifacts. Here however, it will only be used as originally introduced and actually trademarked by Tony Buzan (Buzan and Buzan, 1996). A mind-map always has one central topic in the middle, extending from which labeled branches and sub-branches are drawn (see Figure 2.2). Instead of distinct nodes and links, mind-maps only have labeled branches. The relations between these branches cannot be specified. Structurally, a mind-map is a connected directed acyclic graph with hierarchy as its only type of relation. A mind-map is a simple tree of labeled items, which provide an easy-to-understand hierarchical structure. Mind-mapping is an established technique for brainstorming, outlining, note taking and clustering. Mind-maps are however not suitable for relational structures between items because they are constrained to the hierarchical model.

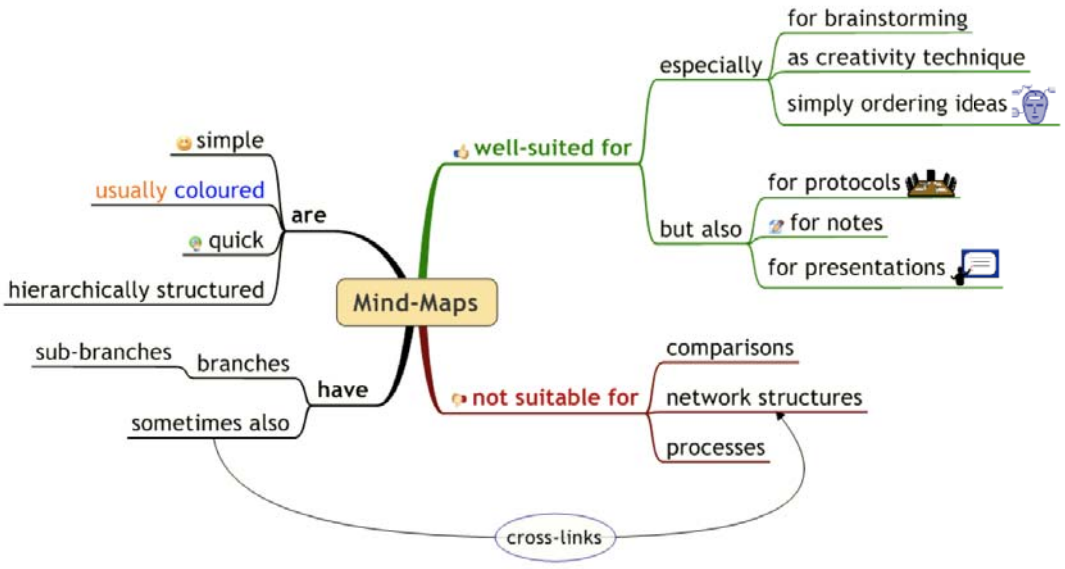

FIGURE 2.2: Example mind-map about mind-maps

Some of the more widely used software tools for mind-mapping are FreeMind [5], a free open-source mind-mapping tool with an active developer community, MindManager [6], the feature-rich market leading commercial mind-mapping software, Nova-Mind [7], another flexible commercial tool and MindMeister [8], a web-based mind-mapping service that features real-time collaborative mapping. MindMeister is also commercial, but the first few maps are free.

2.3.2 Concept Maps

Concept maps, as introduced by Joseph Novak and Gowin (1984), emphasize the interrelations of items. Concept maps consist of labeled nodes and labeled edges linking all nodes to a connected directed graph. Concept maps have proven useful to represent complex subject matters especially with a focus on the interrelations of items. The basic *node and link* structure of a connected directed labeled graph also forms the basis of many other modeling approaches. Many of them use the same basic structure but with more formal types of nodes and links, e. g.: entity-relationship diagrams, semantic networks, and what has been called "knowledge mapping" (O'Donnell et al., 2002) and later "TCU-NLM" (Texas Christian University Node-Link Mapping) (Dansereau, 2005). The helpful effects of this general kind of node-and-link mapping for many application areas have been found in numerous studies, many of them summarized by O'Donnell et al.

[5] http://freemind.sourceforge.net/
[6] http://mindjet.com/
[7] http://www.novamind.com/
[8] http://www.mindmeister.com/

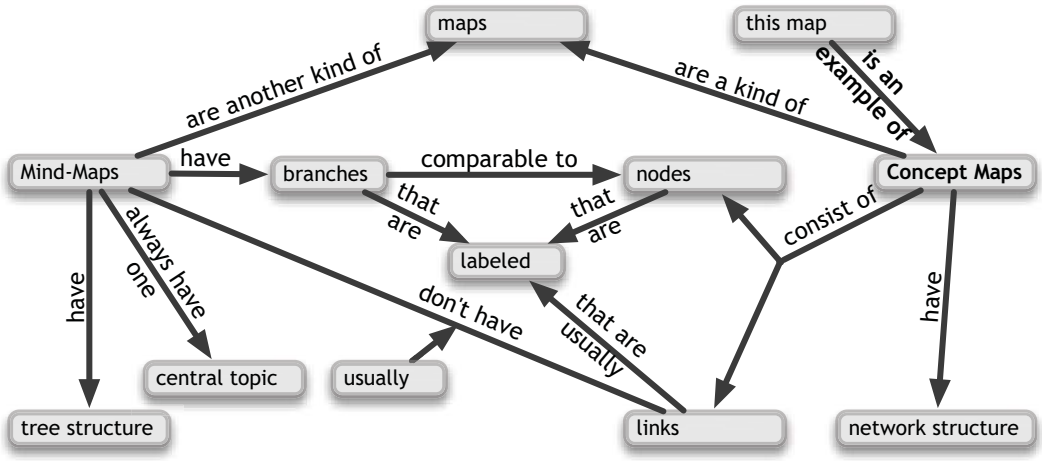

FIGURE 2.3: Example concept map about mind-maps and concept maps

(2002). However, these more general graphs are not as easy to handle as mind-maps because explicitly specifying all the relations is more laborious, which makes them less suitable for simple tasks like note-taking or brainstorming.

Prominent concept mapping tools are the following: cMapTools [9], a closed source but free software developed by the Florida Institute for Human & Machine Cognition, Inspiration [10] and SMART Ideas [11], both commercial dedicated concept mapping tools. SmartDraw [12] and ConceptDraw [13] are versatile graph drawing software that provide functionality for node and link diagrams – and so do many other graphics or presentation tools, e. g., OmniGraffle [14], CorelDraw [15], PowerPoint [16] and many others ...

2.3.3 Spatial Hypertext

A concept that evolved somewhat later out of hypertext research is called "spatial hypertext". The spatial hypertext paradigm in its pure form as described by Marshall and Shipman (1993) expressly abandons the concept of explicitly interrelating objects. Instead, it uses spatial positioning as the basic structure. Because, as Kolb (2001, p. 3) argues,

> "It would seem that linguistically labeled typed links and paths-patterns could carry more kinds of connections than spatial arrangements, if link types could

[9] http://cmap.ihmc.us/
[10] http://www.inspiration.com
[11] http://smarttech.com/smartideas
[12] http://www.smartdraw.com/
[13] http://www.conceptdraw.com/
[14] http://www.omnigroup.com/products/omnigraffle/
[15] http://www.corel.com/
[16] http://office.microsoft.com/en-us/powerpoint/

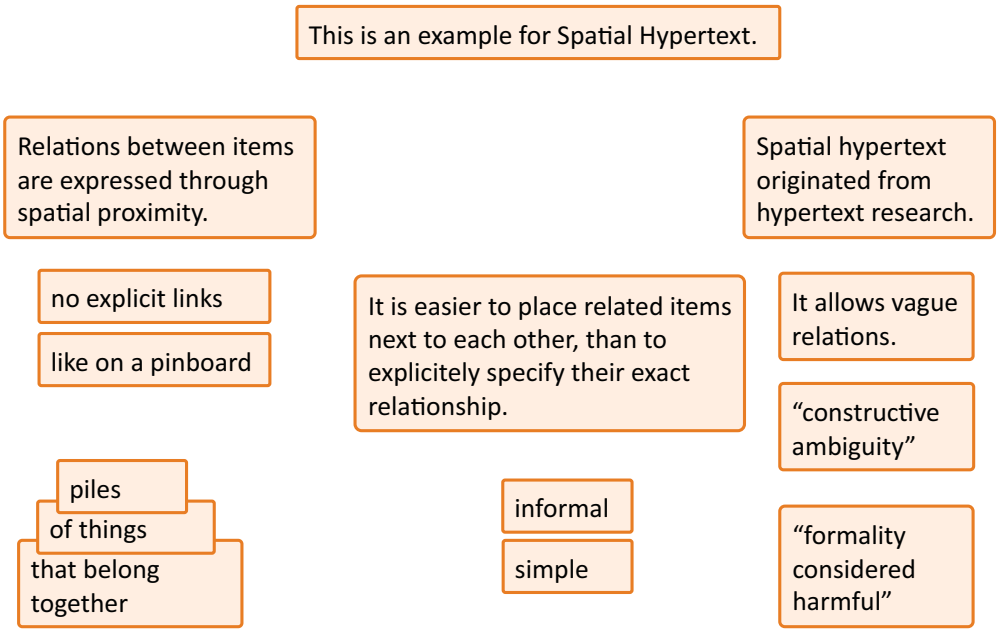

FIGURE 2.4: Example spatial hypertext about spatial hypertexts.

be conveniently indicated. This may be so, but spatial arrangements can accommodate n-ary relations and gradations of meaning relations that are difficult to put into link type labels."

A spatial hypertext is a set of text nodes that are not explicitly connected but implicitly related through their spatial layout, e. g., through closeness and adjacency – similar to a pin-board. As such, a self-contained hypertext can be seen from an overview perspective, by spatially arranging single pages. To fuzzily relate two items, they are simply placed near to each other, but maybe not quite as near as to a third object. This allows for so-called “constructive ambiguity” (Shipman and Marshall, 1999b) and is an intuitive way to deal with vague relations and orders.

The two most prominent software tools for spatial hypertext, both described in Section 4.4 are the Visual Knowledge Builder (VKB) [17] by Frank Shipman et al. (2001), a free research prototype and Tinderbox [18] by Mark Bernstein (2003), a widely used commercial tool.

Although the original puristic form of spatial hypertext abstains from the use of links and does not include the concept of nesting, the approach can easily be extended to include both. In fact, both of the above mentioned tools do that.

[17] http://www.csdl.tamu.edu/vkb/
[18] http://www.eastgate.com/Tinderbox/

2.3.4 Conclusion

All three basic approaches as well as those derived from them have their pros and cons. Unfortunately, none of the approaches combines the main advantages, which would be desirable since many knowledge artifacts go through various differently structured stages: In the life cycle of a project, e. g., a work package may start as a set of keywords scribbled on a scrap of paper, go through several diagrammatic sketches, an outline for a proposal, a detailed hierarchical work plan (e. g., a Gantt chart), an architecture diagram and later an outline of a report. Since no single tool supports these various structures, the contents need do be re-entered many times and there are typically no links between the different representations of one and the same thing across tools and media.

Table 2.1: Pros and cons of the three basic mapping approaches

	Mind-Maps	Concept Maps	Spatial Hypertext
Structural analogy to content	✓	✓	✓
Simple hierarchical topology	✓	–	–
Representation of interrelations	–	✓	–
Constructive ambiguity	–	–	✓
Scalability	–	–	–

An overview over these and related approaches, their cognitive psychological foundations, studies about their effectiveness along with an evaluation of corresponding software tools has been given by Haller (2003). It contains a comparison of these approaches and 13 software programs for certain typical knowledge mapping tasks.

2.4 Zooming User Interfaces (ZUI)

The work on zooming user interfaces dates back to 1978, when William Donelson (1978) from MIT described SDMS (“Spatial Data Management System"), a sophisticated hardware setup of a multi-media room based on then state-of-the-art technologies like analog videodisk projection with a frame buffer for still images. SDMS did not yet feature animated transitions between views, but the data space consisted of a set of hierarchically structured visual views, as depicted in Figure 2.5. Consistent with more recent findings (Bederson, 1999), Donelson already notes in this early research that with the lack of smooth transitions, it was hard for users to maintain orientation in the information space.

The issue of smoothly animated zooming in information environments was addressed by the development of zooming software frameworks: Pad (Perlin and Fox, 1993) as well as

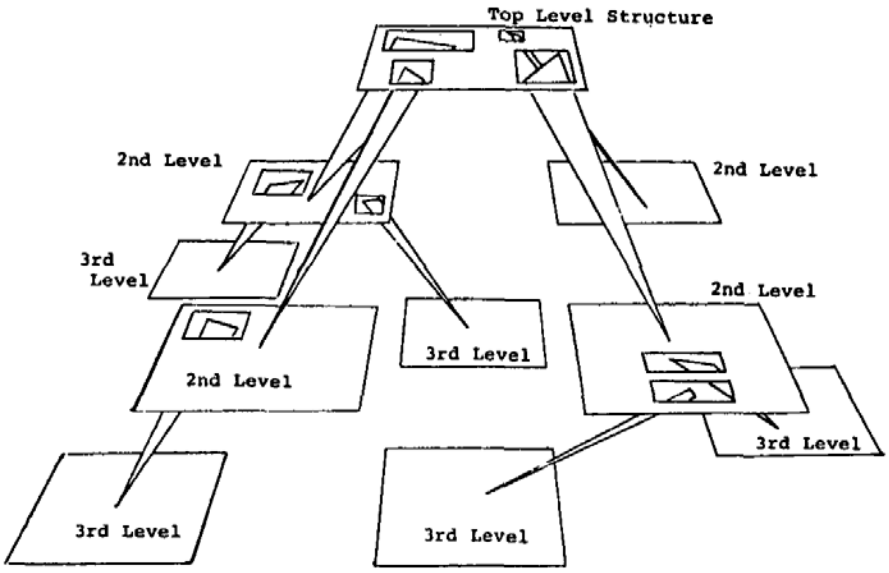

FIGURE 2.5: Tree-Model of the SDMS Data Base: Visual content organized in a hierarchical structure with higher levels having hyperlinks ("ports") to lower levels. (Figure taken from (Donelson, 1978, p. 208))

its successor Pad++ [19] by Ben Bederson et al. (1996), were the seminal developments in the area of zooming UIs. They have been used in various applications and also as a Web browser capable of showing the viewed web pages and their link-structure from a bird eye's view. In a study where participants had to perform browsing tasks in order to answer some questions, subjects using Pad++ were 23% faster than those using Netscape (Bederson et al., 2002). This showed that large zoomable information surfaces can act beneficially as hypertext front-ends. The work on Pad++ led to the development of the Java-based *jazz* framework (Bederson et al., 2000) and later *Piccolo* (Bederson et al., 2004), now community maintained as *Piccolo2D* [20], a scene-graph toolkit in Java and .Net that supports the development of 2D structured graphics programs, in general, and, in particular, zoomable user interfaces. Piccolo2D is the UI framework used in the current iMapping implementation and many other tools. [21]

Other related work in the area of ZUI is discussed in Section 4.4 (iMapping Related Work).

Cockburn et al. (2008) give a broad overview over different techniques to visually integrate context and detail information and summarize numerous evaluation studies comparing such approaches like overview windows, fish-eye views and zooming techniques. Their differences, advantages and implications are discussed in more detail in the requirements Chapter in Section 3.2.3. Why iMapping is actually based on zooming, is explained in Section 4.2.3 in the design chapter.

[19] http://www.cs.umd.edu/hcil/pad++/
[20] http://piccolo2d.org/
[21] http://www.piccolo2d.org/applications/index.html

2.5 Semantic Desktop

The foundations and benefits of semantic technologies in general have been well described in the literature (Berners-Lee et al., 2001; Davies et al., 2006a; Staab and Studer, 2009; Krötzsch et al., 2007; Haller, 2010b). In their essence, they can be used to represent information in a way formal enough to enable automated processing like logical reasoning and integration of formal data from heterogeneous sources. In several so-called **Semantic Desktop** (Sauermann et al., 2009) projects, such semantic technologies have been applied to personal information environments. They allow the interlinking of existing desktop objects like contacts, e-mails, events, tasks, files and notes, and to do that in a comprehensive way that they can be accessed and acted upon with semantic technologies. A semantic desktop system can, e. g., answer queries like *a list of all e-mails received from participants of the last project meeting.*

2.5.1 Semantic Desktop Projects

Founded in 2000, **DeepaMehta** [22] (Richter et al., 2005) was one of the first semantic desktop systems and it is still actively being developed by a small group. On a graphical canvas, DeepaMehta lets the user freely specify semantic relations between typed information items on a topic maps basis. It provides a graph-based UI in a thin client. Once an item (or relation) has been specified (in a topic map), DeepaMehta keeps it in a background repository on the server, independent of whether it is still part of an actual topic map. This separation between the structural model and visual model is also adopted in iMapping, because it allows multiple (visual) instances of an item to be used in different contexts or locations – much like hard links in a Unix file system.

More focused on ontology-based functionalities and based on actual semantic web standards, the IRIS project (Cheyer et al., 2005), later open sourced as OpenIris [23] as well as **Haystack** [24] (Huynh et al., 2003) have been large US research projects in the field of semantic desktops.

The European *Social Semantic Desktop* project **Nepomuk** [25] yielded a whole stack of Semantic Desktop Ontologies [26] and a heterogeneous pool of tools and reference implementations. The first iMapping Prototypes and also most of the current implementation have been developed as part of the Nepomuk project. For more information on semantic

[22] http://deepamehta.de/
[23] http://en.wikipedia.org/wiki/OpenIRIS
[24] http://groups.csail.mit.edu/haystack/, http://simile.mit.edu/hayloft/, http://en.wikipedia.org/wiki/Haystack_(MIT_project)
[25] http://nepomuk.semanticdesktop.org
[26] http://www.semanticdesktop.org/ontologies/

desktop systems in general, see Sauermann et al. (2009). How iMapping and QuiKey relate to the Nepomuk context is described in the project deliverables D1.2 (Völkel et al., 2008) and D1.3 (Haller et al., 2009).

2.5.2 Conceptual Data Structures (CDS)

Another outcome of the Nepomuk Semantic Desktop project is CDS: The *Conceptual Data Structures* framework, described by Völkel and Haller (2009) and refined by Völkel (2010), serves several purposes:

1. It is a flexible semantic data model providing a set of crucial structural primitives that can be extended by the user. CDS is a lightweight top-level ontology, providing relations that naturally occur in common knowledge artifacts (c. f. Figure 2.6). It is designed to bridge the gap between unstructured content like informal notes and formal semantics like ontologies by allowing the use of vague semantics and by subsuming arbitrary relation types under more general ones. By that, it is suitable for representing knowledge in various degrees of formalization in a uniform fashion, allowing gradual elaboration.
2. The java-based CDS-API serves as a back-end for both iMapping and QuiKey, providing basic querying and reasoning functionality.
3. CDS supports interoperability, as the data can be directly accessed and modified by CDS based tools like QuiKey or the *Hypertext Knowledge Workbench* (HKW) [27] or can be exported to other formats like plain RDF through generic CDS converters.

CDS offers several kinds of different items:

– *NameItems:* consisting of a (typically short) unique string that can be used to identify it (like, e. g., the name of a wiki page or a file in a folder).

– *ContentItems:* An item that can consist of any content with a MIME type. Typically, plain text or STIF [28], a type of simplified HTML are used.

– *Relations:* Relation types that also have unique names and can occur in statements. Each relation has a defined inverse relation and can be have subrelations and superrelations.

– *Statements:* A triple – linking two items through a relation.

[27] http://semanticweb.org/wiki/HKW
[28] http://semanticweb.org/wiki/Structured_Text_Interchange_Format

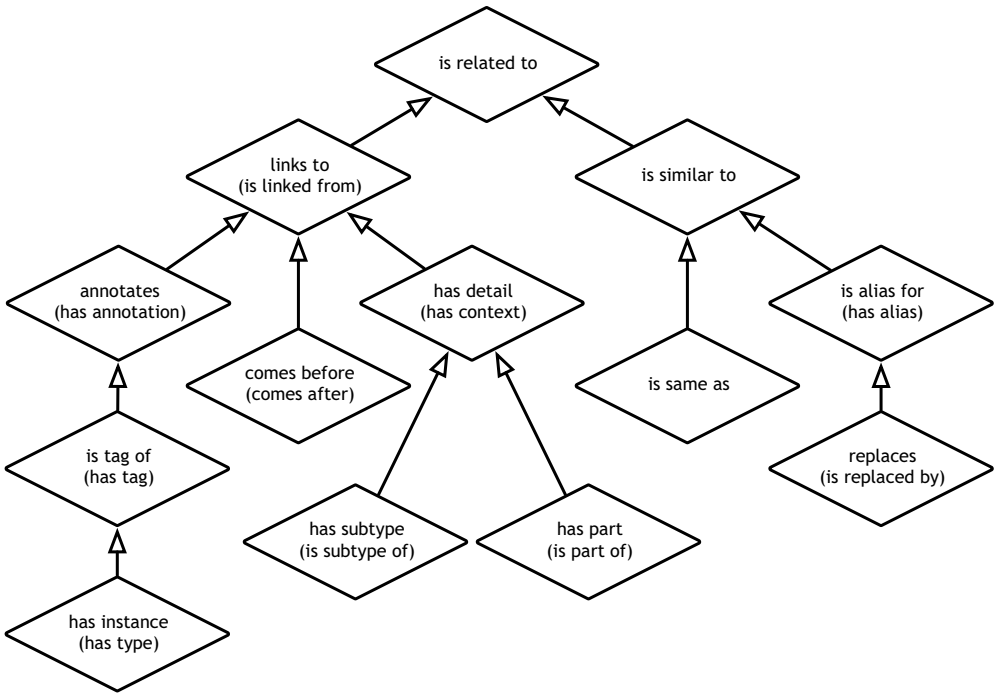

FIGURE 2.6: CDS relation hierarchy. Inverse relation names in parentheses.

CDS, iMapping and QuiKey, all initially developed in the Nepomuk project, have somewhat co-evolved. CDS has been used as back-end and data model for iMapping and QuiKey.

For more information on CDS in relation to the Nepomuk project see (Völkel et al., 2008), for a comprehensive overview and evaluations see Max Völkel's thesis (2010).

2.6 The Seven Tasks for Visual Information Environments

For the design and evaluation of visual information environments, Ben Shneiderman (1996) identified the following seven tasks:

Overview Gain an overview of the entire collection

Zoom Zoom in on items of interest

Filter Filter out uninteresting items

Details-On-Demand Select an item or group and get details when needed

Relate View relationships among items

History Keep a history of actions to support undo, replay, and progressive refinement

Extract Allow extraction of sub-collections and of the query parameters

The four first points also form Shneiderman's well-known Information Seeking Mantra: "Overview first, zoom and filter, then details-on-demand". These four central points are all targeted at reducing perceived complexity. This is so important, because of the before mentioned limitations of the human mind to represent more than 4–7 items in working memory. For tools used in knowledge intensive work, it is thus crucial to impose as little cognitive overhead on the user as possible (Conklin, 1987). To deal with more than approximately 4 items a time, items have to be grouped into clusters, which are then mentally represented as a single object (Cowan, 2001; Miller, 1956; Anderson, 2005). This principle of abstraction is one of the most basic heuristics people use to deal with complex information (Vester, 2002; Dörner, 2003). It is also inherent to context-and-detail techniques like the Levels-of-Detail (LOD) approach, which means that objects in focus are shown in more detail whereas distant objects are shown in less detail in order to reduce complexity – be it for ease of computation or simply to not confuse the user with an overload of visual information.

Chapter 3

Requirements

To guide the design of visual tools for personal knowledge management in general, and iMapping in particular, a set of requirements was compiled, to be used as evaluation criteria for existing visual knowledge mapping techniques and tools, but also as requirements for the design of novel ones like iMapping as described in Chapter 4.

The generation of these requirements was based on several sources:

- From the results of comparing existing techniques and tools (Haller, 2003)
- Extensive *use* of such tools
- Common user interface design principles as suggested especially by Shneiderman (1996); Shneiderman and Plaisant (2004)
- Further considerations of scalability and the possibilities of modern technology

As a theoretical instrument for designers of notational systems, the Cognitive Dimensions framework (Green, 1989; Blackwell et al., 2001; Blackwell, 2006), helps to reflect on certain aspects of such systems. These dimensions were expressly not designed as a fixed set of recommendations but as a loose bunch of neutral dimensions, e. g., *Visibility* and *Viscosity* (resistance to change). While definitely related work, not all dimensions are relevant to this thesis. Those that directly correspond to the requirements are referred to in the respective sections. The Cognitive Dimensions framework is often abbreviated "CDs" – not to be confused with CDS, the semantic data model used for both tools in this thesis called Conceptual Data Structures, explained in Section 2.5.2.

Some of the requirements established here may appear self-evident in some respect, but it should be noted that none of the existing approaches and (mostly commercial) tools evaluated by Haller (2003) fulfilled all of the criteria and vice versa, none of the criteria was fulfilled by all of the tools and techniques.

The list of requirements is split into two parts: Section 3.1 with conceptual requirements, directed rather at the general visual mapping technique, and Section 3.2 with more technical requirements, directed rather at concrete software implementations.

Without the last requirement of *mobility*, this list is also published by Haller and Abecker (2009).

3.1 Requirements for Visual Mapping Techniques

3.1.1 Free Placing

The main point of visual knowledge mapping is to organize information spatially. For a user to be able to use his own cognitive map (Tolman, 1948) for orientation in his information space, it is desirable that information objects can be positioned freely according to the user's preference. Items should also maintain their positions, at least relative to their surrounding, when the map is modified (e. g., extended), otherwise orientation by remembering positions will barely be possible. Free Placing is also required to support the informal and lightweight technique of Spatial Hypertext, as introduced in Section 2.3.3.

3.1.2 Free Relations

The importance of linking information items has been stressed in many disciplines: In complex problem solving, it is key to understand the interrelations between phenomena and agents (Dörner, 2003; Vester, 2002), learning success depends largely on the ability to relate the newly learnt to one's prior knowledge (Ausubel et al., 1980; Reigeluth, 1983), and also Shneiderman's (1996) fifth of the seven tasks is "Relate: View relationships among items".

Every PKM tool should thus allow to represent interrelations between all items. This can happen and should be supported at a whole range of explicitness and formality (Millard et al., 2005):

Formalized Links are links that are typed with a standardized relation. The use of such fixed sets of standardized relations have been proposed e. g., by Jüngst (1998) and studied in depth by O'Donnell et al. (2002). Especially when the information in the map is to be processed by semantic technologies, e. g., on a semantic desktop system (Sauermann et al., 2005; Völkel, 2010), formally defined relation types need to be used. However, the use of *only* formalized relations limits users' expressivity. Furthermore, studies have shown that users often do not stick to the defined meaning of these relations, but rather use them like informal ones (Gaines and Shaw, 1995).

Informal Labeled Links often suffice, since the precision and automated processing facilities offered by semantic technologies is in many cases not needed. Forcing users to use formal relations would then only result in the abovementioned effect of wrong use, leading to incorrect results. Also, it would either be a waste of time and effort, since it is harder to decide on a formal level or, even worse, it could result in the loss of information when users choose to not use the tool because it imposes unnecessary efforts on them. It should thus be possible to just draw informal links between items and freely label them, like it is done with the widely used method of concept maps and many other types of everyday diagrams.

Unlabeled Links that simply represent a relationship that is not further specified, are valuable for the same reason as above. Something like a plain arrow often suffices, e. g., when its meaning is clear from the context. Forcing the user to enter an explicit label, like some systems do, would both increase the user's cognitive load and the visual complexity of the map. One of the reasons why mind-maps are so wide-spread and easy to use, is certainly the fact that relation types between nodes do not have to be specified. Also, Wiegmann et al. (1992) have found out that users with lower verbal abilities can achieve better learning results with visual knowledge maps when using unlabeled links.

Free Nodes finally, that do not have any explicit relation to other nodes should also be possible. Again, in the same line of reasoning, it would impose unnecessary cognitive overhead to force a user to explicitly connect every new item with existing ones (Shipman and Marshall, 1999a), like some tools do, that rely on a connected graph model. Allowing unconnected items that can just be scattered around and informally arranged, has proven useful e. g., for unobstructed brainstorming, where *Premature Commitment* (as in the Cognitive Dimensions vocabulary, Blackwell et al. 2001) would hinder the flow of ideas and their step-by-step structuring. When *Free Placing* is given, relations between items can also be implicitly expressed through

their spatial layout like in spatial hypertext (Shipman and Marshall, 1999b), or can be added later in a separate elaboration step.

How these different types of links are maintained and displayed is not discussed here. For visual tools, one obvious way would be through graphical arrows that visually connect items. However all levels discussed above are e. g., also possible with text-based hyperlinks.

Another valuable possibility of visually integrating a related item would be transclusion, i. e. , to include the content of the linked item directly instead of only refering to it.

To sum up: The mapping technique should allow for user-defined visual languages tailored to the domain at hand. It should support such visual languages by facilitating the process of choosing pre-existing relation types; nevertheless, informal link labels should also be possible. Using links and link labels at all should be optional.

The mark on the *Hidden Dependencies* scale of the Cognitive Dimensions framework should be low here, meaning that relationships between items should be easily visible. However, always showing all relations might lead to serious visual clutter (sometimes referred to as *spaghetti syndrome*). Thus, it is desirable that certain relations are only displayed on demand (c. f. below Sections 3.1.4 and 3.2.2).

3.1.3 Annotations

Annotations [1] can be all sorts of marks or notes that, strictly speaking, do not belong directly to the actual content. In the Cognitive Dimensions framework, such annotations would rank as *Secondary Notation*, meaning extra information in means other than formal syntax.

Annotations can be used for highlighting, e. g., through marking a certain area with color. They can also just be text notes carrying additional information like explanations, details or personal remarks that do not belong to the map itself. For clarity reasons, it should be possible to hide and show annotations on demand. When an annotation is currently not visible, its existence should still be indicated, in order to avoid it to fall into oblivion.

[1] In this thesis, the term *annotation* is not used in the sense of semantic annotations, which denote enriching existing content with formal metadata, but rather in the sense of Secondary Notation as in the Cognitive Dimensions framework (Blackwell et al., 2001).

In many software programs, text annotations are shown in an area separate from the map. However, this has three disadvantages:

1. The space for that is usually occupied even when no annotation is shown – this wastes valuable screen space.
2. Only one annotation can be seen at a time.
3. Because of the split attention effect, additional cognitive effort is needed to connect the annotation with item it refers to, compared with showing annotations in the immediate vicinity of the corresponding items (Chandler and Sweller, 1992).

To avoid that, annotations should either be placed in the direct vicinity of the items they refer to, or visually linked to them e. g., by a connecting line (Mazarakis, 2009).

3.1.4 Hierarchy, Abstraction and Overview

Especially when maps are used extensively, e. g., for personal knowledge management, the issue of overview gains major importance. Ben Shneiderman's famous "visual information seeking mantra" (Shneiderman, 1996; Shneiderman and Plaisant, 2004) that he found to be globally relevant in visual information systems is: "*Overview* first, zoom and filter, then *details on demand*". A central aspect of getting an overview over something and seeing its overall structure, is the ability to abstract it into clusters. For a user to zoom into a specific part of a map, and for a system to decide which data is considered a detail only to be shown on demand, are both largely facilitated when items are organized in a hierarchical model. Also, the last of Shneiderman's seven tasks, "Extract: Allow extraction of sub-collections . . . " is facilitated, when, through a hierarchical model, there is a clear distinction of what parts are to be extracted.

Abstraction is also one of the core Cognitive Dimensions. And one of the core benefits of mind-maps is their inherent hierarchical model that provides an easy way to structure contents.

The mapping technique should thus support to arrange items in levels of hierarchy in a way that when large collections of items are displayed, their macro-structure becomes salient. To that end, the technique should not only support to express fine structure (i. e. relations between single items) but also content structures on higher levels of abstraction, e. g., between groups of items:

Clustering – From the cognitive psychology of memory, clustering or *chunking* is known as the fundamental classical way to make large amounts of information items manageable for our limited capacity working memory (Miller, 1956; Anderson, 2005). Translated to visual mapping, this means a technique should allow clustering several items into a chunk. This can be done implicitly e. g., by arranging them according to gestalt principles (Metzger, 1953), or explicitly in a way that a number of single items are assembled as sub-items of a group. Such a group would then itself be *one* item that from a higher perspective can be treated as a unit. Like this, it becomes possible to have *single* links or annotations referring to *multiple* abstracted items e. g., to a complete process that is represented by an item on its own, which, in turn, is detailed through several sub-items.

Sub-Maps – Taking this principle of clustering to the next level, and regarding whole maps as clusters, leads to arranging whole (sub-)maps in superordinate (meta) maps. And for three reasons, these sub-maps should not just be referred to with a hyperlink but actually integrated in the superordinate map:

- Sub-Maps can be recognized visually by their unique form rather than by words, which is more appropriate since maps are visual artifacts.
- By the use of sub-maps, it is possible to represent not only links inside or between whole maps but also between single items *across* maps.
- The user can zoom into and out of detailed regions (sub-maps) coming from an overview perspective in order to visually integrate these details with their more general context – much in the sense of the zoom metaphor in Reigeluth's (1999) Elaboration Theory of Learning and as confirmed to facilitate orientation (Bederson, 1999).

3.1.5 Scalability

If a knowledge management technique is to be applied to whole information repositories, e. g., as a personal knowledge base, it has to be conceptually fit to handle large numbers of items. Successfully used in this area are usually non-digital text-based systems like Niklas Luhmann's "Zettelkasten" (German for slip box, a system of cross-referenced paper cards, see Luhmann 1982) or today's Wikis that have found widespread use in recent years. Unfortunately, it is very hard to gain overview over such collections, because in these approaches, only one item is presented at a time, and only this item's cross references (links) lead to connected items. It would be desirable to combine the scalability of these approaches with the overview of visual, zooming-based methods.

3.2 Requirements for Visual Mapping Tools

While the above criteria mainly refer to mapping techniques as such, the following requirements are directed rather at concrete software implementations. They are about human factors and specifically about avoiding cognitive overhead, i. e., all cognitive load of the user that is not directly related to the actual content, but rather to dealing with the software as such. In knowledge-intensive activities, this is even more crucial than otherwise, in order to leave as much of the user's limited working memory to the actual task at hand. For the user to be able to fully concentrate on a map's content, the software should aim to relieve him or her of all actions not directly related to the content. Operating the software should in general not involve more interaction than absolutely required. From this goal, the following requirements can be derived.

3.2.1 Simple Editing

This refers to the Cognitive Dimension *Viscosity* (resistance to change). In order to support reuse and elaborative maintenance of a knowledge model, the mapping system should have a low viscosity, i. e. it should be easy to make changes in the artifact:

Brainstorming – To easily and quickly capture thoughts, it should be possible to add a new item with one single (or double) mouse click or one keystroke. The label should then be entered without any further intermediate steps. Multiple keywords (or sentences) should be entered in succession, separated only by a single keystroke and recognized by the software as separate items that can later be refined or have more structure added.

Extending Maps – In order to be able to elaborate or extend existing maps, it should be possible to add new items in any position. Specifically for approaches that support free placing, this may not be trivial, since there might not be enough space initially. Here, the software should aid the user as good as possible, e. g., in rearranging existing items in the immediate vicinity to make space for the additional ones. Hereby it is important that the existing items maintain their relative positions at least roughly, so they still correspond to the initial layout and thereby to the user's cognitive map.

Incremental Formalization and Structure Evolution – As Shipman and Marshall (1999a, p. 349) write: "Systems should be designed to support the process of incremental formalization and structure evolution as tasks are reconceptualized". If a PKM system supports both informal and more formal content structures like it

is demanded above (c. f. Section 3.1.2) and described by several other authors e. g., (Abecker, 2004; Völkel, 2010; Braun et al., 2007, 2008), it should also facilitate the stepwise elaboration and formalization of the contents. In order to support the full life cycle of knowledge artifacts, this is important: An information item may start as an informal note or key word, the entering of which should not be obstructed by unnecessary formality. Later, it may be part of an informal hierarchical bullet list. After several cycles of refinement, the same item might end up well defined and formally interrelated with other concepts of a semantic knowledge model. If this process of incrementally formalizing content is not well supported, it leads to what is called *Premature Commitment* in the Cognitive Dimensions framework (Blackwell et al., 2001) and will likely make the use of formal semantics fail (Völkel, 2010).

3.2.2 Filter and Focus

The second point in Shneiderman's information seeking mantra, *zoom and filter*, addresses the process of deliberately narrowing the focus to the essential. Mapping software should allow filtering out contents that are temporarily not of interest, in order to avoid distracting the user with unnecessary complexity. This can happen in different ways: in the sense of adjusting a focus where the unfocused becomes hidden or reduced, or in the sense of filtering, where objects are faded in or out due to certain properties they carry. Reducing the display to the essential increases the Cognitive Dimension *Visibility*.

3.2.3 Integration of Detail and Context

Every information environment that presents a larger amount of items than what can be easily overseen by a user has to deal with some sort of hierarchical clustering (c. f. Section 3.1.4). This introduces a difference between contextual and detail information (which can be a very soft distinction and may depend on the current point of view). The problem that arises through this, is how to integrate these details with their surrounding context in a way that users are able to grasp the structure of the contents and do not loose orientation.

Reigeluth's (1999) elaboration theory of instruction stresses the importance of putting new contents into context before going into details. These acts of orientation, of integrating details into their context, can place a challenge especially when dealing with large or complex collections of information. For a mapping tool that means: How can the user be supported in not loosing context orientation while dealing with details of a map that is too large to fit on one screen? Whichever approach is used, it should at any point be

possible to easily regain orientation and see the larger context with minimal interaction effort.

To reduce visual complexity at a given level of abstraction, it can be helpful to fade out information that is currently not of interest. However, to maintain the context it might be better to still show some or all surrounding items, but in a less detailed way in order to avoid unnecessary complexity and waste of screen space (Chimera and Shneiderman, 1994). In this **Levels of Detail** (LOD [2]) approach, the whole greater context is initially visible and details are only revealed on demand or automatically, when an item is in focus. When applied to mapping techniques, overview can be improved by letting the user choose the level of detail of each area. Like that, e. g., the essential elements of a (sub-) map can be made more salient by fading out the lower details. *Details on demand* is also the third part of Shneiderman's (2004) mantra. To relieve the user of this manual adaptation of LOD, this can also be done automatically:

For techniques that change the presentation of objects according to their size, the term *semantic zoom* has been coined, although this has nothing to do with semantic technologies in the sense of formalized information. With a semantic zoom, far away objects are usually displayed in a simplified manner, e. g., an article could be represented by its title only. When zoomed into, more bibliographic information could be shown, then also the abstract and finally the whole article.

So-called *Fish-Eye Views* (FEV) form another class of techniques, where LOD of certain areas are adjusted according to a degree-of-interest (DOI) function based on the current focus. FEVs are described later in Section 3.2.3.

Cockburn et al. (2008) give an overview over different categories of techniques to visually integrate context and detail information, and summarize numerous evaluation studies comparing such approaches. Their main three categories are described in the following sections:

Additional Overview

Cockburn et al. (2008) call this category "Overwiev+Detail". They list several studies suggesting that, for certain visual search tasks, it could be beneficial to show an additional overview window – similar to that known from Google Maps (see Figure 3.1). However, they also note:

[2] In this work, *LOD* stands for *Levels of Detail*, not for *Linked Open Data*, for which it has recently been used a lot, especially in the semantic web community.

FIGURE 3.1: Overview Window: A Screenshot of Google Maps featuring a small additional window in the bottom right corner, that gives a broader overview and marks the current view port of the larger detail view.

> "Spatial separation demands that users assimilate the relationship between the concurrent displays of focus and context. Evidence that this assimilation process hinders interaction is apparent in the experiments of Hornbæk et al. (2002), who note that 'switching between the detail and the overview window required mental effort and time.' Baudisch et al. (2002) made similar observations of the costs associated with split attention (Chandler and Sweller, 1992) with overview+detail interfaces." (p. 25)

Fish-Eye Views

Fish-Eye Views (FEV), first described by Furnas (1986) and later more generally by Keahey (1998), are techniques that – similar to an optical fish-eye lens as in Figure 3.2 – show a certain area in great detail while compressing the surrounding context in a way that it is still visible but takes much less space, in favor of the area in focus. There are many different specializations of the abstract FEV approach. Sarkar and Brown, e. g., have developed a specific FEV technique for graphs, as shown in Figures 3.3 and 3.4.

Cockburn et al. (2008) file fish-eye views under the category "focus+context", of which they cover the biggest part. Such focus+context techniques circumvent the abovementioned split attention effect by showing the details *in place*, i. e., in their native visual context. Again, research still seems incoherent, with some studies favoring fish-eye views

FIGURE 3.2: Fish-Eye Lens: Photo shot with a physical fish-eye lens. The picture is distorted so that the flower in the focal area is shown in much detail as compared to the rest of the picture that is very compressed.

over techniques like zooming or additional context windows and some not, where "fisheye was slowest and least preferred" (Cockburn et al., 2008, p. 21). A specific drawback of FEVs is the moving target problem: Because FEVs dynamically distort the view according to the current focus, and because the focus is usually continuously shifted during visual search activities, the target point of a visual search also keeps moving – which makes it harder to pin-point.

While FEVs can surely be helpful in certain situations when dealing with large amounts of visual information, and while they often initially provide a stimulating user experience, they also have shortcomings for other cases. FEVs should only be used as an optional device and only in variants that are well adapted to the general visual paradigm used.

For some further discussion of the usefulness of FEVs for iMapping in particular, see Section 4.2.3.

Smooth Zooming

Both of the aforementioned approaches (additional overview and fish-eye views) show details and context simultaneously and thus have to spatially integrate them somehow. Zooming as an approach does not have this problem because contexts and details are

FIGURE 3.3: Fish-Eye Views of Graphs: An undistorted graph view of the Paris metro network – without fisheye effect. Marked is the station that is in focus of the FEV below in Figure 3.4

.

separated by time. The conceptual integration has to be done mentally by the user. And, as several studies have shown, this mental integration is largely facilitated if the zooming transitions are *smoothly animated* (i. e. continuous and steady) instead of simply jumping to the target view (Donelson, 1978; Bederson, 1999). Cockburn et al. (2008) cite several studies, where although smooth animation in several studies did not improve task completion times, it did help users' spatial memory and facilitate to form a spatial model of their information space – this seems to hold for zooming environments but also for non-zooming ones. They write:

> "Animating the transition between zoom levels can dramatically reduce the cognitive load (but fine-tuning the duration of the animation is important)." (Cockburn et al., 2008, p. 26)

This is not surprising, because when the picture jumps too fast or in big leaps – like the "zoom" functions of many contemporary visual tools (from web browsers to text processors and viewers) are implemented – visual continuity is lost and reorientation is required from the user. The same holds for scrolling and panning. Smooth transitions come closest to what the natural sense of spatial orientation has been optimized for by evolution: in real life, a person gets a continuous coherent visual stream while physically changing perspectives when approaching details or taking a step back from something to see it in its context. Moving in the map should be similar to moving in natural environment because only automated cognitive processes do not interfere with controlled

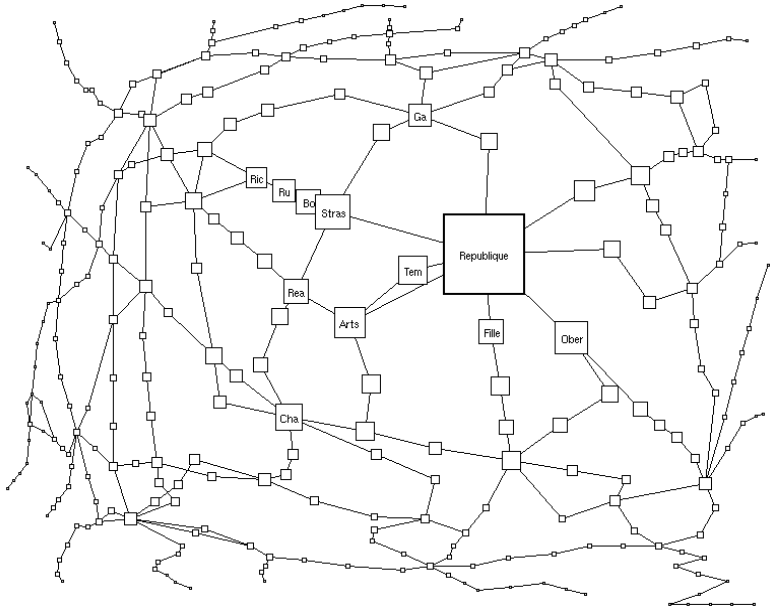

FIGURE 3.4: Fish-Eye Views of Graphs: FEV of the graph in Figure 3.3. The vicinity of the station in focus is shown larger and in more detail (with captions) than the rest of the graph. More distant parts of the graph are compressed to the margins of the view. (Both figures taken from Sarkar and Brown 1992a by permission of the Systems Research Center of Digital Equipment Corporation in Palo Alto, California)

ones (Shiffrin and Schneider, 1977). Especially in PKM settings, when very large maps are being built over several years, maintaining a good spatial model of the map seems more crucial than in shared information visualization environments, where the actual spatial layout is novel to the user or changes without the user's interaction.

The history of zooming user interface frameworks has been described above in Section 2.4. Why and how iMapping is actually based on zooming, is explained in the iMapping chapter in Section 4.2.3.

3.2.4 Accessing External Content

Many external knowledge resources are electronically available – locally like files, or remotely like web pages. For these it should not only be possible to be *referred to* from a map – like a reference in a paper book, they should also be *accessible* from the map, and thereby from their thematic context. This can be done at least by hyperlinking to them, which would still impose a context switch on the user when the referred media would open in another viewer (e. g., a web browser). Because every context switch causes cognitive overhead, ideally they should be avoided – as technology permits – by integrating appropriate viewers into the mapping software. In this way, external visual media (like pictures, web pages, documents or even videos) can be viewed in context.

3.2.5 Interoperability

Most people already use other software for various aspects of their personal or organizational knowledge management, or will do so in the future. Also, different tools are suited for different tasks, which are usually interrelated across types and modalities. For example, an e-mail message can be related to a process described in a map and to an event represented in a calendar tool. Because of that, mapping tools, like personal knowledge tools in general, can increase their usefulness greatly by allowing the interchange of content with other such tools. This can happen in various ways, e. g., by connecting to interfaces of existing tools, by using file formats like XML that are easy to process by third parties or through the use of semantic technologies which even allow interchange at the level of concepts and meaning. However, the use of formal semantics is not considered a requirement for all visual knowledge mapping applications.

3.2.6 Mobility

Because the need for personal knowledge management is not constrained to desktop use, comprehensive PKM solutions should include mobile use cases. This is especially crucial when problem solving processes are to be supported, because of the incubation effect (Wallas, 1926; Friedenberg and Silverman, 2005; Dodds et al., 2004; Ellwood et al., 2009), which is that problem solutions and fruitful ideas are known to often occur in situations where the individual is cognitively engaged in activities *other* than consciously working on the problems solution.

With nowadays' pervasiveness of mobile IT devices, a PKM system should also allow mobile access to one's personal information repositories. While, with current technology, it is difficult to support the whole range of functionalities on mobile devices, especially those that provide overview and spatial editing, mobile tools should at least support targeted search, editing details and especially the capturing of ideas in a fast and effortless manner.

Chapter 4

iMapping

This chapter describes how iMapping as a general *technique* (Section 4.1) and the actual *tool* (4.2) have been designed to fulfill the requirements established in Chapter 3. A clear distinction between technique and tool is hard to make. However, any tool that fulfills the requirements described in the technique section can be considered an *iMapping tool*, even if issues discussed in the tool section are resolved differently. Also, while the technique is described comprehensively, the tool section does not include a complete specification of the software, but rather covers the most crucial design solutions. Note however, that not all design solutions described here have been implemented. Section 4.3 briefly describes the current implementation of the iMapping application and also gives an overview of which features have been implemented in which version. Section 4.4 gives an overview on related work that has not yet been covered in the foundations chapter.

The iMapping technique as such has first been outlined by Haller et al. (2006); Haller (2006). Later, some of the design decisions have been published by Haller and Abecker (2010a). The iMapping technique and tool have been described briefly by Haller et al. (2010) and more comprehensively by Haller and Abecker (2010b) [1].

4.1 Design of the iMapping Technique

This section describes the core features of the general iMapping technique along with their rationale. The design goal was to create a visual mapping technique that unites the advantages of the major existing mapping techniques introduced in Chapter 2 and fulfills the requirements explained in Chapter 3. In summary, these are:

[1] This paper has received the Ted Nelson Award and has been chosen for further publication as Haller and Abecker (2010c)

Core advantages of existing approaches

- Visual knowledge representation with structural analogy to content (as common to most visual mapping techniques)
- Simple hierarchical overall topology (as in mind-maps)
- Representation of interrelations (as in concept maps)
- Allowing constructive ambiguity (as in spatial hypertext)

Requirements for visual Knowledge mapping techniques

- Free placing
- Free relations (in various degrees of explicitness)
- Annotations
- Overview / Abstraction / Hierarchy
- Scalability

The requirements for concrete mapping tools are addressed in the subsequent Section 4.2.

This section is structured as follows: The first three sub-sections explain the *basic structure* (Section 4.1.1), how *linking* is managed (Section 4.1.2) and how *visual tangle is avoided* (Section 4.1.3). The last two describe how *annotations* can be presented (Section 4.1.4) and point out how *other visual techniques integrate* well with iMapping (Section 4.1.5).

4.1.1 Basic Structure

Starting point was the requirement of *free placing*. Like in spatial hypertext, items can be freely created at or moved to any position in the map. An iMap can be seen as a virtually infinite pin board. Usually (e. g., for personal note-taking or idea management), these items will be short text passages. The size can vary from just a keyword to a short note or whole paragraphs. Depending on the implementation, other item types could be anything from pictures, over generic files and web pages to usual PIM objects like contacts, events or to-dos (c. f. Sections 4.1.5 and 4.2.6).

Creating and adding content to an iMap is done by clicking anywhere in the map and typing some text. It is always possible to add informal text items. While typing, existing items matching the current content should be proposed for re-use if desired (see *Re-Using Existing Items* on p. 39). Although allowing for semantic knowledge management, it has been a fundamental design decision to never force the user to specify any semantics. Content can later be refined and formalized incrementally.

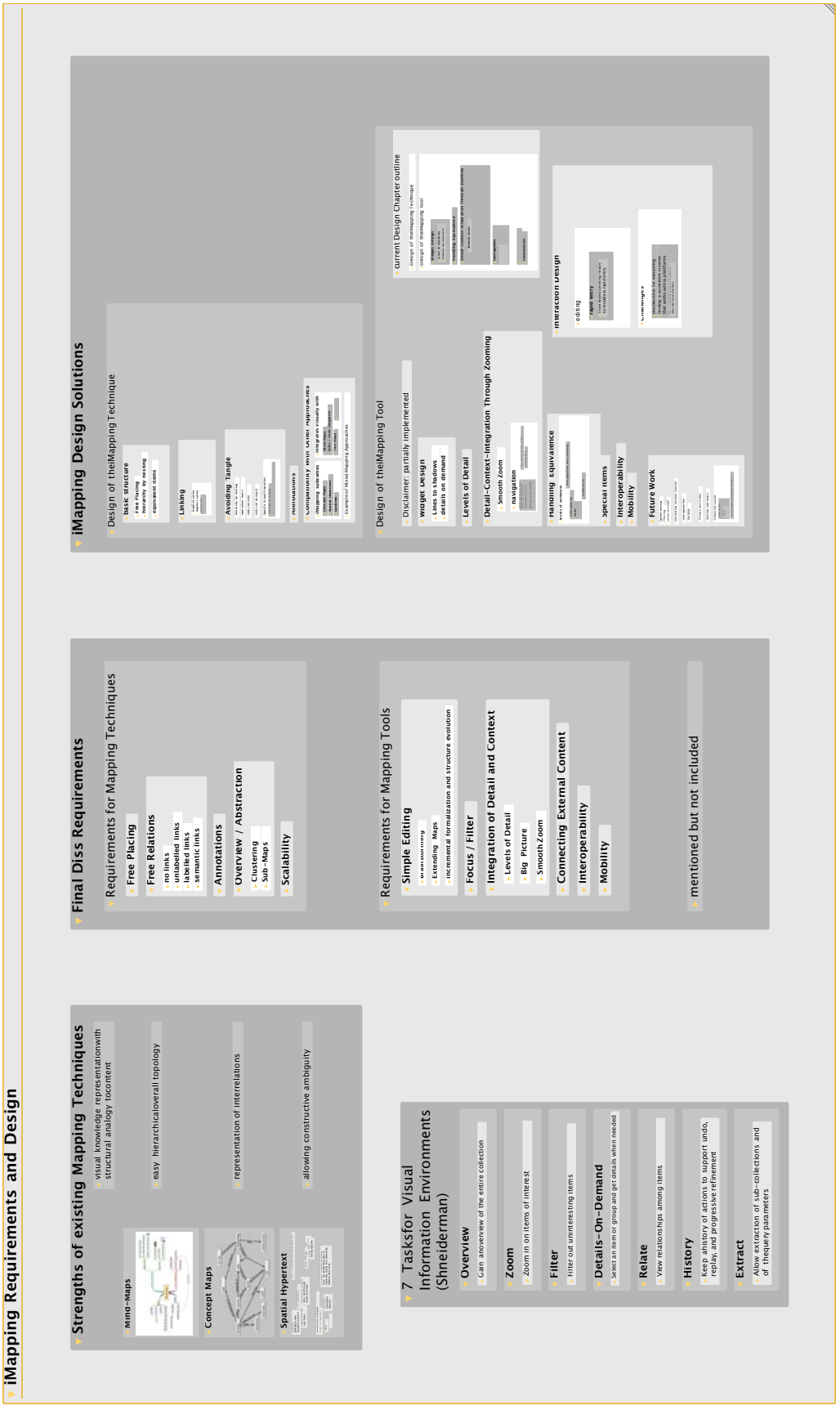

FIGURE 4.1: Example iMap: A sub-map of a big iMap about this thesis.

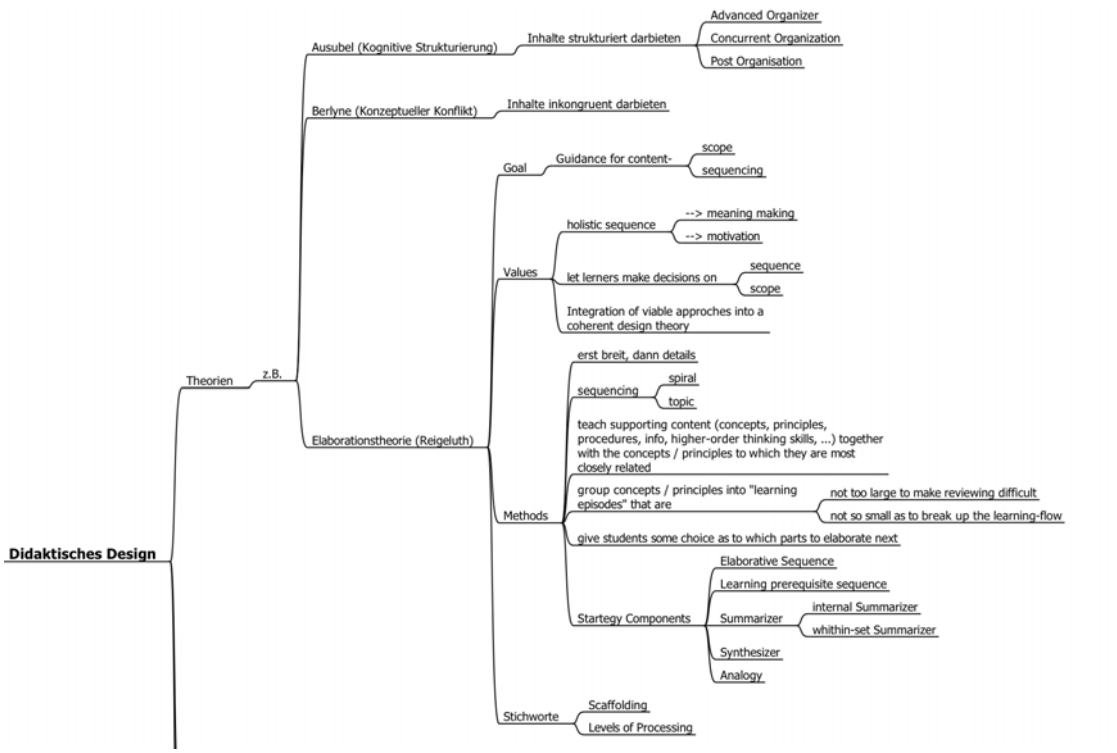

FIGURE 4.2: Limited Spatial Expressivity in Mind-Maps: Clipping from a large mind-map – all sub-branches point into the same direction.

Free placing also allows for informal clustering (requirement described in Section 3.1.4), simply by moving related items closer together. A more discrete, explicit way of clustering or making sub-maps can be achieved by nesting:

A **basic hierarchical structure** is represented by visual inclusion: nested items are shown inside one another (see Figure 4.1). Compared to a classical tree view like it is used, e. g., in mind-maps (c. f. Figure 4.2), this has the following benefits:

- It allows for *Free Placing* thus leaving more freedom to place items according to gestalt principles (Metzger, 1953) (like, e. g., grouping).
- Node-and-link representations (like in classical concept maps and many other visual languages) are still possible without visually interfering with connecting lines used for the hierarchical structure.
- Nesting by inclusion is closer to natural orientation where details are part of their surrounding: In real life, when we want to see the big picture, we take a step back to see the surrounding context of something. The concept of nesting by inclusion is also already widespread, e. g., in the use of parentheses, Venn diagrams, tree-maps and other visual languages like plate notation [2]

[2] http://en.wikipedia.org/wiki/Plate_notation

– The layout principle stays the same on all levels of the hierarchy. In concentric trees like mind-maps, on the contrary, all sub-branches point into the same direction. This largely determines the layout thus leaving less freedom of expressivity to the user (see Figure 4.2). In an iMap, each item and each part of the map can be treated like an unlimited self-contained sub-map. As explained in Section 3.1.4, this facilitates *overview and abstraction*.

Although this kind of nesting is not new, traditional, paper-and-pencil based mapping techniques can not cover many levels of hierarchy like this. Computer-based mapping approaches, however, allow virtually infinite depths of nesting. In Zooming User Interfaces (ZUI) like iMapping, transitions between levels of hierarchy are made with a smooth zooming function that allows users to swiftly change perspective from overview to any detail or back. What level of nesting is best and how many items should be on the same level, depends on the actual use case and the inherent content structure. In any case, the iMapping technique opens up a virtually infinite amount of space for iMaps to grow over time, e. g., when used as personal information repositories. That addresses the requirement of scalability.

Re-Using Existing Items In large maps, it is often the case that some items are relevant in several contexts. For example, one and the same person can be part of a group of colleagues in the context of work, a participant in a certain project, a guest in the context of planning a party and also a close friend another context. In classical graph-based mapping techniques, this example would entail the problem of where to position the item representing this multi-facetted person. And, whatever the solution, the item would still only be present in the vicinity of *one* of the related contexts. All other contexts would need to be connected with long-range links.

In an iMap, one and the same logical item can have several visual occurrences – one in each relevant context. Such *equivalent items* have the same content and share a common identity in the back end. Logically, they are the same.

In terms of interaction, any item can be duplicated into other contexts in a copy-and-paste manner. Alternatively, to avoid unnecessary navigation, it is possible to fetch duplicate existing items by simply re-using their name in a new context and confirming that re-use of an existing items is intended: The user will be prompted to decide whether he wishes to create a logically separate item or to re-use the existing one which would then have multiple visual instances. This principle has been called “bring-to-me navigation” by Jörg Richter et al. (2005).

4.1.2 Linking

To establish link structures, several different ways of interrelating items in an iMap can be distinguished:

Implicit linking – Following the principle of spatial hypertext, items can be loosely placed in spatial relations to one another without explicitly linking them at all. As described by Marshall and Shipman (1993) and refined by Francisco-Revilla and Shipman (2005), these spatial relations can be parsed to extract implicit relations, like sequence, grouping or hierarchy.

Explicit links between items can take four different forms:

Labeled links: If no pre-existing relation type is suitable, the user can always just enter the full label of the link to be displayed, e. g., along the arrow. In the reference implementation, that creates a new relation type in the back-end.

Unlabeled links: Links do not have to be labeled at all.

Semantic statements: Relating items in a subject-predicate-object manner without the need to actually draw a link between them in space can be done in different ways: Either by listing them in a optional table of links (see Figure 4.3), or with an additional tool like QuiKey (see Section 5.2. In either case, the user gets a choice of existing relation types selectable by an incremental text search, thus supporting reuse of existing relation types and avoiding misspellings.

Hyperlinks: Links do not have to be drawn as arrows; hyperlinks go from within the text content of an item to any other item.

In an informal formative evaluation study, testers preferred graphic links and the additional link area the way they are described here to other variants and methods like showing more links, or animating linked items.

4.1.3 Avoiding Tangle

In classic graph-based approaches, nodes usually have to be arranged in a layout that minimizes edge-crossings in order to reduce visual complexity. Of course, arranging a map with the goal to minimize line crossings leaves less freedom for arranging it by other criteria like content structure or gestalt principles (on gestalt theory see Metzger 1953). Even with line crossings minimized, maps with large amounts of nodes and links often suffer from tangled links (a. k. a. *spaghetti syndrome*). In iMapping, this problem is addressed in several ways:

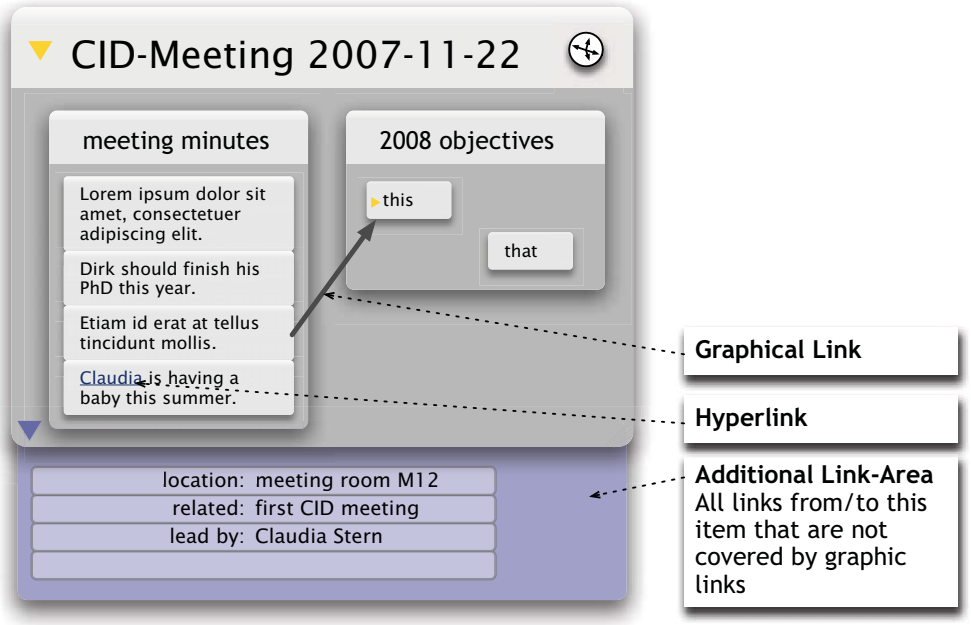

FIGURE 4.3: Different Ways of Linking: Graphical links, hyperlinks and a table of additional links (mock-up).

Equivalent items: The possibility to use one and the same logical item in several places also alleviates especially the need of long distance graph edges (compare Figures 4.4 and 4.5).

Links as lists: As described above, and shown in Figure 4.3, a table of links can also replace graphical links where needed.

Specific Visual Properties Instead of Flat Graph Graph structures like entity-relationship models are capable of representing a very wide range of information structures. Even contents without an *inherent* graph structure can often be broken down into subject-predicate-object triples and thus become represented as a graph. That is also the reason why modern generic knowledge representation languages like RDF (Manola and Miller, 2004) are based on such a graph structure. If such a graph is to be rendered visually, without further knowledge of the content's semantics, it appears natural, to display it as a flat graph with each node one separate entity and each *triple* [3] a labeled arrow between two such entities. However, such graphs have a higher visual complexity and are thus harder to read compared to what can be done with some knowledge about the meanings of the properties (i. e. relation types) used. Figure 4.4 shows a small example of a concept map style semantic net, laid out as a simple graph. Figure 4.5 shows the same structured information in iMapping style, with less connecting lines needed.

[3] In the semantic web community, subject-predicate-object propositions are commonly referred to as "triples".

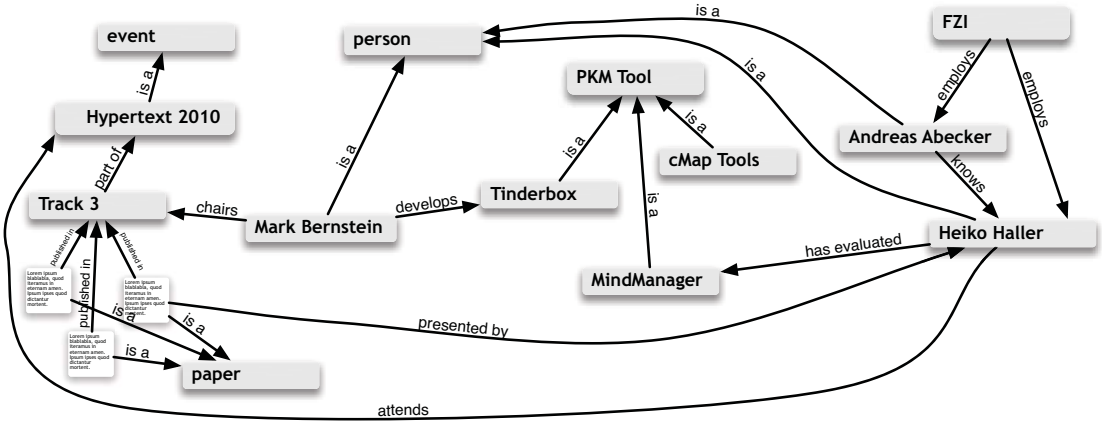

FIGURE 4.4: Tangle Compared: Example of a semantic net rendered as a flat graph.

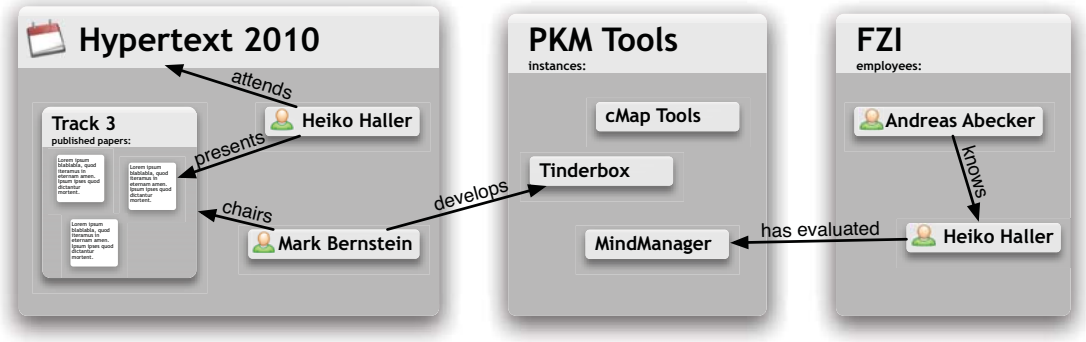

FIGURE 4.5: The same structured information as in Figure 4.4 but as an iMap (mock-up) with less connecting lines needed. Most properties are visualized in ways that are more specific, namely by nesting and by showing known item types with an icon.

Most properties are visualized in more specific ways:

- As argued in Section 4.1.1, displaying hierarchy by nesting, instead of graphical lines like in mind-maps or organization charts, reduces visual clutter. Which of the properties are treated as hierarchical ones is in many cases subjective. But under a personal knowledge management perspective, such decisions are up the user (as opposed to automatic information visualizations, as discussed in Section 2.2).
- Often, groups of items *share* certain properties – e. g., their type or a relation to a common parent item (like the "PKM-Tools" or the "Papers" in Figures 4.4 and 4.5). Both cases can be depicted by nesting without the need of connecting lines between these items and their parent items.
- Another way to show an item's type, is by means of an unobtrusive icon, e. g., in the item's head area. When an item is assigned to a user defined type that does not have an icon associated, the same principle can still be used with text, e. g., by putting the type behind the item's name in parenthesis: Mark Bernstein [Person]

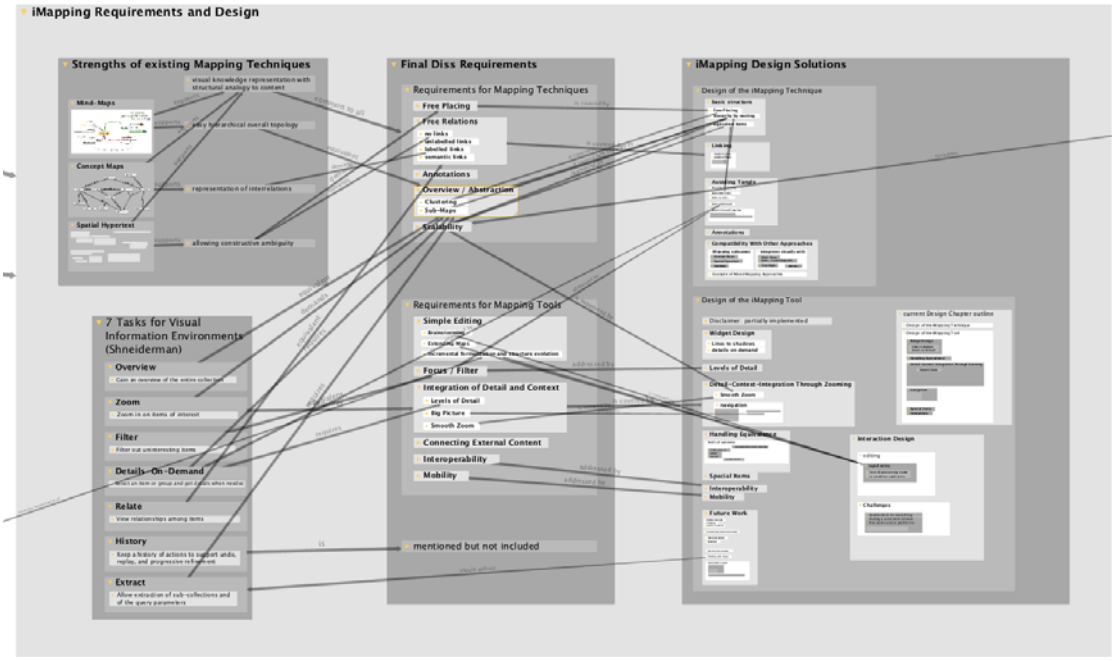

FIGURE 4.6: Tangled Web Syndrome: When all links in a dense graph are visible, it is hard to distinguish them.

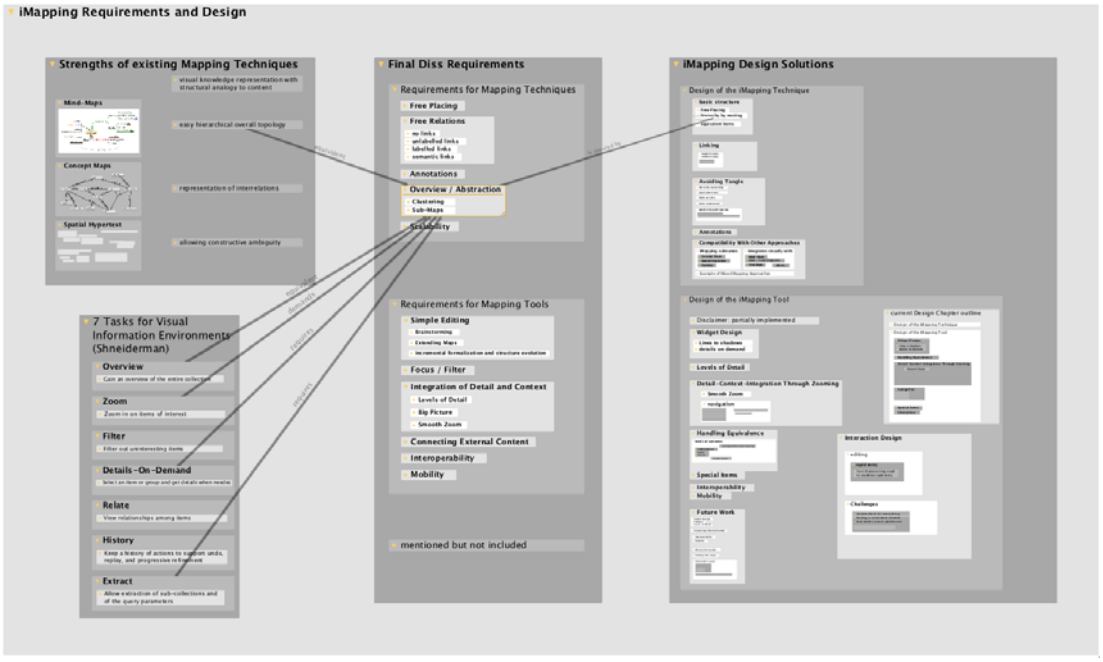

FIGURE 4.7: Links on Demand: Screen shot of the same sub map as Figure 4.6 but with only links from and to one item visible.

This general principle of rendering semantic properties in *specific* ways instead of connecting lines can also be applied to many other semantic and visual properties. In the current implementation, however, the two last ways (icons and shared properties) are not yet implemented.

Links on Demand While the above mentioned measures reduce the number of lines needed, they still do not warrant untangled links, as can be seen in Figure 4.7. A way to further reduce visual clutter, is to apply Shneiderman's (1996) design principle *details on demand*, and only display links when needed. Thereby, the links of current interest are more salient and easier to visually integrate with the nodes they connect. In the current iMapping implementation, by default, an item's graphical links are only displayed when it is selected or hovered over (compare Figures 4.6 and 4.7). Optionally, of course, all links can also be made permanently visible. This also fulfills to some extent the requirement *Filter and Focus* described in Section 3.2.2.

4.1.4 Annotations

As described in the Requirements chapter in Section 3.1.3, in-place annotations should be made possible. There are many established ways and styles for such annotations that are fully compatible with the iMapping technique. As these techniques can be easily transferred to the iMapping technique, there does not appear to be any research needed here. For this reason, no special annotation feature has been implemented up to the current research prototype. However, Figure 4.8 shows some ways in which this could be realized.

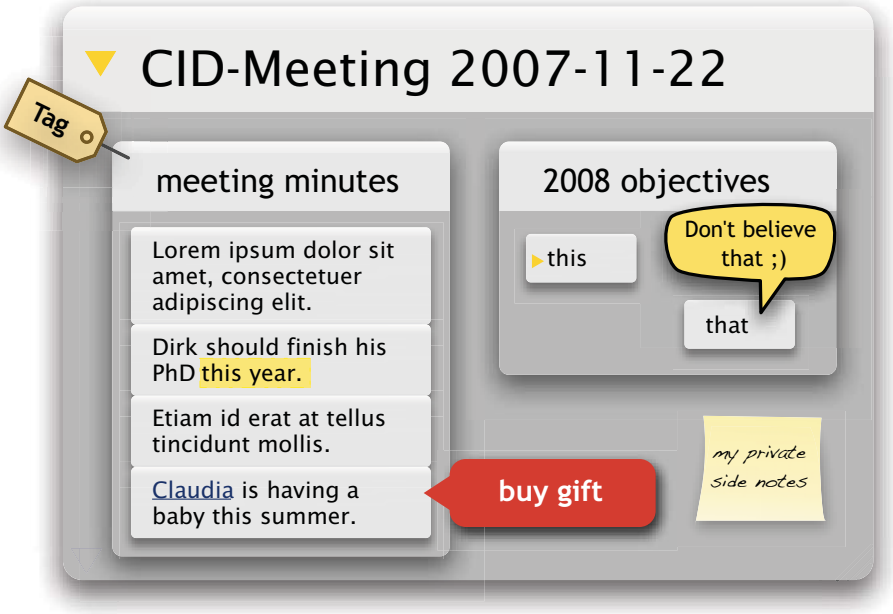

FIGURE 4.8: Annotations: Five possibilities to display annotations in a way so that they stick out from the other *main* content of a map (mock-up).

4.1.5 Compatibility with Other Visual Techniques

Not so much a design decision but a desirable side effect of the iMapping design is the fact that it is conceptually compatible with many other visual techniques. Because an item is basically an empty box, anything that can fit in such a box can be integrated. And because iMapping is based on a multi scale (i. e. zooming) paradigm, the size of the things integrated also does not matter.

Figure 4.9 shows a selection of other visual paradigms integrated seamlessly in an iMap. The first three of the following are actually subsumed by iMapping, the others integrate well:

Spatial Hypertext – Without graphical links and nesting, iMapping basically *is* spatial hypertext.

Concept Maps are a sub set of iMaps: labeled connected directed graphs.

Outlines most commonly have a style of indented labels, as, e. g., in tables of contents. In Figure 4.9, a screenshot of OmniOutliner [4] is used with one item highlighted, which shows that this common indentation style can be transformed into the nested boxes paradigm of iMapping without even changing the layout.

Mind-Maps can be displayed as self-contained sub maps. Their hierarchical structure can even be mapped directly to the item hierarchy used in iMapping. Only the layout and style is different.

Treemaps are an information visualization technique described in Section 4.4. A treemap is a rectangular area recursively subdivided into other rectangular areas that can be zoomed into. As such, treemaps fit very well into iMaps.

Euler Diagrams [5] as well as their more widely known special case **Venn diagrams** [6] use overlapping areas to represent sets and their relationships. As shown by the blue and orange polygons in Figure 4.9, this diagramming paradigm blends nicely with the iMapping technique and introduces another way to depict cases, where an item can have multiple parents – without the use of *equivalent items*.

[4] http://www.omnigroup.com/products/omnioutliner/
[5] http://en.wikipedia.org/wiki/Euler_diagram, http://www.eulerdiagrams.com/
[6] http://en.wikipedia.org/wiki/Venn_diagram

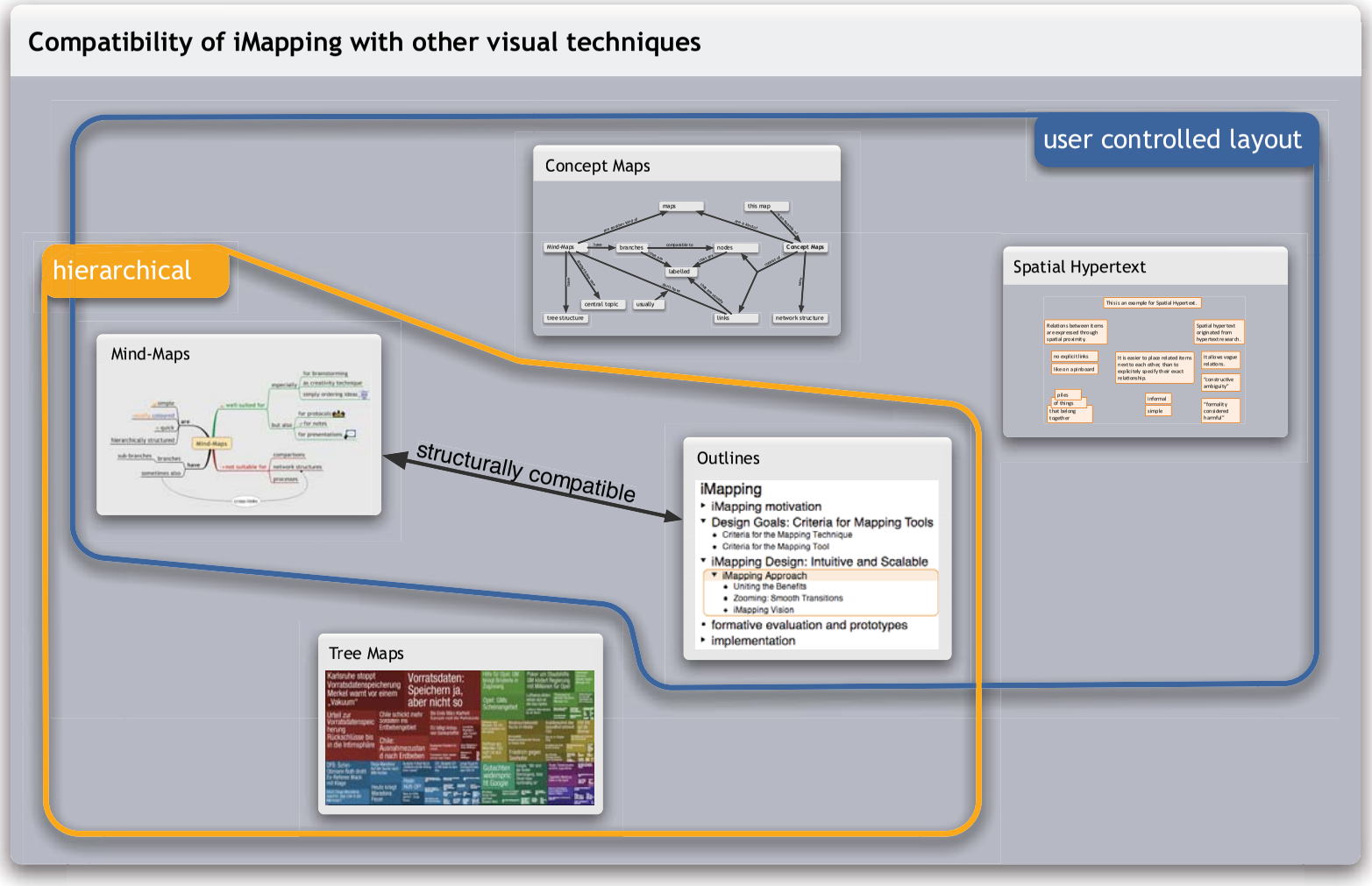

FIGURE 4.9: Example of how different other mapping and visualization techniques integrate well with iMaps – namely: Venn diagrams, outlines, treemaps, mind-maps, concept maps and spatial hypertext

4.1.6 Conclusions

The design of iMapping as a general mapping technique fulfills all of the requirements for visual mapping techniques as posed in Section 3.1 (set in italics): The general structure of iMaps, which use nesting as a *basic hierarchy* allows for *free placing*. The various ways of linking allow *free relations* on all levels of formality from formal semantic links to no links at all. Visual complexity through tangled links is reduced in several ways:

1. Presenting the hierarchy through nesting leaves out the need for connecting lines.
2. The use of equivalent items relieves the need for long-range links.
3. The possibility to show Interrelations in list form leaves graphical links to where they are explicitly wanted.
4. The amount of visible links can be greatly reduced by displaying links on demand only.
5. The principle of using specific visual attributes for certain types of relations additionally reduces the amount of generic graphical links.

The visual paradigm leaves room for many styles of *annotations*. Since the basic iMap item is a simple scalable rectangle, iMapping is conceptually compatible with many other information visualization and knowledge visualization techniques, some of which are actually completely subsumed by iMapping.

As Table 4.1 shows in overview, iMapping as a general mapping technique introduces a scalable approach and unites the advantages of the classical mapping techniques.

Further design decisions that are rather implementation specific and that are based on the requirements of actual knowledge mapping *tools* are described in the next section. A comprehensive overview of all requirements and how they are conformed with is presented in the discussion chapter in Section 7.1.

TABLE 4.1: iMapping in comparison with the three basic mapping techniques. (This is Table 2.1 with an additional column for iMaps.)

	Mind-Maps	Concept Maps	Spatial Hypertext	iMaps
Structural analogy to content	✓	✓	✓	✓
Simple hierarchical topology	✓	–	–	✓
Representation of interrelations	–	✓	–	✓
Constructive ambiguity	–	–	✓	✓
Scalability	–	–	–	✓

4.2 Design of the iMapping Tool

This section deals with those parts of the design work that are not so much part of the abstract technique of iMapping, but rather addresses the concrete implementation. However, as mentioned before, the distinction is fuzzy. It describes design decisions for the current iMapping tool – namely, the visual appearance of items (Section 4.2.1), how different forms of equivalence are handled (Section 4.1.1), why and how context and details are integrated through zooming (Section 4.2.3) and the basic interactions for navigation (Section 4.2.4) and editing (Section 4.2.5). The last section describes other possible special item types, e. g., to integrate external information resources or query results (Section 4.2.6). For an overview of what has been implemented in the current version, see Section 4.3.

The requirements from Chapter 3 that are to be fulfilled by the design of the iMapping tool are

- Simple Editing
- Filter and Focus
- Integration of Detail and Context
- Accessing External Content
- Interoperability
- Mobility

4.2.1 Visual Item Design

Visual Item Structure

As Figure 4.10 shows, the visual representation of an item consists of the following parts:

- A *head* area at the top, that carries the item's text content if it has any. If it is dragged, the item is moved.
- A triangular *expansion icon*, that indicates whether the item is expanded or collapsed. If it is clicked, the expansion status is toggled. If the item is expanded, its *belly* is shown. If the item is collapsed, only its head is visible. If an item contains children that are currently hidden because it is collapsed, this is indicated by the presence of the expansion icon. Otherwise it is only shown on demand.
- A *link drag handle*. Links can be made by dragging it onto another item.

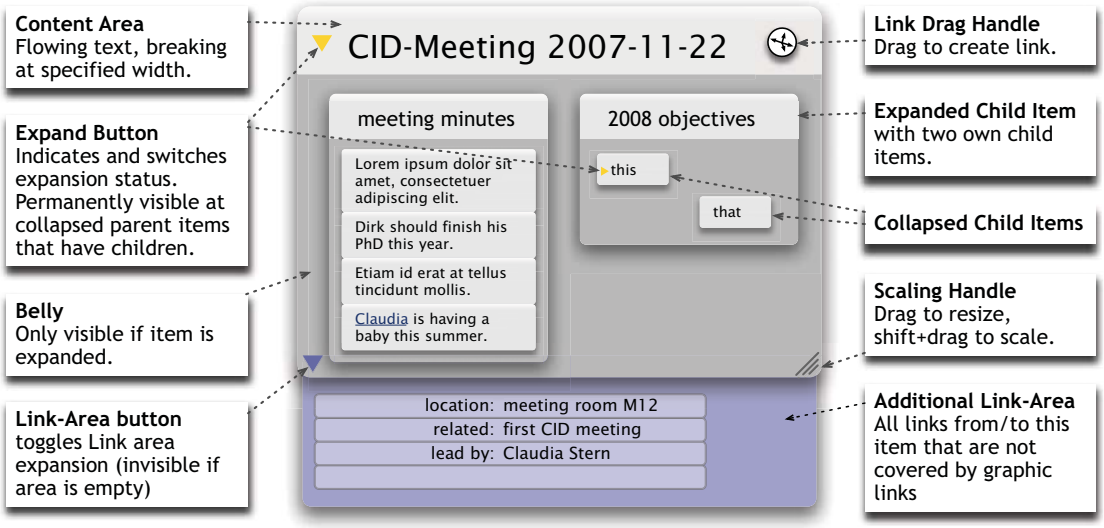

FIGURE 4.10: Item Structure (Mock-Up): Visual Elements of items. Link and resize drag handles are only visible when the item is hovered over with the mouse.

– The belly of an item holds its children items. In any view, the background of the currently visible items is the belly of their parent item. Therefore, if an item's belly is dragged, the whole map is moved in the view port [7]. This is a common *direct interaction* way of panning [8] and scrolling.

– The *resize handle* at the bottom right of an item is used to resize the item, e. g., to make more space for the children. If the resize handle is dragged with the Shift key pressed, the item is scaled instead, i. e., its belly is resized proportionally so that it appears larger or smaller, including the children. See Figure 4.11

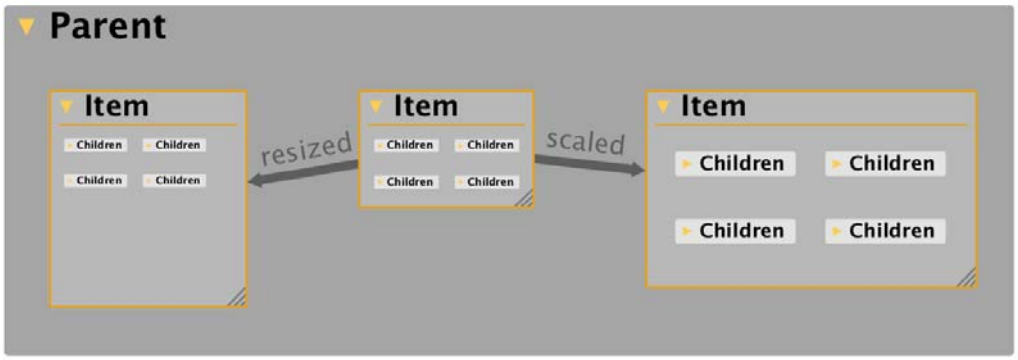

FIGURE 4.11: Resizing: The original item is in the middle, its *resized* version is on the left and its *scaled* version on the right.

[7] The so-called *view port* shows the currently visible part of a map. It could be seen as the virtual *window* through with the map is viewed. Scrolling and panning (s. b.) change the offset between map and viewport.

[8] *Panning* is the horizontal equivalent to scrolling.

– The *link area icon*, like the expansion icon can be used to selectively display the optional link area.

– The *link area* displays additional logical links of the item that are not displayed as arrows, e. g., because they were made in different contexts at an equivalent item. Such links can also be created here by entering the relation type and target in the empty last line of the link table.

Details on Demand

In order to keep visual complexity at a minimum, the *details on demand* principle is also applied here: Per default, only an item's head, belly and expansion icon are shown. All other elements are only shown when the mouse is hovering the item – just like the graphic links.

Also, *levels of detail* can be adjusted by *collapsing* and *expanding* items to decide whether an item's sub-items are to be displayed or not. These two presentation modes serve the purpose of reducing visual clutter and saving screen space: Items containing many children can be *collapsed* such that the *belly* containing the children is not visible. However, this technique only really saves screen space if the gained space is used by other items. This can be achieved with the following technique that could be called **magnetic lists**:

Similar to what is known from collapsible tree views like they are common in outliners and file explorers, a group of items can be arranged vertically adjacent in a list structure. When one of these items is expanded, instead of overlapping its sibling items, those items below the expanded one are moved down. If the item is collapsed, the same siblings move back up, so that the list always remains intact and fully visible. An example of such a magnetic list is shown in Figure 4.12.

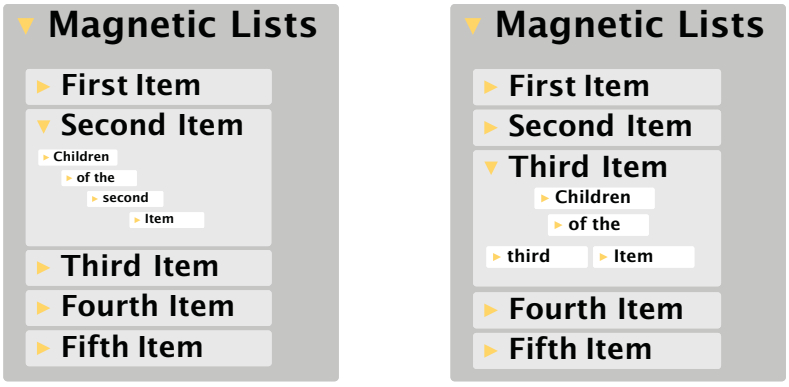

FIGURE 4.12: Magnetic Lists: Items that are grouped in a list structure can be expanded and collapsed individually but always stay adjacent to one another in order to avoid wasting screen space. Without this technique, either expanded items would cover some of their siblings, or much more space would be needed.

An automatic way of adjusting levels of detail is so called *semantic zooming* (s. a. Section 3.2.3), where items are displayed differently depending on their size.

A simple way to do this is to restrict the rendering depth, so that items that are too small to be read are not shown at all. This improves rendering performance and reduces visual complexity. However, apart from smoother animations, which can also be achieved differently, the expected benefit for the user is low or even negative, because this kind of rendering is unnatural: In real life, far away details my become tiny and blurred, but they do not suddenly disappear. And even visual details that are hard to recognize may help visual recognition of the parent items.

A more advanced way of semantic zooming is to render items differently: Items whose head is too small for their text content be read can use their belly area to display the text. At least for short items with many children, like titles or named groups, this improves readability. A drawback of this approach is, that the visual substructure of items is not visible anymore. This can be alleviated by only transparently overlaying the alternative views as demonstrated in Figure 4.13.

FIGURE 4.13: Semantic Zoom: Levels of detail are adapted depending on how large an item is rendered. For items that are too small to read in detail, their text is displayed in large text size covering the whole belly area. (Mock-up overlaying original screen shot)

Item Styles

The visual appearance of items should take account of the following goals:

1. Items should be clearly distinguishable from their parents.
2. The style should be robust to infinite recursive nesting.
3. The head area should be clearly distinguishable from the body, at least when interaction behavior differs, which is the case in the current design.
4. Visual complexity should be as low as possible in order not to distract from the actual content and its structure.

Three alternatives of item stylas are shown and discussed here:

Boxes with Borders This simple style, as depicted in Figure 4.14, uses only lines to designate the borders of items. This style is easy to implement and efficient to render, and it has been used in early iMapping prototypes. However, it turned out that this style sometimes makes it hard to distinguish the hierarchical item structure, especially areas where many parallel lines occur. An explanation for this could be that it might require more cognitive effort to parse the actual item structure when only borders are salient and the item areas need to be deduced from the border structure.

Progressive Shading This style, as depicted in Figure 4.15, omits explicit item borders and uses different shades to distinguish items from their surrounding. Informal surveys have shown that most users prefer this style over the one mentioned above and find it easier to read. Visual complexity is very low here and implementation effort is moderate. One drawback of this style is, that without further consideration, it only covers between 4 and 10 levels of hierarchy, because each level needs to be clearly lighter than its parent level. Also, this results in very dark tones for high levels of hierarchy, which, in turn, yields weak contrasts to the texts.

In the current implementation, which uses this style, this issue is addressed in two ways: 1) Colors (and thereby brightness levels) can be set manually. 2) Automatic coloring wraps around, so that children of an item with the highest brightness level are set to a medium level. It is important to have automatic coloring, because users should not be bothered to decide on color every time they create a new item. However, this current solution still is not optimal for several reasons:

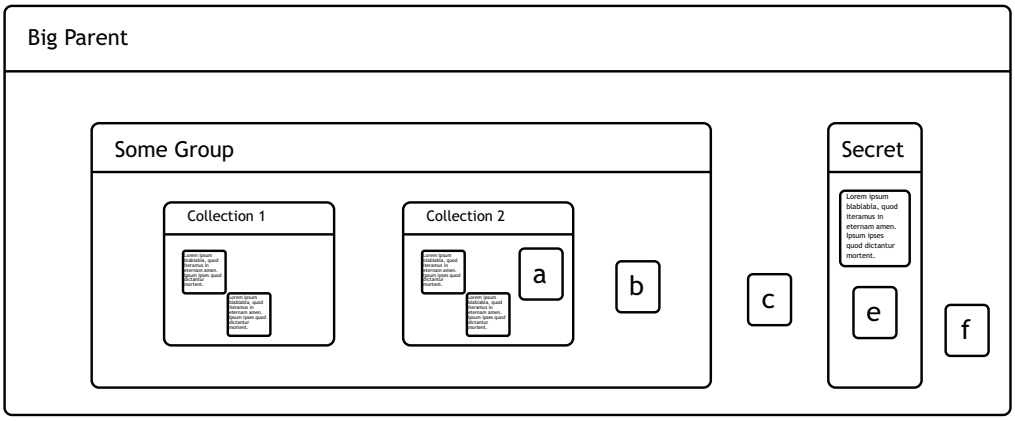

FIGURE 4.14: Boxes with Borders: Easy to implement but not so clear to view.

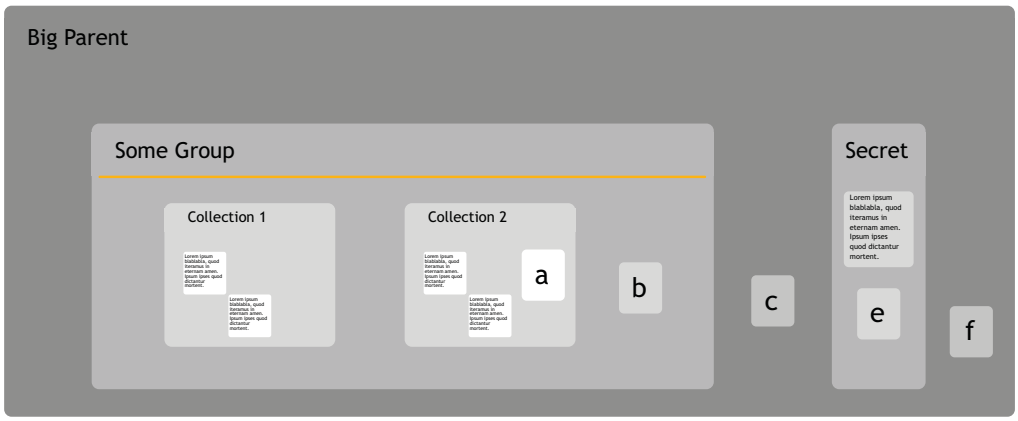

FIGURE 4.15: Progressive Shading: Low visual complexity, but inconvenient for automatic coloring in deep hierarchies.

FIGURE 4.16: Shadows: Visually favored style but hard to implement in a high performance way.

1. Visually, continuous hierarchies are arbitrarily intersected where color jumps back to a darker shade after the maximum brightness (white) has been reached.

2. Automatic coloring does not always yield results satisfactory to the user, which leaves users bothering about manual coloring and violates the principle of avoiding cognitive overhead.

3. No algorithm could be found to compute levels of brightness automatically for a given base color in a way that gave a satisfactory harmonious coloring scheme. This led to the need of designing a color palette by hand, which also turned out non-trivial if a certain esthetic degree is to be met. The importance of aesthetics in software design has been stressed, among others, by Hassenzahl (2006, 2008); Hoffmann and Krauss (2004).

Shadows An item style that does not have the drawbacks of the two styles described above is the one depicted in Figure 4.16: Item borders are distinguished by casting a shadow onto their parent's body thus creating the impression of items being slightly elevated from their parent item. Shadows have become quite ubiquitous in modern UI design and they are generally well received by users, as this is a comparatively natural way of distinguishing objects from their surroundings. This style scales well for infinite hierarchies, as the base color of an item can be the same like its parent's. This also leaves more freedom of manual coloring to the user while never requiring it, since color is not needed to provide enough contrast. Users who have been shown mock-ups of the shades style in informal inquiries, have expressed a clear preference for this style above the others.

The only drawback of the shadowing approach is that it needs considerably more computing power for rendering and that it is comparably harder to implement in a way that still allows smooth animations and looks natural. This is the reason why it has not been used in the current implementation.

4.2.2 Handling Equivalence

When equivalent items (as described in Section 4.1.1) are used, several kinds of equivalence can occur – depending on the data model used. Apart from merely technical ones, the following three are relevant for the iMapping tool as it is currently designed and based on CDS:

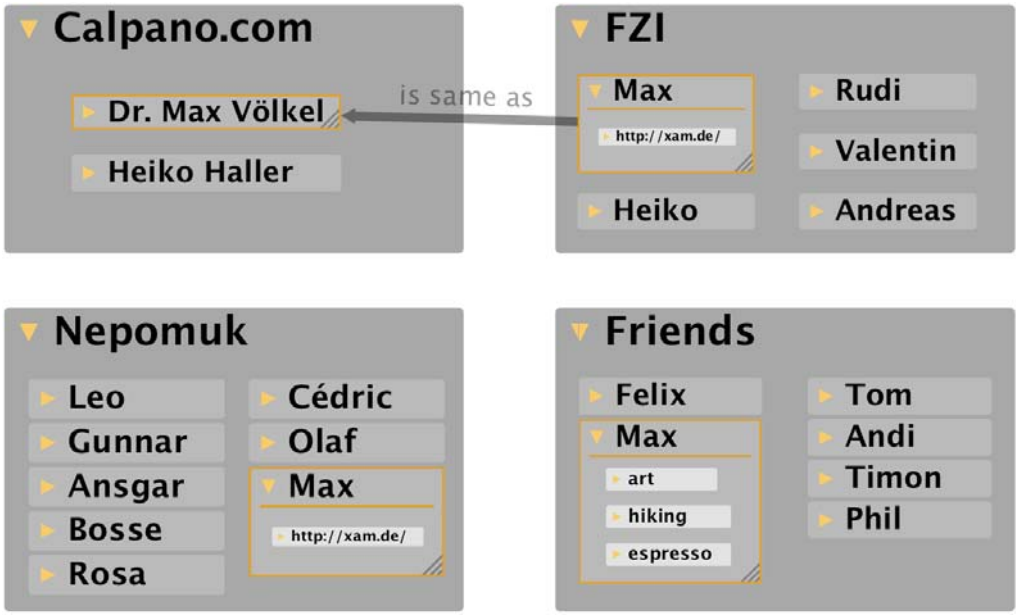

FIGURE 4.17: Different Kinds of Equivalence: The item Max in FZI has a *mirror* in Nepomuk, an *equivalent item* in Friends and a logically *same* item in Calpano.com

Visual Equivalence Items that share the same meaning, the same content, and the same visual appearance including all children items and their layout, are called *mirrors*. Their only difference is their position in the map and thereby typically their parent, i. e. their context. To be precise, also their scale and expansion status may differ. But the essential thing is that mirrors – if they are expanded – look exactly alike. In Figure 4.17, the two Max items in FZI and Nepomuk are mirrors. This is the tightest form of equivalence described here.

Structural Equivalence Items that share the same meaning and the same content but may differ in visual appearance are called *equivalent items*. They represent the same logical item (i. e. same CDS item) but occur in different contexts, can have different shapes, and also different children items. On the CDS level, however, these equivalent items are one and the same. In Figure 4.17, the *Max* item in *Friends* is an equivalent of the other *Max* items. In the iMapping tool, equivalent items can be created in two ways: Either by using the keyboard shortcuts for copy and paste (Ctrl+c and Ctrl+v[9]), or by reusing their names: When a new item is created, while its text content is entered, items matching the text are suggested for re-use and can be picked from a list.

[9] cmd instead of ctrl on Macs

Logical Equivalence Different items that have nothing in common (different CDS items, i.e. different visual appearance, different text content, different URIs [10]) except denoting the same thing, can be linked by using the CDS relation *is same as*. From an iMapping perspective, these are separate items connected by a link. A fully implemented CDS reasoner would treat these items as the same. However, although specified, this functionality is not implemented in the current CDS back end. In Figure 4.17, the item *Dr. Max Völkel* is stated to be the same as the *Max* items.

4.2.3 Integrating Details and Context Through Zooming

In iMapping, hierarchy is modeled by nested boxes as detailed in Section 4.1.1. Traversing such hierarchies is done by smooth zooming. An overview over zooming user interfaces (ZUI) has been given in Section 2.4 in the Foundations chapter. The general pros and cons and requirements of zooming in comparison to other techniques for integrating context and details have been discussed in Section 3.2.3 in the Requirements chapter.

For iMapping in particular, zooming offers the following advantages over the other techniques for integrating context and details that have been discussed by Cockburn et al. (2008) and in Section 3.2.3:

- Zooming is the only technique that leaves the representation of the map intact instead of breaking it apart to different levels of abstraction.
- It scales over deep hierarchies whereas other techniques do not: Both fisheye views (FEVs) and additional overviews are basically restricted to showing only two levels of detail, i. e. the detail view and the overview. If these levels lie too far apart, integration becomes lost: If a detail region covers an area that corresponds to one pixel in the overview, intermediate levels are missing to integrate both views. Similarly, a FEV is limited to magnification factors that still allow to target every detail area in the coordinate system of the overview. With too much magnification, FEVs become unusable, since moving the focus only one pixel would already pan the detail area to far. FEVs usually apply magnification factors roughly between two and fifteen. iMapping however needs to be able to deal with magnification factors of in the range of thousands. The author's iMap, e. g., already spans a magnification range of roughly 2 000 after 15 months use.).

[10] A URI (Uniform Resource Identifier) is a unique text string of a certain style that is used especially in semantic technologies to identify objects. See `http://en.wikipedia.org/wiki/Uniform_Resource_Identifier`.

– Cockburn et al. (2008, p. 24) summarize three studies: "Nekrasovski et al. (2006) recently compared performance in hierarchical tree browsing tasks using zooming and distortion-oriented focus+context techniques, both with and without an overview. Contrary to their predictions, the pan-and-zoom technique was significantly faster than focus+context. Although the presence of an overview did not impact task times, subjective measures of physical demand and enjoyment favored an overview; these findings are in agreement with the findings of Hornbæk et al. (2002). Donskoy and Kaptelinin (1997) also showed that zooming outperformed a fisheye view for tasks involving identification and drag-and-drop manipulation of file icons."

This is why the iMapping tool is based on a zooming function that allows both smooth and virtually infinite zooming. The interaction with this ZUI is described below.

4.2.4 Navigation

The user can freely zoom using the scroll wheel of a mouse – Just like it is known to many people from Google maps. The fix point of the zooming action is set to the current mouse position. Like that, it is also possible to zoom to areas other than the current center.

Another way that is usually faster and more precise is *targeted zoom*, where a selected item is directly zoomed to. While theoretically leaving less freedom to users, in most cases targeted zoom is easier to use, because neither tracking nor fine adjustments are needed in order to get to the desired view. This can be done either by double clicking any item or selecting it and pressing *Enter*. Zooming out is done either by double clicking the parent item at the margins of the window or by pressing *escape*.

A side-effect of the zooming approach is that when zoomed in on an area, its visual context is lost. To get back to an overview perspective, additional navigation is required. The interaction effort for reaching an overview perspective and coming back to the details at hand, is reduced like this: When zooming out with the *Esc* key instead of actively selecting parent items, the selection is not changed. Like that, a user can zoom out several levels of hierarchy by repeatedly pressing *Esc*, and then back to where he was in one move by pressing *Enter*. To zoom *all* the way out in the same way, can be done by pressing *Shift + Esc*.

As mentioned above, panning and scrolling is done in a direct interaction way by simply dragging the map, which can be done by dragging any background part, i. e. the belly area of any item. When the head area is dragged, the respective item is moved instead.

Apart from these interactions that are directly targeted at adjusting the view, there are several cases where the view automatically adapts to the selection, such that the selected item is always fully visible in a well readable size:

- When a link arrow is double-clicked, the item on the other end of the link is selected and zoomed and panned to if necessary.
- When items are selected by keyboard (arrow keys), the view port also follows, so that selected items are always visible.
- When an item is selected in QuiKey [11] as a result of searching, browsing or a query, and the selection is confirmed with *Enter*, the respective item is selected and made visible in the iMap.

All in all, any transition to another place in the iMap is carried out by smooth panning and zooming in order to maintain a sense of spatial orientation while moving through the map.

4.2.5 Editing

Text Editing

The text content of any item can be edited by *Ctrl + double clicking* [12] its text area (head) or by pressing *Ctrl + Enter* when it is selected. New items can be created by *Ctrl + double clicking* in any empty space, i. e. any item's belly area. Both cases enter an edit mode, that can be left in several ways:

- When pressing *Enter* or clicking anywhere outside the edit field, the item is saved with the new text content.
- *Esc* discards the new text and returns to the state before the edit action.
- Pressing the *rapid editing* shortcut *Ctrl + Enter* while in edit mode saves the current item and creates a new item in the same parent just below the current one. Like this, a series or new items can be entered one after another, resulting in a list of items.
- Similarly, pressing *Ctrl + Shift + Enter* while in edit mode saves the current item and creates a new *child* item of the current one. Like this, hierarchical lists can be made with keyboard only interaction.

[11] QuiKey is explained in Chapter 5

[12] On Macs, the *cmd* key has the role the *Ctrl* key has on Windows and most Linux systems. To conform to this standard, for all uses of the *Ctrl* key, on Macs the *cmd* key is used instead.

Whenever a new item is created, existing item names are proposed that match the text that is being entered. These existing names appear as a list under the input field. If one of the existing items is selected, an equivalent of this item is created. This facilitates the reuse of existing items, which is especially desirable when content is modeled semantically. The matching rules that determine which items are proposed are those developed for QuiKey, as described in the next chapter in Section 5.2.1. The same advanced auto-completion method is used to select relation types when creating or editing links.

Auto Grow

When existing Maps are extended with new items, a frequent problem is that there is not enough space for the new content. In approaches that use constrained and largely automatic layouts, like the tree visualizations of many mind-mapping applications, it is trivial to insert nodes without destroying the layout. Approaches that give the user the power of free placing, on the contrary, usually also require the user to manage the complete layout on his own. Restructuring the layout of a map only to be able to add new content, however, is considered cognitive overhead and should be avoided. Any *automatic* re-layouting, on the other hand, entails the drawback of changing the layout chosen by and familiar to the user. In the iMapping Tool, this problem is addressed as follows: If items are added or change size (due to editing or resizing) in a way that they would not fit into their parent anymore, one of the following two adaptations are made, as demonstrated in Figure 4.18.

- If there is enough space around the parent in the direction where more space is needed, the parent automatically grows, in order to keep enclosing its children items. This growing may also propagate up multiple hierarchical levels.
- If there is not enough space, the parent scales down its children area so that there is more space available *inside*, while maintaining the same shape and layout to the outside. Like this, the spatial arrangement among the children items remains untouched as well as the spatial arrangement on the hierarchy level above.

Linking

Drawing a link between two items is done by dragging a linking icon from the source to the target item. Alternatively, one can drag from anywhere in the source item while holding the *Alt* key.

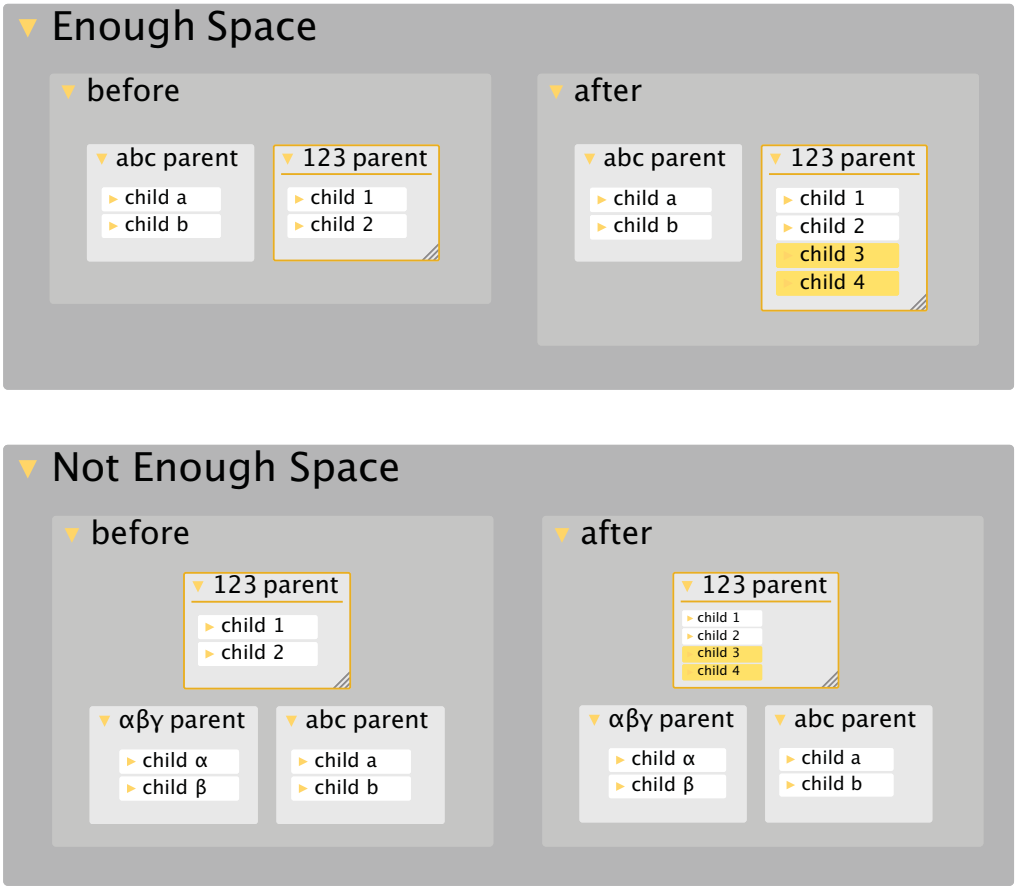

FIGURE 4.18: Auto Grow: Depending on the available surrounding space, parents either grow or scale down in order to make room for more children.

A link can have one of three statuses:

- Visible on Demand (default, as described before)
- Always Visible
- Hidden (a link between the logical items that is not visually represented as arrows in the map)

Such *hidden* links are listed in the link area as described in Section 4.1.2. Such *additional links* can stem from various sources:

- Visual links from or to equivalent items
- Former visual links that have been hidden
- Links that have been made via QuiKey (see Section 5)
- Links that have been specified in the link area, as described below.

The last line of the link area is always empty. Here, new links can be entered by filling out the *relation type* and the *target item* column. Selecting from existing items and relation types is facilitated by an auto-completion feature described in Section 5.2.1.

4.2.6 Special Items

This section summarizes a number of additional item types that have been envisioned or requested but are not implemented in the current version, because this would demand a high amount of implementation work and – while definitely useful in everyday work – is not needed to develop or evaluate iMapping as such.

Accessing External Content

As demanded in Section 3.2.4, various kinds of external information resources should be able to become integrated in iMaps. Such external information items should be distinguished into the following three categories:

Adopted Content is content that originates from external sources that are so tightly integrated, that the content can been viewed *and edited* inside the iMapping application. Examples could be contact data like phone numbers that originally stem from the system address book and are also written back there in order to synchronize with other applications and in order to keep data consistent.

Presentable Content is content that can be transcluded (c. f, Section 2.1). Many types of information are rather simple to transclude, even if they cannot be modified by the application. It can be represented by thumbnail previews that can be zoomed into to view the full content. A double click opens the content in the system default editor or external viewer. Examples include web pages, pictures, PDFs and e-mails.

Alien Content is content of unknown type, such as generic files in a proprietary format. Items are visualized by a generic file icon and the file name.

The distinction between these three classes of content is not inherent in the content as such but rather a matter of which adaptors are implemented.

Complex Items

Other special items can represent structured information in more specific ways than the generic nested graph. An item representing a task, e. g., can display its status, priority, due date etc./ in a form layout specific for task items, like it is shown in Figure 4.19.

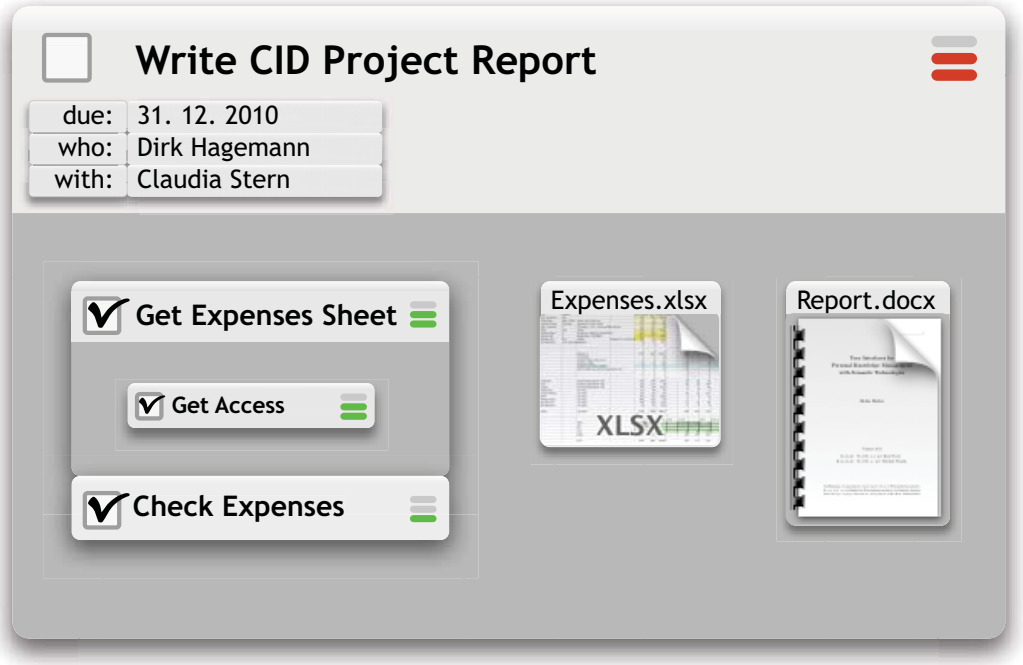

FIGURE 4.19: Tasks and Files: Mock-up of how other item types could look like. Nested tasks with completion status (checkmark) and priority (colored bars), Files are shown with thumbnail previews.

Query Items

Instead of fixed content, a query item always displays the current results of a query, as it is known e. g., from so called inline queries in Semantic MediaWiki (Krötzsch et al., 2007). There can be queries of the internal data model (i. e. CDS queries in the current implementation) or external web services. Depending on their structure, results can be displayed as lists, tables, or scatter-spaces (Waldeck and Balfanz, 2004), which allow interactive exploration of multidimensional data points.

A special case of local query items are items are *semantic collections*: These are query items, in which additional existing items can be *added* to the result set, with the effect that they are automatically semantically annotated such that they are *made* an actual correct result of the query. For example, if an item *Max* is dragged into a semantic collection that represents *all friends of Heiko that live in Karlsruhe*, then statements are added, that *Max is a friend of Heiko* and *Max lives in Karlsruhe*. Without further specification, such semantic collections are only possible for atomic queries and *AND* queries, because in most other cases it would not be clear, which statements need to be added exactly. Semantic collections are a more general form of what was shown in Figure 4.5 and explained in Section 4.1.3.

4.2.7 Conclusions

All requirements for concrete knowledge mapping software as introduced in Section 3.2, have been addressed by the design of the iMapping tool:

Simple editing is possible in the following ways: Items can be added anywhere with little interaction. *Brainstorming* and *extending maps* are facilitated by the *rapid entry* shortcuts that allow entering successive items and by the *auto grow* feature that recruits additional space when needed. *Incremental formalization and structure evolution* is supported in many ways:

- Items can easily be created anywhere without any formal overhead.
- The hierarchical structure can easily be changed later.
- Items can be interlinked informally in several ways.
- Relation types can be specified later.
- Relation types can be semantically refined later (e. g., by defining superrelations and subrelations).
- Semantic collections allow formal semantic annotation by simply moving an item (or its equivalent item) into the collection.

Focusing is provided in an implicit way by the zooming approach: When zoomed into a detail, contexts vanish. When zoomed out, too far away details become small. Focusing can be supported even further through semantic zooming. Additionally, a user can adapt the level of detail by manually expanding and collapsing items. *Filtering* can be done with QuiKey (see Chapter 5). Filtered map views remain a subject of future work (see Section 7.2).

The *integration of context and detail* is covered by a zooming function that allows smooth transitions between overview and detail views. It also features the option to quickly zoom all the way out and back in, in order to regain global orientation.

Although not implemented, many examples have been given of how *external content* can be integrated as adopted, presentable or alien content.

Although no runtime adaptors to other software are currently in use, *interoperability* is facilitated by the use of CDS as a flexible formalism specifically designed for PKM interchange. Also, all data is stored in open formats like RDF and XML that are easy to reuse and that can be extended for other future uses.

Although no special *mobile* versions have been developed, iMapping as a technique and the interaction concept of the tool seems appropriate for mobile use for several reasons:

1. iMapping makes fundamental use of zooming, which has proven useful esp. on mobile devices with touch screens: Like the iPhone, most smartphones nowadays have zooming viewers for many kinds of contents including maps, web pages, and documents.

2. The screen design, which only uses one coherent content area without toolbars or auxiliary panes, results in very good screen real estate – a prerequisite for mobile use.

3. The fine-grained content structure, which is encouraged through the iMapping design, is also favorable for mobile use because it can be browsed in a targeted manner instead of reading longer texts.

4. In most mobile devices, text entry is cumbersome. The auto-completion algorithm that is applied wherever existing content is reused, is borrowed from QuiKey, which has proven very interaction efficient (see Section 6.2).

Note that not all of these features are implemented in the current version of the software. The implementation status of each feature discussed in this chapter is presented in Section 4.3.2. A comprehensive overview of all requirements and how they are addressed by the design and implementation of iMapping and QuiKey is given in Section 7.1. Future directions, how the design of iMapping could be improved further, is discussed in Chapter 7.2.

4.3 Implementation of the iMapping Tool

The current implementation of the iMapping tool is open source and based on Java.

This section briefly describes the software component architecture of the current iMapping tool (Section 4.3.1) and gives an overview of the current implementation status of the features discussed so far (Section 4.3.2).

4.3.1 Software Component Architecture

As depicted in Figure 4.20, the iMapping tool consists of the two components *ui* and *core*. Like all other components mentioned here they are available in source code [13] as well as in compiled form [14] via the software build management system Maven [15].

[13] http://svn.imapping.info/
[14] http://mvn.imapping.info/
[15] http://maven.apache.org/

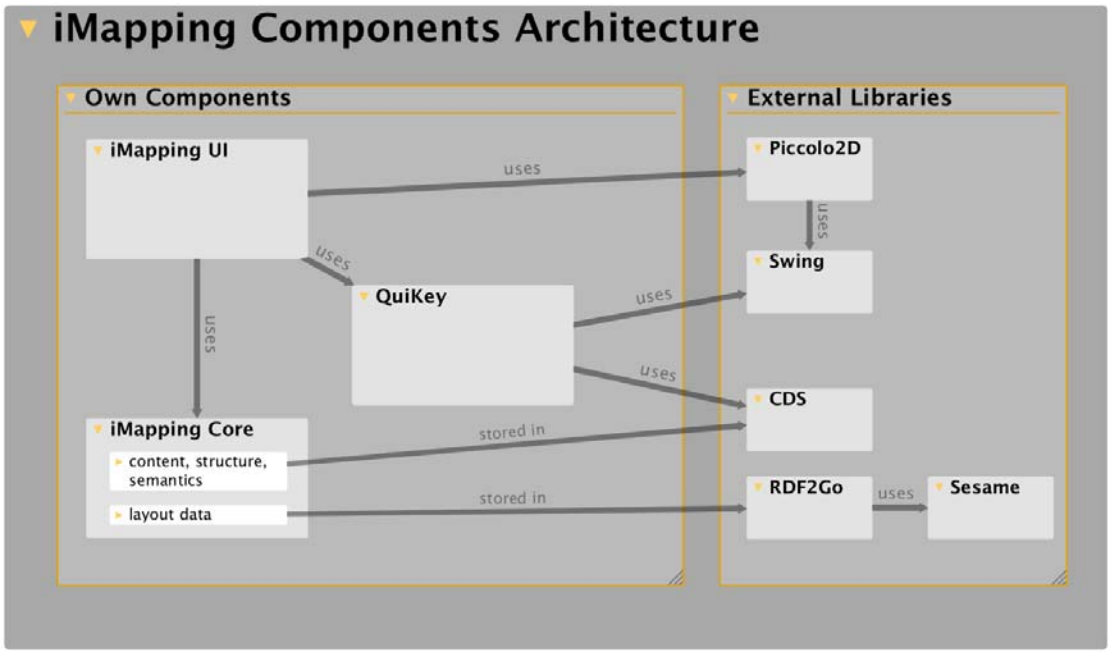

FIGURE 4.20: iMapping Architecture: Rough overview over the main components (ui, core) and their external dependencies.

iMapping UI

The iMapping UI is based on the zooming user interface framework Piccolo2D [16] that provides a 2D scene-graph with efficient zooming and scaling capabilities. The Java version of Piccolo2D in turn is based on Java Swing / AWT. The history of Piccolo2D has been outlined in Section 2.4.

In favor of getting the more basic core features mature enough for user testing, the goal of infinite scalability was deferred in the development agenda. So, in the current implementation, the whole content of the map is loaded into main memory, which limits the size of the Map to several thousand items for a memory limit of 1 GB. The exact memory usage depends of course on the amount of content, links and equivalents per item. The highest impact on memory consumption has the use of images, which has not been optimized for, so far. To give an order of magnitude, for the largest known iMap to date with 4 668 items, the application consumes 600 MB of memory without images and 1.2 GB of memory when 30 images are loaded, each of roughly one megapixel in average.

Another big factor of memory consumption is image caching: In order to improve rendering performance during animations, items are cached as rendered images so they only need to be rendered in detail when they are displayed largely. Like that, smooth animations of arbitrarily large maps are possible, as memory permits. Without image caching, the map mentioned above consumes around 400 MB less. Without this optimization,

[16] http://piccolo2d.org/

rendering performance, while sufficient for smaller maps (in the range of hundreds of items) is an issue for large iMaps (with thousands of items): When zoomed fully out of a large iMap with all items expanded, theoretically every single item needs to be rendered. For larger iMaps, like the author's one mentioned above, without caching, the current UI takes roughly one second to completely render the Map on an Apple MacBook Pro with a 2.4 GHz Intel Core 2 Duo CPU – even when small items are not displayed below a threshold where they become too small to read. For animations to appear as smooth, however, 30 animation frames per second are required and thus image caching is used despite its costly memory consumption.

iMapping Core

The back end component, which is named *core*, comprises the actual business logic, deals with persistence and keeps the model consistent, as explained below.

Historically, the CDS back-end used to be based on the semantic desktop infrastructure provided by Nepomuk (introduced in Section 2.5): All data was stored in the semantic data store shared by all Nepomuk components in order to enable interoperability. The binding to this semantic infrastructure has been dropped for performance reasons in order to be able to test the UI features in acceptable quality. However, all data is still stored in semantically defined formats: The actual content and its structure (items, links, relation types) is stored in CDS (Völkel, 2010, introduced in Section 2.5.2). The visual layout information (item positions, sizes, expansion status etc.) is stored in RDF [17]. The RDF schema used for iMapping is published online [18] and is included in Appendix E.

The use of RDF is handled by tools of the *semweb4j* [19] suite, which allows to generate a Java API from this semantic data model and to read and write the RDF data at runtime through this Java API. This is done through RDF2Go, a tool of the semweb4j suite. RDF2Go also acts as an abstraction layer for RDF stores. The actual RDF store used in the current implementation is Sesame [20]. The technical mapping of items' visual iMapping metadata in RDF to their corresponding CDS items is done by using the same URIs. The core component is responsible for keeping these two models consistent.

[17] The "Resource Description Framework" (Manola and Miller, 2004) is flexible language that allows to semantically define data models. It is widely used in the semantic web community and for linked open data.

[18] `http://svn.imapping.info/info.imapping.core/src/main/resources/imapping.n3`

[19] `http://semweb4j.org/`

[20] `http://www.openrdf.org/`

iMaps are stored in a single file with the extension *iMap*. This file is a zip compressed container for the two files: *content.cds.xml* which holds the content in the XML serialization of CDS, and *layout.rdf.nt* which holds the layout information formulated in the N-Triples format [21], a text serialization of RDF.

4.3.2 Implementation Status

The current implementation of the iMapping tool covers all basic functionality. Some more advanced features that would require a high implementation effort have been postponed in favor of bringing the basic iMapping principles to a level of usability where they can be evaluated and started to be used.

Tables 4.2 and 4.3 give an overview of which features have been implemented in which version. They are structured according to the order and section in which they were discussed. The three versions listed are: the one used for the comparative evaluation in spring 2009 (see Section 6.1), the one used for the summative evaluation in summer 2010 (see Section 6.3) and the current one from spring 2011.

TABLE 4.2: List of features that have been described in Section 4.1 Design of the iMapping Technique. Indicated are their implementation status for the versions used in the user studies (described in Sections 6.1 and 6.3) and the current version.

Feature	Comp. Eval.	Summ. Eval.	Feb. 2011
Basic Structure			
Free Placing	✓	✓	✓
Hierarchical Nesting	✓	✓	✓
Linking			
Labeled Graphical Links	✓	✓	✓
Additional Link Area	–	–	✓
Hyperlinks	–	(✓) [H]	–
Annotations			
Visually Different Annotation Items	–	–	–
Text Highlighting	–	(–) [T]	(–) [T]
Compatibility with Other Visual Techniques			
Spatial Hypertext	✓	✓	✓
Concept Maps	✓	✓	✓
Outlines, Mind-Maps, Treemaps	–	–	–
Euler-/Venn Diagramms	–	–	–

[H] Hyperlinks through wiki syntax were interpreted and browsable as visual (arrow) links that could optionally be displayed. However, they were not clickable in the text. Due to malfunction of the wiki parser used, this feature had to be disabled.

[T] Although not explicitly supported by the tool, it is possible to highlight text passages by the use of a technical trick.

[21] http://www.w3.org/2001/sw/RDFCore/ntriples/

TABLE 4.3: List of features that have been described in Section 4.2 Design of the iMapping Tool. Like in the table above, indicated are their implementation status for the versions used in the user studies (described in Sections 6.1 and 6.3) and the current version.

Feature	Comp. Eval.	Summ. Eval.	Feb. 2011
Visual Item Design			
Collapse / Expand	✓[C]	✓[C]	✓[C]
Resize	✓	✓	✓
Scale	✓	✓	✓
Item Colors	–	–	✓
Magnetic Lists	–	–	–
Semantic Zoom	–	–	–
Linking			
Thick Links (easy to click)	–	✓	✓
Links on Demand	✓	✓	✓
Hidden Links	✓	✓	✓
Show All Links	✓	✓	✓
Permanently Visible Links	–	–	–
Link Dragging Icon	–	–	–
Handling Equivalence			
Visual Equivalence	–	–	–
Structural Equivalence	–	✓	✓
Logical Equivalence	_[E]	_[E]	_[E]
Integrating Details and Context Through Zooming			
Smooth Zoom (manual)	✓	✓	✓
Smooth Zoom (targeted)	✓	✓	✓
Zoom All Out and Back	✓	✓	✓
Fisheye Views	–	–	–
Navigation			
Panning and Scrolling	✓	✓	✓
Keyboard-Navigation	–	✓	✓
Following Links Bidirectionally	–	✓	✓
Editing			
Wiki Syntax for Text Formatting	–	✓	✓
Wiki Syntax for Semantic Linking	–	✓	_[W]
Using System Clip Board for Text	–	–	✓
Special Items			
Background Images	–	✓	✓
Transcluding Other External Content	–	–	–
Linking External Content	–	–	–
Query Items	–	✓[Q]	✓[Q]
Task Items	–	–	–

[C] Indicating children items of collapsed items is not implemented: Expand icons are permanently visible in the current version.

[E] While it is possible to state the logical equivalence of two items in iMapping, the current CDS back-end does not interpret this relation as specified. Note however, that *logical equivalence* is the loosest form of equivalence in iMapping and is also implied by *structural equivalence*, which is correctly handled.

[W] Due to malfunction of the wiki parser used, this feature had to be been disabled.

[Q] Implemented in QuiKey, see Chapter 5.

4.4 iMapping Related Work

Related work that iMapping is based on, including the history of zooming user interface (ZUI) frameworks has been described in Chapter 2. Work related to QuiKey can be found in Section 5.4. Here, additional approaches and tools are covered that are in some respect comparable to iMapping.

This section is structured into the categories PKM Tools, Zooming Tools, ZUI Frameworks and Other although some tools are hard to categorize or fall on several categories. Figure 4.21 at the end of this section shows a categorized overview of related work discussed here, in Chapter 2 and Section 5.4.

PKM Tools

There is a broad variety of personal knowledge management tools – both research prototypes and industrial ones and they are too many to be listed here. A comprehensive overview on such tools and their history is given by Davies et al. (2005); Davies (2011).

The following few are some of the most closely related to iMapping:

Tinderbox by Mark Bernstein (2003) is a commercial tool and probably the most widely used spatial hypertext editor. It is a mature tool rich in useful features and it is the most similar to iMapping. While it seems to be mainly targeted at supporting authors' writing processes, it is also being used for personal knowledge management. Its structural approach is in many aspects comparable to iMapping – e. g., it is mainly based on freely placeable text-items that can be interlinked and nested into each other and that can have user-defined types of properties. Tinderbox even features so-called "agents", which are persistent queries comparable to query items as explained in Section 4.2.6.

In contrast to iMapping, in Tinderbox, the focus is on one hierarchy level at a time. Nested items are regarded rather as separate sub-maps that need to be expanded one at a time. Also the intended way of use is to keep separate files for separate projects. This differs from the iMapping vision, which is targeted to support a continuously growing personal knowledge repository where everything can be semantically linked to everything else, that is represented on one infinite canvas and where several levels of hierarchy are seen simultaneously and transitioned fluently. For more information about Tinderbox, see `http://www.eastgate.com/Tinderbox/`.

Visual Knowledge Builder (VKB) by Frank Shipman et al. (2002) is a free research tool implementing the pure Spatial Hypertext approach, including nested sub-maps. It also features a spatial parser that is capable of recognizing certain spatial arrangements of items, such as lists and groups. VKB differs from iMapping in several respects – most notably in that iMapping does also feature explicit visible links between items and a zooming facility to ease nested navigation. VKB is available for download, along with further information from http://www.csdl.tamu.edu/vkb/.

Popcorn (Davies et al., 2006b) is an experimental personal knowledge base tool that combines the concept mapping approach with the principle of transclusion (Nelson, 1995) and is in some points quite similar to iMapping. However, it does not use a zooming approach to visually bind single sub-maps together and it does not use semantic technologies.

OneNote is a mature note taking application by Microsoft. It supports the integration of many types of contents and also allows some spatial layouting, however not to the extent of most other visual mapping tools. It also does not seem to allow specifying interrelations between information items. Also, OneNote does not provide an overview perspective or any kind of nesting and zooming. For further information on MS OneNote, see http://office.microsoft.com/en-us/onenote/.

Zooming Tools

Since the iMapping concept has first been published (Haller, 2006; Haller et al., 2006), several other projects also started to take the nesting and zooming approach to the domain of PKM in a wider sense. This is not to say that they have been influenced by the iMapping concept, but that the field of ZUI in PKM is being covered by several independent approaches. These include the ZOIL project and the tools Raskin and Akinyo. In contrast to iMapping, they all provide a ZUI as an additional interface to existing content like files, websites or PIM objects. However, they do not allow interrelating items or building personal knowledge models (formal or informal) in a narrower sense of PKM.

Raskin, inspired by and named after late user interface expert Jeff Raskin, is a zooming file browser for the Mac. Raskin displays the file system tree as nested boxes with preview thumbnails for most file types. For more information about Raskin, see http://www.raskinformac.com/.

Akinyo is based on Adobe Flash[22] and runs in a web browser environment. Like iMapping, it offers recursively nested collections of items that can be assembled to large zoomable maps. Item types can be text notes but also office documents, pictures and even web pages – all of which are represented by zoomable thumbnail previews. In contrast to iMapping, Akinyo does not allow the interlinking of items, neither formal nor informal. It also foregoes a search function. Akinyo's development status is currently declared as beta. For more information, see http://akinyo.com/.

Prezi is a zooming presentation tool also based on Adobe Flash. Like iMapping, it can be used for zooming-based presentations that feature an overview perspective from which details can be zoomed into. While Prezi is more mature than iMapping and supports the use of rich media content, it is not suitable as a PKM tool, since it is not built for maps larger than for a single presentation and editing is optimized for creative use of text and media rather than for efficient information entry and maintenance. For more information on Prezi, see http://prezi.com/.

aiSee is a graph visualization software that is specialized on large graphs with up to hundreds of thousands of nodes. It supports many different graph layout algorithms and also features several techniques to integrate context and details, including overview windows, FEVs and zooming. However, aiSee does not support the interactive authoring of graph structures but focuses on rendering preexisting data. For more information about aiSee, see http://www.aisee.com/.

Jambalaya is a visualization plug-in for the ontology editor Protégé[23]. It uses a ZUI and – among others – a view based on nested boxes to visualize ontologies. Being part of an ontology editor, Jambalaya does not support the use of informal content structures and is not designed for PKM. Like iMapping, Jambalaya is based on Piccolo2D. For more information on Jambalaya, see http://www.thechiselgroup.org/jambalaya.

Google Maps is Google's widely known online geographical mapping application. Though not a PKM application, it has made ZUI available to the wide public and has made the use of the scroll wheel for zooming a de-facto standard in ZUI interaction. Google Maps can be used online at http://maps.google.com/.

[22] http://www.adobe.com/flashplatform/
[23] http://protege.stanford.edu/

ZUI Frameworks

For the implementation of the iMapping tool, the ZUI framework Piccolo2D [24] has been chosen, because, at the time, it was the only mature ZUI framework available, it is open source and based on Java, which most of the Nepomuk software was based on. Piccolo2D has been described in Section 2.4. Other comparable ZUI frameworks include the following, which are also shown in Table 4.4 in comparison.

Table 4.4: Zooming User Interface Frameworks in Comparison

Framework	Status	Language	Open Source
Piccolo2D	mature	Java, C# .NET [a]	✓
ZVTM	beta	Java	✓
ZOIL	alpha	C# .NET	✓
Seadragon	mature	C# .NET	–

[a] The development of the .NET version of Piccolo2D appears to have been abandoned.

ZVTM (Pietriga, 2005), also a Java based open source library that also supports displaying PDF [25] and some SVG [26] content. At least when the implementation work for iMapping began in 2006, ZVTM did not seem stable enough yet.

ZOIL as described by Jetter et al. (2008) is a research project that develops and investigates user interaction techniques in a ZUI environment for personal information management. Among other techniques, it includes semantic zooming and ways to browse external data. The ZOIL software framework is based on C# technology and is hosted open source at `http://zoil.codeplex.com/`.

Seadragon / Deep Zoom – Microsoft's Seadragon [27] technology should be noted, that now powers the *Deep Zoom* [28] facility in Microsoft's internet application framework Silverlight [29] and the impressive ZUI information visualization application Pivot [30], which can be seen in action in Gary Flake's TED conference talk [31]. Seadragon and Silverlight were not considered for iMapping because they are neither open source software nor were they available in 2006.

[24] `http://piccolo2d.org/`
[25] `http://en.wikipedia.org/wiki/Portable_Document_Format`
[26] `http://en.wikipedia.org/wiki/Scalable_Vector_Graphics`
[27] `http://en.wikipedia.org/wiki/Seadragon_Software`
[28] `http://www.microsoft.com/silverlight/deep-zoom/`
[29] `http://en.wikipedia.org/wiki/Silverlight`
[30] `http://en.wikipedia.org/wiki/Microsoft_Live_Labs_Pivot`
[31] `http://ted.com/talks/gary_flake_is_pivot_a_turning_point_for_web_exploration.html`

Other

Treemaps are a visualization technique for quantitative hierarchical data. In a recursive, space-filling way, a rectangle is hierarchically partitioned into smaller nested rectangles in a way that the sizes of the rectangles correspond to the quantitative dimension that is being visualized. This technique is commonly used e. g., to give an overview of the usage of storage space in file systems, where the total size of a directory results from the sizes of its files and subdirectories.[32] While treemaps are also based on nested boxes and are browsed by zooming, the technique significantly differs from iMapping in two ways: (a) it is an information visualization technique (i. e. it is used to display pre-existing structured data and not to interactively build structured knowledge bases) and (b) it is a space filling technique, where existing space is entirely subdivided, leaving no slack space between parent and children items. A comprehensive overview on treemap based research and tools is given by Shneiderman and Plaisant (2009).

A general data model for nested graphs is described by Poulovassilis and Levene (1994).

ZigZag by Ted Nelson (2006) is on the one hand also an interactive visualization system for typed and networked information structures. It is not a knowledge mapping tool since instead of user-defined lay-outs, it uses fixed layout algorithms to display and edit the structured information from different perspectives. On the other hand, ZigZag is also a data format. How it compares to other graph-based data formats has been described by Goulding et al. (2010). The official home page of ZigZag is `http://www.xanadu.com/zigzag/`.

Conclusions

Both PKM and ZUI tools are finding increasing popularity in recent years. And most aspects of iMapping can also be found in at least one other tool. However, bringing together nesting and zooming with graph representations seems to be a novel approach. Applying this approach to offer semantic technologies for PKM in particular fills a hitherto unoccupied gap, as can be seen in Figure 4.21.

[32] Treemap tools for file systems are e. g., Konqueror for KDE/Linux (`http://www.konqueror.org/`), Disk Inventory X for the Mac (`http://www.derlien.com/`) or WinDirStat for Windows (`http://windirstat.info/`).

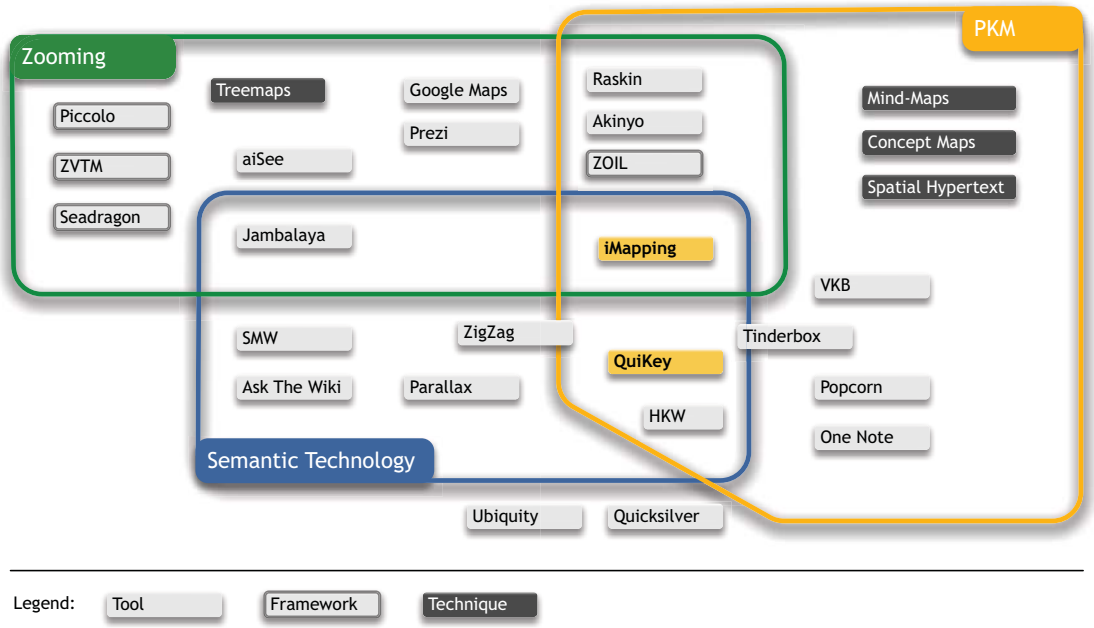

FIGURE 4.21: Related Work: Tools, techniques and frameworks in relation to iMapping and QuiKey. Tools that are not described in this section are covered by QuiKey's related work in Section 5.4. Items that overlap category borders could not be clearly classified, Tinderbox, e. g., does not support formal semantic modeling and reasoning, but does offer persistent structured queries over user-defined relation types.

Chapter 5

QuiKey

This chapter introduces QuiKey, a supplementary tool that complements the visual iMapping approach. Like with iMapping, QuiKey is the name of the interaction *concept* as well as of the first *tool* that implements this concept.

QuiKey offers interactive fine-grained access to structured information sources in a light weight text based user interface. It is designed to be highly interaction efficient for searching, browsing and authoring semantic knowledge bases as well as incrementally constructing complex queries.

This chapter introduces QuiKey (Section 5.1), describes the underlying interaction design (Section 5.2) and briefly its current implementation (Section 5.3). QuiKey as an interaction concept has first been introduced by Haller (2008a,b). An empirical evaluation of the actual QuiKey tool, comprising a comparative GOMS analysis and a user study, which confirms interaction efficiency has been published by Haller (2010c) and is reported in the evaluation chapter in Section 6.2.

5.1 Introduction to QuiKey

While iMapping focuses on overview and intuitive use through a visual approach, it can also be a burden when every item needs to be deliberately positioned and when items need to be located before they can be edited or referred to. Hence, it is desirable to complement such a visual tool by techniques that provide map-independent access to the same structured information. QuiKey is such a tool: It is specifically designed to fill the above-mentioned gap.

QuiKey is a kind of smart semantic command-line that allows to browse, query and author semantic knowledge bases in a step-by-step manner. It combines ideas of simple interaction techniques like auto-completion, command interpreters and faceted browsing to form a generic, extensible user interface for graph structured knowledge bases. Among other things, QuiKey offers an incremental search function that allows jumping to any item in an iMap with just a few keystrokes.

QuiKey's main design goals are (a) efficient interaction and (b) avoidance of errors – both syntax errors and typographical errors. It is targeted at covering the following use cases:

- Text search (finding items that contain certain text strings)
- Targeted search of linked information (e. g., finding someone's phone number or someone's friend's e-mail address)
- Information entry (e. g., adding a new contact and linking her to an existing project)
- Set-based browsing (e. g., going from *all projects financed by the EU* to *all members of these projects*)
- Formulating simple queries (e. g., a list of people that live in Portugal)
- Incrementally constructing complex queries from simple ones (e. g., a list of people that live in Portugal and are members of projects financed by the EU).

QuiKey is greatly inspired by *quicksilver*. Quicksilver [1] by Nicholas Jitkoff is a kind of advanced application launcher for the Mac that has gained a lot of popularity due to its versatility and efficiency: With very few keystrokes, quicksilver can open files and applications and trigger a large variety of common actions, not only on any files but also on specific information objects: Depending on the plug-ins installed, it can e. g., manage play-lists in iTunes, send files via e-mail or dial a contact's phone number.

QuiKey is the attempt to adapt and transfer quicksilver's highly efficient interaction paradigm to the semantic desktop. Both tools allow browsing structured information models. However, while Quicksilver is for finding and acting upon certain desktop objects, QuiKey is a generic authoring and query tool for graph-based knowledge bases.

[1] http://qsapp.com/

5.2 Design of QuiKey

The primary design goals of QuiKey were to make interaction as efficient as possible, and to avoid typing- and syntax mistakes. Interaction efficiency means that for a given goal, the interaction required should be reduced to a minimum. While avoiding of errors is generally desirable, it is especially important for semantic systems, where even minimal errors may lead to complete failure. While interaction efficiency is generally desirable (Raskin, 2000), it is especially crucial for mobile devices where typing is usually cumbersome. This is in line with the secondary design goal to have a minimalistic screen design, which is also suitable for constrained screen space like in mobile devices.

The following principles have guided the interaction design of QuiKey:

- Everything can be done with the keyboard alone. While mouse interaction is always possible, it is never required. This avoids unnecessary costly switches between input media, also referred to as *homing* in user interaction literature (Card et al., 1983).
- There is only one mode for everything. Searching, browsing, authoring and querying is all done in the same consistent way of interaction. Modelessness is generally regarded desirable in interaction design (Raskin, 2000; Apple Inc., 2008; Nielsen, 1994).
- Short feedback cycles to reduce error-proneness. All parts of a complex interaction (items, relations, query operators and such) are implicitly or explicitly selected from lists of existing things, and the structure of the current operation is reflected visually. Like this, misspellings or syntax errors are greatly avoided and if they occur, they are easy to notice. This is in line with (Nielsen, 1994, p. 154) who found that "Seeing/pointing vs. remembering/typing" and "Feedback timely and accurate" are among the top three "Heuristics to explain the serious usability problems".

QuiKey is organized around the notion of *parts*. A part can be an existing item, a relation, a new text string or a command. Depending on the types and order of the parts entered, it is decided what action to take. The following are the main functionalities of QuiKey. As explained below, they are tightly interwoven.

5.2.1 Text Search

The simplest functionality, which is also usable without any understanding of the structure of semantic knowledge models, is full text search. While search terms are entered, a list of ranked results is displayed based on a set of matching rules explained below. The

best hit is pre-selected, so that any item can be addressed by mere text entry without the need to use any special keys for syntax elements or selection. Of course, other items can also be selected via arrow keys or mouse. In QuiKey, text search results are ranked according to the following ordered list of **ranking rules**:

1. Matching the complete, coherent search string is better than matching separate search words.
2. Matching full words is better than matching prefixes only.
3. Matching prefixes of words is better than matching arbitrary substrings only.
4. Matching the beginning of an item is better than matching it anywhere else.
5. Matching several search words in the right order is better than random order.
6. The shorter the item, the better.

These rules served as a basis to put together the following ordered list of text **matching patterns**:

1. Perfect match between search string and item
2. Full search string matches full words at the beginning of the item
3. Full search string matches at the beginning of the item
4. Full search string matches full words
5. Full search string matches at the beginning of a word
6. Separate search words match full words in right order and beginning of item
7. Separate search words match full words in the right order
8. Separate search words match full words in any order
9. Separate search words match prefixes in right order and beginning of item
10. Separate search words match prefixes in the right order
11. Separate search words match prefixes in any order
12. Separate search words match substrings in right order and beginning of item
13. Separate search words match substrings in the right order
14. Separate search words match substrings in any order

Each line of the example screen shot in Figure 5.1 corresponds to one of these matching patterns.

Most other autocomplete implementations do not match on infixes, which e. g., would not match the string *genre* to relation names like *has_genre* or *HasGenre*, both common ways to name relation types in more technically oriented communities. Some other implementations only match the letters of the search string in the exact order. Quicksilver,

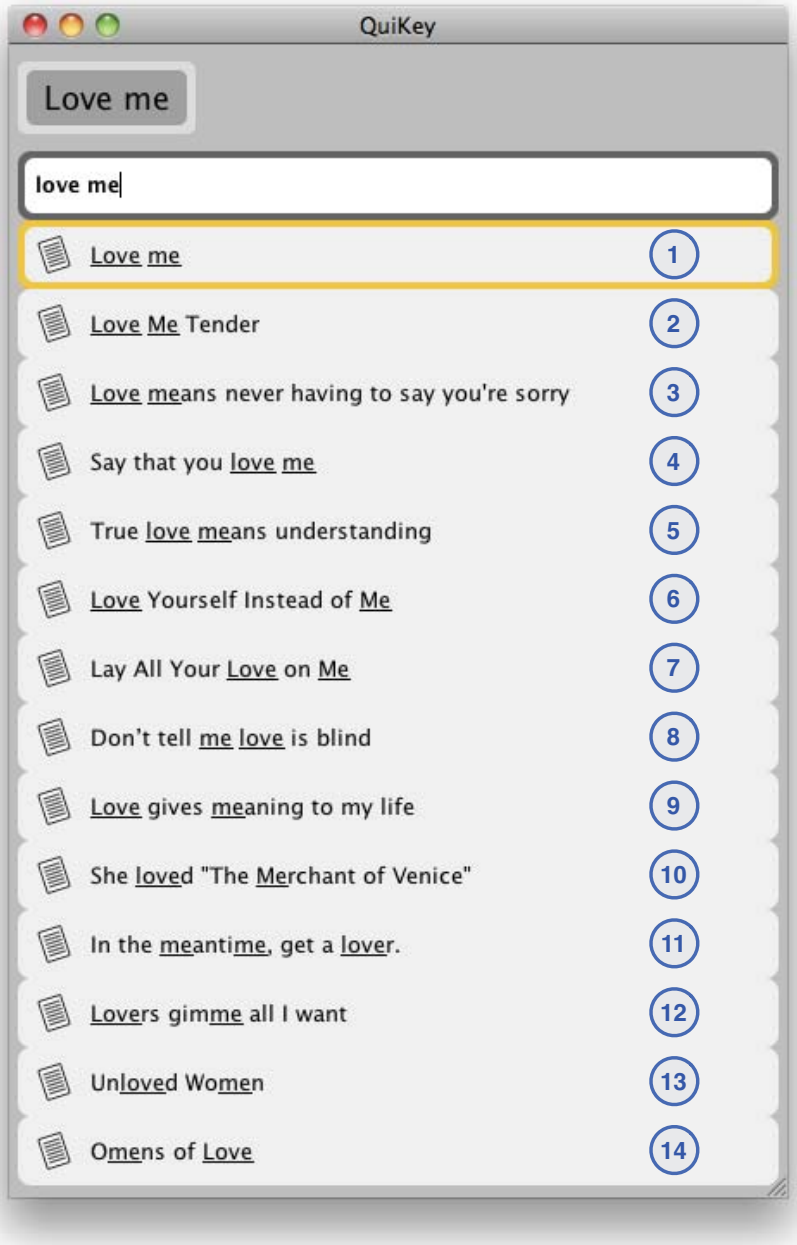

FIGURE 5.1: QuiKey Text Search (screen shot): Items that match the search string *love me*. Each item shown here is an example for one of the 14 matching patterns.

e. g., does not distinguish separate search words and will e. g., not match *Ontology Web Language* to *Web Ontology Language*.

This text search functionality is used throughout QuiKey wherever an item, relation type or command needs to be identified and it is one of the factors that make QuiKey efficient. For example, to bring the item *Michael Jackson* to the top position out of 43 270 named entities of the data set used in the evaluation (described in Section 6.2), it suffices to type `m jac`[2]. Note that most of the following examples of user input are spelled out for

[2]This `typewriter font` is used to denote literal keyboard input.

the sake of clarity, although the actual input needed is usually much shorter due to the text matching function explained above.

5.2.2 Browsing

Starting with an item selected by text search, a user can navigate the knowledge base through its graph structure hop by hop by hitting the tab key.

When an item is jumped to, all corresponding statements (triples) about this item are displayed – sorted by relation types. When an item and a relation are selected (e. g., Madonna→has album) [3], all statements matching this pattern are displayed as in Figure 5.2. From there, any statement can be selected to jump to the target object of the statement (in our example a particular album). Like this, a user can browse from entity to entity with the pattern *item→relation→item→relation→item ...* (e. g., `Madonna→has album→Like a Virgin→has genre→Pop music ...`).

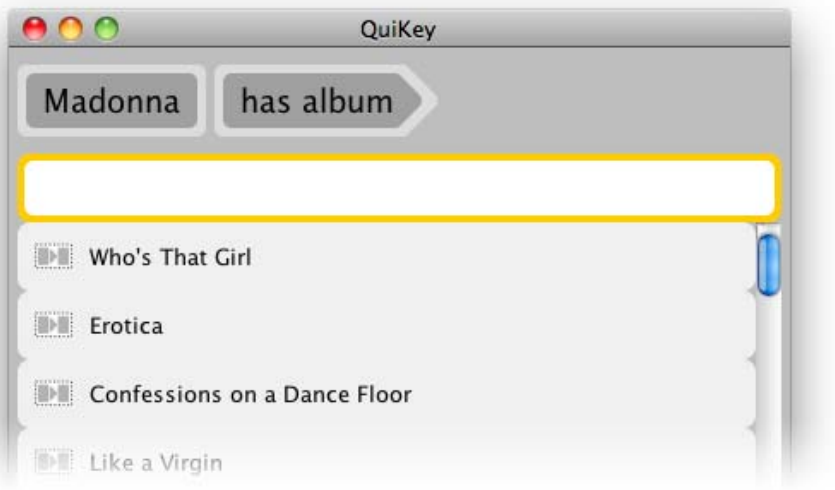

FIGURE 5.2: Browsing: A list of all albums of Madonna is shown after selecting the item *Madonna* and the relation type *has album*.

Set Based Browsing is a term coined by Huynh and Karger (2009). It denotes a way of browsing, where one moves from a set of items to a related set of items instead of stepping from one item to the next as in classical browsing. In QuiKey, this can be done by using the pattern *item→relation→relation ...* with any number of consecutive relation types. The result is a set of all items that have the specified relation to any of the items in the set before.

For example, `Madonna→has album→has genre` returns a list of all genres of all albums of Madonna, as shown in Figure 5.3.

[3] The arrow symbol → is used here to separate input parts, which is done with the tab key in QuiKey.

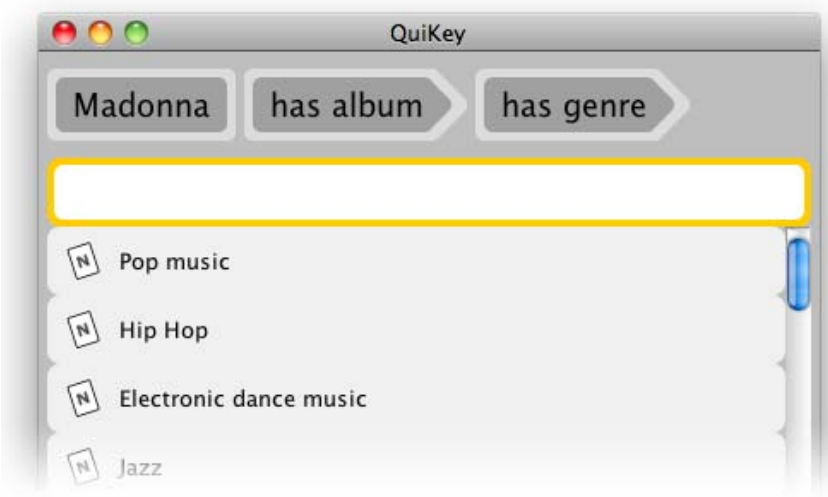

FIGURE 5.3: Set-based Browsing: Going from the above set of all albums of Madonna to all genres of these albums.

5.2.3 Queries

Constructing complex, possibly nested, queries over structured or semi-structured data with a text-based query language, like querying an RDF graph (Manola and Miller, 2004) with the query language SPARQL (Prud'hommeaux and Seaborne, 2008) or formulating ASK Queries in Semantic MediaWiki (Krötzsch et al., 2007) is difficult: Every slight syntax error or misspelling makes the whole query fail or (worse) return unintended results. And there is usually no feedback as to where the error lies because complex queries are formulated and evaluated as a whole only. QuiKey tackles these two common problems:

Misspellings and syntax errors are largely avoided because instead of requiring the user to write a whole query in a complicated syntax, in QuiKey, simple queries are constructed by browsing interactively, selecting from existing items and without the need of syntactical characters.

Queries in QuiKey are such that they always yield one plain result set (as opposed to tables or graphs).

In QuiKey, like in faceted browsing, the border between browsing and query construction is blurred. In fact, the two above browsing examples already form semantic queries.

Queries that can be formulated by browsing as explained above can be persisted by simply appending a name under which the query should be saved as another *part* – prefixed by a question mark. For example, coming from the situation in Figure 5.2, `Madonna→has album→?Madonna's albums` saves a new query item named *Madonna's albums* that represents all of Madonna's albums.

Other query patterns that cannot be constructed by browsing, can be formulated like explained below. Three different kinds of queries are possible:

Basic Queries are simple triple patterns of the *subject – predicate – object* form, where one of the three positions is undefined. The result set consists of all items (or relation types) that match the undefined part. So, instead of the example above, the same result can be obtained with the pattern `?Madonna's Albums→is album of→Madonna`. This is needed especially when QuiKey's interaction concept is used on a data model like RDF, which does not support the use of inverse relation types.

Chain Queries are the equivalent to set based browsing, where relation types are *chained* successively as defined above in Section 5.2.2. Coming from the example in Figure 5.3, a query name can be appended like above to save the query. Similarly, the reverse pattern `?Madonna's Genres→is genre of→is album of→Madonna` saves a query item representing all genres of all albums of Madonna.

Complex Queries can be constructed by combining existing saved queries with the logical operators *and* and *or* like it is shown in Figure 5.4. Like any other set of items obtained by browsing, also complex queries can be saved by appending a query name prefixed by a question mark. Thereby, more complex queries can be constructed step by step out of existing ones.

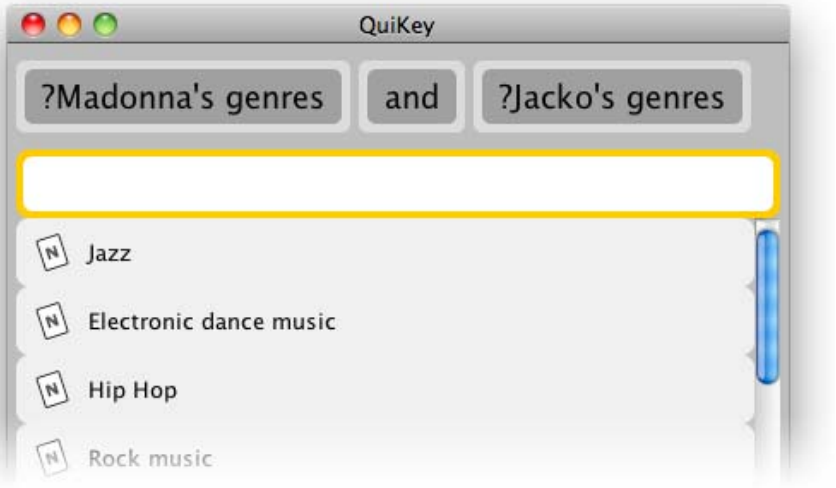

FIGURE 5.4: Complex Queries: Two saved queries are intersected to yield a list of all genres that Madonna's albums have in common with Michael Jackson's albums.

In all of the browsing and query patterns described so far, it is also possible to replace any explicit item by a *set* of items. Sets of items can either be defined intensionally by an existing query item or extensionally by an explicit list of items (separated by *ctrl+,*). In the resulting queries, the elements of these sets are interpreted disjunctively, as illustrated in the following two examples:

The pattern `Erotica, Like a Virgin, Ray of Light→has genre→` will yield a list of all genres of either of the three albums listed, i. e. the genres of (Erotica OR Like a Virgin OR Ray of Light), like it is shown in Figure 5.5.

If the query item *Madonna's Genres* already exists (as defined above), the pattern `Madonna's Genres→is genre of→is album of→?Artists like Madonna` will store a new query item *Artists like Madonna* that represents all artists that have a album that has a genres that is also the genre of at least one of Madonna's albums.

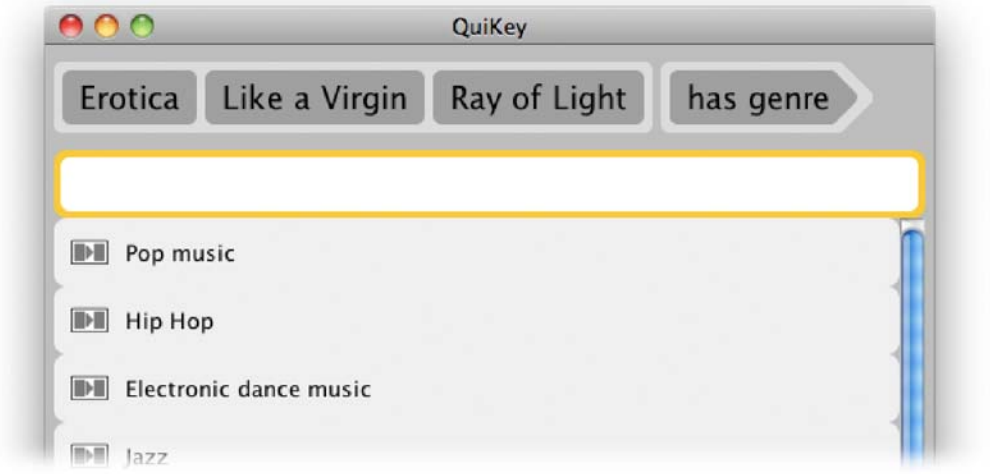

FIGURE 5.5: Defining Sets Extensionally: The items *Erotica*, *Like a Virgin*and *Ray of Light* are treated as a set. All genres of any of the items are displayed.

While the interaction concept of QuiKey could be extended to comprise a wider range of query constructs, the expressivity of the tool so far is limited to the patterns defined above.

In summary, QuiKey allows to define sets of items through

- Basic queries (triple patterns: $S, P, ?X$; $?X, P, O$ or $S, ?X, O$)
- Chain queries ($?X, P_1, P_2, \ldots, P_n, ?X$ or $S, P_1, P_2, \ldots, P_n, O$)
- Combining queries conjunctively (Q_1, AND, Q_2)
- Combining queries disjunctively (Q_1, OR, Q_2)

Here, S and O denote items at *subject* and *object* positions respectively; P denotes relation types at the *predicate* position. ?X stands for the name of a the query item to be

constructed (which also acts as a placeholder in these S–P–O patterns), and Q stands for an existing query item.

In the terms of SPARQL, the expressivity of QuiKey can be described as *UNIONs of acyclic Basic Graph Patterns with only one output variable.*

In EBNF [4], the set of possible queries can be defined as follows, where, in the resulting query definitions, I is an item, R is a relation type, and Q is a placeholder for the undefined position (like $?X$ in the above examples). $\cup$ and $\cap$ have their common set-theoretic meaning (The meaning of the resulting patterns is explained below.):

$$
\begin{aligned}
\mathrm{V} &= \{q, i, r, s\} \\
\mathrm{T} &= \{I, R, Q, \cup, \cap\} \\
\mathrm{S} &= \{q\} \\
\mathrm{P} &= \{q \mapsto (q \cup q), \\
&\quad\ q \mapsto (q \cap q), \\
&\quad\ q \mapsto Qri, \\
&\quad\ q \mapsto irQ, \\
&\quad\ q \mapsto iQi, \\
&\quad\ r \mapsto R, \\
&\quad\ r \mapsto rR \\
&\quad\ i \mapsto I, \\
&\quad\ i \mapsto (q), \\
&\quad\ i \mapsto s, \\
&\quad\ s \mapsto I \\
&\quad\ s \mapsto I \vee s\}
\end{aligned}
$$

Here, conjunctions and disjunctions of existing queries are denoted by $q \cup q$ and $q \cap q$. Basic queries where the subject position is asked for are denoted by QRi, for the object position it is iRQ. iQi denotes a query where the relation types of all statements between the two items (subject and object) is asked for. Any position of r can be filled either by a single relation type R for basic queries or by any number of repeated Rs for chain queries. All positions of i can be filled either by a single item I, an extensionally defined set of items s or by another query q.

[4] http://en.wikipedia.org/wiki/Extended_Backus-Naur_Form

5.2.4 Authoring

With QuiKey, knowledge bases can be altered in the following different ways:

Adding Items To add a new text item to the knowledge base, it is enough to just type the text and press enter. The newly created item is automatically saved in the CDS back-end. In order to be able to access items created in QuiKey, they also get created in the associated iMap inside a special *inbox* item. From there, they can be moved to any place in the map when needed.

Adding Statements To make statements about existing items, the statement can be entered in a subject→predicate→object pattern, separated by tab-keys. So, for example, `Michael Jackson→has album→This Is It` adds this statement to the knowledge model. Only that the user does not even have to type in the whole labels because parts that are already known can be chosen from the suggestions list while typing in auto-completion manner. So, for this example, it is actually enough to type `m jac→h albu→This Is It`.

Adding Items and Statements Together If not all three parts in such a statement are known objects, the respective items or relation types are also added to the knowledge base. So, in the above example, if *This Is It* is not a known item, it is directly created, together with the statement that it is an album of Michael Jackson. To avoid accidentally creating new items instead of reusing existing ones, the respective part is highlighted in yellow during interaction, i. e. before the action is executed. This can bee seen in Figure 5.6.

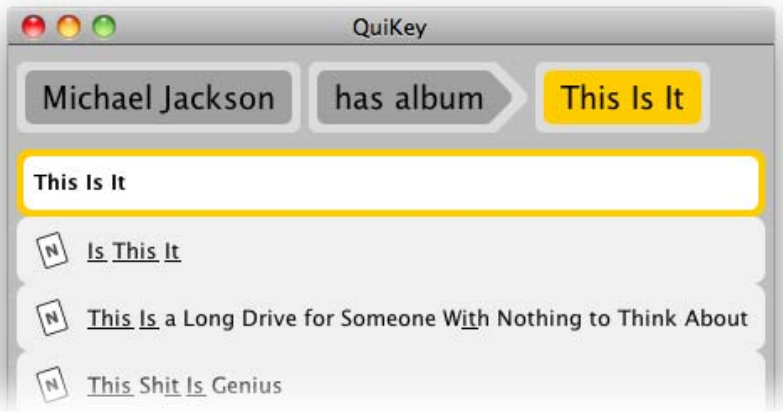

FIGURE 5.6: Adding Several Objects in QuiKey: A new *item* "This Is It" is about to be created, together with the *statement* that it is an album of Michael Jackson.

Furthermore, certain actions like editing and removing existing items can be done with *commands* like `Michael Jackson→rename to→Michael Joseph Jackson`.

5.2.5 Conclusion

With the actions described above, a knowledge graph can be woven in separate simple steps in an ad-hoc fashion. Creating and interlinking items and relation types is done in the same interaction efficient manner as searching and browsing the graph. Simple semantic queries can easily be constructed and saved, and complex semantic queries can be constructed by combining existing ones, simple or complex.

All this can be done without the need to display, browse or modify the visual graph of an iMap, although the same knowledge base can be used and edited with both tools simultaneously.

Apart from requiring the user to think in triple patterns, which is a prerequisite for dealing with graph based knowledge bases, cognitive overhead and especially interaction overhead are reduced to a minimum. And even without that, QuiKey can simply be used as a powerful text search tool to jump to any iMap item directly.

5.3 Implementation of QuiKey

This section briefly describes how QuiKey is integrated with iMapping and touches some implementation issues regarding performance and scalability.

Like iMapping, QuiKey was initially developed as part of the semantic desktop project Nepomuk[5]. As envisioned by Wiil (2005), with CDS (introduced in Section 2.5.2), iMapping and QuiKey, although technically independent tools, use a common semantic back-end. Thereby both tools can access the same data model simultaneously. The only point where both tools interact with each other, is that QuiKey is deployed to complement iMapping. It can be invoked from the iMapping application with the keyboard shortcut *Ctrl+F*, which is the standard shortcut for search features on all common platforms. Conversely, QuiKey can trigger iMapping to zoom to an item that has been picked in QuiKey. However, QuiKey can also be used as a stand alone tool to work on CDS knowledge bases independently.

[5] http://nepomuk.semanticdesktop.org

Like iMapping, QuiKey's implementation is based on Java. Its UI is based on Swing. QuiKey is available online as open source code [6] and in compiled form [7].

For QuiKey, performance is critical. Interaction efficiency, QuiKey's main design goal, is not worth much when system response times are too long. Performance is a challenge in two areas: a) in the auto-completing text search and b) for query answering.

Auto-Completing Text Search The CDS back-end, which stores all text content, offers an in-memory inverted sub-string index, which serves QuiKey's text search functionality. The preliminary set of matching items is then ranked by QuiKey according to the matching rules described in Section 5.2.1.

For performance reasons, the search depth of QuiKey's matching rules can be adjusted by the user. However, even with all rules enabled, text searches with up to 4 search words are executed well below one second on a knowledge base with 43 270 items on a 2.4 GHz Intel Core 2 Duo processor – with up to 3 search words, results are usually perceived as instantaneous.

Response times deteriorate dramatically with 5 or more search words due to the combinatorial explosion of search word combinations: To execute the matching patterns described in Section 5.2.1, the current implementation uses regular expressions [8]. Regular expressions, however, do not allow to match several search strings in arbitrary order (which is needed for the matching patterns 8, 11 and 14). Therefore, for each of these three patterns, a number of $x!$ regular expressions is first generated and then executed (x being the number of separate search words). However, there is room for future optimization: For matching patterns with arbitrary order, instead of using regular expressions, a programmatic implementation should bring significant gains in performance, if it filters down the preliminary set sequentially for all search words.

A more global approach to optimize system response times is to limit the number of matching patterns executed to only the top few patterns that are needed to obtain the top-k number of results, where k is the number of items needed to fill the current window height. Like this, in many cases, the lower ranking patterns, that are also less efficient, only get executed if the result list is yet too short.

A complementary approach to cut down on response times is to let the text-matching run in a separate thread, which terminates as soon as either the top k results are reached *or* as soon as new user input is received. This avoids spending time on obsolete queries.

[6] http://svn.imapping.info/info.quikey.lab/
[7] http://mvn.imapping.info/quikey/
[8] http://en.wikipedia.org/wiki/Regular_expression

With regard to software architecture, this multi-threaded design is more laborious and so far has not been needed for the model sizes and use cases where QuiKey was deployed.

Query Answering Although the general interaction concept of QuiKey could also be applied to more expressive query languages, the current tool only implements the following patterns, as described in Section 5.2:

- Basic queries
- Chain queries
- Combining queries conjunctively
- Combining queries disjunctively

The processing of semantic queries, which includes subclass, subrelation and inverse relation reasoning, is completely done by the CDS reasoning engine of the back end and thus not subject to the implementation of QuiKey. For all observed uses of QuiKey, so far the also semantic query response times have been well below one second.

5.4 QuiKey Related Work

Quicksilver, which QuiKey is mainly inspired from, has been described above in Section 5.1. This section describes additional related work. However, apart from *Ubiquity* and *Ask The Wiki*, these works are not related to QuiKey in the way that they are conceptually similar approaches. Instead, they are described here because they have been referred to in various sections of this thesis, e. g., as comparisons for the evaluation of QuiKey, which is detailed in Section 6.2.

Ubiquity Ubiquity has been an experimental user interface realized as a web browser plug-in by Aza Raskin at Mozilla Labs. It uses a language-based command-line interface to act upon web content. Ubiquity allows to locally alter web pages, create instant mashups, send e-mails, and so on. While it also uses an auto-completing command interface to navigate and author content structures, it is targeted at handling web content and using web services rather than cohesive knowledge models. For more information on Ubiquity, see `http://mozillalabs.com/ubiquity/`.

Parallax by David Huynh and Karger (2009) is probably the most convenient user interface to date to explore large amounts of structured data. Parallax is an experimental front end to Freebase,[9] a website that offers a large amount of open structured data. A video[10] about parallax nicely explains the benefits of semantic search in general and parallax in particular. It features the notion of set-based browsing, where navigation takes place from one set of things to another related set of things (e. g., from Michael Jackson's albums to their genres). It partly addresses the same use cases as QuiKey (querying large graph based knowledge bases through navigation), and features a visually much richer user interface (many navigation options per view, picture content, thumbnail previews, etc.). While it may be more intuitive, Parallax is also more restricted in the structure of the queries (e. g., intersection queries do not seem possible) and it does not allow authoring the underlying knowledge base or saving complex queries. QuiKey has been compared to Parallax in the evaluation study detailed in Section 6.2. Parallax can be used online at `http://www.freebase.com/labs/parallax/`.

Semantic MediaWiki (SMW) by Markus Krötzsch et al. (2007) is an extension to MediaWiki[11], originally targeted at enriching Wikipedia[12] with semantically defined content. MediaWiki is the software that Wikipedia is based on. SMW introduces into MediaWiki the functionality of semantically defining article contents like links and data. It also offers the possibility to store persistent *inline queries*, that are displayed as embedded result lists and tables. SMW is one of the most widely used tools for semantic modeling and it powers hundreds of websites in a multitude of domains.[13] SMW+[14] is a set of extensions for SMW targeted at facilitating processes of editing, querying and data maintenance. QuiKey has been compared to SMW in the evaluation study detailed in Section 6.2. Both iMapping and QuiKey have been compared to SMW+ in an evaluation study by Völkel (2010), which is briefly summarized at the beginning of Chapter 6. For more information on Semantic MediaWiki see `http://semantic-mediawiki.org/`.

Ask The Wiki developed by Thanh Duc Tran and Daniel Herzig (Haase et al., 2009), is an extension to Semantic MediaWiki (Krötzsch et al., 2007). It adds a search interface that will take plain keywords as in classical keyword search and tries to interpret them as structured queries according to the underlying schema and data graphs. Ask The Wiki then presents the user with a choice of structured queries using the given keywords. While

[9] `http://www.freebase.com/`
[10] `http://vimeo.com/1513562`
[11] `http://www.mediawiki.org/`
[12] `http://www.wikipedia.org/`
[13] `http://semantic-mediawiki.org/wiki/Sites_using_Semantic_MediaWiki`
[14] `http://wiki.ontoprise.de/`

Ask The Wiki also constructs semantic queries without the use of syntax characters, it differs from QuiKey in several ways:

1. Query construction is not interactive but post-hoc – the user sees the interpretation of his input only after it is submitted. This makes errors more probable.
2. There is no definitive way to formulate a query for a given information need. The user has to rely on the system to correctly guess what he means. While for novice users this may allow intuitive use, expert users cannot formulate a precise query.
3. Ask The Wiki is a query interface only. It does not allow browsing or editing content or saving queries.

A demo of Ask the Wiki is available online (for the Firefox browser only) at
http://semanticweb.org/wiki/Special:ATWSpecialSearch.

HKW (Hypertext Knowledge Workbench) by Max Völkel (2010) was a browser based reference implementation of a generic CDS editor. Like iMapping and QuiKey, it has been developed in the Nepomuk project. Other than iMapping and QuiKey, it allowed explore and edit every aspect of CDS, including e. g., statements that referred to other statements etc. HKW has been compared to SMW+, iMapping and QuiKey in an evaluation study by Völkel (2010), which is shortly outlined at the beginning of Chapter 6. For more information on HKW, see
http://semanticweb.org/wiki/Hypertext_Knowledge_Workbench.

Chapter 6

Evaluation

Both iMapping and QuiKey have been evaluated in several ways: During the development process, repeated formative evaluations (Scriven, 1991) have guided design decisions for both tools. After the most important of the hereby gathered issues had been tackled, both tools were evaluated in summative ways. This chapter describes several studies targeted at evaluating different aspects of the usability of iMapping and QuiKey:

First, the general usability of iMapping was checked against a state of the art mind-mapping application in a comparative user study reported in Section 6.1.

For QuiKey, first it has been shown that, for the intended types of tasks, significantly less interaction steps are needed than for comparable tools. This has been done by means of a theoretical (GOMS) analysis. Secondly, a user study has confirmed that users actually do use QuiKey mostly in such optimal ways and that it can in fact easily be used for both authoring, browsing and querying a CDS knowledge base. This two-part study is reported in Section 6.2.

In order to test the overall usability of the two tools with regards to scalability, and to justify their coexistence, they have been evaluated in a long term user study involving very large maps, which is reported in Section 6.3. An additional user study comparing iMapping and QuiKey to other semantic modeling tools is also briefly summarized there in Section 6.3.5.

Lastly, in order to gain insights on actual use, user-made iMaps were collected in a contest. Statistics about these iMaps are reported in Section 6.4, a selection of these iMaps is included in Appendix A.

6.1 iMapping – Comparative Evaluation

With the iMapping prototype from early 2009, a comparative evaluation study was carried out in order to specifically test the suitability of the nesting and zooming approach for personal knowledge management tasks on a larger scale, because this seems to be the biggest difference to the classical approaches discussed in Section 2.3. For that purpose, the iMapping prototype was compared with the then current version 7 of MindManager [1], the most wide-spread state-of-the-art mind-mapping application.

6.1.1 Method

In order to test reasonable map sizes without requiring several hours per session, participants were presented with pre-filled maps, one in each of the two applications. Both maps contained the same content in the same structure: A small biological taxonomy – once as a classical mind-map (see Figure 6.1) and once as an iMap (see Figure 6.2). Because the mind-map paradigm is not designed for large maps, a limited map size of only 100 nodes was chosen that was still easily manageable in MindManager as well as a limited number of eight cross-links to avoid unfair visual complexity in the mind-map paradigm that is not designed for highly interrelated structures.

Each participant of the comparative user study used both tools. In order to avoid sequence effects, the order of the tools was balanced out by alternating it. Each participant went through the following process:

1. Short introduction to the first tool, covering basic functionalities and interactions needed for the tasks. Participants were asked to try out the interactions and explore the map and the tool until they felt confident in using it.
2. Participants were asked to carry out a set of tasks. The time taken for each task was measured for comparison. The tasks were
   - adding certain new items to specific places in the structure
   - rearranging items in the hierarchy
   - interlinking existing items
   - finding remote items by mere visual search and reading out all their connections
3. In the end, without being able to look at the map, subjects where asked a question like “How many categories of plants were there in the map, and which ones were they?”. The percentage of right answers was recorded.

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

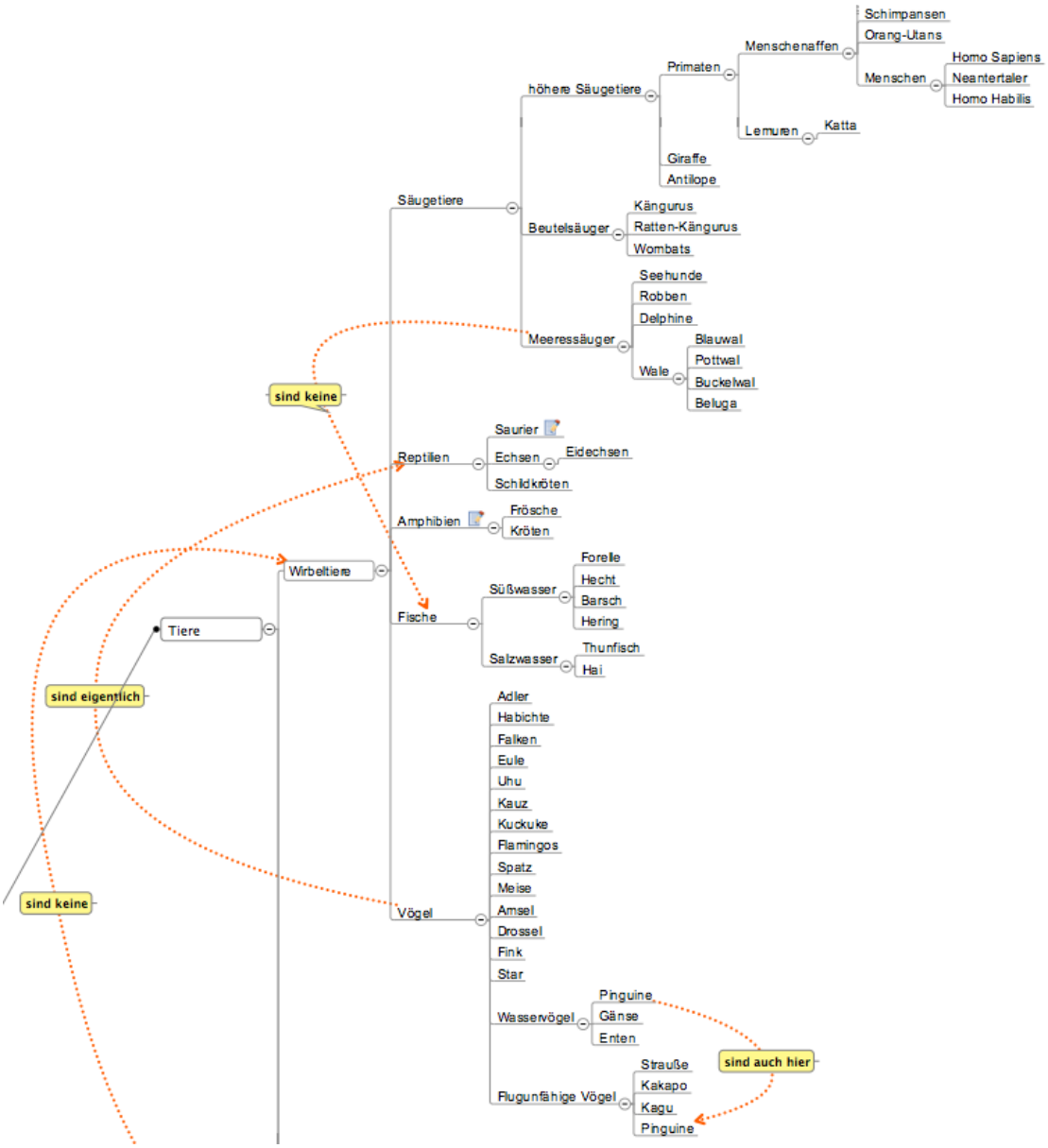

FIGURE 6.1: Mind-Map Used for Comparative Evaluation: Part of the mind-map containing a small biological taxonomy in MindManager).

4. Introduction to the second tool (as above).

5. Second set of tasks (as above).

6. After that, participants were asked for their spontaneous impressions as well as for explicit ratings of the tools in several aspects on a scale from 1 to 5 (German school grades).

Although structurally equivalent, the two sets of tasks differed slightly to avoid content-related learning effects. To also avoid effects of sequence, the order of the tools was

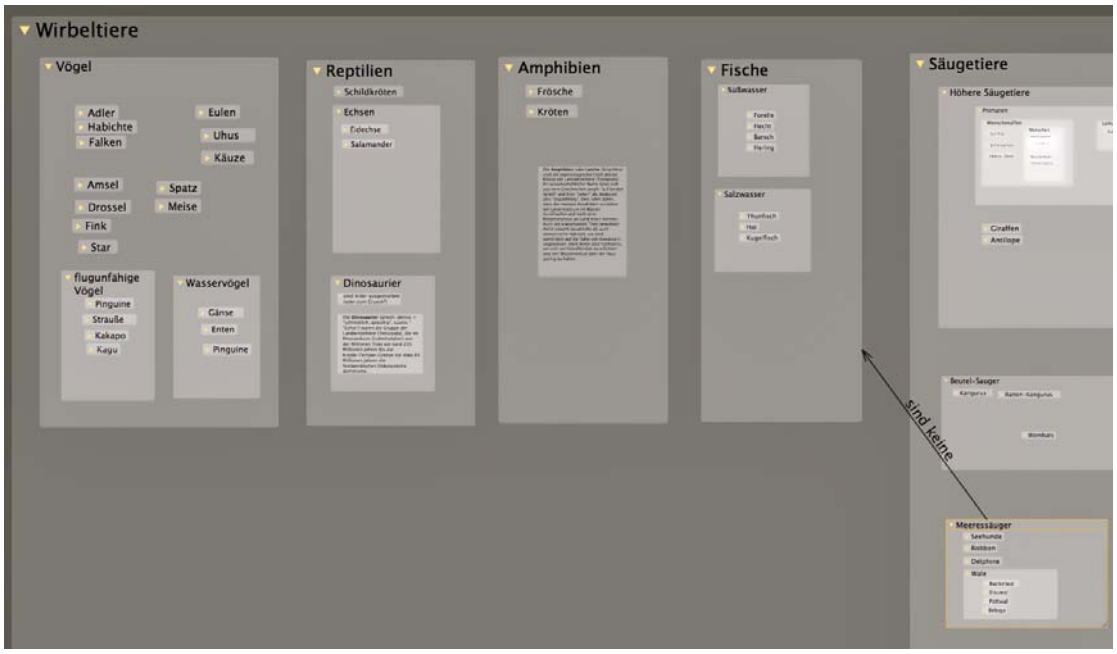

FIGURE 6.2: iMap Used for Comparative Evaluation: Part of the iMap used for evaluation containing the same taxonomy.

balanced out, so each of them came first for every second participant. To avoid cross-over effects of tasks with tools, the pairing of task sets with tools was also balanced out. The original list of tasks can be found in Appendix D.1.

The sample of participants consisted of 12 rather heterogeneous individuals (6 male and 6 female), aged 23 – 64. Their primary occupations were: painter, journalist, interactive media engineer, photo model, students (English and biology, computer science, psychology, economical engineering), PhD students (health-care, economics, logistics, semantic web).

The average time each user needed for the different kinds of tasks and the ratings given by the users were compared between the two tools and tested for statistical significance with a dependent t-test for paired samples.

6.1.2 Results

There were no effects of gender or age on any of the measures and there was no statistically significant difference in the interaction times measured. However, as Table 6.1 shows, there was a clear tendency of all editing tasks (adding, rearranging and interlinking of content) being carried out slightly faster in the iMapping application. This could be explained by users' comments that in the iMap, the topology was clearer and easier to grasp, which is also reflected in the ratings (see Table 6.2).

Drawing links between items was faster in MindManager. However, this finding is somewhat distorted by the fact that participants were told not to try and label these links in MindManager, since it is rather complicated there. In iMapping, they all did put labels in the links because a popup suggested relation types right after dragging the link. Another reason is that in the prototype tested, creating a link was only possible by holding down the alt-key, which was hard to remember and led to mistakes. [2]

When people were challenged to find information about an item outside their current view, this was especially hard because these items were not where most users expected them. Because it was the visual concept that should be tested, participants were asked not to use the search function. Instead, they had to rely on mere visual search. In the iMap, it took some participants over a minute to spot the target item, resulting in a longer average search time for iMapping. This can be explained by the fact that with the given size of the two maps, in MindManager, which by default does not decrease font size for items deep down the hierarchy, there was a zooming level, where the whole map was still readable with very little scrolling needed. This made it easier to overlook all the content in the mind-map, while in the iMap, deeply nested items were too small to read from an overview perspective. In order to remain fair on the side of MindManager, which is not designed for very large maps that vastly exceed the size of one screen, only this moderate map size was used for comparison. When used with larger maps, the mind-map should become harder to overview, while in the iMap it should not get worse, since deeper hierarchies would be hidden from the overview in the tiny areas and thus not increase visual complexity for a given zoom-level. While this claim remains to be proven by further investigation, this kind of visual search can be avoided in real usage, since QuiKey offers an efficient search functionality, that allows to find any item fast without scanning the map. While not systematically tested in this study, those users that asked for a search function were presented with QuiKey's basic search function and uniformly appeared very satisfied by it.

When asked to remember the number and names of certain items (how many categories of plants and categories of vertebrates there were in the map), recall was nearly twice as good from the iMap. This is probably due to the more characteristic layout allowed by iMapping compared to MindManager, where all sibling items appear as a vertical list. This would also explain why the difference in remembering *what* these actual categories were was actually much smaller.

The time until users said they were familiar enough with the tools and the content to begin with the tasks was longer for iMapping. It could be argued that this longer practice

[2] Based on this insight, to make linking more intuitive, it is planned to introduce a linking button in future versions instead of requiring to use modifier keys for linking.

TABLE 6.1: iMapping vs. MindManager – Interaction Times in Comparison: Mean times and percentages for groups of tasks in comparison (rounded values) – None of them are statistically significant even at the 5 % level.

	MindManager	**iMapping**	**difference**
Practice time	5.8 min	8.6 min	2.8 min
Add new text items	28.0 sec	23.6 sec	−4.4 sec
Rearrange items in hierarchy	25.5 sec	21.8 sec	−3.7 sec
Draw links	27.4 sec	25.7 sec	−1.7 sec
Find items and read links	25.9 sec	33.0 sec	7.1 sec
Remember number of items	33.3 %	62.5 %	29.2 %
Remember names of these items	30.8 %	37.1 %	6.3 %

time accounts for the higher efficiency in certain tasks and even through a mere exposure effect for the better ratings. However, although frequent users of MindManager were avoided, 75 % of the participants had used MindManager or similar tools before, so the overall exposure to mind-mapping applications and even MindManager itself was higher than for iMapping in most cases. Several participants even explicitly stated that although they prefer the iMapping concept, using the mind-map was easier for them because it was more familiar, and that they expect that, after getting used to it, iMapping would be more efficient for them than mind mapping.

The exact mean times and percentages in comparison can bee seen in Table 6.1.

The average subjective ratings, which were given as German school grades for both tools, are listed in Table 6.2. iMapping was rated better in every surveyed aspect and almost one full grade better in average. Most of these findings were statistically significant – in particular: *overview* and *aesthetics* were rated better in iMapping, which was also perceived to be more suitable for *note taking* and *personal knowledge management in general*. Especially in the overall rating, iMapping was rated *good* in average as opposed to MindManager, which was only rated *satisfactory*.

The good rating for *overview* is also backed by many spontaneous comments that were made during testing and that go in the direction of iMapping providing an easier way to get overview over especially the coarse structure: through the general approach of nested items, the possibility to arrange them freely and through the facility to zoom all the way out and back in again with only two keystrokes (as explained in Section 4.2.4).

The good rating for *aesthetics* came somewhat as a surprise, since three of the participants (all female, by the way) had vehemently complained about the lack of color in the predominantly grey iMapping prototype. But even so, the current visual design of the iMapping application seems to be quite well-received.

TABLE 6.2: iMapping vs. MindManager – Subjective Ratings in Comparison: Mean ratings given as German school grades for both tools: iMapping is rated better in every category surveyed. (1 = very good, 5 = insufficient).

	MindManager	**iMapping**	**difference**	**sig. (p)**
Navigation	2.4	2.1	0.3	.275
Overview	3.0	1.9	1.1**	.003
Structure / interrelations	2.7	2.1	0.6	.270
Aesthetics	3.3	2.2	1.1*	.038
Look and feel	2.5	2.0	0.5	.065
Suitability for brainstorming	2.5	2.0	0.5	.137
Suitability for note-taking	3.0	1.9	1.1***	.001
Suitability for personal KM	3.3	2.0	1.3***	$< .001$
Overall rating	3.0	2.0	1.0***	$< .001$
Average	2.9	2.0	0.9***	.001

* Statistically significant at the 5 % level
** Statistically significant at the 1 % level
*** Statistically significant at the 0.1 % level

The best rating and highest advantage has been given for the aspect *suitability for personal knowledge management in general*. However, this is not surprising, since MindManager is not designed for very large maps whereas it was a core requirement for iMapping to be able to scale up to extremely large maps as they can evolve over years of personal knowledge management.

All in all it is a satisfactory result that iMapping is comparable to MindManager in objective measures of time needed for interaction for typical tasks, and in subjective measures even superior to MindManager, which can be regarded as state of the art, has millions of users and is on the market since 1994.

Since the intended usage scenarios for iMapping exceed those overlapping with MindManager, some hypothesized advantages could not be tested in this setting and are subject to ongoing investigation. Namely: Long-term use for personal knowledge management, the use of much larger maps with thousands of items, the use of QuiKey and using semantic links and queries.

6.2 QuiKey – Query Building Evaluation

The main claim of QuiKey's interaction design, to be highly interaction efficient, has been evaluated in two phases. A set of tasks has been defined, by means of which the respective tools could be compared (Section 6.2.1). With these tasks, first, a comparative interaction analysis was carried out according to the KLM-GOMS method, where QuiKey was measured against state of the art semantic search interfaces (Section 6.2.2). Second, the outcomes of the GOMS analysis were validated by a user study, where actual interaction times were measured (Section 6.2.3).

6.2.1 Tasks and Data Source

The assignment that was chosen as a basis for the evaluation was to construct a conjunctive query that yields the answer to the question: *Which musical genres do Madonna and Michael Jackson have in common?* This assignment fulfills the following requirements:

- It can be decomposed into single steps that cover a wide range of QuiKey's functionalities (searching for items, browsing their properties, constructing simple and complex queries and adding new items).
- It is comparable to other tools that offer similar functionality.
- It is easy to understand because the musical domain is common knowledge.
- A large amount of structured data is publicly available.

In fact, an export from freebase[3] of the music domain has been taken (artists, albums, genres etc.). It was filtered down to yield a data set of 26 115 items that was highly interconnected to allow for complex semantic queries but small enough to run smoothly with the current CDS back end which was designed to handle personal knowledge management data fast and in memory rather than large imported data sets.

This goal can be broken down to the following sub-tasks, which have been used for comparison in the GOMS analysis and the user study:

1. Finding out what albums Madonna has made (text search, browsing)
2. Finding out what genres these albums have (set browsing)
3. Saving this as a persistent query (saving query)
4. Doing the same (1–3) for Michael Jackson
5. Intersecting these two saved queries (constructing complex query)

[3] http://www.freebase.com/

6.2.2 GOMS Analysis

KLM-GOMS is a method developed by Card, Moran & Newell (1983) to estimate the time it takes a user to complete simple interactive tasks using a keyboard and mouse. [4] To that end, each of the tasks is deconstructed in depth into single actions such as keystrokes, mouse movements, mouse clicks, mental operations and homing (moving the hand from one input device to the other). For each of these atomic actions, average execution times have been experimentally identified (Card et al., 1983), that can be used to estimate completion times for average experienced users.

With this method, QuiKey was compared to Semantic MediaWiki (SMW, Krötzsch et al., 2007), and Parallax (Huynh and Karger, 2009), both introduced in Section 5.4. SMW with its ASK query language is probably the most widely used semantic knowledge modeling tool that also allows to construct complex and persistent queries. Parallax is probably the most convenient user interface to date to explore large amounts of structured data.

Comparisons were made once for the most efficient way each tool could theoretically be used for the task and once for the typically expected way (e. g., query code was formatted with white space, query names were more verbose (like "Jacko genres" instead of "MJgen") and search words were spelt out instead of only to the point necessary.

Results: The resulting estimates of interaction time needed are shown in Table 6.3. Unfortunately, in Parallax it does not appear to be possible to intersect existing queries or sets. Saving a current query also seems to be not possible. However a straightforward way how this would be done in Parallax' interaction paradigm is apparent and so this way was guessed as an approximation. As can be seen in Table 6.3, QuiKey is theoretically more interaction efficient than any of the two compared tools on any of the tasks. This means that, learnability, interaction styles and error-proneness aside, it is theoretically possible to complete these tasks faster in QuiKey than in the other tools. What remains to be shown is that QuiKey can actually be *used* in this efficient way by real users.

6.2.3 User Study

In order to determine whether real users can use QuiKey as efficiently as it is designed to be, 16 participants performed the set of tasks defined in Section 6.2.1. All participants were familiar with semantic technologies in general. 3 of the participants were female, 13

[4] For an overview see http://en.wikipedia.org/wiki/KLM-GOMS

TABLE 6.3: Results of the GOMS Analysis: Estimated interaction times for the sub-tasks in comparison. Theoretically minimal interaction paths and typically expected ones are computed separately. Overall results for QuiKey are bolded for comparison with Table 6.4.

	SMW		**Parallax**		**QuiKey**	
Task	typical	minimal	typical	minimal	typical	minimal
One-time overhead per query	8.2	8.2	0	0	0	0
Elementary query Madonna	13.2	13.0	10.8	10.8	7.5	6.1
Elementary query M. Jackson	16.4	16.1	13.1	13.1	8.1	6.4
Increment to chain query $^{\times 2}$	18.3	17.2	7.2	6.6	3.5	2.9
Increment to save $^{\times 2}$	19.4	19.4	8.2	6.5	7.5	5.8
Intersection query	21.0	21.0	n/a	n/a	10.7	7.6
Overall time w/o intersection	113.2	110.4	54.8	50.3	**37.5**	**30.0**
Overall time incl. intersection	134.2	131.4	n/a	n/a	**48.2**	**37.6**

$^{\times 2}$ The two sub-tasks marked with $\times 2$ were needed twice in the complete task, so they are also counted twice in the overall times.

male. Their age ranged from 23 to 36. 14 of the 16 participants had never used QuiKey before and only one was familiar with it.

Each participant went through the following process:

1. Short introduction to QuiKey: Basic functionalities and interactions that are needed for the tasks were explained and demonstrated. Participants were asked to try out the interactions and explore the model and the tool until they felt confident in using it. (This took up to 8 minutes.)
2. A screen capture tool was started, which recorded screen content, audio, keystrokes and the users via screen camera.
3. Participants were then asked to carry out the set of tasks. Instructions were given sequentially one by one, whenever the previous step was completed.

Later, keystrokes and interaction times were determined from the captured video with a precision of 18 frames per second.

Results: The mean number of keystrokes and interaction times are listed in Table 6.4 along with their confidence intervals[5]. The last column lists the minimal interaction times any user had needed to complete the task. Since these values come from different users they do not add up to the overall times. The overall minimum times instead reflect

[5] http://en.wikipedia.org/wiki/Confidence_interval

the time of the overall single fastest user. These minimum times are included in the table because they prove for each task how fast it *can* actually be done.

TABLE 6.4: User Study Results: Mean times and keystrokes needed for sub-tasks (with 95% confidence intervals)

	Keystrokes		**Measured time**		
Task	Mean	±CI	Mean	±CI	Minimum
Elementary query Madonna	10.8	±1.1	9.2	±1.2	4.8
Elementary query M. Jackson	12.7	±1.9	8.9	±1.5	4.3
Increment to chain query [×2]	7.6	±0.4	7.1	±0.8	3.2
Increment to save [×2]	18.7	±2.6	8.7	±1.2	2.5
Intersection query	20.1	±2.7	16.0	±3.1	8.1
Cum. interaction time w/o intersection			**50.1**	±6.4	**31.8**
Cum. interaction time incl. intersection			**66.1**	±9.1	**38.9**

[×2] The two sub-tasks marked with ×2 were needed twice in the complete task, so they are also counted twice in the overall times.

Comparing the bolded values shows that the overall GOMS estimation of minimal interaction times were very close to the actual minimal times. This supports the GOMS estimations in general. Mean interaction times were somewhat longer than estimated for typical use. This may well be due to the fact that most of the participants were first time users. Also, during user testing it was noticed that search words were often spelt out completely instead of only to the extent needed (e. g., `Michael Jackson` instead of `m jac`). This is because users do not check results after every keystroke while typing known words. However in situations where keyboard interaction is more costly, like on mobile devices or touch screens, for motion impaired users, or for long or unfamiliar words, the use of shorter search strings becomes more relevant.

6.2.4 Conclusions

QuiKey's main design goal of interaction efficiency has been reached: The GOMS analysis has shown by theory, that the assignment can be fulfilled with considerably less interaction steps in QuiKey as compared to SMW and Parallax. This is true for both the overall assignment (Finding out what genres the albums of Madonna and Michal Jackson have in common) as well as for all single tasks on the way to construct the conjunctive query (namely: text search, browsing, set browsing, saving queries and intersecting saved queries). The subsequent user study has confirmed that QuiKey is actually used, even by novice users, in an efficient way close to the theoretical predictions.

This finding is backed by the outcomes of the User Experience Questionnaire reported in Section 6.3, where QuiKey has earned very high ratings on perceived efficiency.

6.3 Summative User Study

The goal of the summative evaluation study was to test the overall usability of iMapping and QuiKey and to justify their coexistence. It should be shown, that both tools can be used effectively for the tasks they were designed for, and that both of the tools have specific strengths and weaknesses that complement each other well.

The first idea was to simply let users freely use the tools for their own work during two months, and then to observe them performing specific tasks in a laboratory setting, in order to get comparable measures for certain types of tasks for both tools respectively. However, although there were some first active users even before the study, it turned out that most of the initial 34 study participants did not adopt the tools right away for their own daily productive work – which would have been necessary in order to gain the desired sizes of user-generated maps. For that reason, the summative user study was then split into two parts:

1. A systematic user study: In order to gain systematically comparable data, registered participants were provided with an existing large iMap, in which they were to perform certain homework tasks during a two months period, before they were invited to participate in a final lab session, where they were observed carrying out a standardized set of tasks.

2. Additionally, in order to gain insights on how actual users construct iMaps in their daily work, all registered users were encouraged to submit their largest iMap in an open contest.

For both parts of the study, participation was incentivized by the possibility to win an iPad, which was raffled among all participants in a lottery drawing.

This section describes the hypotheses for the systematic user study, along with their operationalizations (Section 6.3.1), the study design (Section 6.3.2), the process of data analysis (Section 6.3.3) and finally the results of this user study. An analysis of the user-generated maps is given in Section 6.4.

6.3.1 Hypotheses and Operationalizations

Since some major aspects of the tools have already been covered by the earlier studies (i. e. basic iMapping navigation and editing in Section 6.1 and query building with QuiKey in Section 6.2), the goal of this summative evaluation study was to confirm suitability for different kinds of information seeking tasks.

Below, for the sake of coherence, hypotheses are explained together with their operationalizations. Therefore, the types of tasks used in the study are introduced *before* the hypotheses, since they are needed to understand the operationalizations. The ten hypotheses are then presented in three groups: General Usability, iMapping vs. QuiKey and iMapping specific Hypotheses.

The two main types of tasks that have been examined are: *finding linked information* and *getting vicinity*. They have been chosen, because they are believed to be frequent PKM tasks and the two tools are hypothesized to differ significantly in the efficiency of handling these tasks.

Finding Linked Information What is meant here is to find an item that is linked to another item that can be directly targeted. An example is: "What do Tigers eat?" This type of task can be split into two sub tasks: 1) finding the *known* item (in our example the tiger) and 2) navigating to the *target* item that is linked from there (in our example the sheep, see Figure 6.3).

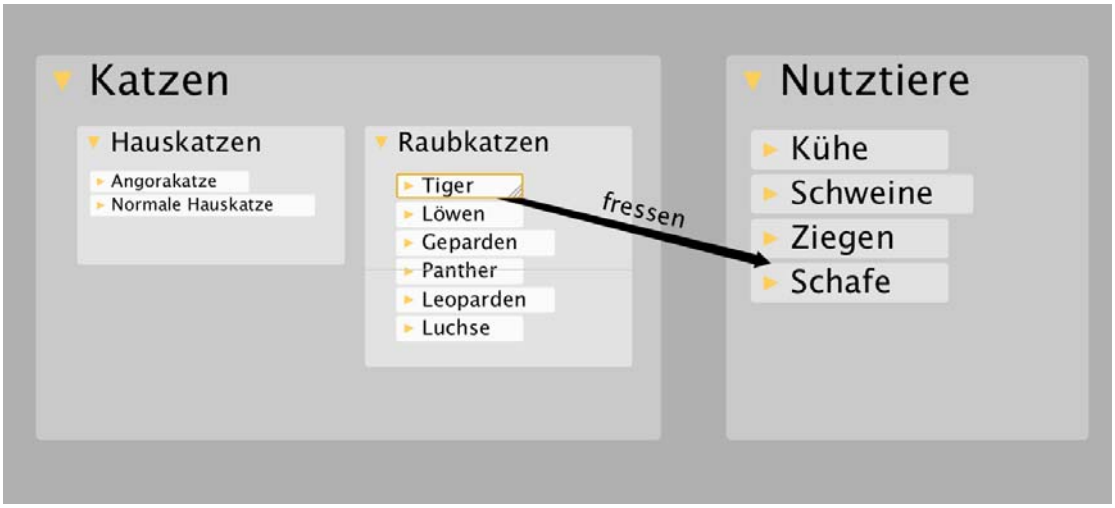

FIGURE 6.3: Finding Linked Information: What do tigers eat? Screenshot of the map part that has to be found for this task.

The interaction times for these two sub tasks were measured separately. The variables are called *finding* and *target delta* (the additional time needed to get to the actual target from there) respectively. Other examples tasks of this type were: "What is the homepage of the Journal of Digital Information" or "What is the telephone number of the Louvre?"

Overview For the general concept overview, two different variables were measured – discovering the *context* and discovering the *vicinity* of an item:

Context tasks required to find out the context two hierarchy levels above a given item. Questions for the *context* type of task were either rather direct, like – in the above example, given the item *sheep* ("Schafe"): "What is the super-class of that? – And of that?" (see Figure 6.4). Or, a less guiding question of the same type was – given a school class: "In what village is the school?".

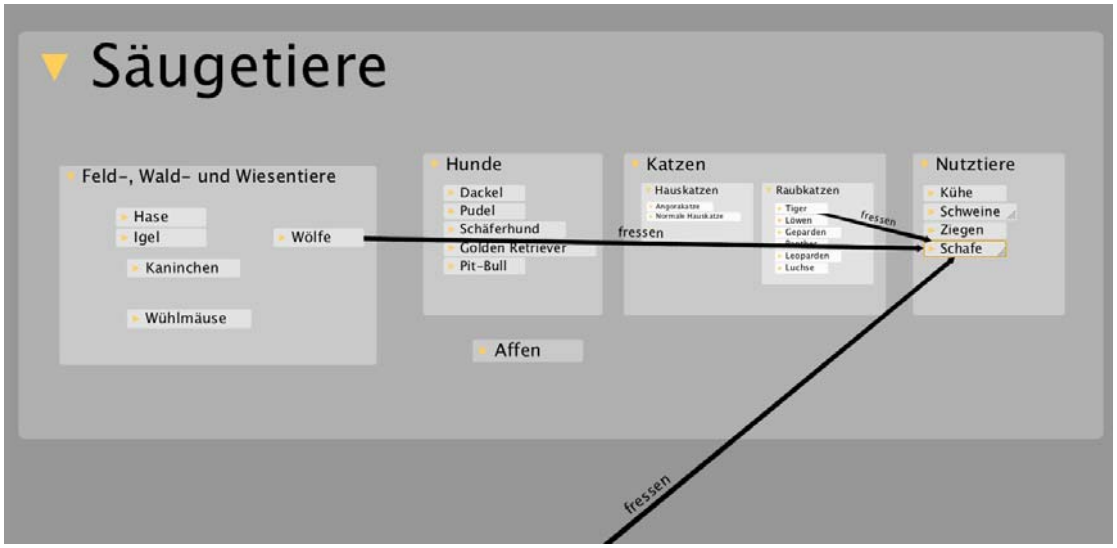

FIGURE 6.4: Context Task: What is the super-class of sheep? And of that?

Vicinity tasks required to get an overview over a number of items closely related to the given one. One such question e. g., was, given *Heinz*, a boy in a school class: "Who are the girls in his class?" (see Figure 6.5).

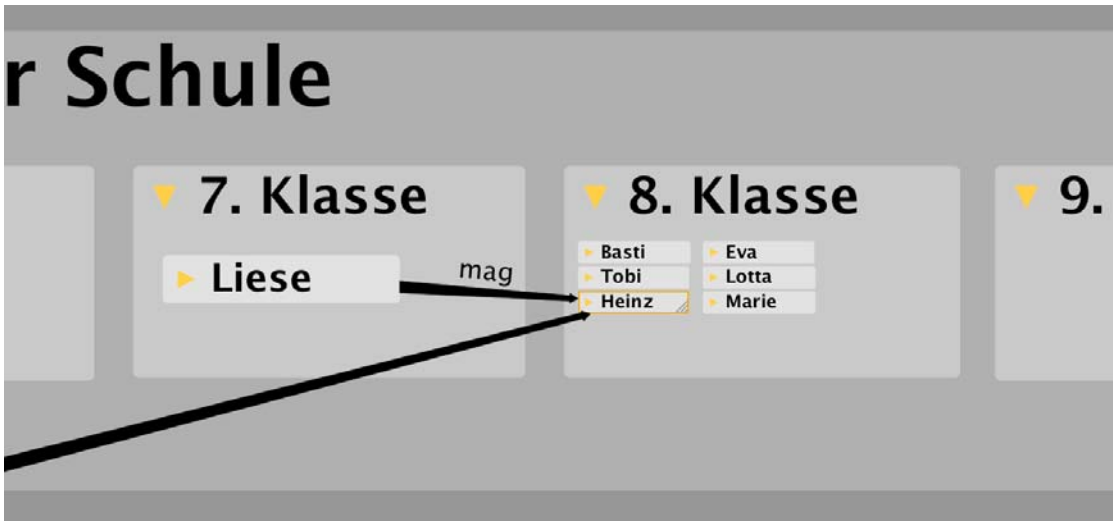

FIGURE 6.5: Overview Task: What are the girls in Heinz' class?

For the general overview concept, these two variables have always been used together, i. e. their mean value.

Here are the ten hypotheses, arranged in the three groups: General Usability, iMapping vs. QuiKey and iMapping specific Hypotheses:

General Usability

H1: All tasks can be successfully achieved with each of the tools alone. The tasks described above are in principal achievable in both tools, although some tasks may be easier achieved in one tool or the other. The hypothesis is that each of the sub-tasks described above (*finding, target delta, context, vicinity* can be achieved with either of the tools without error.)

H2: User experience is positive for both tools. *User experience* (often abbreviated *UX*) is a theoretic construct that has, in recent years, to some extent, replaced the concept of mere *usability*. In ISO 9241-210 (2009), it is defined as "a person's perceptions and responses that result from the use or anticipated use of a product, system or service". UX widens the focus of investigation from mere functional aspects to a more holistic perspective that additionally includes usefulness and hedonic aspects like attractiveness. A widely used instrument to measure user experience, is the four dimensions *AttrakDiff* questionnaire by Hassenzahl et al. (2003); Hassenzahl (2006). For this study, however, the more recent questionnaire *UEQ* by Laugwitz et al. (2006, 2008) was used that covers the following six dimensions: *Attractiveness, Perspicuity, Efficiency, Dependability, Stimulation* and *Novelty*. Especially the aspect of perspicuity [6] is vital for information systems that handle complex information like heterogeneous semantic knowledge models.

H3: Efficiency is perceived as good in both tools, especially in QuiKey. Because many design decisions for both tools were targeted at increasing efficiency, it is expected that perceived efficiency (as measured by the UEQ, explained above) should be positive for both tools. Since interaction efficiency was the primary design goal for QuiKey and has already been confirmed by the evaluation study described in Section 6.2, perceived efficiency is expected to be especially high for QuiKey.

iMapping vs. QuiKey

According to the nature of the two tools, they are expected to differ in the interaction times needed, depending on the type of task at hand:

[6] Definition of perspicuity according to Merriam-Webster Online: Plain to the understanding especially because of clarity and precision of presentation.

H4: Finding items is more efficient in QuiKey. Finding an item for which at least some part of the content is known should be much faster in QuiKey as opposed to finding it by visual navigation in iMapping. Thus, interaction times for the subtask *finding* should be significantly lower with QuiKey than with iMapping.

H5: Finding linked information is more efficient in QuiKey. As detailed above, finding linked information can be split into two subtasks. While H4 covers the first subtask (*finding*) alone, H5 targets the combined task as a whole. It is hypothesized, that *finding + target delta* would be faster in QuiKey, because QuiKey only shows information matching the current query, whereas in a comprehensive interface like iMapping, visual information is much more complex and has to be cognitively filtered by the user in order to find and follow the desired link.

H6: Getting overview is more efficient in iMapping. Since overview was a specific design goal of iMapping but not QuiKey, it is hypothesized that overview tasks (both *context* and *vicinity*) be more efficient in iMapping than in QuiKey.

H7: For combined Tasks, using both tools together is more efficient than using either of them alone. For tasks that require both targeted search and overview (*finding + target delta + context + vicinity*), it is hypothesized that it is most efficient to use both tools in combination. This might seem trivial at first sight, but for this to be the case, even when H5 and H6 are true, users would still need to pick the right tool for each part and switching tools must not cost more times than it saves.

iMapping specific Hypotheses

H8: Visually finding items is easier in familiar environments. In order to find out to what degree familiarity helps visual search and orientation in iMapping, all tasks were presented in two different conditions: a) in places of the map that were known to the participants from the preceding preparatory assignments and b) in places that were completely new to them. The hypothesis is that with iMapping, tasks are completed faster in familiar environments of a map. This should be the case for all of the above tasks, but especially so for visual search (*finding*).

H9: Exploring link structures is more efficient with links on demand than with all links visible In order to test the usefulness of the links-on-demand method, the following tasks were used, one with all links visible, and one with links on demand:

"Which of the requirements in the list to the right are met by *all three* of the approaches on the left?", "Which of the requirements in the list to the right are met by *none* of the approaches on the left?" (see Figures 6.6 and 6.7). The hypothesis is, that in the links-on-demand condition, users will be much faster than in the condition with all links visible.

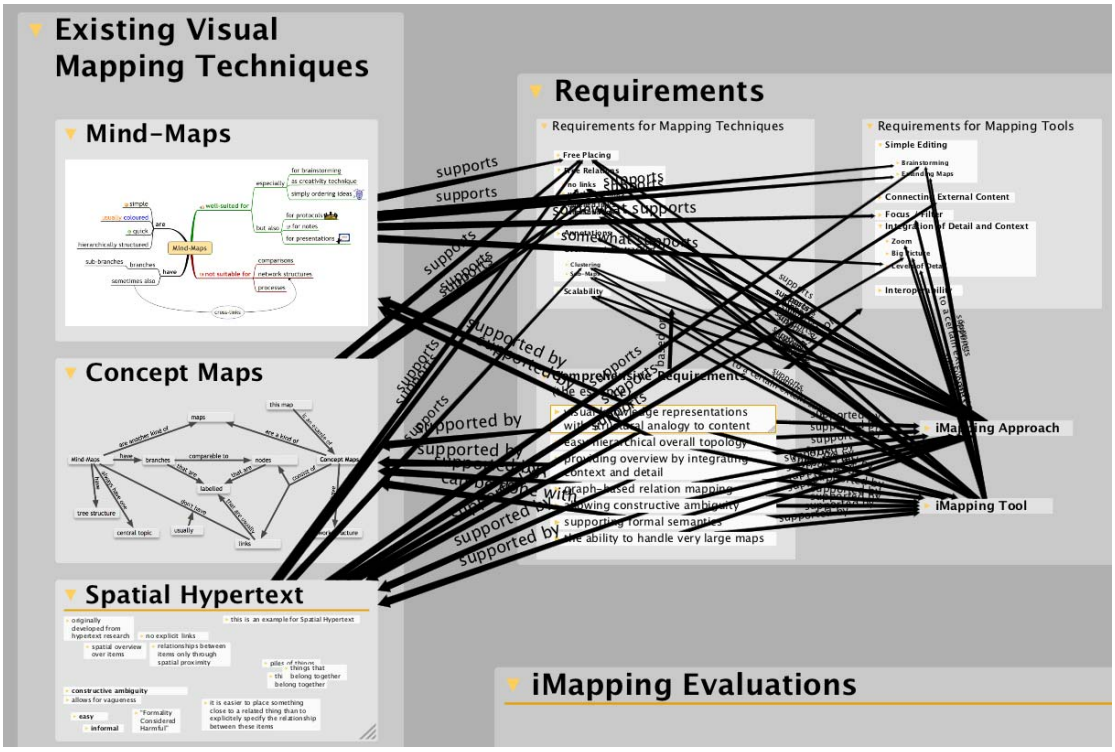

FIGURE 6.6: All Links Visible: Screenshot of the map part for H9. In this condition, users are expected to be slower and more error-prone compared to the version below in Figure 6.7.

H10: Reading more complex structures can be done efficiently in iMapping. One additional task was tested without a concrete hypothesis other than that it can be successfully completed: To read the answer of what would be a combined query involving class hierarchy and link structures. The questions asked in the context were: "Apart from tigers, which other animals eat sheep? And what is the lowest common super-class of these three?" Although strictly this is not a hypothesis, it will be labeled *H10* here for reasons of referability.

6.3.2 User Study Design

The systematic user study consisted of two parts: a preparation phase ranging between one week and five months, and a concluding interview. During the preparation phase, participants were successively given assignments in order to ensure familiarity with the

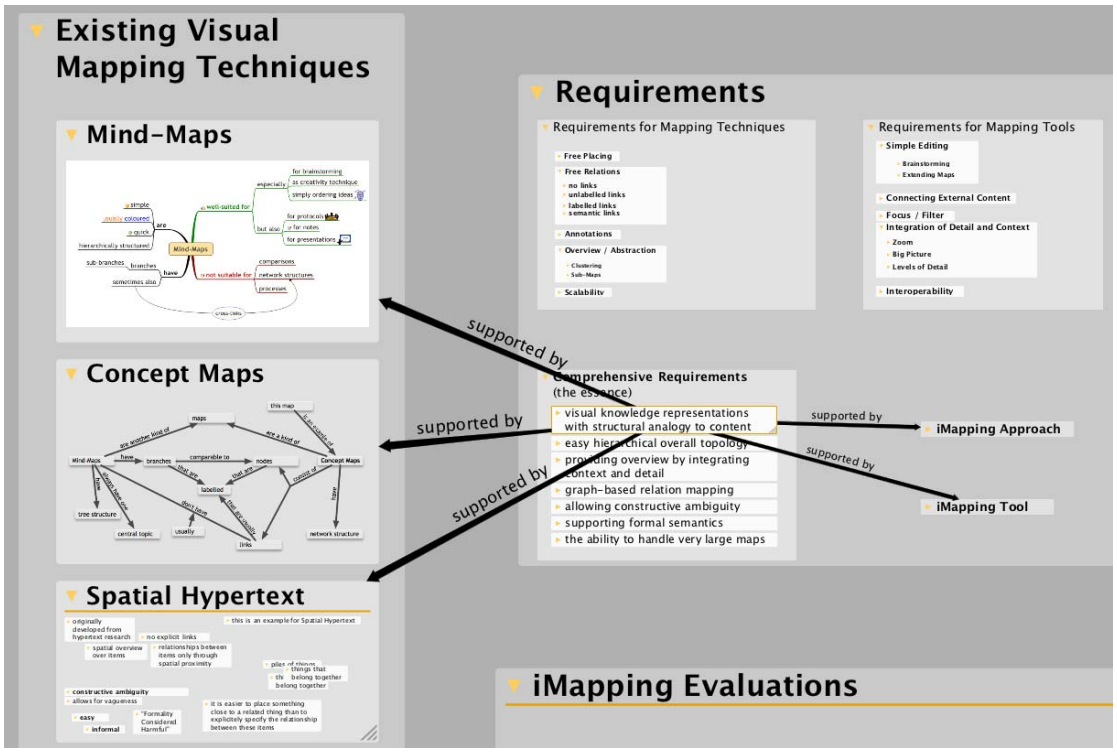

FIGURE 6.7: Links on Demand: Screenshot of the map part for H9. In this condition, users are expected to be faster and less error-prone compared to the version above in Figure 6.6.

tools and the map. In the concluding interview, participants were observed while completing further assignments. Also they were asked to provide their log files and complete user experience questionnaires.

The rest of this section will successively describe the sample of participants, the preparatory phase and the data that was obtained before reporting on its analysis (Section 6.3.3) and outcomes (Section 6.3.4).

Participants

Initially, 34 participants were recruited from the following three groups: 10 undergraduate students from various backgrounds, all working at FZI Karlsruhe, 13 PhD students from FZI and the Karlsruhe Institute of Technology, and 11 external participants recruited from the growing user base. As common in such studies, it turned out that after the begin of the study, unfortunately, around half of the participants dropped out – some by apologizing for having overestimated their available time, others by simply never responding. To obtain a sufficient number of participants, some were recruited half way through the study, and some only finally responded after being offered to catch up by a comprehensive set of preparatory tasks that could be completed in around half an hour.

The constitution of the sample of participants that finally participated in the complete study was the following:

- N = 18
- 4 undergraduate students, 5 PhD students, 9 externals
- age ranging from 22 to 47 with an average of 31 years
- 2 female, 16 male
- 7 experts, 9 intermediates and 2 complete novices to semantic technologies
- 7 also participating in the big maps contest (see Section 6.4)

Preparatory phase

After participants were recruited, they were sent a download link and initial instructions. They received 9 successive assignments, about one per week. Together with the first assignments, links to introductory online video clips were sent, that explained usage of the basic functionality of the two tools. [7]

The actual assignments were designed in order to assure two things: a) that the participants were familiar with the basic functionality of the tools and b) that they were familiar with certain parts of the map where some of the final assignments were to take place (see Hypothesis 7 in Section 6.3.1). All assignments were rather short, so they could be achieved in less than 5 minutes each.

The functionalities covered by the assignments were:

in iMapping:

- Adding items
- Collapsing and expanding items
- Moving and nesting items
- Resizing items
- Panning navigation
- Zooming navigation
- Drawing links
- Following links

[7] The three introductory video clips were:
iMapping basics: http://www.youtube.com/watch?v=TYy-CfWNS00
dealing with links: http://www.youtube.com/watch?v=TYswv9lMW7w
QuiKey basics: http://www.youtube.com/watch?v=pFHEqEhmB4w

in QuiKey:

- Searching for items
- Adding new items
- Adding new links
- Navigating the link structure

All of these functionalities, and some more, were introduced by the introduction videos. A list of the assignments is included in Appendix-D.2

Collection of Data

The final interviews, where data was collected were structured as follows:

- The participant was welcomed and asked for any problems encountered during the assignments or further use of the software.
- It was checked whether the participant had all interaction skills necessary for the final assignments.
- As final assignments, the participant was asked questions that required finding the answers using the tools. As detailed below, some of the answers were to be given by using only one of the tools. Participants were observed and timed while searching for the answers. When a question had not been answered after one minute, which sometimes was the case for visual search in iMapping, hints were given to the participant as to where to look for the desired area. When a wrong answer was given, the answer was rejected by the experimenter and time continued counting until the right answer was found.
- After the assignments, the participant was asked to anonymously complete the User Experience Questionnaire (UEQ) online – once for each of the two tools.

The final assignments were clustered into seven blocks, as listed in Table 6.5:

The exact wording of these questions is included in Appendix D.3.

Each of the blocks 1–6 was to be answered in one of the following three conditions: a) either with QuiKey alone, b) with iMapping alone or c) with both tools allowed. In condition c, the participant was allowed to choose for herself which tool to use for which task.

TABLE 6.5: Structure of the Evaluation Assignments: Six structurally equivalent sets of three questions each (one of them including two additional questions for H10), plus one block with the two questions for H9 (see Section 6.3.1).

Familiar vs. Unknown	**Block**	**Task Type**
Familiar parts of the Map	Block 1	Find linked information Discover vicinity See context
	Block 2	Find linked information See context Discover vicinity
	Block 3	Find linked information See context Discover vicinity
Unknown parts of the Map	Block 4	Find linked information See context Discover vicinity
	Block 5	Find linked information See context Discover vicinity Additional linkage question Additional context question
	Block 6	Find linked information See context Discover vicinity
–	Block 7	Link structure question 1 Link structure question 2

In order to avoid any systematic biases through difficulty differences between the blocks, or because some block may be easier to handle with one of the tools, the assignment of conditions to blocks was completely balanced out, so that after all 18 sessions, each block had been done in each of the three conditions exactly six times.

Also the sequences of the conditions as well as the combinations of blocks per condition were balanced out, so that tool sequence effects can also be ruled out and good comparability between conditions can be assumed.

The two questions in block 7 (see Section 6.3.1) have also been balanced against the two conditions compared, so that each task was completed 9 times in each of the conditions.

Interaction times needed to answer the questions were measured using a stopwatch.

6.3.3 Data Analysis

In order to test hypotheses H1–H10 introduced in Section 6.3.1, the following variables have been computed from the raw data:

For H1, no computation or calculations were necessary, see Section 6.3.4 for discussion.

For H2 and H3, to compute the results of the UEQ, the preconfigured MS Excel spreadsheet was used that comes included with the questionnaire material. It aggregates the 26 seven-step item scores to 6 scales and an overall score, each ranging from -3 to $+3$. Since they are not comparative hypotheses, confidence intervals have been computed instead of significance tests.

For H4–H8, the variables *find*, *target delta*, *context* and *vicinity* have been computed as well as their *sum* across the following conditions:

iMapping / *QuiKey* / *mix*: denoting what tool was allowed to be used
familiar / *unknown*: denoting what parts and items of the map were used.

H4–H9: The level of measurement for interaction times is of course a ratio scale, which would suggest using parametric significance tests like t-tests or an ANOVA [8]. A prerequisite of these however is that the data is normally distributed on all scales. While some of the variables by visual judgment of the distribution graphs clearly were, others clearly were not, which is confirmed by the Kolmogorov-Smirnov test, which was run for all variables used. The test revealed the variables *QuiKey find, QuiKey context, QuiKey vicinity, iMapping target delta, iMapping vicinity* and *unknown iMapping vicinity* as significantly deviating from a normal distribution. The exact values for all variables can be found in Appendix C. For that reason, the nonparametric Wilcoxon signed-rank test (Wilcoxon, 1945) has been used for all comparative measures that allow for paired testing. For additional computations across groups of users for H7, where pairing was not possible, the Mann–Whitney U variant of the Wilcoxon test was used (see Section 6.3.4).

Because the robustness of the parametric t-test against violations of the normality prerequisite is under debate (Sawilowsky and Blair, 1992), t-tests were also computed for reference. An overview over the outcomes of both parametric and non-parametric tests for all comparisons tested can be found in Appendix C.

[8] "Analysis of Variance", see `http://en.wikipedia.org/wiki/Analysis_of_variance`

6.3.4 Results and Discussion

Below, the results of H1–H10 are presented, each along with some short discussion. Like in Section 6.3.1, they are grouped into General Usability, iMapping vs. QuiKey and iMapping Specific Measures.

General Usability

H1 All tasks given to the participants have been completed successfully, with two exceptions:

1) In the task comparing the readability of link structures with all links visible vs. links shown on demand only, 10 of the 18 participants gave one or more wrong answers, in the condition with all links visible. This result is conform to H9 (Exploring link structures is more efficient with links on demand than with all links visible), and will be discussed there in Section 6.3.4. In the condition where links were shown on demand only, which is also the default setting, all participants answered the question without errors. Thus, this does not contradict H1.

2) However, 11 of the 18 participants needed to receive help after taking more than 60 seconds to find the desired item in the condition with *familiar* map parts and only visual search in iMapping allowed. In all other conditions, even with *unknown* map parts and iMapping only, all participants were able to complete the tasks without help. Possible reasons of this counterintuitive result are discussed below with H7 (Visually finding items is easier in familiar environment).

That several users actually completed all tasks in all conditions on their own and without error would strictly prove the hypothesis. However, since it obviously cannot be generalized, and since visually finding items in iMapping took an overall average of 59 seconds, even with help received, **H1 needs to be rejected and its statement modified as follows: While all tasks can theoretically be completed in both tools, especially visual search can be very cumbersome and should be complemented by a text search facility.**

H2 The User Experience Questionnaire (UEQ, introduced in Section 6.3.1) has been filled out by all 18 participants. As can bee seen in Figure 6.8 and Table 6.6, both tools are rated between +1 and +2 on each of the UEQ scales. These scales range from −3 to +3, and, as Schrepp and Laugwitz (2010) assure, values between 1 and 2 can be interpreted as "very good" compared to other software tested with the UEQ. This **confirms H2: User experience is positive for both tools.**

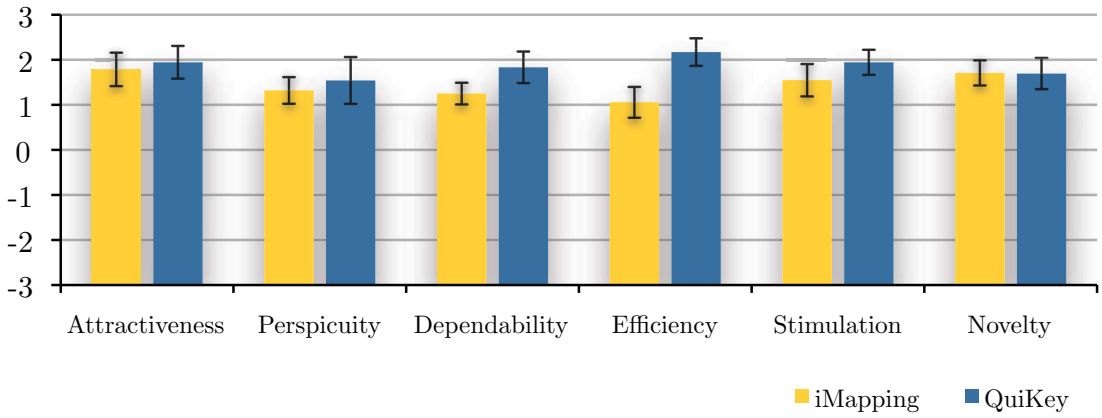

FIGURE 6.8: Positive User Experience: Average UEQ scores with confidence intervals for the two tools in comparison.

H3 As expected, the highest ratings have been given on the *efficiency* scale for QuiKey, which is in fact the only score greater than +2. This confirms the results of the GOMS analysis described in Section 6.2. As for iMapping, the efficiency value is the lowest of all ratings received. This may be due to a variety or reasons: a) the slow animation performance of the iMapping tool in large maps like the one used for the final lab session, b) the difficulties many users had in finding some items by visual search (discussed below with H8), the long start-up and loading times for the iMapping tool, or even a contrast effect to QuiKey. As discussed above, the value +1 is however still clearly in the positive range. Thus, also **H3 can be confirmed: Efficiency is perceived as good in both tools, especially in QuiKey.**

TABLE 6.6: UEQ Results: Mean scores, 95 % confidence intervals and standard deviations of the six UEQ scales and the overall rating for both tools in comparison.

	iMapping			**QuiKey**		
Scale	Mean	±CI	SD	Mean	±CI	SD
Attractiveness	1.79	±0.37	0.80	1.94	±0.36	0.79
Perspicuity	1.32	±0.30	0.64	1.54	±0.52	1.13
Dependability	1.25	±0.24	0.52	1.83	±0.35	0.76
Efficiency	1.06	±0.34	0.74	2.17	±0.31	0.66
Stimulation	1.55	±0.36	0.78	1.94	±0.28	0.60
Novelty	1.71	±0.28	0.60	1.69	±0.35	0.75
Overall	1.48	±0.21	0.45	1.86	±0.27	0.59

iMapping vs. QuiKey

H4 The average time taken in iMapping to find the desired item was 59 seconds. For QuiKey, it was 3.1 seconds.

The unexpectedly long time needed in iMapping is partly due to the fact that the areas of the map that were supposed to be *familiar* to the participants, in fact, for many of them, were not. This is discussed below with H8.

Even when only the values from the *unknown* condition are compared, which caused participants less trouble and where all targets were found without help, the average search times are still 22 seconds for iMapping as opposed to 2.9 for QuiKey. Table 6.7 shows the exact values along with some descriptive statistics and p-values. **The data strongly support H4: Finding items is more efficient in QuiKey.**

TABLE 6.7: Finding is Faster in QuiKey: Mean times, 95 % confidence intervals, standard deviations and range of the variable *find* for both tools in comparison – both overall and *unknown* condition only. p-values are calculated with a Wilcoxon test for paired samples.

Tool	Condition	Mean	±CI	SD	Min	Max	p
iMapping	overall	58.6	±18.8	57.4	8	214	
	unknown	22.1	±6.9	14.9	8	60	
QuiKey	overall	3.1	±0.6	2.0	1	8	$< .001$
	unknown	2.9	±0.8	1.7	1	8	$< .001$

H5 Keeping in mind, that finding linked information as defined in Section 6.3.1 is composed of the two variables *finding* + *target delta* (combined in the variable *target*), it is not surprising that also for this combined measure the difference between the tools was huge and significant in the same magnitude as *finding* alone (9 seconds for QuiKey and 64 for iMapping, $p < .001$, see Table 6.8).

Target delta is the amount of time it took to find the linked information from the point where the first item (that linked to the target) was already found. Looking at the variable of this sub-task alone, it turns out, that although the average values for both tools hardly differ (5.5 and 5.7 seconds), the medians show that actually most participants were faster in iMapping for this sub-task: The median for iMapping was 2 seconds as opposed to 5 seconds for QuiKey. The nearly equal averages were caused by extreme outlier values of two participants that had taken 40 and 62 seconds for this sub-task. The Wilcoxon significance test however, which is rather robust against such outliers, confirms the advantage of iMapping for *target delta* with $p = .01$.

In conclusion, while **H5: Finding linked information is more efficient in QuiKey (as a combined task) can be maintained,** it can be said that for the sub-task of following the link structure, iMapping has been slightly more efficient.

TABLE 6.8: Finding Linked Information is Faster in QuiKey: Median and mean times, 95 % confidence intervals, standard deviations and range for the combined variable *target* (find + target delta) and the sub-task *target delta* alone for both tools in comparison. p-values are calculated with a Wilcoxon test for paired samples.

Tool	Variable	Median	Mean	±CI	SD	Min	Max	p
iMapping	target	36.5	64.3	±20.4	62.4	10	241	
	target delta	2.0	5.7	±3.8	11.7	1	62	
QuiKey	target	8.0	8.6	±1.3	3.9	3	19	< .001
	target delta	5.0	5.5	±1.0	3.0	3	10	.012

H6 Getting overview has been measured by two variables: *context* and *vicinity*, the former being the time to recognize (grand)parent items in a hierarchy, the latter being the time needed to see all surrounding sibling items. As can bee seen in Table 6.9, getting overview was achieved about 4 times faster in iMapping for both measures. Since the difference is statistically significant, **H6 can be maintained: Getting overview is more efficient in iMapping.**

TABLE 6.9: Getting Overview is Faster in iMapping: Mean scores, 95 % confidence intervals, standard deviations and range of the variables *context* and *vicinity* for both tools in comparison. A Wilcoxon test for paired samples was used to calculate p-values.

Tool	Task	Mean	±CI	SD	Min	Max	p
iMapping	context	6.5	±1.7	5.3	0.5	24	
	vicinity	2.7	±0.7	2.1	0.5	12	
QuiKey	context	28.4	±9.8	30.1	4	126	< .001
	vicinity	11.8	±4.2	6.8	3	78	< .001

H7 At first glance, as can be seen in Figure 6.9, times taken for the whole blocks (*finding* + *target delta* + *context* + *vicinity*) were shortest in the condition where users were free to choose which tool to use for which sub-task. Mean overall times were 73 seconds for iMapping, 49 seconds for QuiKey and 33 seconds when both tools could be used, and differ significantly (p = .01 comparing the latter two).

When only the *unknown* condition is taken into account, where visual search was not as unrealistically cumbersome, (as discussed below, with H8), this difference, while still present, diminishes to only 5 seconds, which is statistically insignificant.

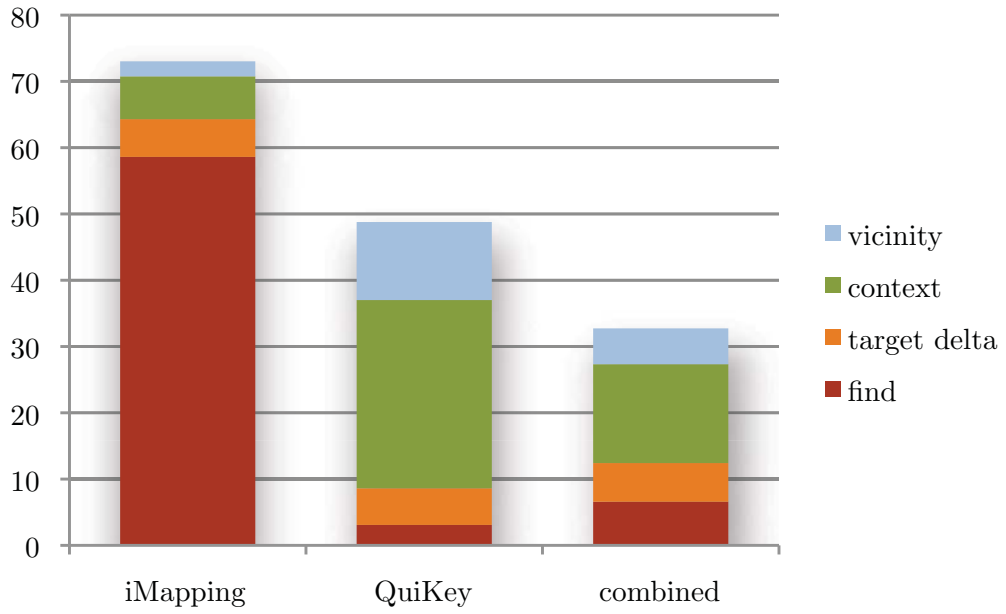

FIGURE 6.9: Overall Interaction Times in Seconds: For finding items, QuiKey is much faster. Overview (context and vicinity) is easier in iMapping. For overall performance, participants were fastest when allowed to use both tools combined.

However, it was observed that participants often did not use both tools in combination even when this was allowed. Those participants that *did* switch tools turned out to be significantly faster on the whole block than those that did not (overall 58 seconds vs. 80, $p = .049$) and even more so in the *unknown* condition where those 9 participants that used both tools when allowed took only half the time needed by the 9 others: 18 seconds as opposed to 36 ($p = .005$), as can be seen in Figure 6.10.

In conclusion, **this confirms H7: For combined Tasks, using both tools together is more efficient than using either of them alone.**

iMapping Specific Measures

H8 In the case of this study, this hypothesis could not at all be confirmed. In fact, participants took much longer to find items in the areas that were supposed to be *familiar* to them (95 seconds on average) as opposed to finding items in areas of the map they had never seen before (22 seconds).

The reasons however became obvious during the final observed assignments and lies in a flaw of the design of the study:

The *unknown* parts of the map were hierarchically structured in a way that could be easily understood: One of them was a geographically sorted "World Map". The Louvre could be found by browsing *World Map* → *Europe* → *France* → *Paris* → *Louvre*. Another

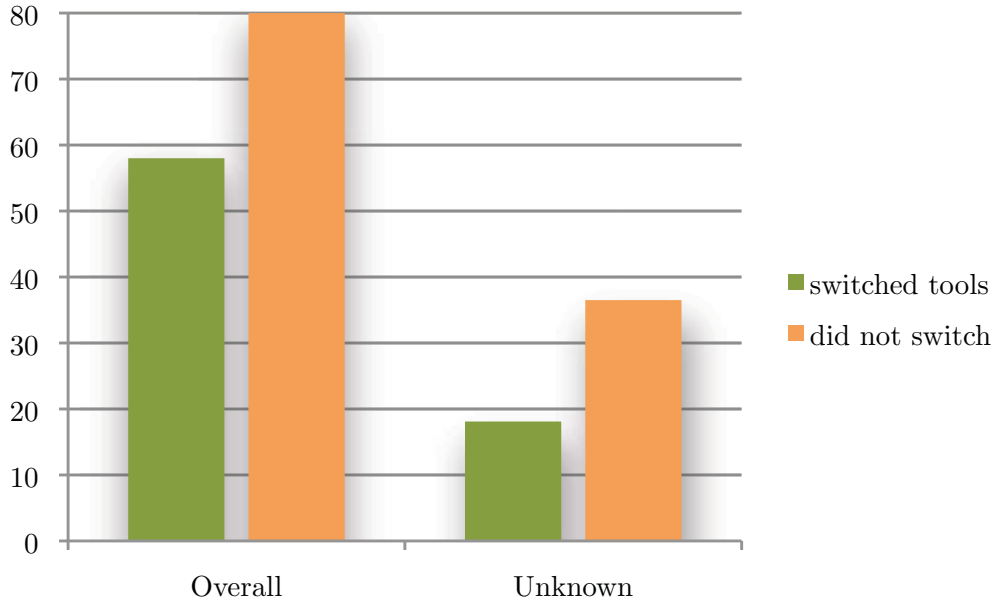

FIGURE 6.10: Switching Tools: Overall interaction times in seconds, grouped by whether participants actually used *both* tools when they were allowed to. Those that did use both tools for a combined block of assignments were faster. This holds across conditions (familiar and unknown) but also for the *unknown* condition alone.

was a rudimentary sketch of the animal kingdom that – although not biologically correct – was easy to browse: The tiger e. g., could be found by browsing *Animals* → *Vertebrate* → *Mammals* → *Cats* → *Cats of Prey* → *Tiger*. Also, the top level entry points to these new parts were pointed out to participants when they were introduced to the setting. These easy to browse structures were chosen so participants had a chance at all at finding the desired items although the parts of the map were unknown to them.

The other parts in contrast, that were supposed to be *familiar* to the participants, were less obvious to find. One journal, e. g., had to be found by browsing *Dissertation* → *Publication Planning* → *Journals* → *Primary Publication Targets* → *Journal of Digital Information*. A citation about the "homo oeconomicus" had to be found by browsing *Conferences* → *September 2009 Conferences* → *Vision Summit 2009* → *Keynotes* → *Prof. Götz Werner* → *"Den Homo Oeconomicus . . . "*. It was expected that the participants were familiar with these map parts, because they had already been touched by the preparatory assignments. It turned out, however, that visiting certain areas of the map two or three times was simply not sufficient to let people remember where they were located in the large map.

In conclusion, **H8: Visually finding items being easier in familiar environments could not be confirmed.** However, due to the explanation above, it should also not be completely rejected but rather left open for further research.

H9 In the example tested, with links shown on demand only, all participants were significantly faster than in the condition where all links were always shown. The average times taken to complete the task were 10 seconds vs. 60 seconds ($p < .001$). Also, while, in the links-on-demand condition, all participants gave correct answers only, at least 10 out of 18 participants gave one or more wrong answers in the all-links-visible condition. (The exact number is not known here because in the first interviews, errors had not been recorded. The average number of errors among the 14 participants where they were recorded, is 1, $p = .004$). Consequently, **H9 is confirmed: Exploring link structures is more efficient with links on demand than with all links visible.**

H10: Reading more complex structures can be done efficiently in iMapping. All participants were able to answer both parts of the question. For the first part, from one and the same the same item, several links had to be followed that pointed to targets that could, in most cases, only be made visible by panning and/or zooming. For the second part, the lowest common parent items of these three targets was to be determined, which lay between two and four hierarchy levels up, depending on which of the link targets had been focused last (see H10 in Section 6.3.1 for the concrete example).

TABLE 6.10: Reading Complex Structures: Mean scores, 95 % confidence intervals (CI), standard deviations (SD) and range of the time taken for the two parts of the task.

Task	Mean	±CI	SD	Min	Max
Follow links	8.9	±2.7	5.8	1	21
Find common parent	5.4	±1.9	4.1	0.5	19

6.3.5 Conclusions

In conclusion, all hypotheses except H1 and H8 were confirmed by the data:

General user experience is very good for both tools on all 6 scales of the user experience questionnaire UEQ (H2). As a part of that, efficiency is also perceived as good in both tools and especially so in QuiKey (H3). This is in line with an additional long term user study by Völkel (2010), who compared the three CDS-based tools iMapping, QuiKey and HKW to SMW+ in an effort to evaluate the suitability of CDS for building personal semantic knowledge models (HKW and SMW+ have been introduced in Section 5.4). In this comparative study, the usability of iMapping has been rated 1.6 in German school grades[9] and QuiKey 2.0 as opposed to 2.0 for SMW+ and 2.8 for HKW.

[9]German school grades range from 1 (very good) to 5 (insufficient)

Finding items is much more efficient in QuiKey as opposed to mere visual search and navigation in iMapping (H4). Because of that, finding *linked* information as a *combined* task is also more efficient in QuiKey (H5), although the isolated sub task of following links has been performed faster with visual links in iMapping by most participants. By contrast, getting overview is more efficient in iMapping (H6). For more complex tasks that combine the above kinds of sub tasks, using *both* tools together is more efficient than using either of them alone (H7).

Visually exploring link structures of highly interlinked items is much efficient and less error prone when links are displayed on demand only, as opposed to all links visible (H9).

While all tasks can theoretically be completed in each of the tools alone, especially visual search can be very cumbersome and should be complemented by a text search facility (H1 refined).

That visually finding items should be easier in familiar environments (H8) could not be confirmed. However, this hypothesis should not bee seen as refuted, because it seems that what was foreseen to count as familiar in fact for many users was not familiar.

Additionally, it has been shown that reading certain more complex structures, like simultaneously following several links and determining the common context of their targets, can also be done efficiently in iMapping (H10).

All in all, it can be said that iMapping is especially suitable for exploration tasks coming from a known item or area. However, for localizing items in the map and for the targeted obtention of information, QuiKey is more efficient and thus a valuable complement to the visual iMapping tool.

6.4 User Generated Maps

In order to gain insights on how users actually use the tools, especially if and how they build large iMaps, an open contest was carried out where everyone could submit his largest iMap.

This section first describes the contest and the topics of the maps that were obtained. Then it gives some descriptive statistics on these maps and describes to what extent semantic features have been used. A selection of the actual iMaps that were submitted can be found in Appendix A.

6.4.1 The Contest

The Big Maps Contest was carried out during the same period as the summative user study described in Section 6.3. At the time, 81 individuals were registered as iMapping users and all of them received an invitation to participate in the contest. These registered users were all those that had asked for the (otherwise hidden) download link in the past half year. Of course, this includes many people that have never really used the tools. In order to give an incentive, it was promised to include the submitters of the three largest iMaps in the lottery drawing for an iPad, together with the participants from the user study mentioned above.

From the 81 users that received the invitation to participate in the contest, 10 submitted a map. One other user reported to have an iMap in use, which he could not submit because it contained company secrets. One of the submissions is not considered because it consisted of software-generated files that contained converted imports of parts of the gene ontology [10]. Since it had not been user generated with the either of the two tools, it does not give any insight on actual map and tool usage.

6.4.2 Topics of the submitted iMaps

The nine iMaps varied largely in size, depth, style and topics: Four were private maps covering everyday information items like

- To-do lists
- Course planning notes
- Product ideas
- Japanese vocabulary

[10] http://www.geneontology.org/

The other five maps were more domain specific:

- About the design of QuadCopters (Figure 6.11)
- Concepts of a vampire story
- Draft of a software tool
- Lists of countries and international organizations
- The history of space science

The most sophisticated iMap submitted to the big maps contest is shown in Figure 6.11. All other user generated maps, except those containing personal information, are included in Appendix A in overview and with some close-up clippings.

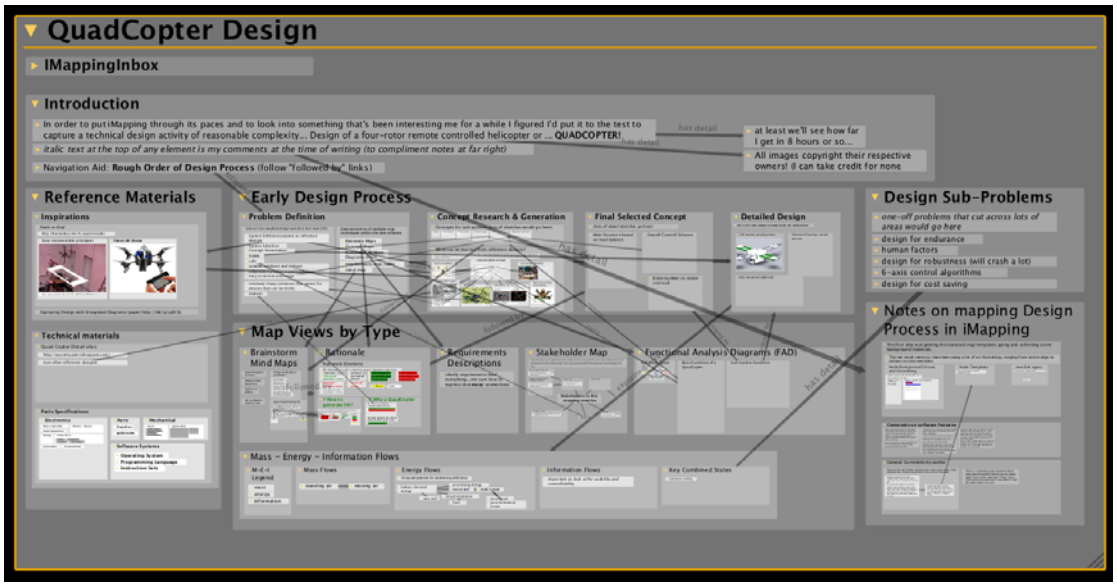

FIGURE 6.11: On the Design of QuadCopters: One of the biggest and certainly the most multifaceted iMap submitted to the Big Maps contest, here shown with all links visible. An overview over more user generated maps and some close-up clippings can be found in Appendix A.

6.4.3 Map Characteristics

An overview over some statistical measures of these nine user generated iMaps can be seen in Table 6.11, which also shows the same measures for the author's own personal iMap – the map that was also used in the systematic user study described in Section 6.3.

6.4.4 Use of Semantics

To determine the extent to which iMapping users make use of the semantic modeling features, the following statistical measures of their iMaps give some insight:

TABLE 6.11: Statistics on User Generated Maps: Aggregated map characteristics for the nine user maps (average, standard deviation and range) and corresponding values for the author's own iMap.

Measure	Mean	SD	Min	Max	Author's
No. of iMap items (incl. equivalents)	309.3	232.9	51	840	4 668
No. of equivalent items	79.8	141.5	0	338	2 497
No. of user-made relation types	19.6	21.5	0	59	240
No. of links with user-made relation types	58.0	62.3	0	184	674
No. of links with built-in relation types	151.3	230.5	0	746	4 494
No. of visual links	119.0	119.9	0	336	1 599
No. of additional links	90.3	246.3	0	746	3 569
Average no. of links per item	0.7	0.7	0	2.4	1.1
Average item length (characters)	38.8	37.8	10.5	130.4	30.1
Maximum item length (characters)	906.8	1 363.3	24	4 246	1 361
Average hierarchical depth	3.4	1.6	1.1	6.1	6.6
Maximum hierarchical depth	6.2	2.9	2	11	11

The average number of 20 user-made relation types per user shows that users create their own relation types. The three times higher number of links with these user-made relation types (58, find both in Table 6.11), shows that they also re-use them. However, none of the nine user-generated iMaps contained user-made relation types that had explicitly specified superrelations or subrelations of other relation types. Also, only one user had renamed some of the default inverse relation names, which would be useful when links are used on both directions not only for visual browsing but also for semantic modeling or queries. This suggests that none of the participants of this contest has built very sophisticated semantic models that would include a hierarchy of relations.

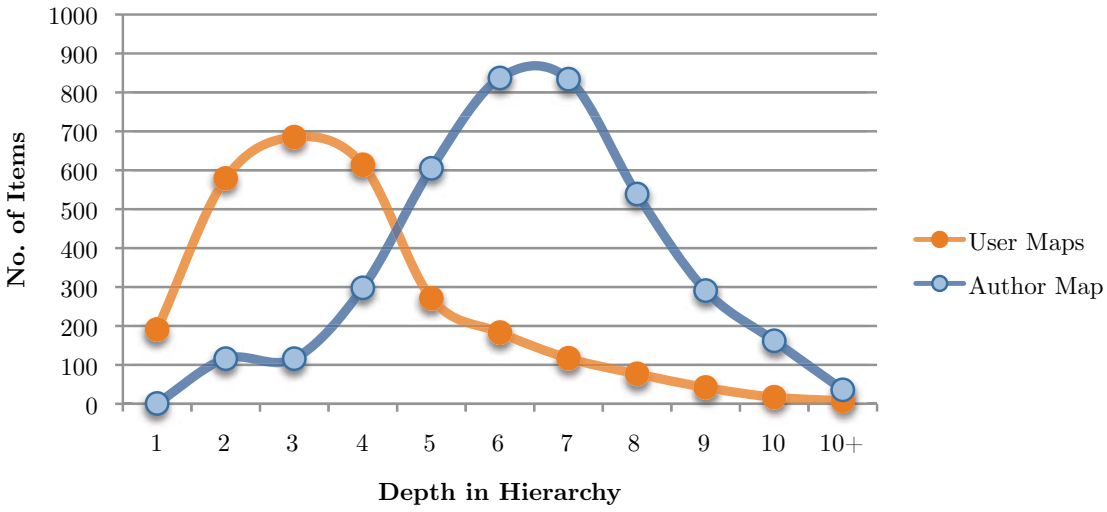

FIGURE 6.12: Hierarchical Depth of Items: Distribution of items' hierarchical depth – summed up over all nine users' maps (which had grown over 1–10 weeks) compared to the author's map (which has grown over 16 months).

On the other hand, in a survey, 5 out of 10 actual users reported to use and value at least one of the semantic features (like inverse relations, subtypes/subrelations, queries or RDF export). Two more users reported to use these features but would not really miss them. This survey had been sent out to 100 registered iMapping users with the incentive of receiving the latest version of the tools after filling it out. Of these, 26 users actually responded. The 10 of these respondents counted above were those that had stated to seriously use the tools regularly or at least sometimes, as opposed to only exploring it or not using it at all.

6.4.5 Conclusions

The fact that 12% of all registered iMapping users actually sent in their iMaps, suggests that the application is actually being used, which is also in line with the responses to the online questionnaire. That, after only two months, most of these maps contained several hundreds of items in hierarchies of up to eleven levels deep, furthermore proves that iMapping can successfully be used to maintain large collections of information items. Also, the high number of links shows, that iMapping supports the construction of link structures well. These findings are backed by the author's own experience of using both iMapping and QuiKey as every-day PKM tools for one year and a half and thereby building a well cross-linked and continuously growing iMap of several thousand items.

The relatively rare use of the semantic features may be due to poor documentation and instruction for these features. If they contribute to improved PKM, remains subject to future investigations.

Chapter 7

Conclusion

This last chapter has three sections: Section 7.1 gives an overview of how the requirements have been fulfilled by the design of iMapping and QuiKey. Section 7.2 gives an outlook on possible future work and Section 7.3 finally summarizes the results and contributions of this thesis.

7.1 Conformance with Requirements

This section gives an overview on how the iMapping technique combines the core benefits of the other techniques discussed in Chapter 2, how the requirements defined in Chapter 3 have been met by the design of the iMapping technique, the design of the iMapping tool and its actual implementation (all described in Chapter 4) and how the design and implementation of QuiKey also fulfill their design goals and functionalities as described in Chapter 5.

Table 7.1: iMapping and Other Techniques in Comparison: iMapping unites the core benefits of the three basic mapping approaches, as discussed in Section 2.3.

	Mind-Maps	Concept Maps	Spatial Hypertext	iMaps
Structural analogy to content	✓	✓	✓	✓
Simple hierarchical topology	✓	–	–	✓
Representation of interrelations	–	✓	–	✓
Constructive ambiguity	–	–	✓	✓
Scalability	–	–	–	✓

As detailed in Section 4.1 and summarized here in Table 7.1, iMapping as a technique unites the core benefits of the three basic visual knowledge mapping techniques as introduced in Section 2.3: Both concept mapping and spatial hypertext are completely

TABLE 7.2: iMapping Requirements and Solutions: List of requirements from Chapter 3 and how they are addressed by the design and implementation of iMapping and QuiKey.

Requirements for Techniques	**Design Solution**	**Implemented**
Free Placing	✓ User defined layout	✓
Free Relations		
Formalized Links	✓ Arrows with formal relation types	✓
Informal Labeled Links	✓ Arrows with free relation types	✓
Unlabeled Links	✓ Arrows without label	–
Free Nodes	✓ Items without arrows	✓
Annotations	✓ See mock-ups (Section 4.1.4)	–
Hierarchy, Abstraction, Overview	✓ Nesting & zooming	✓
Scalability	✓ Infinite space through nesting & zoom	✓

Requirements for Tools	**Design Solution**	**Implemented**
Simple Editing		
Brainstorming	✓ Rapid entry shortcuts	✓
Extending Maps	✓ Auto grow	✓
	✓ Magnetic lists	–
Incremental Formalization	✓ (see Section 4.2.7)	✓
Filter and Focus		
Focus	✓ Zoom to fill window	✓
	✓ Expand / collapse	✓
	✓ Semantic zoom	–
Filter	✓ Semantic queries in QuiKey	✓
Integration of Detail & Context		
Local	✓ Smooth zoom	✓
	✓ Easy hierarchy navigation	✓
Global	✓ Zoom to top and back	✓
Accessing External Content	✓ Adopted content	–
	✓ Presentable content	Images only
	✓ Alien content	–
Interoperability	✓ Open and extendable formats	✓
Mobility	✓ Fine granular items	✓
	✓ Zooming suitable for small screens	✓
	✓ Content-only screen design	✓
	✓ Minimalist interface in QuiKey	✓
	✓ Efficient text interaction in QuiKey	✓

subsumed by the iMapping technique. It thus combines the possibility of graph-based representations of interrelation, with the constructive ambiguity of unconnected freely placed items. At the same time, iMaps provide a simple hierarchical overall topology like mind-maps. As this is realized by deep zooming and nesting, the iMapping approach scales up to very large maps that are conceptually unlimited in size. All in all, visually modeling content in analogy to its inherent structure is supported by the relatively unrestricted visual modeling capabilities.

Table 7.2 lists the requirements defined in Chapter 3 and shows by which design solutions they have been addressed. All of these design solutions have been explained in Sections 4.1 and 4.2. An additional column indicates which of these design solutions are implemented in the current version of the software. A more detailed overview on the implementation status of all design solutions has been given in Tables 4.2 and 4.3 in Section 4.3.2. As can be seen, all requirements have been addressed by one or several design solutions. And most of them have also been implemented in the current version of the two tools. Those that have not been implemented so far do not affect the core concept and have not been necessary for evaluating the technique or for basic usage of the tools.

TABLE 7.3: QuiKey Requirements and Solutions: All requirements have been addressed and implemented.

Design Goals	**Design Solution**	**Implemented**
Efficient interaction	✓ Finding items by advanced auto-completion	✓
Avoidance of errors	✓ Selecting instead of spelling	✓
	✓ Short feedback cycles	✓
	✓ Interactive query construction	✓

Functionalities	**Design Solution**	**Implemented**
Text search	✓ Advanced autocomplete matching patterns	✓
Targeted search of linked info.	✓ Item → relation → item → relation ...	✓
Information entry		
Creating new items	✓ `New content`, Enter.	✓
Creating new links	✓ Item → relation → item, Enter.	✓
Creating new relation types	✓ Item → `new relation name` → item, Enter.	✓
Editing item content	✓ Item → *rename to* → `new content`, Enter.	✓
Set-based browsing	✓ Item → relation → relation ...	✓
Formulating simple queries	✓ is done by browsing.	✓
Saving queries	✓ Item → relation → `?name of query`	✓
Incr. constr. complex queries	✓ Saved query → *AND* → other saved query	✓

Table 7.3 summarizes how the two design goals for QuiKey have been met and by what interaction patterns the designated functionalities have been covered. All functionalities of QuiKey that have been discussed have also been implemented.

For both iMapping, and QuiKey, areas that require further research and design work are discussed below in Section 7.2.

7.2 Outlook

This section touches on some areas of possible future work.

Future Work on iMapping

Some features like annotations, query items or magnetic lists have not been implemented so far (see Section 4.3.2) although they have been already been designed. In other areas, however, more research and design work would be needed in order to further improve iMapping. The following are some of these areas.

Spatial Search and Filters Search result sets from either keyword search or semantic queries could be shown in the map or in a spatial distribution derived from their positions in the map. Spatial filters could limit the search and queries to certain *areas* of a larger map.

Spatial Parsing Like described by Marshall and Shipman (1993), a spatial parser could extract implicit structures like sequence and proximity and make them explicit to provide additional functionalities, e. g., automatic serialization of a set of items.

Formalization Aid An unobtrusive annotation assistant could propose incremental formalizations. Such suggestions could come from spatial parsing or could be guessed from pre-existing semantic information – e. g., when an item representing a person is dragged into an item representing an event, probable semantic annotations would be *participant of* or *organizer of*.

Fish Eye Views It could be investigated, whether FEVs (as discussed in Section 3.2.3) that are specialized for nested graphs (Noik, 1993) could improve the visual exploration of iMaps.

Technology Assessment Another interesting field of research that has not been addressed in this thesis is the question raised in Chapter 1: How do certain aspects of knowledge media *influence* (distort, limit, support) the thinking processes they support?

Future Work on QuiKey

Topics for future work on QuiKey include

- Providing a way to interactively construct complex queries from scratch without the need to name and save intermediate queries.
- Supporting more expressive query operators including negation.
- Explaining the meaning of query items using language patterns.
- Displaying short explanations during interaction, about what is going to happen, before an action is executed.
- Giving unobtrusive confirmations on completed actions.
- Improving text matching rules during use by adapting their ranking.
- Automatically learning shortcuts for frequently used items.
- History and undo functions.
- Using QuiKey as an additional front-end to existing semantic tools like Semantic MediaWiki.
- Developing a mobile version, where some of QuiKey's minimalist interactions have a bigger benefit, because typing is more costly.

7.3 Summary

This thesis has described the design of two novel interaction approaches for personal knowledge management: iMapping, a visual knowledge mapping technique based on nesting and deep zooming, and QuiKey, a lightweight interactive text-based tool for text search and fine-granular access to structured content.

The goal of this thesis is to improve tool support for personal knowledge management (PKM) in a cognitively adequate way (Chapter 1). To this end, existing PKM and knowledge mapping techniques have been analyzed, in particular the seminal techniques mind-mapping, concept mapping and spatial hypertext. Together with modern computational approaches like zooming user interfaces (ZUI) and semantic desktop technologies, they form the foundations (Chapter 2) of this work.

Based on these seminal approaches, two sets of requirements have been extracted, one more generally for the design of visual knowledge mapping *techniques* for PKM, and one more concretely for the design of such *tools* in particular (Chapter 3). These requirements can be used as evaluation criteria or as a guideline for the design of future visual knowledge mapping techniques and tools.

Chapter 4 has described the design and implementation of iMapping: The design of the general iMapping technique (Section 4.1) and the design of the concrete iMapping tool (Section 4.2) are based on the requirements mentioned above.

The iMapping technique combines the core advantages of the seminal existing approaches, namely:

- Maps with a structural analogy to their content
- A simple hierarchical overall topology
- Graph-based representations of interrelation
- The freedom to model informally with constructive ambiguity

Moreover, unlike the classical approaches, the iMapping technique scales up to large map sizes that are conceptually infinite by using a structure based on nesting and zooming (Section 4.1.1). This is crucial for dealing with large amounts of information items as they accumulate over the years of practicing PKM. iMapping also allows to interlink information items in various ways ranging from informal associations to formally defined semantic links (Section 4.1.2) and provides several means to avoid visually complex tangled links (Section 4.1.3). Through its structure based on nested boxes, the iMapping technique is also conceptually compatible to many other visual techniques, some of which it completely subsumes (Section 4.1.5).

Additional to developing iMapping as an abstract technique, an actual iMapping tool has been designed and implemented. The design challenges for this iMapping tool, like navigation issues, integrating details and contexts and different kinds of item equivalences have been described in Section 4.2 and its implementation in Section 4.3.

QuiKey (described in Chapter 5) is a lightweight text-based tool, which complements this visual mapping approach. QuiKey offers several functionalities to access graph-structured knowledge bases:

- Text search based on advanced auto-completion
- Browsing structured content from single item to single item
- Browsing sets of semantically interrelated items from set to set
- Editing the knowledge base in several ways
- Constructing and saving simple semantic queries
- Constructing complex semantic queries from existing simple ones

QuiKey's contribution is to provide these functionalities in a way that is highly interaction efficient.

Both tools have been implemented in Java and are based on semantic desktop technologies such as the Conceptual Data Structures framework. They are open source and run on all major platforms. To date, the tools have been downloaded more than 100 times and are in frequent use by at least 10 people.

In order to prove the usefulness of the interaction concepts of iMapping and QuiKey, several evaluation studies have been carried out, where a total of 66 users were presented with the implementations of the two tools (Chapter 6). Their main results can be summarized as follows:

- Both tools received very good ratings on the User Experience Questionnaire (Section 6.3.4). Especially QuiKey has been confirmed to be highly interaction efficient – theoretically (Section 6.2.2) as well as empirically (Section 6.2.3) and subjectively (Section 6.3.4). But also iMapping is in its interaction efficiency comparable to market leading mind-mapping software and even was subjectively preferred over such by all 12 participants asked (Section 6.1).
- Visually exploring link structures is much more efficient when links are shown on demand only, as it is done per default in iMapping as opposed to having all links visible at the same time.
- While all tasks could be completed in each of the tools alone, QuiKey is more efficient for finding items and iMapping is better for getting an overview. For combined

tasks that require both searching and overview, using both tools in combination is more efficient than using either of them alone (Section 6.3.4).

– An analysis of iMaps that have been sent in by actual iMapping users has shown that iMapping can in fact successfully be used to model and maintain large collections of interrelated information items. iMaps have been used for quite diverse topics ranging from personal notes and vocabulary over story writing to aerospace construction design (Section 6.4).

Semantic knowledge management technologies might in the future be able to significantly increase interoperability between applications even across context boundaries and they might leverage the creative and productive power of knowledge workers. However, if knowledge management does not start at the personal level and with providing immediate benefit to the *individual* knowledge worker, it is questionable whether semantic knowledge management systems will ever spread very far.

iMapping-based user interfaces could be a step forward to make highly structured knowledge management easier and more intuitive. Especially for large knowledge repositories, with highly interlinked information items of different levels of formalization, iMapping-based user interfaces can make knowledge management more intuitive, on one hand, and more powerful, on the other hand, because it provides a medium that fosters the use of visual orientation in a flexible and easy way and yet supports more structured knowledge modeling in just the desired degree of formalization.

The contribution of this thesis is fivefold:

– The set of requirements
– The iMapping technique
– The iMapping tool
– The QuiKey concept
– The QuiKey tool

While the two tools introduced in here are under continued development, it is encouraged to transfer the design solutions developed in this thesis to other applications, so the general approach can find more widespread application.

For up to date information on iMapping and QuiKey and to download their latest implementations, see `http://imapping.info/` and `http://quikey.info/`.

Appendix

Appendix A

User Generated Maps

The following pages show an overview over user generated iMaps submitted by participants of the Big Maps Contest detailed in Section 6.4, as well an iMap by the iMapping development team and the author's own iMap. Each iMap is presented in overview, i. e. fully zoomed out, and with at least one zoomed in detailed section.

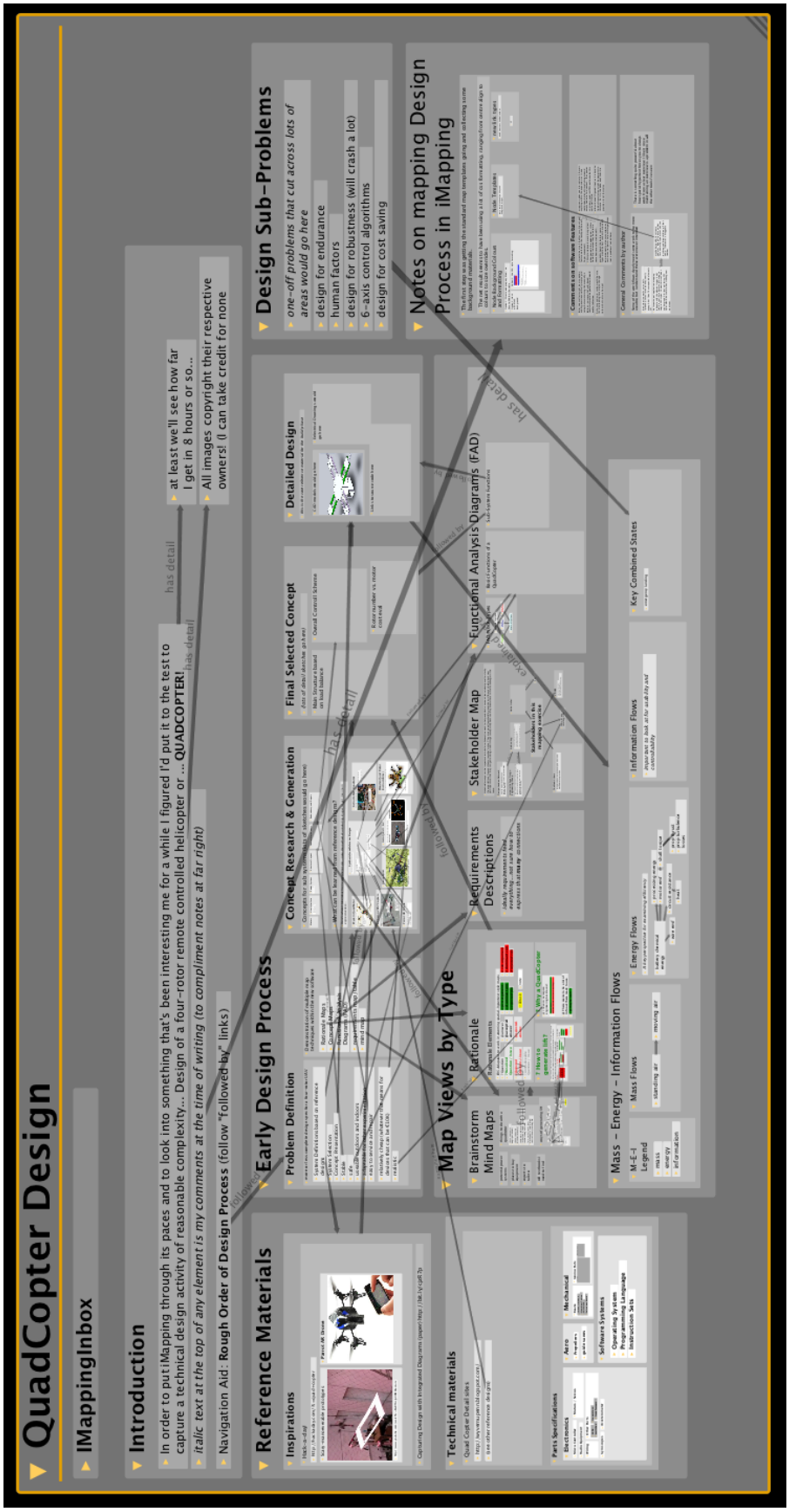

FIGURE A.1: On the Design of QuadCopters: One of the biggest and certainly the most multifaceted iMap submitted to the Big Maps contest by Nathan Eng, here shown with all links visible. Details of this map are shown in subsequent figures.

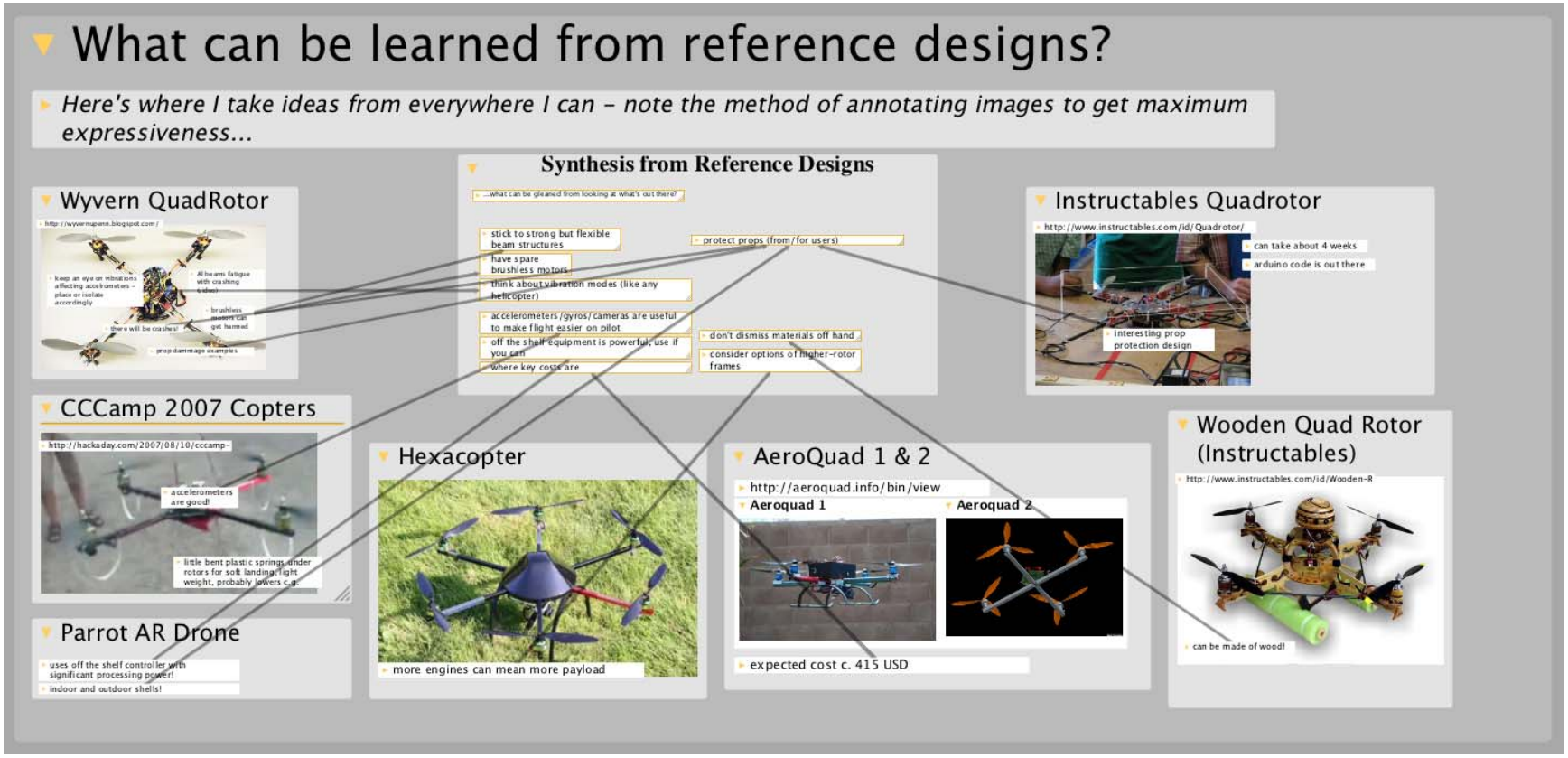

FIGURE A.2: Annotated Images: Clipping of the QuadCopter iMap showing a list of lessons learned with references to accompanying images.

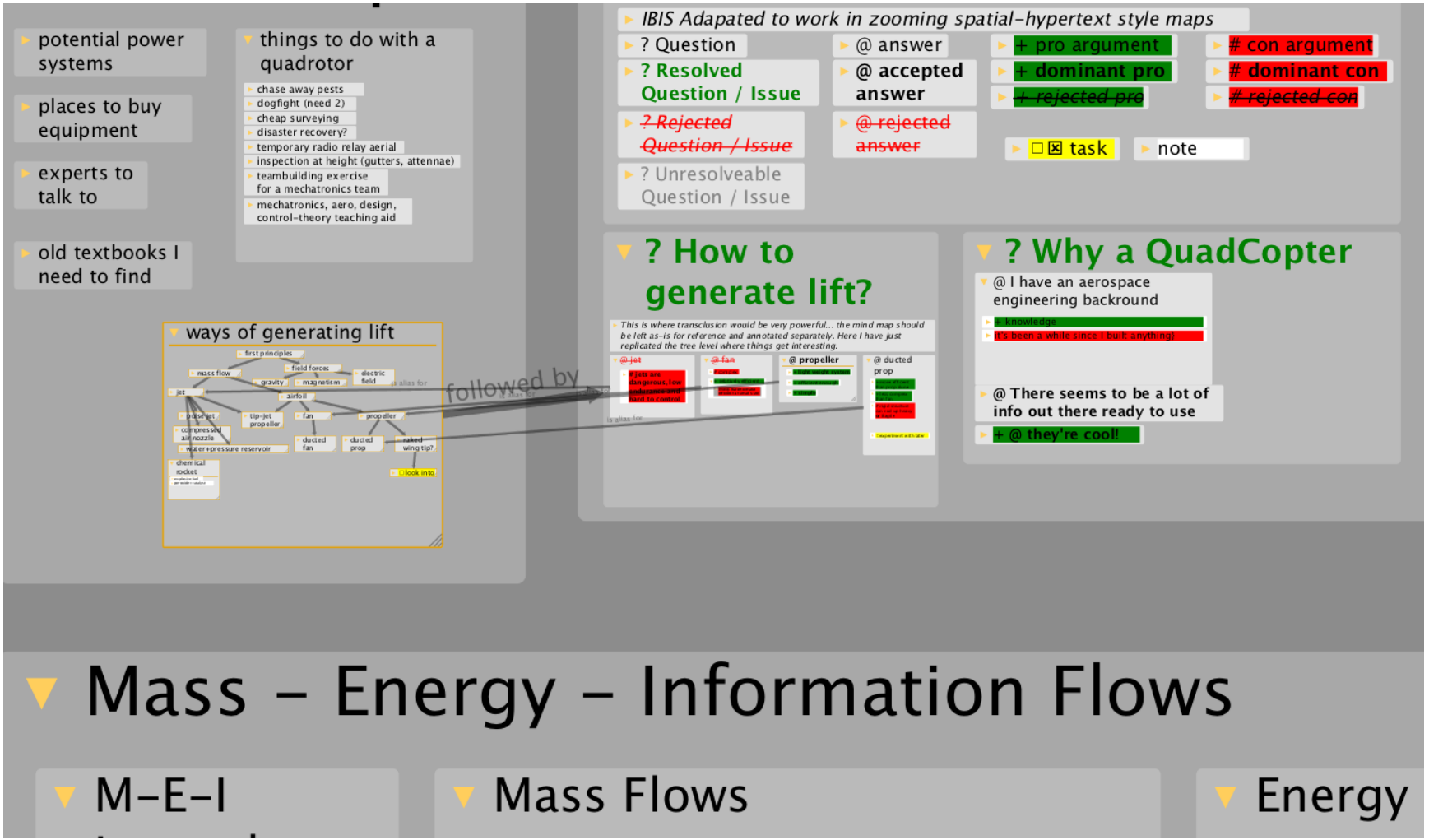

FIGURE A.3: Issue Mapping: Clipping of the QuadCopter iMap showing an example of how IBIS-based (Kunz and Rittel, 1970) issue mapping (Okada et al., 2008) adapted for iMapping by the user.

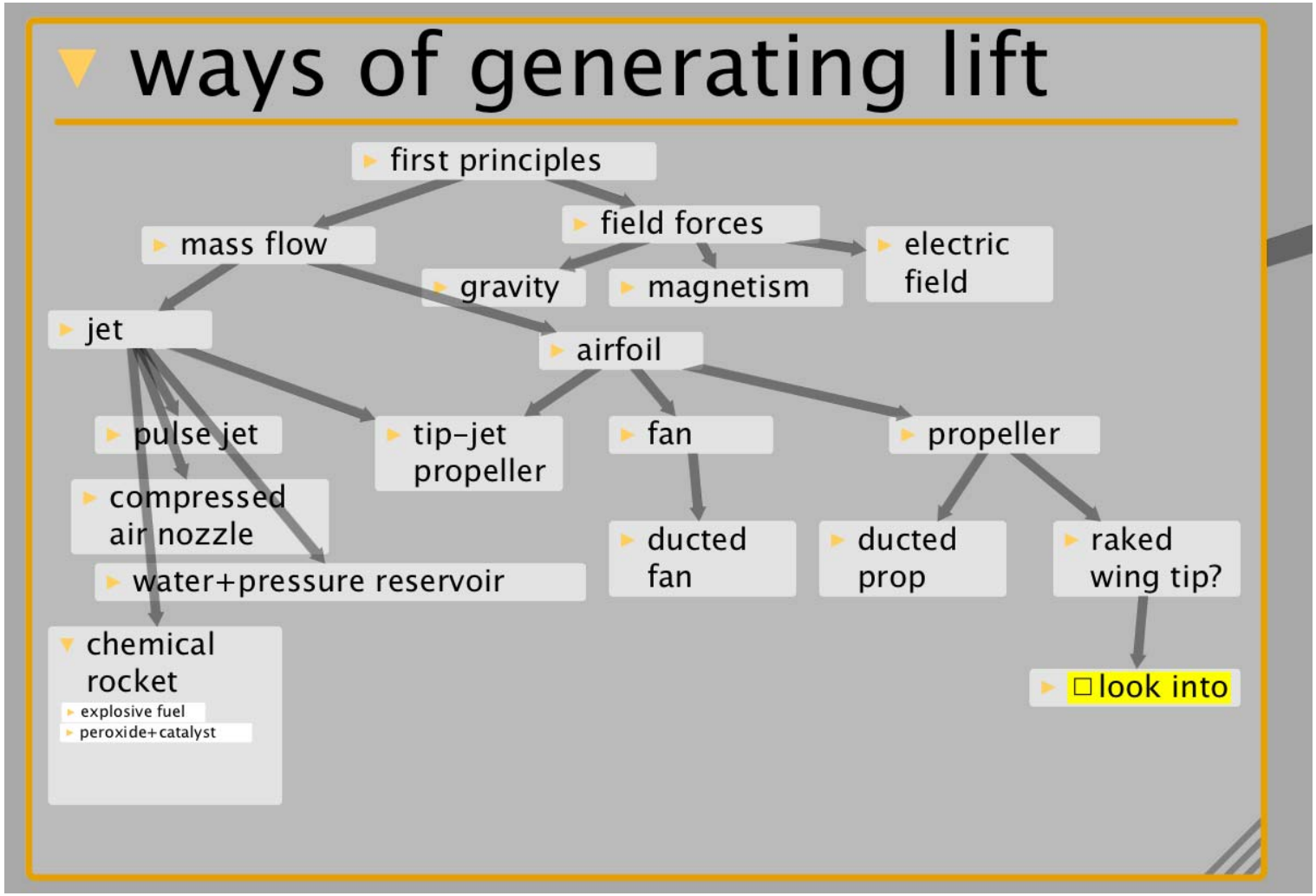

FIGURE A.4: Generating Lift: Clipping of the QuadCopter iMap.

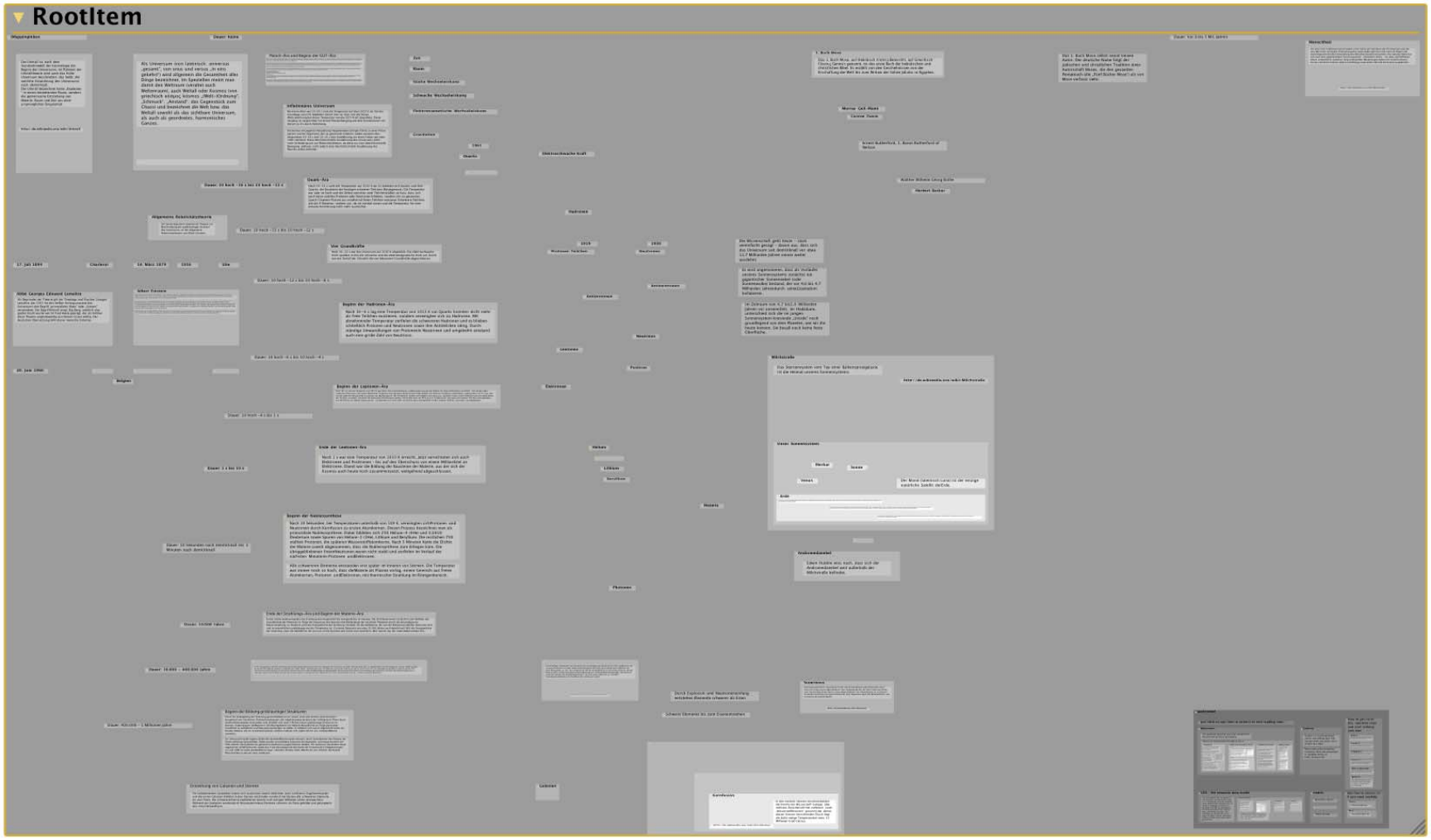

FIGURE A.5: The Universe: iMap about the history and science of the universe. Detail below. (Note how the original start Map that comes included with the tool is kept for reference in the bottom right corner)

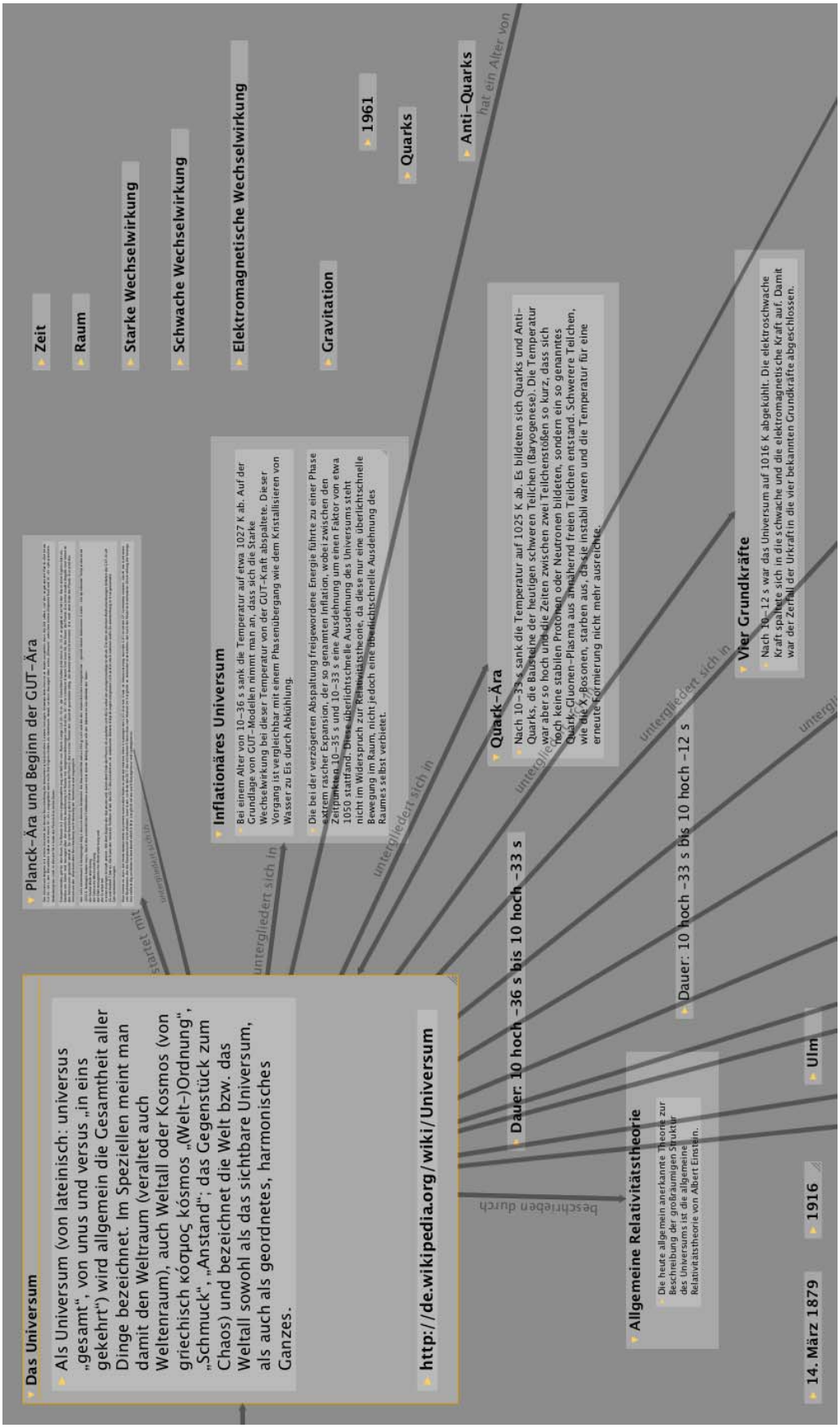

FIGURE A.6: Linked Instructional Text: Detail of the Universe iMap showing instructional text passages spatially arranged and highly interlinked. (Only links from and to one central item are shown here)

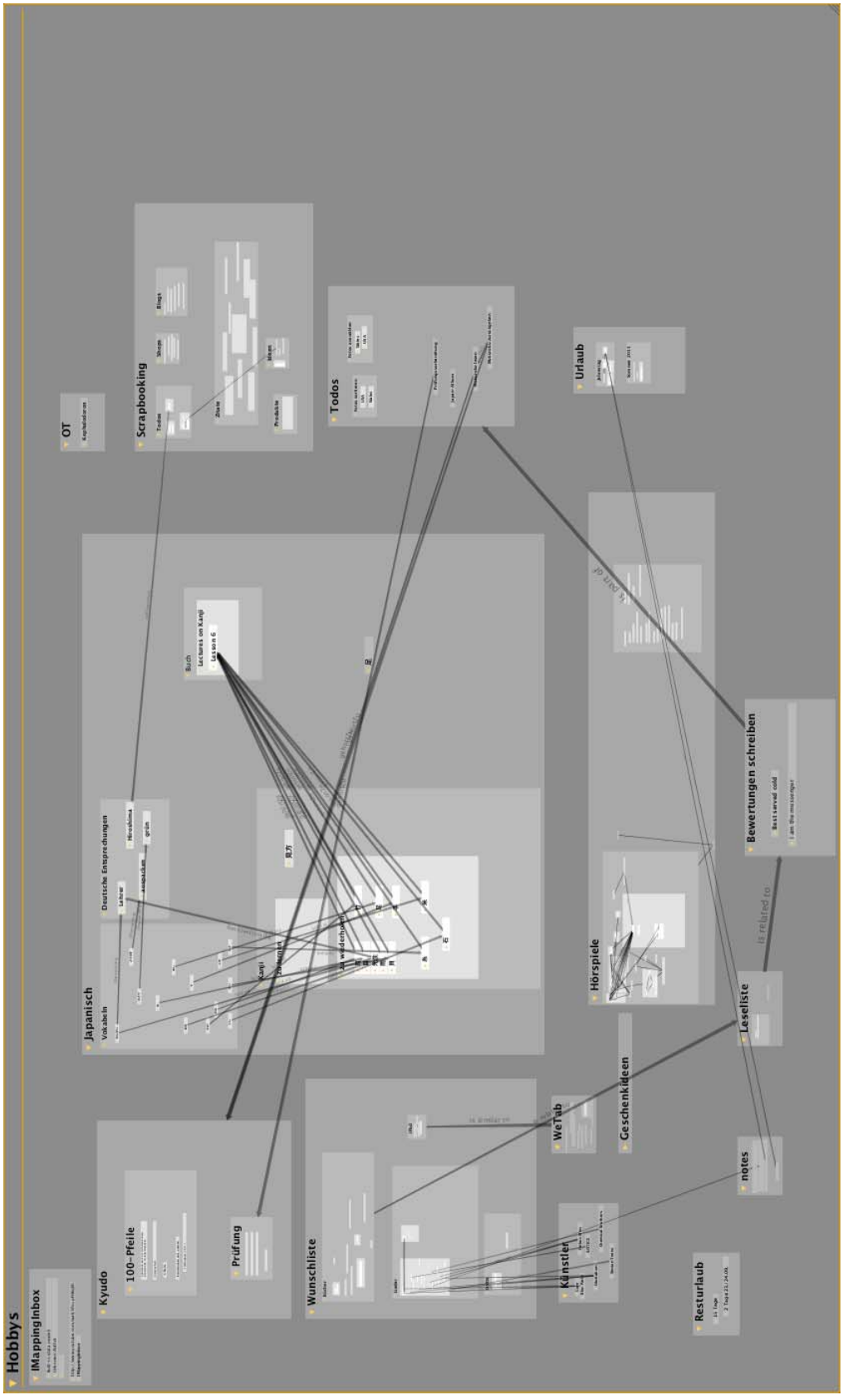

FIGURE A.7: Personal Notes: A genuine PKM iMap about everyday notes – from hobbies, personal tasks, and reading lists to training notes and Japanese vocabulary.

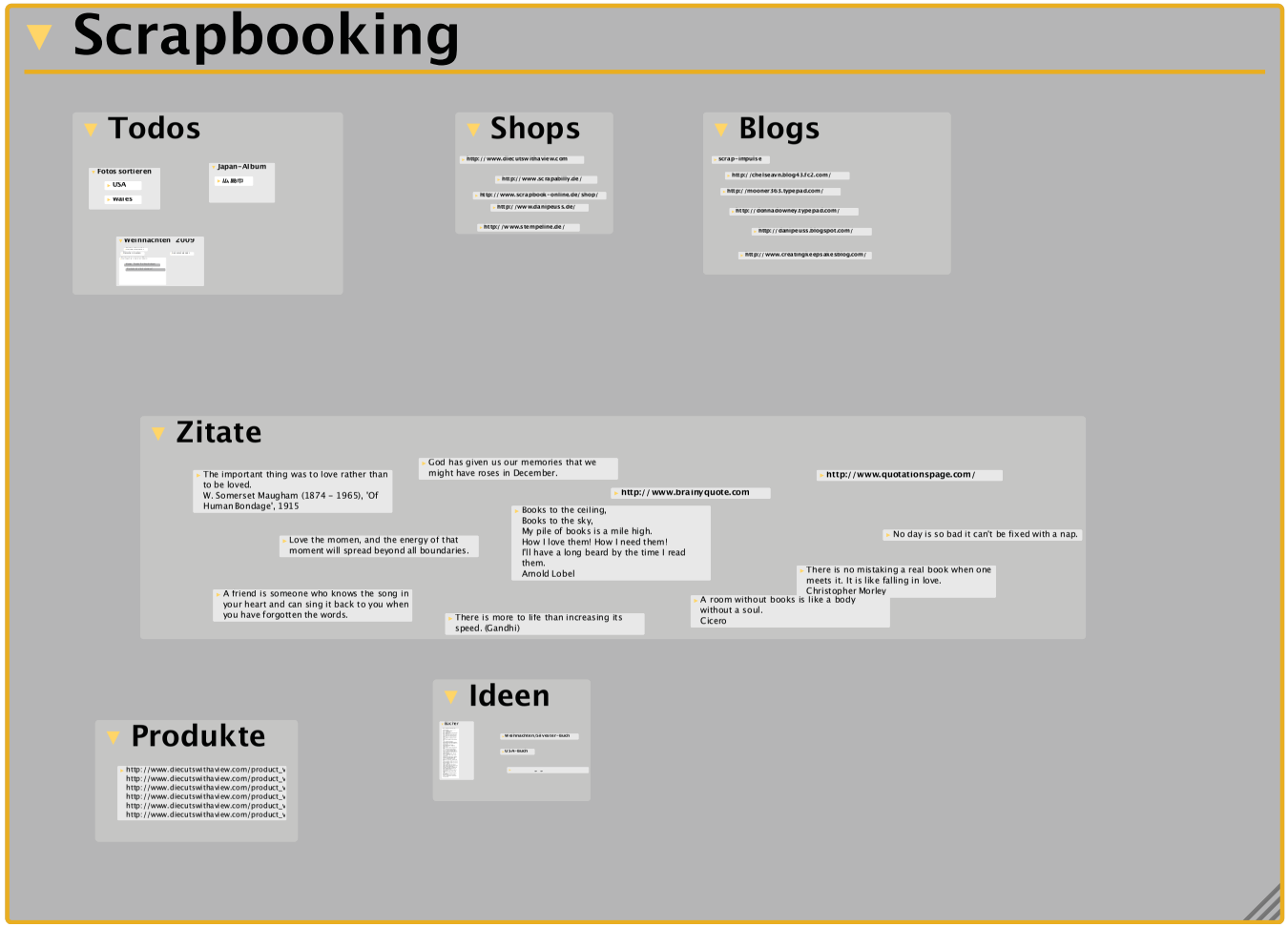

FIGURE A.8: Personal Notes: Detail clipping from the map in Figure A.7

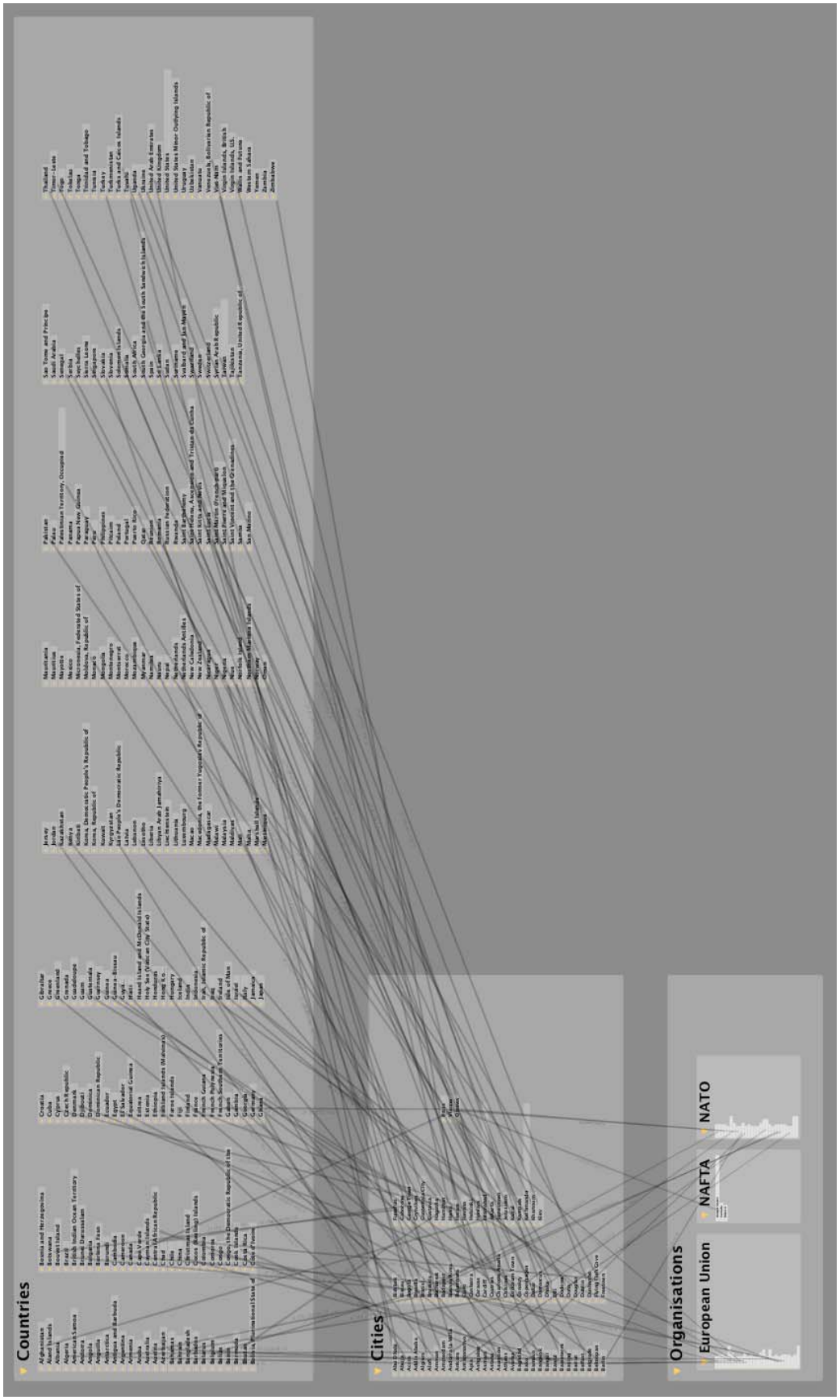

FIGURE A.9: Countries: iMap of countries, their capitals and international organizations. Detail below.

FIGURE A.10: International Organizations: Clipping of the countries iMap showing confederations of states. Some countries occur in several confederations as equivalent items.

FIGURE A.11: Vampire Story: An early iMap with only one hierarchy level. Simply a collection of concepts for a story. For a later state of the same iMap, see next figure.

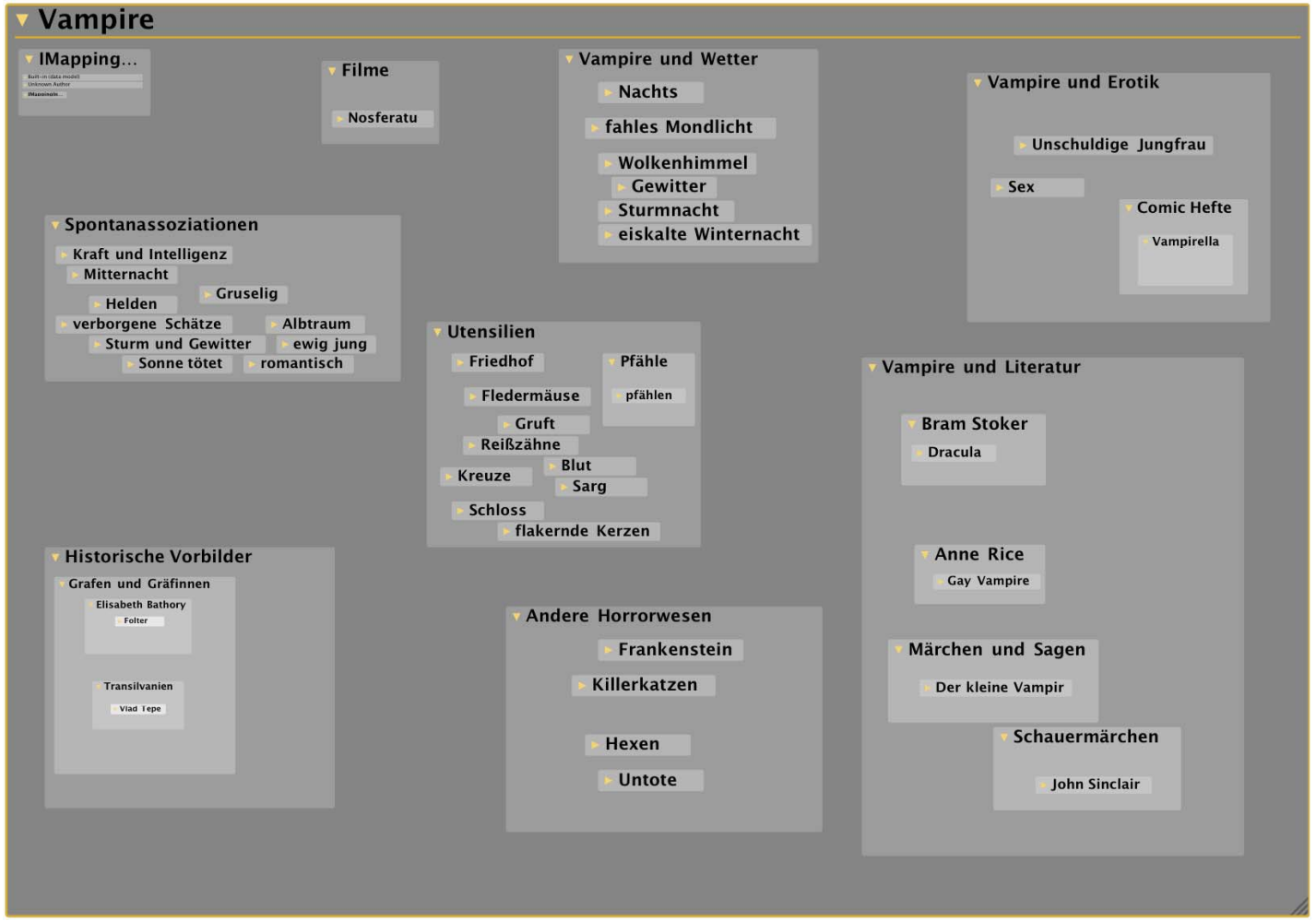

FIGURE A.12: Vampires Clustered: The same iMap as before but at a later state of work. Concepts have been clustered up to three hierarchy levels deep.

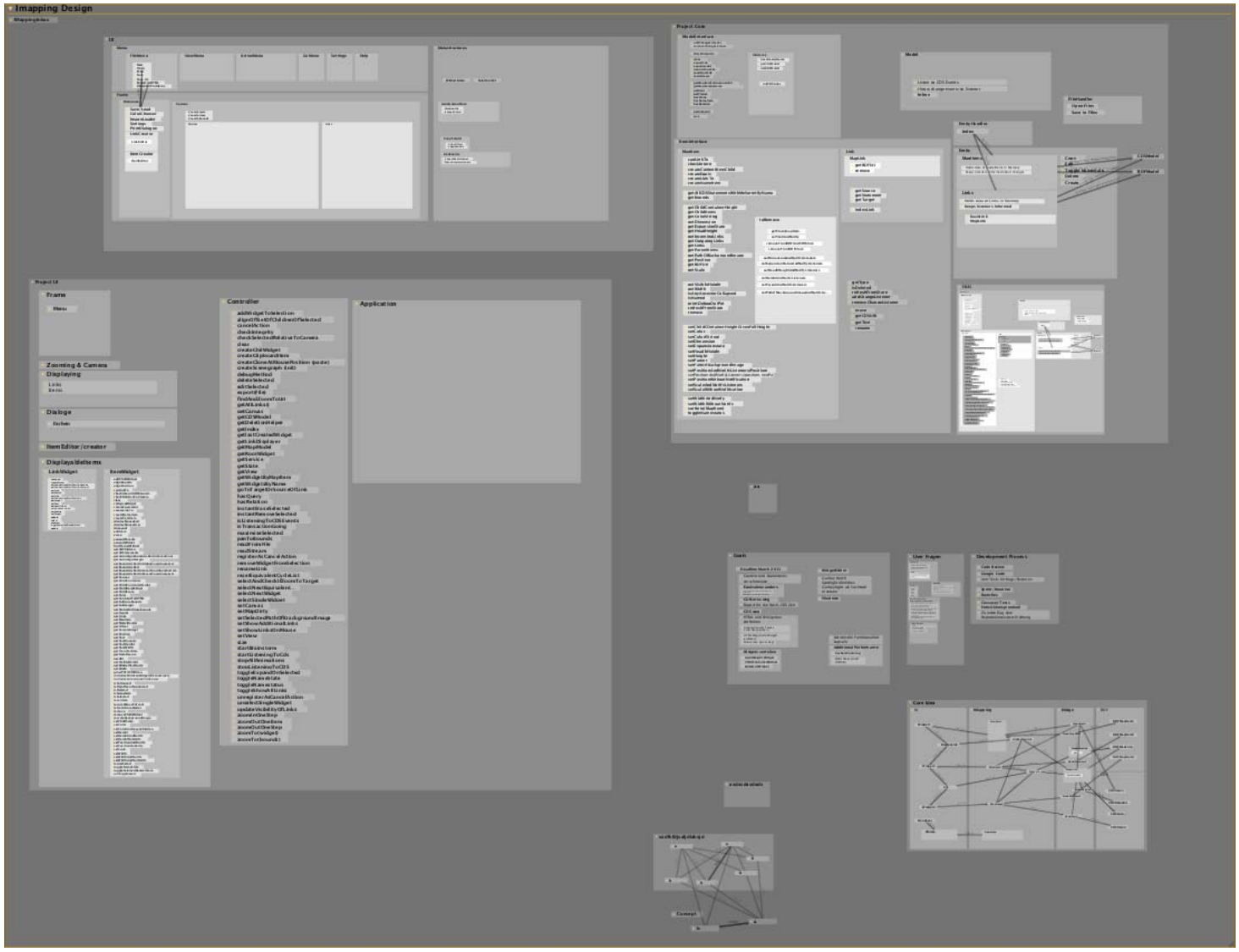

FIGURE A.13: iMapping Architecture: An iMap used by the software developers of the iMapping tool to get an overview over the current and planned software architecture.

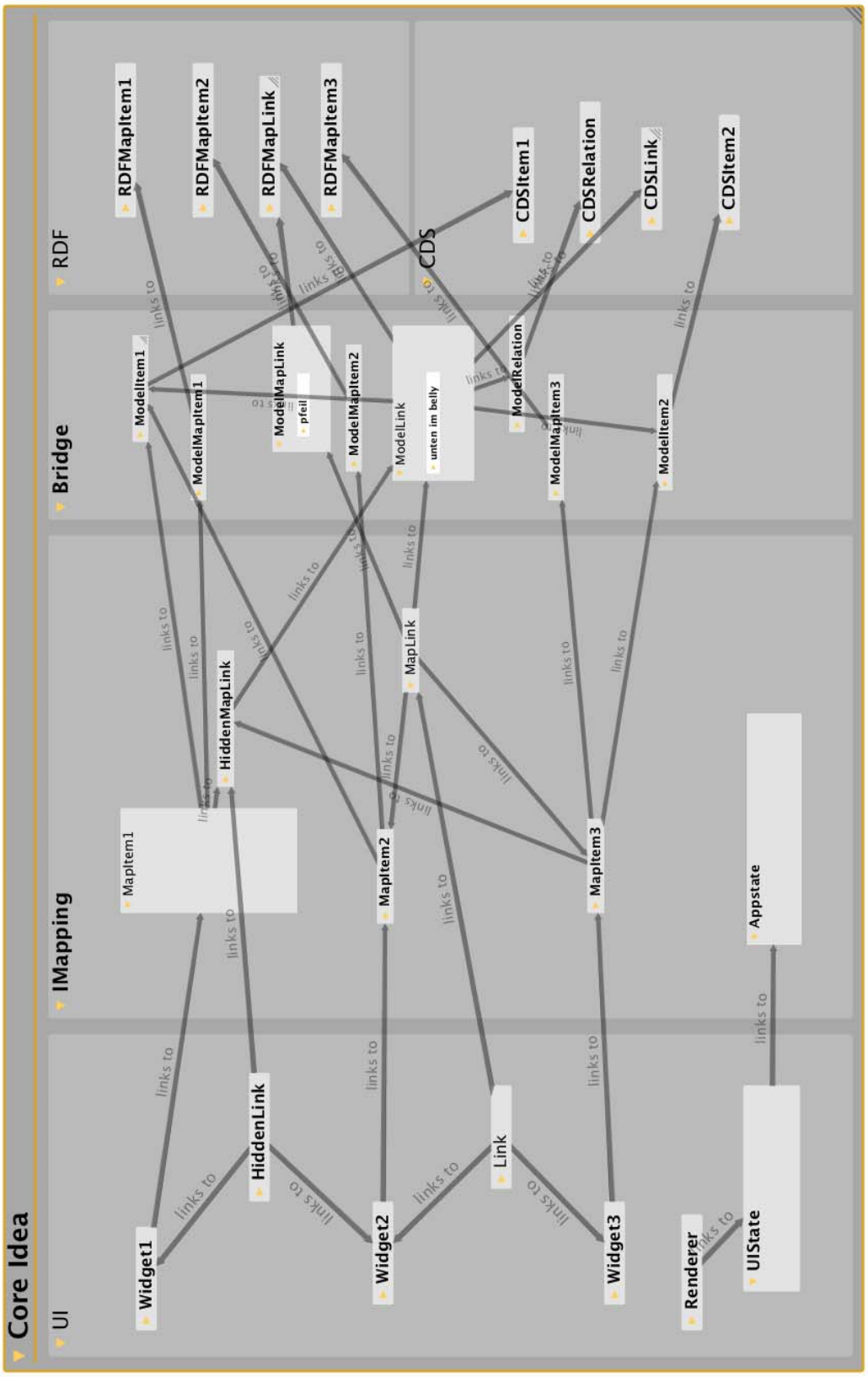

FIGURE A.14: iMapping Architecture: Detail clipping from the map in Figure A.13.

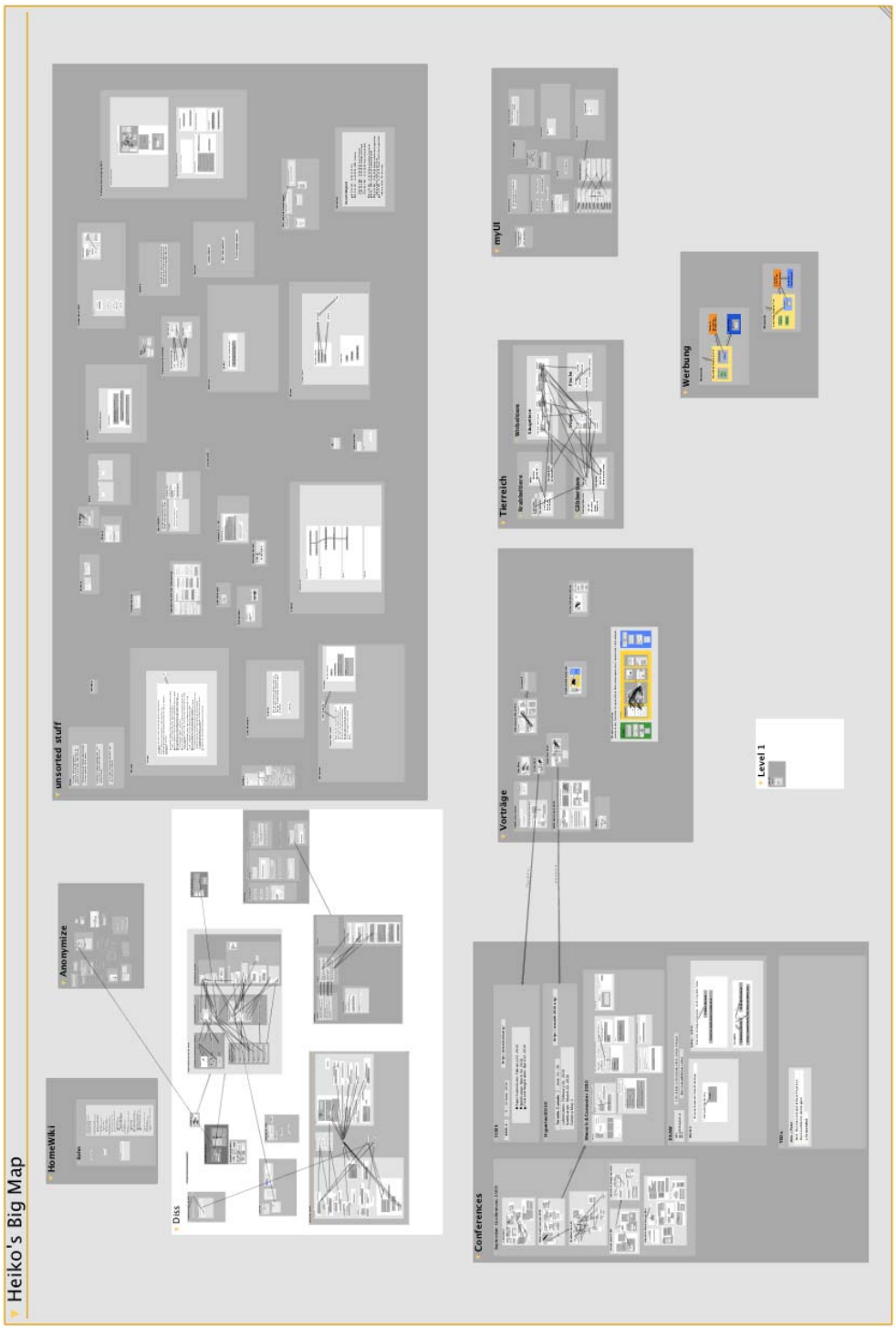

FIGURE A.15: The Author's Map: The biggest iMap to date. Used for PKM and Presentations for 16 Months. Printed to a size where everything would be readable, this iMap would be more than 100 meters high and wide. For statistics see Table 6.11.

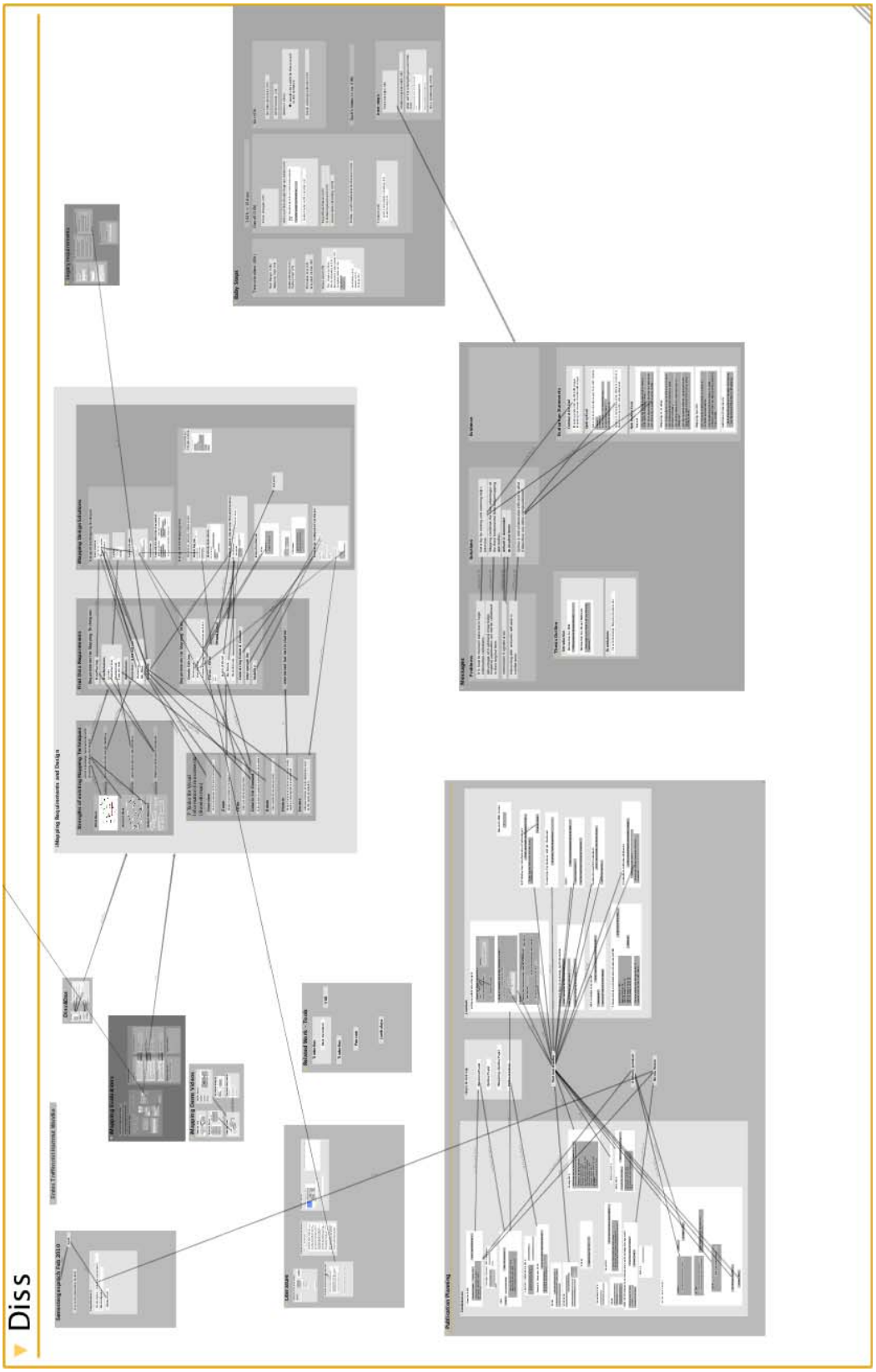

FIGURE A.16: This Thesis: Detail clipping from the map in Figure A.15. Most screenshots in this thesis are from this area.

Appendix B

Interpretation of UEQ Scores (Personal E-Mail Messages)

Original Inquiry

From: Heiko Haller [mailto:heiko.haller@fzi.de]
Sent: Mittwoch, 10. November 2010 19:29
To: Laugwitz, Bettina; Held, Theo; Schrepp, Martin
Subject: UEQ - durchschnittliche Scores?

Hallo,

vielen Dank fürs das Entwickeln des UEQ - ich habe ihn gerade für die Softwareevaluation meiner Diss eingesetzt.
Um die Ergebnisse besser interpretieren zu können, würe es mich allerdings interessieren, ob Ihr Vergleichsdaten dazu habt, wie andere Tools abschneiden.

Sind Werte zwischen 1.1 und 2.2 gut oder ist das Durchschnitt?

Könnt Ihr die Durchschnittswerte Eurer Evaluationserhebungen rausrücken? Habt Ihr noch weitere, von anderen Tools oder wisst Ihr, wer den UEQ noch angewendet hat? Am liebsten irgendwo in der thematischen Nähe von Semantischer Software für persönliches Wissensmanagement :)
Ist da irgendwas veröffentlicht?

Hoffnungsvolle Grüße -
Heiko

P.S. Meine Egebnisse - bei Interesse gerne auch Rohdaten - stelle ich natürlich gern zur Verfügung.

--
Heiko Haller
http://www.fzi.de/mitarbeiter/haller
tel: +49-721-2085100, fax: +49-721-9654-959
==========
FZI Forschungszentrum Informatik an der Universität Karlsruhe (TH)
Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Deutschland
Stiftung des bürgerlichen Rechts, Az: 14-0563.1 RP Karlsruhe
Vorstand: Rüdiger Dillmann, Michael Flor, Wolffried Stucky, Rudi Studer
Vorsitzender des Kuratoriums: Ministerialdirigent Günther Leßnerkraus
==========

Response

From: Schrepp, Martin
Sent: Thursday, November 11, 2010 16:21
To: Heiko Haller
Cc: Held, Theo; Laugwitz, Bettina
Subject: RE: UEQ - durchschnittliche Scores?

Sehr geehrter Herr Haller,
wir haben zwar eine Reihe von Vergleichswerten, leider können wir Ihnen diese nicht zugänglich machen. Zum Teil handelt es sich um Untersuchungen zur Attraktivität von verschiedenen SAP Softwareprodukten, die natürlich rein intern sind und nicht öffentlich zugänglich gemacht werden können. Andere Untersuchungsergebnisse sind uns von Anwendern des UEQ nur für interne Untersuchungen zur Konsistenz der Skalen zur Verfügung gestellt worden. Auch diese Daten können wir natürlich nicht weitergeben.

Allerdings kann ich ihnen sagen, dass Werte > 1 für eine gute Qualität bzgl. der jeweiligen Skalen sprechen. Man hat bei Fragebögen dieses Typs immer eine gewisse Tendenz zur Mitte (d.h. viele Personen vermeiden eher extreme Bewertungen), so dass man schon wegen dieser Antworttendenz nicht mit Werten deutlich >2 rechnen kann (selbst wenn das Produkt sehr gut ist).

Falls wir Werte >1 messen spricht das bei unseren Produkten für eine gute bis sehr gute Qualität (ich weiss auch von anderen Anwendern, die ihre Produkte regelmäßig mit dem UEQ untersuchen, dass diese eine analoge Interpretation vertreten). D.h. wenn Sie für alle Skalen Werte zwischen 1.1 und 2.2 gemessen haben, können Sie sehr zufrieden sein. Das sind dann schon Spitzenwerte!

Ich hoffe, dass Ihnen diese Antwort weiterhilft. Wir wären natürlich sehr an Ihren Daten interessiert. Wir werden diese natürlich absolut vertraulich behandeln und nur zur Überprüfung der internen Konsistenz unserer Skalen verwenden. Könnten Sie mir die Daten zur Verfügung stellen und evtl. einige Hintergrundinfos zur untersuchten Software?

Viele Grüße,
Martin Schrepp

Appendix C

Data-Analysis: Additional Material

The following two pages show additional statistics on the data obtained in the summative long-term user study detailed in Section 6.3:

Table C.1 shows significance tests for the normality of distribution of the variables used. Because many of them are not normally distributed, only results of the nonparametric Wilcoxon test have been reported throughout in the result section (Section 6.3.4).

For reasons of reference and comparability, Table C.2 shows results for both the Wilcoxon *and* the t-test for all variables mentioned in the study.

C.1 Distributions

Table C.1: Kolmogorov–Smirnov Test for Normality of Distribution: Some Variables significantly deviate from the normal distribution.

Variable	**D**	**p**
QuiKey find	0.239*	.033
QuiKey Target Delta	0.187	.160
QuiKey Context	0.295*	.004
QuiKey Overview	0.247	.025
Unknown QuiKey find	0.264	.161
Unknown QuiKey Target Delta	0.173	.656
Unknown QuiKey Context	0.205	.439
Unknown QuiKey Overview	0.319	.052
iMapping find	0.205	.097
iMapping Target Delta	0.344***	< .001
iMapping Context	0.155	.362
iMapping Overview	0.246*	.026
Unknown iMapping find	0.254	.196
Unknown iMapping Target Delta	0.238	.258
Unknown iMapping Context	0.249	.216
Unknown iMapping Overview	0.402**	.006
QuiKey sum	0.192	.523
iMapping sum	0.133	.906
Mix sum	0.160	.744
Unknown QuiKey sum	0.218	.313
Unknown iMapping sum	0.160	.744
Unknown mix sum	0.252	.203
QuiKey sum (tool-mixers only)	0.253	.426
iMapping sum (tool-mixers only)	0.160	.920
Mix sum (tool-mixers only)	0.254	.359
Unknown QuiKey sum (tool-mixers only)	0.097	1
Unknown iMapping sum (tool-mixers only)	0.382	.145
Unknown mix sum (tool-mixers only)	0.171	.916

* Statistically significant at the 5% level
** Statistically significant at the 1% level
*** Statistically significant at the 0.1% level

C.2 Significance Tests

TABLE C.2: Wilcoxon and T-Tests for All Comparisons Made – along with corresponding effect sizes (D). As can be seen the two significance tests largely yield comparable results, exept for *Target delta; QuiKey − iMapping* and *iMapping overview; familiar − unknown*.

Comparison	**D**	**Wilc.**	**t-Test**
Find; QuiKey − iMapping	-55.50	$< .001$	$< .001$
Unknown find; QuiKey − iMapping	-19.17	$< .001$	$< .001$
Target; QuiKey − iMapping	-55.69	$< .001$	$< .001$
Unknown target; QuiKey − iMapping	-17.00	$< .001$	$< .001$
Target delta; QuiKey − iMapping	-0.17	.012	.930
Target delta; QuiKey − iMapping (outliers adjusted)	1.97	.002	.002
Unknown delta; QuiKey − iMapping	2.17	.006	.003
Context; iMapping − QuiKey	-21.94	$< .001$	$< .001$
Unknown context; iMapping − QuiKey	-6.47	.003	.007
Overview; iMapping − QuiKey	-9.51	$< .001$	$< .001$
Unknown overview; iMapping − QuiKey	-10.74	.002	.025
Sum; mix − QuiKey	-32.09	.014	.012
Unknown sum; mix − QuiKey	-5.43	.151	.408
Unknown sum; mix − iMapping	-5.22	.266	.241
Mix sum by toolmix (dependent)	-21.29	.049	.117
Unknown mix sum by toolmix-unknown (dependent)	-18.39	.005	.049
Sum; mix − QuiKey (tool mixers only)	-41.73	.011	.014
Unknown sum; mix − QuiKey (mixers only)	-8.19	.066	.039
Unknown sum; mix − iMapping (mixers only)	-10.50	.033	.086
iMapping sum; familiar − unknown	80.89	$< .001$	$< .001$
iMapping sum; familiar − unknown (excluding find)	7.89	.052	.080
iMapping overview; familiar − unknown	0.36	.049	.625
Time; links-on-demand − all-links-visible	-49.94	$< .001$	$< .001$
Errors; links-on-demand − all-links-visible	-1.00	.004	.001

Appendix D

Evaluation Assignments

The following pages list the assignments used in the user evaluations. Since all user studies have been carried out in Germany, they are all in German.

Appendix D.1 shows the sheet used by the experimenter to carry out the comparative user study detailed in Section 6.1, which compared iMapping to MindManager. Note that it is in informal keyword style only.

Appendix D.2 lists the preparatory assignments sent out one by one to participants of the long-term user study reported in Section 6.3.

Appendix D.3 lists the assignments used in the final session of the long-term user study.

D.1 Assignments for the Comparative User Study (iMapping vs. MindManager)

Name:
Datum, Uhrzeit
Reihenfolge der Tools:
Reihenfolge der Aufgaben:

Briefing: Ablauf
MindManager o.ä. schon mal benutzt?

Üben
navigate
add
edit
rearrange
link
ganz raus zoomen
explore

Set A
Tool:
Reihenfolge:
overall practice time:

Add: Koala als neuer Beutelsäuger
Add: Champignon als neuer Pilz (richtige Stelle)
Brainstorm: Höhere Säugetiere: Löwe, Tiger Elefant
Cluster: Raubkatzen, die da rein
Rearrange: Meeressäuger auch höhere Säugetiere
Link: Löwe frisst Antilopen
Find: was wissen wir über Pilze?
Find: welche Wale kennen wir?
remember without map:
Wieviele Klassen von Wirbeltieren hatten wir?
welche?

Set B
Tool:
Reihenfolge:
overall practice time:

Add: Molch als neue Amphibie
Add: Petersilie als neue Pflanze (richtige Stelle)
Brainstorm: Höhere Säugetiere: Löwe, Tiger Elefant
Cluster: Raubkatzen, die da rein
Rearrange: Dinosaurier sind Echsen
Link: Adler fressen Frösche
Find: Was wissen wir über Tintenfische?
Find: Welche Süßwasserkennen wir?
remember without map:
Wieviele Klassen von Pflanzen hatten wir?
welche?

Bewertungen

iMapping
Spontane Reaktionen:

Schulnoten:
Navigation: Wie leicht kommt man zur gewünschten Stelle?
Übersichtlichkeit
Struktur: Wie gut erkennt man die Struktur,
d.h. die Beziehungen zwischen den Dingen?
Ästhetik: Wie schön?
Wohlfühlfaktor: Wie gut fühlt sich die Benutzung an?
Tauglichkeit für Brainstorming
Tauglichkeit für Note-Taking
Tauglichkeit für PKM allgemein
Gesamtnote

MindManager
Spontane Reaktionen:

Schulnoten:
Navigation: Wie leicht kommt man zur gewünschten Stelle?
Übersichtlichkeit
Struktur: Wie gut erkennt man die Struktur,
d.h. die Beziehungen zwischen den Dingen?
Ästhetik: Wie schön?
Wohlfühlfaktor: Wie gut fühlt sich die Benutzung an?
Tauglichkeit für Brainstorming
Tauglichkeit für Note-Taking
Tauglichkeit für PKM allgemein
Gesamtnote

D.2 Preparatory Assignments for Summative User Study

First Assignment

Legt bitte irgendwo in Eurer Map ein Item an namens "Schafkopf Stammtisch".

Und in diesem dann folgende vier Namen:
Sepp
Schorsch
Wrtlbrmpft
Lansdberger Dominikus

Second Assignment

Neben dem "Schafkopf Stammtisch" bitte folgende 2 Städte anlegen:
Hintertupfing
München

Dann bitte Links ziehen von den vier Stammtisch-Gesellen zu den Städten, mit der Relation "wohnt in" (die bei ihrer ersten Verwendung angelegt wird):
Sepp und Schorsch wohnen in Hintertupfing
Wrtlbrmpft und Lansdberger Dominikus wohnen in München

Es sollte dann ungefähr so aussehen wie im Anhang.

Third Assignment

Die Aufgabe für diese Woche ist:

Lege mit QuiKey folgende Statements über die Stammtischgesellen an (als Statements zwischen den bestehenden Items, so wie im Video beschrieben):
Sepp mag Schorsch
Sepp mag Landsberger Dominikus
Sepp hasst Wrtlbrmpft

Change to New Plan

(As explained in Section 6.3)

Neuer Plan Der Plan hat sich inzwischen etwas geändert: Weil ich herausfinden will, ob man sich auch in großen Maps gut orientieren kann, schicke ich Euch hier meine eigene ziemlich große iMap. Sie umfasst über 3000 Items und ich bitte Euch, die weiteren Aufgaben mit dieser Map auszuführen.

Die Informationen aus den bisherigen Aufgaben (Schorsch und die Stammtischgesellen) sind bereits in der neuen Map enthalten - es ist also nichts verloren :).

Wer außerdem inzwischen seine eigene Map pflegt und nutzt kann ja über das File-Menü (Open / Save) zwischen den Maps hin und her wechseln.

New Deal Here is the deal:

1) Unter allen, die alle Aufgaben und die Abschlussbefragung mitmachen und mir ihr Logfile schicken, wird am Ende ein iPad verlost. Dies gilt nur für Euch 33 Teilnehmer der Evaluationsstudie.

2) Außerdem kann Jedermann bis Ende September seine/ihre größte beste iMap einschicken. Die drei größten Maps nehmen zusätzlich an der Verlosung Teil (Details folgen). Das gilt auch für Euch, Ihr könnt damit also Eure Gewinnchance verdoppeln.

Mit einer Gewinnchance von mindestens 1:36 dürfte das die beste iPad-Verlosung sein, die Euch je über den Weg läuft! Und so ein Ding ist cool - es hat schon Applenörgler zu Fanboys gemacht ;-)

Aufgabe Die erste neue Aufgabe für Euch ist jetzt:

Finde die Stammtischgesellen in der großen Map (am besten mit QuiKey) und schau Dich ein bißchen um, was es da so neues gibt.

Die restlichen paar Aufgaben kommen dann jetzt wieder regelmäßg und etwas dichter hintereinander. Und dann würden ab Mitte September die Abschluss-Interviews kommen.

Klappt das?

Fifth Assignment

Einer hat es schon gemerkt: was bei den Stammtischgesellen in der Großen Map noch fehlt sind die Statements aus Aufgabe 3, die Ihr (hoffentlich bereits) mit QuiKey eingetragen hattet.

Also - zum QuiKey wieder aufwärmen - bitte nochmal kurz nachtragen:
Sepp mag Schorsch
Sepp mag Landsberger Dominikus
Sepp hasst Wrtlbrmpft

(Wer nichtmehr weiß wie's geht - Hier nochmal der Link zum Video:
http://www.youtube.com/watch?v=pFHEqEhmB4w)

Außerdem ist ein Neuzugang in der Friedrich Schiller Schule in Klasse 7 einzutragen: Liese.

Und ein Link: Liese mag Heinz (aus der 8. Klasse).

Und hier nochmal die letzten Aufgaben:

Sixth Assignment

in iMapping:

1) Lies in der großen Map nach, was Götz Werner gesagt hat.

2) Erkunde auch ein wenig die Umgebung dessen - da sind noch weitere interessante Zitate

Seventh Assignment

in iMapping - Bei den Stammtischgesellen in Hintertupfing:

1) Bitte in der 9. Klasse der "Jens" anlegen.

2) Einen Link: Marie mag Jens

Eigth Assignment

1) Browse in iMapping (ohne QuiKey) zu: Heiko's Big Map >Diss >Publication Planning >Journals >primary publication targets >Human-Computer Interaction (Informa Journal)

2) Finde über die Links heraus, wer dieses Journal vorgeschlagen hat (Relation "listed"), und was er noch zu diesem Jounal gesagt hat.

Last Assignment

Bitte findet - nur mit QuiKey - folgendes heraus:

Es gibt ein Zitat das mit "There is no Problem..." begint.

1) Wer hat es gesagt?

2) Was hat diese Person gegründet und

3) wem gehört das?

Also einfach vom Zitat um drei Ecken zur Antwort durchhangeln. (Item-Relation-Item-Relation-Item-Relation-Item)

Alle Antworten bitte per Mail an mich.

D.3 Final Interview Assignments for Summative User Study

Block 1

Wie heißt Sohn vom Landsberger Dominikus?
Welche Mädchen in seiner Klasse?
In welchem Dorf ist die Schule?

Block 2

Wer hat über den Homo Oeconomicus gesprochen?
Auf welchem Event?
Wer hat da noch Keynotes gehalten?

Block 3

Wie ist die Homepage vom Journal of Digital Information?
Ist es ein Primär-target?
Welche Journals gibt es noch?

Block 4

Telefonnummer vom Louvre?
In welchem Land ist das?
Welche Regionen gibt es in da noch?

Block 5

Was fressen Tiger?
Zu welcher oberklasse gehört das? Und eins höher?
Was gibt's da noch so?
(iMapping only) von wem wird das noch gefressen? (iMapping only) Was ist die kleinste gemeinsame Oberklasse der 3?

Block 6

Im MyUI Projekt gibt es ein Mitglied Namens Barnabas Tackacs.
Welches ist die Institution von Barnabas?
Welches Workpackage leiten die?
Welche anderen Workpackages gibt es da?

Block 7

Which requirements are supported by all three on the left?
Which requirements are supported by non of the three on the left?

Appendix E

iMapping Back-End – Data Model for Visual Meta-Data

```
# represents spec ver 0.6.8
# changes to 0.6.8: storeImapItem now gets a color attribute
# changes to 0.6.7: store body gets path to a background image
# changes to 0.5.6: add HeadHeight
# changes to 0.5.5: replace property "imap:hasItemWidth" with "imap:hasItemScale"
# changes to 0.5.4: remove property "hasLinkSurroundingItem" aka "hasContainingBody"
#                   owl:Cardinality typo fixed
# changes to 0.5.3: use cds-namespace again but name classes by rdfs:label

@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix cds:   <http://ont.semanticdesktop.org/ontologies/2007/cds#> .
@prefix imap:  <http://ont.semanticdesktop.org/ontologies/2007/imapping#> .

imap:StoreImapItem a rdfs:Class
; rdfs:comment "An iMapping item, carrying the outer properties of the item"
.

imap:StoreBody a rdfs:Class
; rdfs:comment "The Body of an item, carrying all inner properties"
.

cds:Item a rdfs:Class
; rdfs:label "CdsItem"
; rdfs:comment "a cds:Item which we call CdsItem here to be unabiguous"
.

cds:Statement a rdfs:Class
; rdfs:label "CdsStatement"
; rdfs:comment "a cds:Statement which we call CdsStatement here to be unabiguous"
.
```

```
imap:hasBody a rdfs:property
; rdfs:comment "assigns one body to each item"
; rdfs:domain imap:StoreImapItem
; rdfs:range imap:StoreBody
; owl:cardinality "1"
.

imap:representsCdsItem a rdf:Property
; rdfs:comment "Gives you access to content and details of the item"
; rdfs:domain imap:StoreBody
; rdfs:range cds:Item
; owl:cardinality "1"
.

imap:hasParentBody a rdf:Property
; rdfs:comment "points to children Items (CDS:detail)"
; rdfs:domain imap:StoreImapItem
; rdfs:range imap:StoreBody
; owl:cardinality "1"
.

imap:hasPositionX a rdf:Property
; rdfs:comment "relative to SurrImI"
; rdfs:domain imap:StoreImapItem
; rdfs:range xsd:double
; owl:cardinality "1"
.

imap:hasPositionY a rdf:Property
; rdfs:comment "relative to SurrImI"
; rdfs:domain imap:StoreImapItem
; rdfs:range xsd:double
; owl:cardinality "1"
.

imap:hasItemScale a rdf:Property
; rdfs:comment "Scaling factor relative to parent item. Inherits DefaultScaleFactor if not
initially specified"
; rdfs:domain imap:StoreImapItem
; rdfs:range xsd:double
; owl:cardinality "1"
.

imap:hasBellyWidth a rdf:Property
; rdfs:comment "inner width / width of the belly's coordinate system. "
; rdfs:domain imap:StoreBody
; rdfs:range xsd:double
; owl:cardinality "1"
.

imap:hasBellyHeight a rdf:Property
; rdfs:comment "inner height / height of the belly's coordinate system. "
; rdfs:domain imap:StoreBody
; rdfs:range xsd:double
; owl:cardinality "1"
```

```
.

imap:hasHeadHeight a rdf:Property
; rdfs:comment "height of the text-area. "
; rdfs:domain imap:StoreBody
; rdfs:range xsd:double
; owl:cardinality "1"
.

imap:hasBellyPreviewImage a rdf:Property
; rdfs:comment "URI of preview image (.png)"
; rdfs:domain imap:StoreBody
; rdfs:range rdfs:Resource
; owl:cardinality "1"
.

imap:BackgroundPicturePath a rdf:Property
; rdfs:comment "path to a optional background picture"
; rdfs:domain imap:StoreBody
; rdfs:range rdfs:Literal
; owl:cardinality "1"
.

imap:itemColor a rdf:Property
; rdfs:comment "Color of the Item saved as hex string"
; rdfs:domain imap:StoreBody
; rdfs:range rdfs:Literal
; owl:cardinality "1"
.

imap:lastFocusedDetailItem a rdf:Property
; rdfs:comment "points to a detail-item"
; rdfs:domain imap:StoreBody
; rdfs:range imap:StoreImapItem
; owl:maxCardinality "1"
.

# -- expansion status

imap:hasExpansionStatus a rdf:Property
; rdfs:comment "if not set: default= expanded if it has details, collapsed if it has no details"
; rdfs:domain imap:StoreImapItem
; rdfs:range imap:ExpansionStatus
; owl:maxCardinality "1"
.

imap:ExpansionStatus a rdfs:Class
; rdfs:comment "one of expanded, collapsed or SemanticZoom"
.

imap:Expanded a imap:ExpansionStatus .
imap:Collapsed a imap:ExpansionStatus .
imap:SemanticZoom a imap:ExpansionStatus .
```

```
#######Links#######

imap:StoreLink a rdfs:Class
; rdfs:comment "An iMapping link"
.

imap:representsCdsStatement a rdf:Property
; rdfs:comment "refers to the corresponding cds statement"
; rdfs:domain imap:StoreLink
; rdfs:range cds:Item
; owl:cardinality "1"
.

imap:linksFrom a rdf:Property
; rdfs:comment "The source of the link (where the arrow starts)  - corresponds to rdf:subject of
corresponding cds:Statement"
; rdfs:domain imap:StoreLink
; rdfs:range imap:StoreImapItem
; owl:cardinality "1"
.

imap:linksTo a rdf:Property
; rdfs:comment "The target of the link (where the arrow points to) - corresponds to rdf:target of
corresponding cds:Statement"
; rdfs:domain imap:StoreLink
; rdfs:range imap:StoreImapItem
; owl:cardinality "1"
.


# -- Link visibility status  (renamed from "visibility status")

imap:hasLinkVisibilityStatus a rdf:Property
; rdfs:comment "optional. OnDemand if not specified"
; rdfs:domain imap:StoreLink
; rdfs:range imap:LinkVisibilityStatus
; owl:maxCardinality "1"
.

imap:LinkVisibilityStatus a rdfs:Class
; rdfs:comment "one of OnDemand, Permanent or Hidden"
.

imap:OnDemand a imap:LinkVisibilityStatus .
imap:Permanent a imap:LinkVisibilityStatus .
```

List of Figures

Appendix A: User Generated Maps

List of Tables

Abbreviations

PKM	**P**ersonal **K**nowledge **M**anagement
PIM	**P**ersonal **I**nformation **M**anagement
ZUI	**Z**ooming **U**ser **I**nterface
LOD	**L**evels **o**f **D**etail
DOI	**D**egree **o**f **I**nterest
N&Z	**N**esting and **Z**ooming
FEV	**F**ish **E**ye **V**iew
FZI	**F**orschungs**z**entrum **I**nformatik, Karlsruhe
KIT	**K**arlsruhe **I**nstitute of **T**echnology
MIT	**M**assachusetts **I**nstitute of **T**echnology
CDS	**C**onceptual **D**ata **S**tructures
RDF	**R**esource **D**escription **F**ramework
API	**A**pplication **P**rogramming **I**nterface
PDF	**P**ortable **D**ocument **F**ormat
SMW	**S**emantic **M**edia**W**iki
ANOVA	**An**alysis **o**f **Va**riance
SD	**S**tandard **D**eviation
CI	**C**onfidence **I**nterval
H1–H9	**H**ypotheses **1**–**9**
UX	**U**ser E**x**perience
UEQ	**U**ser **E**xperience **Q**uestionnaire
GOMS	**G**oals, **O**perators, **M**ethods, and **S**election rules
KLM-GOMS	**K**eystroke-**L**evel **M**odel GOMS

Bibliography

Abecker, A. (2004). *Business-Process Oriented Knowledge Management: Concepts, Methods and Tools*. PhD thesis, PhD thesis at the Universität Karlsruhe (TH), Fakultät für Wirtschaftswissenschaften.

Anderson, J. R. (2005). *Cognitive Psychology and Its Implications*. Worth Publishers, 6th edition.

Andrews, K. (2010). Information visualisation. Course notes, IICM – Graz University of Technology, Graz, Austria. Available at http://courses.iicm.tugraz.at/ivis/ivis.pdf.

Apple Inc. (2008). Apple human interface guidelines. Technical report, Apple Inc., Cupertino, CA. Available at http://developer.apple.com/library/mac/#documentation/UserExperience/Conceptual/AppleHIGuidelines/.

Ausubel, Novak, J. D., and Hanesian, H. (1980). Psychologie des Unterrichts.

Baudisch, P., Good, N., Bellotti, V., and Schraedley, P. (2002). Keeping things in context: a comparative evaluation of focus plus context screens, overviews, and zooming. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, CHI '02, pages 259–266.

Bederson, B. B. (1999). Does animation help users build mental maps of spatial information? In *Proceedings of Information Visualization Symposium (InfoVis 99)*, pages 28–35.

Bederson, B. B., Grosjean, J., and Meyer, J. (2004). Toolkit design for interactive structured graphics. *IEEE Transactions on Software Engineering*, 30(8):535–546.

Bederson, B. B., Hollan, J. D., Perlin, J., Meyer, K., Bacon, D., and Furnas, G. (1996). Pad++: a zoomable graphical scetchpad for exploring alternate interface physics. *Journal of Visual Languages in Computing*, 7(1):3–31.

Bederson, B. B., Hollan, J. D., Stewart, J., Rogers, D., and Vick, D. (2002). A zooming web browser. In Ratner, J., editor, *Human Factors and Web Development, Second Edition*. Lawrence Erlbaum Associates Inc., Hillsdale.

Bederson, B. B., Meyer, J., and Good, L. (2000). Jazz: An extensible zoomable user interface graphics toolkit in java. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 171–180.

Bederson, B. B. and Shneiderman, B. (2003). *The Craft of Information Visualization: Readings and Reflections (Morgan Kaufmann Series in Interactive Technologies)*. Morgan Kaufmann Publishers, San Francisco, CA.

Berners-Lee, T., Fischetti, M., and Dertouzos, M. L. (1999). *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper, San Francisco, CA.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, pages 34–43. Available at http://www.scientificamerican.com/article.cfm?id=the-semantic-web.

Bernstein, M. (2003). Collage, composites, construction. In *HYPERTEXT 2003, Proceedings of the 14th ACM Conference on Hypertext and Hypermedia, August 26–30, 2003, Nottingham, UK*, pages 122–123, New York, NY. ACM.

Blackwell, A. F. (2006). Ten years of cognitive dimensions in visual languages and computing. *Journal of Visual Languages and Computing*, 17(4):285–287.

Blackwell, A. F., Britton, C., Cox, A. L., Green, T. R. G., Gurr, C. A., Kadoda, G. F., Kutar, M., Loomes, M., Nehaniv, C. L., Petre, M., Roast, C., Roe, C., Wong, A., and Young, R. M. (2001). Cognitive dimensions of notations: Design tools for cognitive technology. In Beynon, M., Nehaniv, C. L., and Dautenhahn, K., editors, *Cognitive Technology*, volume 2117 of *Lecture Notes in Computer Science*, pages 325–341, Heidelberg. Springer.

Braun, S., Schmidt, A., Walter, A., Nagypal, G., and Zacharias, V. (2007). Ontology maturing: a collaborative web 2.0 approach to ontology engineering. In *Proceedings of the Workshop on Social and Collaborative Construction of Structured Knowledge at the 16th International World Wide Web Conference (WWW 07), Banff, Canada*, volume 237 of *CEUR-WS*. Available at http://www.imagination-project.org/upload/OntologyMaturing.pdf.

Braun, S., Schmidt, A., Walter, A., and Zacharias, V. (2008). Using the ontology maturing process model for searching, managing and retrieving resources with semantic technologies. In *Proceedings of the On the Move Conferences – ODBASE 2008*, volume 5332 of *LNCS*, pages 1568–1578.

Buckland, M. (2006). *Emanuel Goldberg and His Knowledge Machine: Information, Invention, and Political Forces (New Directions in Information Management)*. Libraries Unlimited, Westport, CT.

Burkhard, R. A. (2005). Towards a framework and a model for knowledge visualization: Synergies between information and knowledge visualization. In Tergan, S.-O. and Keller, T., editors, *Knowledge and Information Visualization*, volume 3426 of *LNCS*, pages 238 – 255. Springer, Heidelberg.

Bush, V. (1945). As we may think. *Atlantic Monthly*, 176:101–108. Available at: http://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/3881/.

Buzan, T. and Buzan, B. (1996). *The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential*. Plume, New York, NY.

Card, S. K., Mackinlay, J. D., and Shneiderman, B. (1999). *Readings in Information Visualization: Using Vision to Think (Interactive Technologies)*. Morgan Kaufmann Publishers, San Francisco, CA.

Card, S. K., Moran, T. P., and Newell, A. (1983). *Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates Inc. Inc., Hillsdale.

Chandler, P. and Sweller, J. (1992). The split-attention effect as a factor in the design of instruction. *British Journal of Educational Psychology*, 62:233–246.

Cheyer, A., Park, J., and Giuli, R. (2005). IRIS: Integrate. Relate. Infer. Share. *1st Workshop on The Semantic Desktop. 4th International Semantic Web Conference.*

Chimera, R. and Shneiderman, B. (1994). An exploratory evaluation of three interfaces for browsing large hierarchical tables of contents. *ACM Transactions on Information Systems*, 12(4):383–406.

Cockburn, A., Karlson, A., and Bederson, B. B. (2008). A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys*, 41(4).

Conklin, J. (1987). Hypertext: an introduction and survey. *Computer*, 20(9):17–41.

Cowan, N. (2001). The magical number 4 in short-term memory: a reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24(1):87–185.

Dansereau, D. F. (2005). *Node-link mapping principles for visualizing knowledge and information*, pages 61–81. Number 3426 in Lecture Notes in Computer Science. Springer, Heidelberg.

Davies, J., Studer, R., and Warren, P., editors (2006a). *Semantic Web Technologies - trends and research in ontology-based systems*. John Wiley & Sons, Hoboken, NJ.

Davies, S. (2011). Still building the memex. *Communication of the ACM*, 54:80–88. Note: For a more extensive version, see Davies et al. (2005).

Davies, S., Allen, S., Raphaelson, J., Meng, E., Engleman, J., King, R., and Lewis, C. (2006b). Popcorn: the personal knowledge base. In *DIS '06: Proceedings of the 6th conference on Designing Interactive systems*, pages 150–159.

Davies, S., Velez-Morales, J., and King, R. (2005). Building the memex sixty years later: trends and directions in personal knowledge bases. Technical report, University of Colorado. Available at: `http://www.cs.colorado.edu/department/publications/reports/docs/CU-CS-997-05.pdf`. Note: For a shortened and refereed version of this report, see Davies (2011).

Dodds, R. A., Ward, T. B., and Smith, S. M. (2004). A review of experimental research on incubation in problem solving and creativity. Unpublished thesis, Texas A&M University. Available at `http://ecologylab.net/research/publications/DoddsSmithWardChapter.pdf`.

Donelson, W. C. (1978). Spatial management of information. In *Proceedings of 1978 ACM SIGGRAPH Conference*, pages 203–209.

Donskoy, M. and Kaptelinin, V. (1997). Window navigation with and without animation: a comparison of scroll bars, zoom, and fisheye view. In *CHI '97 extended abstracts on Human factors in computing systems: looking to the future*, CHI '97, pages 279–280.

Dörner, D. (2003). *Die Logik des Mißlingens. Strategisches Denken in komplexen Situationen.* Rowohlt Taschenbuch, Hamburg, Germany.

Drucker, P. F. (1999). Knowledge-worker productivity: The biggest challenge. *California Management Review*, 41(2):79–94.

Edwards, D. M. and Hardman, L. (1999). Lost in hyperspace: cognitive mapping and navigation in a hypertext environment. In McAleese, R., editor, *Hypertext: theory into practice*, pages 90–105. Intellect Books, Exeter, UK, UK.

Ellwood, S., Pallier, G., Snyder, A., and Gallate, J. (2009). The Incubation Effect: Hatching a Solution? *Creativity Research Journal*, 21(1):6–14.

Engelbart, D. C. (1962). Augmenting human intellect: A conceptual framework. Technical report, SRI. SRI Summary Report AFOSR-3223 Prepared for: Director of Information Sciences, Air Force Office of Scientific Research, Washington 25, DC. Available at http://www.dougengelbart.org/pubs/augment-3906.html.

Eppler, M. J. and Burkhard, R. A. (2005). Knowledge visualization. In Schwartz, D., editor, *Encyclopedia of Knowledge Management*, pages 551–560. Idea Group Reference, Hershey, PA.

Francisco-Revilla, L. and Shipman, F. M. (2005). Parsing and interpreting ambiguous structures in spatial hypermedia. In *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, pages 107–116.

Friedenberg, J. D. and Silverman, D. G. (2005). *Cognitive Science: An Introduction to the Study of Mind.* Sage Publications, Inc, 1 edition.

Furnas, G. W. (1986). Generalized fisheye views. In *Human Factors in Computing Systems CHI '86*, pages 16–23. Available http://www.si.umich.edu/~furnas/Papers/FisheyeCHI86.pdf.

Gaines, B. R. and Shaw, M. L. G. (1995). Concept maps as hypermedia components. *International Journal of Human-Computer Studies*, 43(3):323–361.

Goodman, D. (1998). *The Complete HyperCard 2.2 Handbook: Fourth Edition (Volume 1) (Complete Hypercard 2.2 Handbook Series)*. iUniverse, Bloomington, IN.

Goulding, J. O., Brailsford, T. J., and Ashman, H. L. (2010). Hyperorders and transclusion: understanding dimensional hypertext. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, HT '10, pages 201–210.

Green, T. R. G. (1989). Cognitive dimensions of notations. In Sutcliffe, A. and Macaulay, L., editors, *People and Computers*, volume V, pages 443–460. Cambridge University Press, Cambridge.

Haase, P., Herzig, D. M., Musen, M., and Tran, D. T. (2009). Semantic wiki search. In *6th Annual European Semantic Web Conference, ESWC2009, Heraklion, Crete, Greece*, volume 5554 of *LNCS*, pages 445–460.

Haller, H. (2003). Mappingverfahren zur Wissensorganisation. Diploma thesis published on KnowledgeBoard Europe http://www.knowledgeboard.com/ Available at http://heikohaller.de/literatur/diplomarbeit/.

Haller, H. (2006). iMapping – a graphical approach to semi-structured knowledge modelling. In Rutledge, L., editor, *Proceedings of the The 3rd International Semantic Web User Interaction Workshop (SWUI2006)*. Poster and extended abstract presented at the The 3rd International Semantic Web User Interaction Workshop. Available at http://people.aifb.kit.edu/hha/papers/iMapping_SWUI2006_paper.pdf.

Haller, H. (2008a). QuiKey. In Bloehdorn, S., Grobelnik, M., Mika, P., and Duc, T. T., editors, *Proceedings of the Workshop on Semantic Search (SemSearch 2008) at the 5th European Semantic Web Conference (ESWC 2008)*, volume 334, pages 74–78. Available at http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-334/paper-06.pdf.

Haller, H. (2008b). QuiKey – the smart semantic commandline (a concept). Poster presented at the European Semantic Web Conference 2008. Available at http://files.heikohaller.de/QuiKey-Poster_ESWC08.pdf. Won Best Poster Award.

Haller, H. (2010a). The case for pkm. In Schroeder, U., editor, *Interaktive Kulturen Workshop-Band*, pages 75–79. Proceedings of the 2nd Workshop on Personal Knowledge Management, PKM2010. Available at http://files.heikohaller.de/The_Case_For_PKM.pdf.

Haller, H. (2010b). Knitting the kNet – towards a global net of knowledge. *Open Journal of Knowledge Management*, 1(1). Available at http://www.c-o-k.de/beitrag/301/.

Haller, H. (2010c). QuiKey – an efficient semantic command line. In *Knowledge Engineering and Knowledge Management by the Masses – 17th International Conference, EKAW 2010*, number 17 in Knowledge Acquisition, Modeling and Management (EKAW), Heidelberg. Springer. Available at http://files.heikohaller.de/QuiKey_Haller_EKAW2010.pdf. Won Best Demo Award.

Haller, H. and Abecker, A. (2009). Requirements for Diagrammatic Knowledge Mapping Techniques. In *Conference Proceedings of I-KNOW'09 and I-SEMANTICS'09*. Full proceedings available at http://i-know.tugraz.at/wp-content/uploads/2009/09/Full-Proceedings_I-KNOW_I-SEMANTICS_2009.pdf.

Haller, H. and Abecker, A. (2010a). Designing a knowledge mapping tool for knowledge workers. In *Proceedings of the 14th international conference on Knowledge-based and intelligent information and engineering systems: Part I*, KES'10, pages 660–669, Heidelberg. Springer.

Haller, H. and Abecker, A. (2010b). iMapping – a zooming user interface approach for personal and semantic knowledge management. In *HT '10: Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 119–128. ACM. Also published as Haller and Abecker (2010c); won Ted Nelson Award.

Haller, H. and Abecker, A. (2010c). imapping: a zooming user interface approach for personal and semantic knowledge management. *SIGWEB Newsletter*, pages 4:1–4:10. Originally published as Haller and Abecker (2010b). Available at http://doi.acm.org/10.1145/1836291.1836295.

Haller, H., Abecker, A., and Völkel, M. (2010). A graphical workbench for knowledge workers. In Cordeiro, J., editor, *ICEIS 2010, 12th International Conference on Enterprise Information Systems, Funchal, Portugal, June 2010*. INSTICC (Institute for Systems and Technologies of Information, Control and Communication).

Haller, H., Völkel, M., and Kugel, F. (2006). iMapping wikis – towards a graphical environment for semantic knowledge management. In Schaffert, S., Völkel, M., and Decker,

S., editors, *Proceedings of the First Workshop on Semantic Wikis – From Wiki To Semantics*. Available at http://people.aifb.kit.edu/hha/papers/iMappingWikis.pdf.

Haller, H., Völkel, M., Abecker, A., Davis, B., Edlund, H., Groth, K., Gudjonsdottir, R., Kotelnikov, M., Lannerö, P., Lindquist, S., Sogrin, M., Sundblad, Y., Westerlund, B., Dragan, L., Scerri, S., Schutz, A., and Varma, P. (2009). Integrated knowledge workbench. Nepomuk project deliverable 1.3, FZI – Forschungszentrum Informatik. Available at http://nepomuk.semanticdesktop.org/xwiki/bin/view/Main1/D1-3.

Hassenzahl, M. (2006). Hedonic, emotional, and experiential perspectives on product quality. In Ghaoui, C., editor, *Encyclopedia of Human Computer Interaction*, pages 266–272. Idea Group, Hershey, PA.

Hassenzahl, M. (2008). The interplay of beauty, goodness, and usability in interactive products. *Human-Computer Interaction*, 19:319–349.

Hassenzahl, M., Burmester, M., and Koller, F. (2003). AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität. In Ziegler, J. and Szwillus, G., editors, *Mensch & Computer 2003. Interaktion in Bewegung*, pages 187–196.

Hoffmann, R. and Krauss, K. (2004). A critical evaluation of literature on visual aesthetics for the web. In *Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, SAICSIT '04, pages 205–209.

Hornbæk, K., Bederson, B. B., and Plaisant, C. (2002). Navigation patterns and usability of zoomable user interfaces with and without an overview. *ACM Transactions on Computer-Human Interaction*, 9:362–389.

Huynh, D. and Karger, D. (2009). Parallax and companion: Set-based browsing for the data web. Technical report, Metaweb, MIT. Available at http://davidhuynh.net/media/papers/2009/www2009-parallax.pdf.

Huynh, D., Karger, D. R., Quan, D., and Sinha, V. (2003). Haystack: a platform for creating, organizing and visualizing semistructured information. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 323–323.

ISO 9241-210 (2009). *Ergonomics of Human-System Interaction – Part 210: Human-centred Design for Interactive Systems.* International Organization for Standardization, Geneva.

Jetter, H.-C., König, W. A., Gerken, J., and Reiterer, H. (2008). Zoil – a cross-platform user interface paradigm for personal information management. In *Personal Information Management: PIM 2008, CHI 2008 Workshop.*

Jones, W. (2010). No knowledge but through information. *First Monday*, 15(9). Available at http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/3062/2600.

Jones, W. and Teevan, J. (2007). *Personal Information Management.* University of Washington Press, Seattle, WA.

Jüngst, K. L. (1998). *Lehren und Lernen mit Begriffsnetzdarstellungen (2. Auflage).* Afra, Frankfurt.

Keahey, A. (1998). The generalized detail-in-context problem. In *Proceedings of the 1998 IEEE Symposium on Information Visualization*, pages 44–51.

Kolb, D. (2001). Places and spaces: Adjacency effects. Position Paper presented on Spatial Hypertext Workshop of the ACM Conference on Hypertext and Hypermedia, Aarhus, Denmark.

Kommers, P. A. M., Jonassen, D. H., and Mayes, J. T., editors (1992). *Cognitive Tools for Learning.* Springer, Heidelberg.

Krötzsch, M., Vrandečić, D., Völkel, M., Haller, H., and Studer, R. (2007). Semantic wikipedia. *Journal of Web Semantics*, 5:251–261.

Kunz, W. and Rittel, H. W. J. (1970). Issues as elements of information systems. Working paper 131, Studiengruppe für Systemforschung, Heidelberg.

Lajoie, S. P. and Derry, S. J., editors (1993). *Computers as Cognitive Tools.* Lawrence Erlbaum Associates Inc., Hillsdale.

Laugwitz, B., Held, T., and Schrepp, M. (2008). Construction and evaluation of a user experience questionnaire. In *USAB '08: Proceedings of the 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for Education and Work*, pages 63–76.

Laugwitz, B., Schrepp, M., and Held, T. (2006). Konstruktion eines Fragebogens zur Messung der User Experience von Softwareprodukten. In Heinecke, A. M. and Paul, H., editors, *Mensch & Computer 2006: Mensch und Computer im Strukturwandel*, pages 125–134.

Luhmann, N. (1982). Kommunikation mit Zettelkästen (Erfahrungsbericht). In et al., H. B., editor, *Öffentliche Meinung und sozialer Wandel*, pages 222–228, Wiesbaden. VS Verlag für Sozialwissenschaften.

Lunn, D., Bernstein, M., Marshall, C., Matias, J. N., Nyce, J. M., and Tompa, F. (2010). Past visions of hypertext and their influence on us today. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, HT '10, pages 315–315.

Manola, F. and Miller, E. (2004). Resource Description Framework (RDF) primer. W3C Recommendation. Available at http://www.w3.org/TR/rdf-primer/.

Marshall, C. C. and Shipman, F. M. (1993). Searching for the missing link: discovering implicit structure in spatial hypertext. In *HYPERTEXT '93: Proceedings of the fifth ACM conference on Hypertext*, pages 217–230.

Mazarakis, A. (2009). Kann zu viel Nähe schaden? – Revisionen zum Split-Attention Effect. In Wandke, H., Kain, S., and Struve, D., editors, *Mensch & Computer 2009: Grenzenlos frei!?*, pages 403–412.

Metzger, W. (1953). *Gesetze des Sehens*. Kramer, Frankfurt.

Millard, D. E., Gibbins, N. M., Michaelides, D. T., and Weal, M. J. (2005). Mind the semantic gap. In *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, pages 54–62.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97.

Nekrasovski, D., Bodnar, A., McGrenere, J., Guimbretière, F., and Munzner, T. (2006). An evaluation of pan & zoom and rubber sheet navigation with and without an overview. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 11–20.

Nelson, T. (2006). A cosmology for a different computer universe: Data model, mechanisms, virtual machine and visualization infrastructure. *Journal of Digital Information*, 5(1). Available at http://journals.tdl.org/jodi/article/view/131/129.

Nelson, T. H. (1995). The heart of connection: hypermedia unified by transclusion. *Communications of the ACM*, 38(8):31–33. http://doi.acm.org/10.1145/208344.208353.

Nelson, T. H. (1999). Xanalogical structure, needed now more than ever: parallel documents, deep links to content, deep versioning, and deep re-use. *ACM Computing Surveys*, 31. Available at http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/60.html.

Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 152–158.

Noik, E. G. (1993). Layout-independent fisheye views of nested graphs. In *in VL'93: IEEE Symposium on Visual Languages*, pages 336–341.

Novak, J. D. and Gowin, D. B. (1984). *Learning how to learn*. Cambridge University Press, New York.

O'Donnell, A. M., Dansereau, D. F., and Hall, R. (2002). Knowledge Maps as Scaffolds for Cognitive Processing. *Educational Psychology Review*, 14(5).

Okada, A., Shum, S. J. B., and Sherborne, T. (2008). *Knowledge Cartography: Software Tools and Mapping Techniques*. Springer, Heidelberg.

Perlin, K. and Fox, D. (1993). Pad: An alternative approach to the computer interface. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, pages 57–64.

Pietriga, E. (2005). A toolkit for addressing hci issues in visual language environments. *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 00:145–152.

Poulovassilis, A. and Levene, M. (1994). A nested-graph model for the representation and manipulation of complex objects. *ACM Transactions on Information Systems*, 12:35–68.

Prud'hommeaux, E. and Seaborne, A. (2008). *SPARQL Query Language for RDF*. W3C Recommendation. Available at http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

Raskin, J. (2000). *The humane interface: new directions for designing interactive systems*. ACM Press/Addison-Wesley Publishing Co., New York, NY.

Reigeluth, C. M. (1983). Meaningfulness and instruction – relating what is being learnt to what a student knows. *Instructional Science*, 12(3):197–218.

Reigeluth, C. M. (1999). The elaboration theory: Guidance for scope and sequences decisions. In Reigeluth, C. M., editor, *Instructional-design theories and models – II: A new paradigm of instructional theory*, pages 425–454. Erlbaum, Hillsdale.

Richter, J., Völkel, M., and Haller, H. (2005). Deepamehta – a semantic desktop. In Decker, S., Park, J., Quan, D., and Sauermann, L., editors, *Proceedings of the 1st Workshop on The Semantic Desktop. 4th International Semantic Web Conference (Galway, Ireland)*, volume 175. CEUR-WS.

Sarkar, M. and Brown, M. H. (1992a). Graphical fisheye views of graphs. Technical report, DEC (Digital Equipment Corporation). Available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.62.2127 This is a longer version of Sarkar and Brown (1992b).

Sarkar, M. and Brown, M. H. (1992b). Graphical fisheye views of graphs. In Penny Bauersfeld, J. B. and Lynch, G., editors, *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '92, pages 83–91. Available at http://doi.acm.org/10.1145/142750.142763.

Sauermann, L., Bernardi, A., and Dengel, A. (2005). Overview and outlook on the semantic desktop. In Decker, S., Park, J., Quan, D., and Sauermann, L., editors, *Proceedings of the 1st Workshop on The Semantic Desktop at the ISWC 2005 Conference*, volume 1.

Sauermann, L., Kiesel, M., Schumacher, K., and Bernardi, A. (2009). Semantic Desktop. In Blumauer, A. and Pellegrini, T., editors, *Social Semantic Web*, pages 337–362.

Sawilowsky, S. S. and Blair, R. C. (1992). A more realistic look at the robustness and type ii error properties of the t test to departures from population normality. *Psychological Bulletin*, 111(2):352 – 360.

Schnotz, W. and Bannert, M. (2003). Construction and interference in learning from multiple representation. *Learning and Instruction*, 13:141–156.

Schrepp, M. and Laugwitz, B. (2010). UEQ – durchschnittliche scores? Personal e-mail messages – included in Appendix B.

Scriven, M. (1991). Prose and cons about goal-free evaluation. *American Journal of Evaluation*, 12:55–62.

Shiffrin, R. M. and Schneider, W. (1977). Controlled and automatic human information processing: II – perceptual learning, automatic attending, and a general theory. *Psychological Review*, 84(2):127–190.

Shipman, F. M., Hsieh, H., Maloor, P., and Moore, J. M. (2001). The visual knowledge builder: a second generation spatial hypertext. In *HYPERTEXT '01: Proceedings of the twelfth ACM conference on Hypertext and Hypermedia*, pages 113–122.

Shipman, F. M. and Marshall, C. C. (1999a). Formality considered harmful: Experiences, emerging themes, and directions on the use of formal representations in interactive systems. In *Computer Supported Cooperative Work*, volume 8, pages 333–352.

Shipman, F. M. and Marshall, C. C. (1999b). Spatial hypertext: An alternative to navigational and semantic links. *ACM Computing Surveys*, 31(4).

Shipman, F. M., Moore, J. M., Maloor, P., Hsieh, H., and Akkapeddi, R. (2002). Semantics happen: knowledge building in spatial hypertext. In *HYPERTEXT '02: Proceedings of the thirteenth ACM conference on Hypertext and hypermedia*, pages 25–34.

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages*.

Shneiderman, B. and Plaisant, C. (2004). *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition)*. Addison Wesley, Reading, MA.

Shneiderman, B. and Plaisant, C. (2009). Treemaps for space-constrained visualization of hierarchies. Technical report, Human-Computer Interaction Lab, University of Maryland. http://www.cs.umd.edu/hcil/treemap-history/.

Spence, R. (2007). *Information Visualization: Design for Interaction (2nd Edition)*. Prentice Hall, Upper Saddle River, NJ.

Staab, S. and Studer, R. (2009). *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, Heidelberg, 2nd edition.

Tolman (1948). Cognitive maps in rats and men. *Psychological Review*, 55:189–208.

Vester, F. (2002). *Die Kunst vernetzt zu denken*. dtv, München. Report to the Club of Rome.

Völkel, M. (2010). *Personal Knowledge Models with Semantic Technologies*. PhD thesis, Karlsruher Institut für Technologie (KIT), Fakultät für Wirtschaftswissenschaften, Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB).

Völkel, M. and Haller, H. (2009). Conceptual data structures for personal knowledge management. *Online Information Review*, 33(2):298–315. Available at http://www.aifb.kit.edu/web/Article1974.

Völkel, M., Haller, H., Bolinder, W., Davis, B., Edlund, H., Groth, K., Gudjonsdottir, R., Kotelnikov, M., Lannerö, P., Lundquist, S., Sogrin, M., Sundblad, Y., and Westerlund, B. (2008). Conceptual data structure tools. Deliverable 1.2, nepomuk consortium. Available at http://nepomuk.semanticdesktop.org/xwiki/bin/view/Main1/D1-3.

Vygotsky, L. and Cole, M. (1978). *Mind in society: The development of higher psychological processes*. Harvard University Press, Cambridge, MA.

Waldeck, C. and Balfanz, D. (2004). Mobile liquid 2d scatter space (ml2dss): A visual interactive information space (ispace) for displaying large amounts of information and allowing simple vision-based knowledge discovery on small screen sizes. In *Proceedings of the Eighth International Conference on Information Visualisation, IV 2004.*, pages 494–498.

Wallas, G. (1926). *The art of thought*. Harcour, New York.

Wiegmann, D. A., Dansereau, D. F., McCagg, E. C., Rewey, K. L., and Pitre, U. (1992). Effects of knowledge map characteristics on information processing. *Contemporary Educational Psychology*, 17(2):136–155.

Wiil, U. K. (2005). Hypermedia technology for knowledge workers: a vision of the future. In *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, pages 4–6.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics*, 1:80–83.

Zins, C. (2007). Conceptual approaches for defining data, information, and knowledge. *Journal of the American Society for Information Science and Technology*, 58(4):479–493.