Diploma thesis assignment

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

Diploma Thesis

Mindmapping using semantic technologies

Bc. Dominik Salai

Supervisor: Mgr. Miroslav Blaško

Study Programme: Open Informatics

Field of Study: Software Engineering & Interactions

May 5, 2014

Aknowledgements

I would like to express a great deal of gratitude to my supervisor Mgr. Miroslav Blaško for supervising and helping me during the conceptual, implementation and writing phases of my diploma thesis.

Declaration

Abstract

Mind maps represent a new revolutionary way of viewing, learning and discovering new information, but suffer in reuseability. However ontologies are more conservative and the formalized information capture approach makes them highly reuseable. This thesis discusses the impact of ontologies on mind mapping techniques. The focus is on maximizing the advantages of both of these different approaches. The methodology of creating reuseable mind maps in ontology driven mind mapping tool Ontomind is presented together with the improvements of the tool to further enhance the user experience. The implemented features are then tested and the methodology is demonstrated on a chosen specific domain to present that the two fundamentally different approaches of information capture can be brought together to benefit one another.

Keywords: mind maps, ontology, Ontomind, methodology, information, knowledge

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human brain is the most remarkable and the most complex structure in the whole known universe. Researchers from all over the world are trying to find out how it works. The relationship between brain and mind still remains one of the most important scientific questions.

In history, mankind has gone through eras of agriculture growth, industrial growth and finally arrived at the age of information and intelligence. That's why the 21st century, is announced to be "The century of Brain" by the Ministry of Education Datuk Seri Hj. Mohamed Khaled Nordin at 14th International Conference of Thinking in Malaysia, 2009.

Nowadays, we live in the world ruled by information. There are outrageous amounts of data and information being gathered or discovered every day. With so many information at our hands, few questions may arise. How to put this information and data into order? Are we and the human brain actually able to cope with such amount of information? Can we learn and understand everything?

Many great thinkers of the world believe, that the solution may be mind maps. During their short existence, the use of mind maps have widely spread across the whole planet, and almost half of the population have heard about this little revolution in thinking which affects the potential of human mind and how we actually use our brains.

Mind maps introduce new approach how to view, learn and discover new information from the individual point of view, but their biggest weakness is reusability. Because, while our brain functions the same across the whole population, the way how information, memories and associations are stored in our minds is individual to everyone of us.

On the other hand there are ontologies, which are great in describing and categorizing the gathered data and information. Unlike the mind maps, ontologies are developed by large groups of people over a long period of time with the idea of shareability and reusability in mind.

One of the possible answers for the questions mentioned above might lie in making the mind maps and ontologies work together.

1.1 Goals

Here, the goals of this thesis are presented, following with the short overview of what is discussed in specific chapters.

- Get familiar with semantic web technologies (OWL, SPARQL-DL) conceptual maps and mind maps.
- Analyze utility of ontology driven mind mapping tool Ontomind for different areas of knowledge management.
- Create methodology to manage different types of knowledge within the tool (e.g. personal vs. shareable, assertional vs. terminological).
- Implement missing features of the tool to fulfill defined methodology.
- Demonstrate methodology and implemented features on chosen domain of interest.
- Test implemented features of the tool.

1.2 Short Overview

In chapter 2 of this thesis, mind mapping technique is introduced, together with the explanation of how and why it is for the best to use mind maps in information technology and a little comparison of mind maps to conceptual maps is presented. Starting with section 2.6 the mind maps are analyzed in different use-cases and their pros and cons are summarized together with the guidelines on how to create a good mind map.

Then in chapter 3, semantic web technologies in form of ontologies are introduced. Their definition, purpose, structure and syntax together with the short description of various types of ontologies and their use is presented. Then, starting with section 3.8 the ontologies are analyzed and their pros and cons are summarized.

Chapter 4 is concerned with Ontomind - a mind mapping software which takes advantage of semantic technologies. First the examination of its concept, then its software architecture and purpose of Ontomind is presented. Starting with section 4.4 a deep discussion of the features of Ontomind is given, focusing on how it takes advantage of ontologies and how it affects traditional mind mapping techniques. Then the pros and cons are summarized in the end.

Chapter 5 is devoted to creating the methodology of how to improve Ontominds ontology driven mind mapping approach. The focus is on bringing back what is great about mind mapping and improving further what has Ontomind done right.

Chapter 6 is concerned with the implementation. First the Ontominds setup process is discussed then in the second part the actual implementation of chosen improvements proposed in previous chapter is described.

Chapter 7 is devoted to the demonstration and testing of the new implemented features. In the first part the specific domain ontology from students life is developed for the purpose of the demonstration. Then in the second part of this chapter the new implemented features are tested according to the predefined test scenarios.

Since the structure of this diploma thesis is not ordinary and the personal contribution to the topic is scattered into different chapters and sections, the text in appendix A and a figure (Fig. A.1) identifies the personal contribution to this topic in a from of a mind map. The structure of this mind map mirrors the structure of the diploma thesis and the personal contribution is marked with the light bulb icon next to the section name.

Chapter 2

Mind maps

"Your brain is like a sleeping giant." - Tony Buzan

In this chapter, first the issue of mind mapping is discussed. A short course of mind maps history is presented and following is the presentation of different use-cases in which mind maps are commonly used. Then a short vision of the future of mind maps is presented. In the end of this first part mind maps are shortly compared to one other, more formal knowledge capture technique called concept mapping, which is often considered as a middle ground between mind maps and ontologies presented in chapter 3. Most of the knowledge used and elaborated on in the first part of this chapter is presented in [3].

Then, in the second part of this chapter starting with the section 2.6 the analysis of the mind maps is done. Their pros and cons are extracted from the typical use-cases and presented together with the proposal of simple guidelines on how to create a good mind map.

2.1 History of mind maps

Mind maps were invented in 60" of the 20th century by a man called Tony Buzan. The whole idea however started a little bit sooner and was driven by his will and hunger to answer how human brain function.

Tony was a student at the university when he started to realize, that the amount of his responsibilities is growing bigger and bigger. He tried to borrow a book about a brain from the university library, but soon discovered that he did not wanted to operate his brain like a brain surgeon, but to understand how it works. He realized that such book probably doesn't exist. That was the moment which lead him to studies of psychology, neurophysiology, theory of information and many more.

During his studies he found out that human brain functions better, when all of its parts can work together in harmony, than when someone uses just one part of his brain for a specific purpose. The amazing discoveries were made by little things, like combining words and colors, because they connect emotional and rational part of the brain. Incorporating images into note taking which comes from the saying: "A picture stands for a thousand words".

Then, during the 70" and 80" he traveled around the world and presented his ideas to many professors, businessmen, academics, students and other people. Right now, Tony is working really hard on improving his computer program called iMindMap[16] which can leverage the use of mind maps in computer science and offers many advantages when it comes to mind map creation.

2.2 Definition of mind maps

Mind maps are defined from many different perspectives. The most layman definition according to [3] is that the mind map is a graphical tool for holistic thinking which supports all of the brain functions - mainly our memory, creativity, learning and all additional thinking.

Making use of the statement mentioned before - "A picture stands for a thousand words", and looking at the following figure (Fig. 2.1), the basic understanding of what is a mind map and how does a mind map look like can be derived.

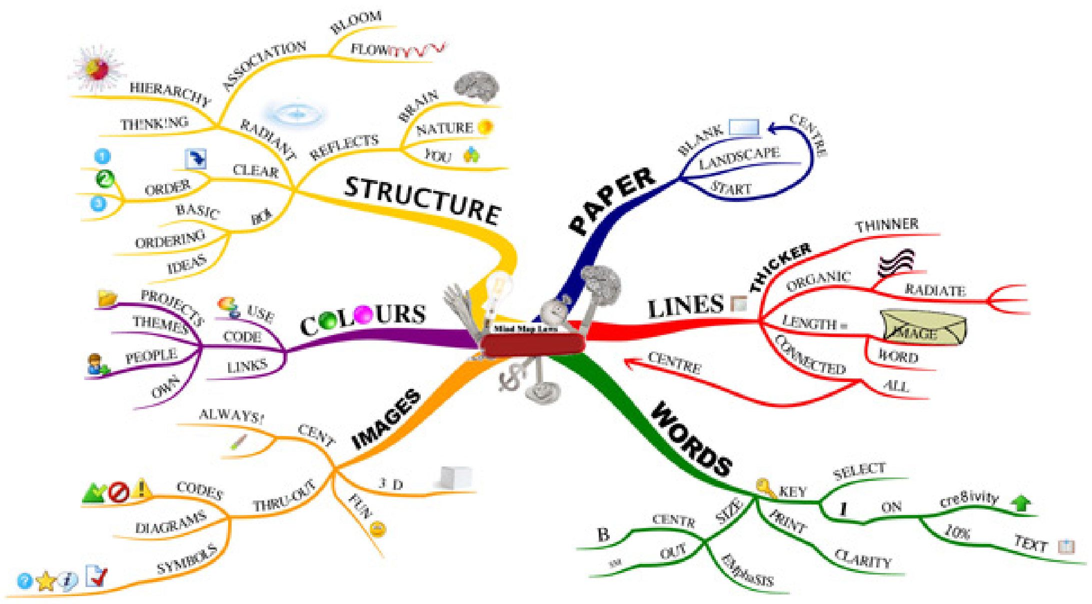

Figure 2.1: Mind map with the structure and mind mapping techniques taken from [18].

Mind map is a visual expression of our radiant thinking[1]. It is a process of how the brain thinks and come up with ideas. By capturing it and giving it a form of mind map the mirror image of what is happening inside of our heads is created.

[1]Deep discussion about radiant thinking is out of the scope of this thesis. Essential explanation of radiant thinking is that it is the way the human brain works. It is connected to the physical structure of the brain and neurons and it stands on the concept of multitude of associations to one word, idea or topic. For further reading about radiant thinking, please refer to [2].

From the presented figure (Fig. 2.1), a more universal definition is formulated. Mind map is a diagram used to graphically outline some information. The associations branching form the central node (the main idea), are hierarchically organized and radiate from the center. The sub-branches of larger branches are words, tasks or other items which are related to the main idea.

Mind maps are also defined formally in [12]. However the formal definition is extensive and it is not essential for the purpose of this thesis.

Nice summing definition of a mind map is presented in [18] - *"Mind Map is a powerful graphic technique which provides a universal key to unlock the potential of the brain. It harnesses the full range of cortical skills – word, image, number, logic, rhythm, color and spatial awareness – in a single, uniquely powerful manner. In so doing, it gives us the freedom to roam the infinite expanses of our brain. The Mind Map can be applied to every aspect of life where improved learning and clearer thinking will enhance human performance.*

2.3 Use of mind maps

In this section the discussion of how can mind maps be used in various scenarios and how can they help with everyday tasks is presented.

2.3.1 Making a decision

Mind maps can be used when making a decision. Something similar to pros and cons list. By putting the objects of the decision in the center of the mind map and branching out every criterion which is important, the structure of key criteria is created. Assignment of colors on branches and nodes or the pros and cons helps to differentiate the two objects.

The advantage of mind map over the pros and cons list would become more apparent when other associations on branches already created are introduced. Then, the two objects can be compared on more specific criterion. An overview of what the brain can come up with when thinking about those objects of interest is created. Such dyadic decision mind map is depicted on the figure (Fig. 2.2) which was made for the purpose of making a decision to move because of the new job.

2.3.2 Organizing information

Mind maps can also be used for note taking or gathering information. By creating a mind map of the topic when trying to take notes, the information is stored as the mirror image of the mind on the paper or in a computer. Notes are written in harmony with how the brain works and thinks, thus enhancing the ability of the brain to easily remember and understand the information.

By taking a look at the mind map, the distinction between what is important and what is not become immediately apparent. In a blink of an eye, the connections and relations between key motives are identified by the brain, giving it a better structure and overview of the topic. Furthermore the mind map can accommodate notes from the presentation (or speaker) and the notes from the author himself (mind map creator) in a non-intrusive way.

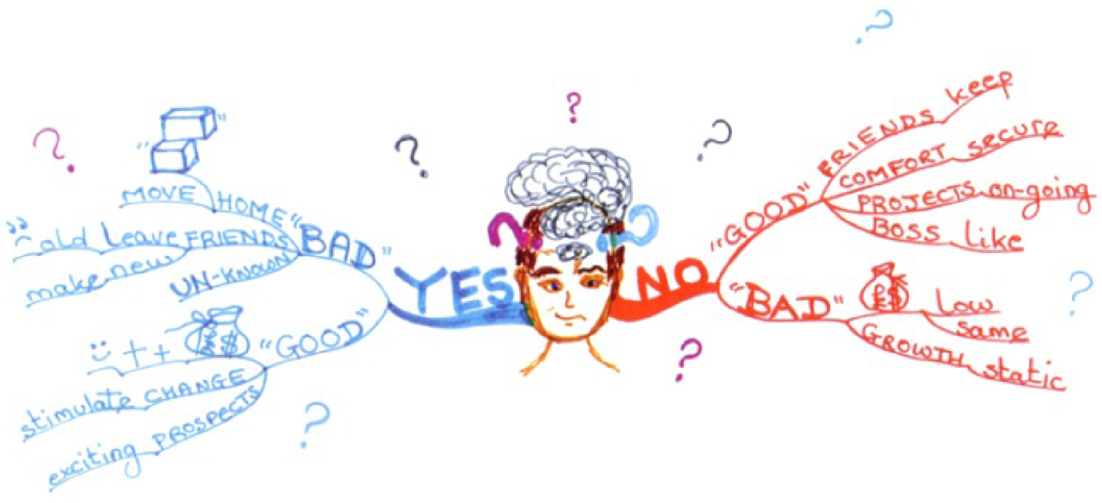

Figure 2.2: Decision making mind map taken from [14].

This way also what the mind has to say to this topic is captured, which aids the brain allowing it a better understanding of the problem at hand.

2.3.3 Being creative

Brainstorming for ideas is the best use-case of the mind maps. By taking advantage of the power of associations used in mind maps, the brainstorming session with colleagues or friends could become generators of enormous amounts of ideas.

Every creative option of the main topic can be searched with mind mapping slice & dice approach. Prejudices on specific thoughts can be erased and mind can be freed of traditional thinking with the mind maps ability to create new associations from already stated ideas and thoughts.

Combination of extraordinary ideas and objects can be achieved by seeing the same topic or object from different perspectives[2].

2.3.4 Time management

Ordinary diaries usually have only one week on one page, and are usually bounded by working hours, which is not the case of mind maps. Mind maps used for time management can capture different time spans, or just an individual event, or mix the two approaches together.

By using mind maps for time management, the identification of what needs to be done over the course of the specific time frame is done immediately. Furthermore the structure of the timetable is better recognized and understood by seeing the whole picture.

[2]This idea creation and combination is extremely helpful and easy in computer environment, where the creators are not tied by the limits of the paper and can expand their ideas almost infinitely. Imagine the possibilities in an environment where the mind map creator can create, mix, edit, reorder, structure and erase his ideas in seconds. Only the mind is a frontier, but mind has no known boundaries.

An example of the time management mind map is shown on figure (Fig. 2.3).

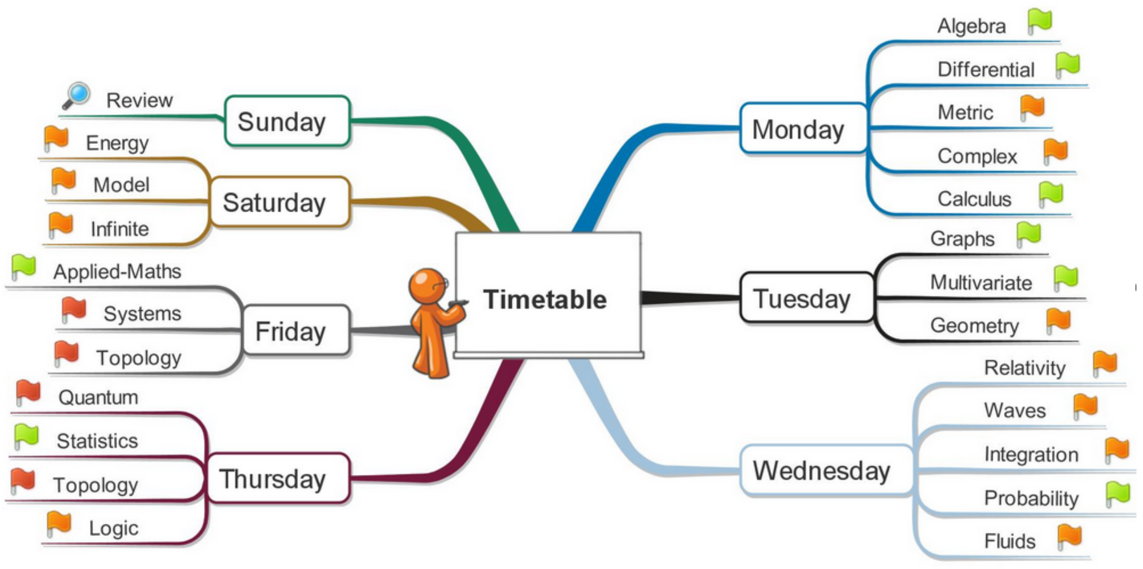

Figure 2.3: Mind map template for organization of week agenda taken from [19].

2.3.5 Presenting

Academics - teachers and students have probably prepared many presentations and know how hard it can be to make a great and visually attractive presentation. Mind maps and even mind mapping computer programs can help them out.

Mind maps can show the whole outline of the presentation. By using only the keywords and images, the presentations in form of a mind map are more visually attractive and focus on the structure of the presentation, thus helping out others to understand the topic better.

Furthermore with the power of associations, presentation mind maps can contain only the key concepts and don't have to focus on actual textual information.

2.4 The future of mind maps

Although the mind mapping technique is not that old, many people all over the world started to realize their importance. Few decades back, mind mappers were forced to draw their mind maps on a piece of paper. In a modern society, where computers are evolving so fast, there are many mind mapping programs that can make life easier.

The infinitely long and infinitely wide piece of paper of computer made mind maps on which can be drawn is one of the best advantages. With so much space there is no need to constraint the associations to those that are most appropriate at the moment. They can be listed all. Furthermore, manipulation, colorization and restructuring of the mind map and its nodes and branches is done with ease in seconds. Such modifications to the mind maps created on paper with crayons would consume too much time, material and effort.

With the power of computer mind mapping programs the presentation of the mind map to others is much easier. Presentations based on mind maps are done in a few clicks. Sharing with our friends and colleagues, converting from one format to other or even linking other mind maps and resources is easy. Analysis of the structure and content of mind maps can help to develop better algorithms for artificial intelligence and actually make robots to understand human brain based on the power of associations.

Because of the growth of computer technologies like desktops, tablets and oversized smartphones, mind maps are facing bright future and probably will take place in peoples lives more and more in upcoming years. With these devices being adopted in schools, mind maps may be used in education, making the learning process more effective and fun.

2.5 Mind map versus concept map

According to [4] a concept map is a graphical tool for knowledge representation and organization. Concept maps include concepts represented by the nodes and relationships between concepts represented by the links connecting nodes which together form a proposition. Concepts are defined as a perceived regularity in objects or events designated by a label. Propositions are meaningful statements about objects or events in the domain and usually contain two or more concepts linked together by relationship labeled by a phrase to form such statements.

A really simple concept map is shown in the figure (Fig. 2.4).

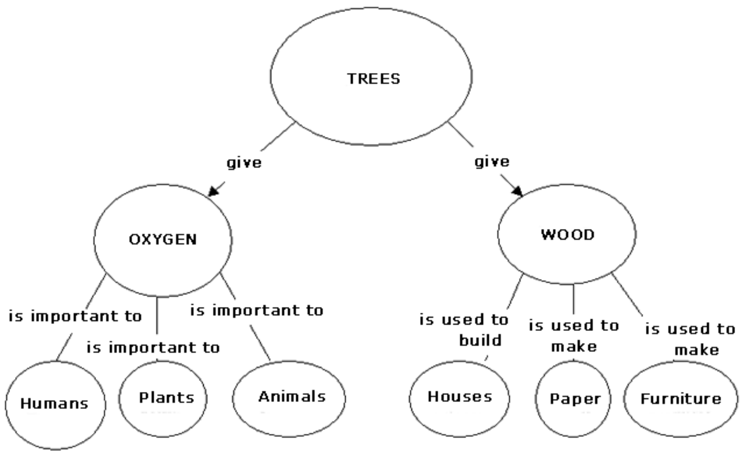

Figure 2.4: Concept map depiciting a specific domain taken from [1].

From the presented figure (Fig. 2.4), the similarities between mind maps and conceptual maps in appearance are obvious. While they share a few similar use-cases like brainstorming and information capture, the structure of concept maps and their purpose is different.

Mind maps usually reflect what the creator thinks about a topic. A single topic. Thus restricting a mind map to have a tree structure and organize information in a radial hierarchy. With such tree structure, nodes of the mind maps are restricted to have just one parent node. Making a link between two nodes in a mind map is just stating that two ideas are somehow connected, but not defining the actual relationship.

Concept maps however have more free form. They usually capture knowledge about a set of concepts and relationships between them, so they are not restricted to just one central node, topic or a concept[3]. Concepts may be organized hierarchically into a tree structure from the most generic one to more specific one, but this feature is used just for readability and knowledge understanding. With such approach and structure they are able to present more complex ideas than mind maps. Furthermore, with the linking of the concepts through relations, concept maps help development of thinking by revealing how individual ideas can form a larger one.

The purpose of concept maps is to capture the information in a more formalized manner than mind maps. Concept maps try to formulate the knowledge in a form of meaningful statements which helps reuseability and readability making them more shareable. Therefore concept maps are used more in use-cases where logic and shareability is needed. For example in:

- lexicon development
- collaborative knowledge modeling
- creating new knowledge
- presenting whole structure of some idea
- creating a shared vision and understanding
- and many more ...

The creation of concept maps is much more time consuming, since it has to focus on the logical hierarchy of concepts and relations between them and not just associating things together like mind maps do. But with additional time spent on such logic, they are more formalized and can be shared and reused much easier than mind maps. Furthermore, the formalization of the knowledge captured in the concept map may be considered as a first step of developing an ontology.

However, a more thorough analysis of conceptual maps is not relevant to the following discussion on the topic of this diploma thesis. For further reading about conceptual maps, please refer to [4].

[3]Computer made mind maps can also have hyperlinks used for navigation between the mind maps, which can possibly form a graph, but typically mind maps have a tree structure

2.6 Mind maps analysis

As presented in section 2.3.3, the power of creativity and expressiveness with use of associations of mind maps is outstanding and cannot be matched with typical note taking techniques for brainstorming purposes. Mind maps are able to generate more new ideas than any other known technique. When using the so called "slice & dice" approach, mind maps can take things apart and look at the main topic from different points of view. Furthermore, with slicing and dicing the subsequent nodes of the mind map the process continues and dives deeper. With such structure of mind maps the big picture of what information is captured is recognized immediately.

The usage of images, colors, different fonts, icons, tags, etc. makes the mind map even more expressive and easy to read by the brain. Icons can have great expressiveness and just by making the font bolder or bigger a whole different meaning is achieved in the context of a topic described by a mind map.

The organic structure of mind map itself is one of the key reasons why mind maps are so helpful. As presented in section 2.3.2 mind maps try to mimic how the brain thinks, works and stores associations and information inside, thus making the mind map and the information stored in it more accessible and understandable for creators, readers and the brain.

Within mind maps, the captured information can be reorganized freely. There is no prescribed hierarchy, no rules, no boundaries to what and how it should be captured. This feature of moving everything around, creating new associations and ideas is also one of the key benefits of mind maps. By making a mind map a personal mirror of the creators brain and thinking the burden of learning new things, taking notes or describing something can be easily handled.

However, the personal mirror of the brain is also a key disadvantage of using mind maps. Personal means that it is meant for just one individual. And with mind maps it is the same. Because the way that each individual associates things with one another, the mind maps make perfect sense just for their creators. While it is sometimes easy to understand some mind maps which are made to express some common knowledge or just as a bare form of note taking, it can be really challenging trying to understand why the specific person chose one word or association over another.

This particular disadvantage becomes more apparent when more experienced mind mappers with their own techniques of taking notes, vocabulary, tags, icons and images are encountered. Even the use of colors can have different meanings in heads of more experienced mind mappers. Looking at such mind map and trying to come up with the big picture might be even impossible for complete beginners or just people that are not in sync with the personal techniques of the mind map creator.

Mind map is a quick and easy way to capture any information, but only to some extent. With the extended amount of information captured, the mind map itself becomes hard to read. Furthermore the mind map branches and nodes can have images, tags, different colors and fonts used, and other means of brain stimuli which may result into confusion when looking at such mind map.

The process of making a mind map is characteristic to each individual and in case of mind mapping beginners it is also characteristic to each mind map creation. The beginners may capture the similar information in a different way or structure. Such repeated reinvention of how the similar information is captured in the mind map might results into another issue of reuseability of the mind map even for a particular individual.

Mind mapping is a graphical technique used to outline some information or knowledge. The use of graphic techniques usually means that it would require more space than just writing down the information and this is also the case of mind maps. Mind maps can accommodate many branches and nodes. Together with lots of images and tags and other visual perks mind maps are not very concise and can take much more space than just ordinary notes.

2.7 Mind maps pros and cons

From the analysis in previous section the advantages of mind maps are summarized:

Creativity - Mind maps are good for creating new ideas during brainstorming sessions.

Slice & Dice - Mind maps are able to capture information about some topic from different points of view.

Quick - Mind maps are quick to create, because they use associations of the brain.

Easy - Mind maps are easy to create, because of the mirroring of the brain thoughts which is more natural.

Adaptation - Mind maps are using organic structure which can adapt to most use-cases.

Management - Mind maps are easy to restructure, because everything can be moved around freely.

Stimulation - Mind maps use visual means like colors, images, fonts, etc. for brain stimulation and associations.

Outline - Mind maps are great for outlining captured information and presenting a big picture of the topic.

Also the disadvantages based on the analysis from the previous section are summarized:

Interpretation - Mind maps are not very precise, thus the information captured can have ambiguous meaning.

Not standardized - Mind maps may capture the same information differently every time.

Concision - Mind maps usually takes much more space than just ordinary notes.

Organization - Mind maps might become very unorganized when lot of different information is captured.

The pros of mind maps are aimed at quick creation of the mind map, high adaptability to use-cases and creativity. Most of the cons of mind maps presented are tightly related to what is known as the biggest problem of mind maps and that is reuseability.

2.8 Create great maps

From the presented pros and cons in a previous section, the guidelines on how to create good mind maps can be derived. Focusing on minimizing the cons and maximizing the pros.

One topic - Mind map should have just one central topic. Slice & dice approach helps to look at the central topic from different points of view. When there is a need of new topic, new mind map should be created. One central topic keeps the brain focused and makes the process of mind map creation straightforward. One topic helps to keep the mind map relatively small, which also helps orientation and readability.

Use colors - Mind map should make use of colors. Colors might help differentiation of points of views when slice & dice approach is used. Colors might help distinguishing important information form not important. It may help with associations of feelings with specific color, and many more.

Use images - Mind maps should use images, icons, tags and other visual means. Following the simple principle, that an image is worth a thousand words, is the best explanation. Human brain can associate much more words with an image, because the image gives the brain not only meaning but also the visual stimuli to create more associations based on color, type or quality of the visual mean.

Be concise - Mind maps should not use long phrases or sentences. This guideline becomes much more important when trying to be creative during brainstorming sessions. The whole sentences are bad because they tightens the field of new associations. Anyone can come up with far more associations to just one word, than to a whole sentence, because the sentence as a whole somehow stops the chain of associations[4].

Own technique - When becoming more experienced and used to making their own mind maps, creators should develop their own vocabulary, tags, acronyms, abbreviations, icons and use different shapes which will bear a specific meaning. Such own technique may have two different impacts on mind map readability and shareability. For individuals which are not familiar with such techniques, it may render the mind map unreadable and useless, but on the other hand to individuals which are familiar with the technique, such mind map is easy to read and to understand it correctly, because the techniques may add clarity to the captured information.

[4]By writing down a whole sentence the associations are narrowed down to follow up the meaning of the whole sentence and thus creating associations within its constraints. The following example will try to describe what is it about. Imagine making a mind map about lemonade. Branching out from central topic and writing down "Momma's lemonade is the best" will somehow focus the subsequent associations with mother's lemonade. It constraints the mind to think about that particular type of lemonade, and not giving it the opportunity to explore the wonderful universe of lemonades.

2.9 Mind maps summary

In the first part of this chapter mind maps were successfully described together with their usage in different scenarios. A future of mind mapping was envisioned and a simple comparison to conceptual maps was presented. In the second part a successful analysis of mind maps has been made together with the identification of pros and cons of mind mapping. Some guidelines for creating mind maps were proposed based on the previous analysis.

The final outcome of the analysis is that mind maps are quick and easy to made, excellent when in the need of creativity, highly adaptable to any use-case and many more. On the other hand, they are very personal and the biggest problem is the reuseability and shareability of the information captured by the mind map.

Chapter 3

Ontologies

"Making computers understand."

Semantic technologies helps computers and programs to understand the meaning of their doings. They try to derive the meaning of the information presented. In essence they can be thought of a new level of intelligent, relevant, capable and responsive interaction with information. According to [5] semantic technologies are algorithms and solutions that bring structure and meaning to information, they help with separation of the signal from the noise. Ontology is a one of semantic technologies used in today world.

In this chapter, first the issue of semantic web technologies in form of ontology is discussed. A short history, then the definition, structure and common syntax of ontology is presented. Following is their purpose in todays world and description of different types of ontologies used in different use-cases.

The second part is devoted to the analysis of ontologies which starts with the section 3.8. The pros and cons of ontologies are extracted from the typical use-cases and summarized in the end.

3.1 History of ontology

Ontology as a term has its roots in ancient philosophy and has many different meanings from which the best one for description would be the meaning - *"science of being"*. It is originally a philosophy branch which deals with the organization of reality and nature. Ontology tries to come up with answers for questions like - *"What is existence?"*, *"What is the being?"*, and many more.

In computer science and artificial intelligence research, ontology represents knowledge representation as a set of concepts. Researchers tries to describe the building blocks of which the objects of the real world are made.

Also as mind maps the ontology studies in computer science are very young. Since the beginning of ontology study in early 90's, it has attracted a lot of attention in knowledge engineering communities. Especially in last few years with the really fast evolving world wide web technologies and a phenomenon called semantic web. However, there are so many opinions what ontology really is that they are starting to confuse. In the following sections the definition of ontology in computer science and artificial intelligence studies as well as various types and where they are used is discussed.

3.2 Definition of ontology

There are many different definitions of ontology. These definitions range from really simple ones with all the meaning behind the words to really thorough descriptions.

In few words, according to [10] and Thomas Gruber, ontology is an explicit specification of conceptualization. This definition is formally accepted in the artificial intelligence community. It means that the ontology is a description of concepts and relationships that can exist for an agent or a community of agents.

Conceptualization can be defined as intensional semantic structure which encodes implicit knowledge constraining the structure of a piece of domain. An ontology is a specification of this structure. Conceptualization is an abstracted and simplified view of the world which will be represented. Every knowledge base, knowledge-based system or agent is committed to some conceptualization, explicitly or implicitly. The set of objects which can be represented is called the *universe of discourse*.

The *universe of discourse* is a set of objects and relationships between those objects which forms a representational vocabulary within which a knowledge-based program represents his knowledge. In computer science and artificial intelligence ontology of a program can be described by defining a set of representational terms. In such ontology, definitions associate the names of entities in the *universe of discourse* with descriptions of what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms. Formally, it can be said that an ontology is a set of statements of a logical theory.

Simply - ontology formally represents knowledge as a set of concepts in particular domain and relations between those concepts. Furthermore, by presenting a shared vocabulary it describes the types and properties of those concepts.

3.3 Structure of ontology

The following figure (Fig. 3.1) shows a really simple excerpt of pizza ontology presented in [11] in a form of a graph. The boxes on the diagram represents concepts of a given domain and the connecting lines and arrows represent relationships.

The presented ontology domain is concerned with pizza. It states that in this world there is a concept of *Pizza*, which consist of *PizzaBase* and *PizzaTopping*. These are yet another concepts of this specific domain. Furthermore there is a simple hierarchy and inheritance in place, stating that for example *PizzaBase* can be specified further to be one of *DeepPanBase* or *ThinCrispyBase*. The same logic applies for the *PizzaToppings*.

From the structural point of view ontologies consists of these main building blocks:

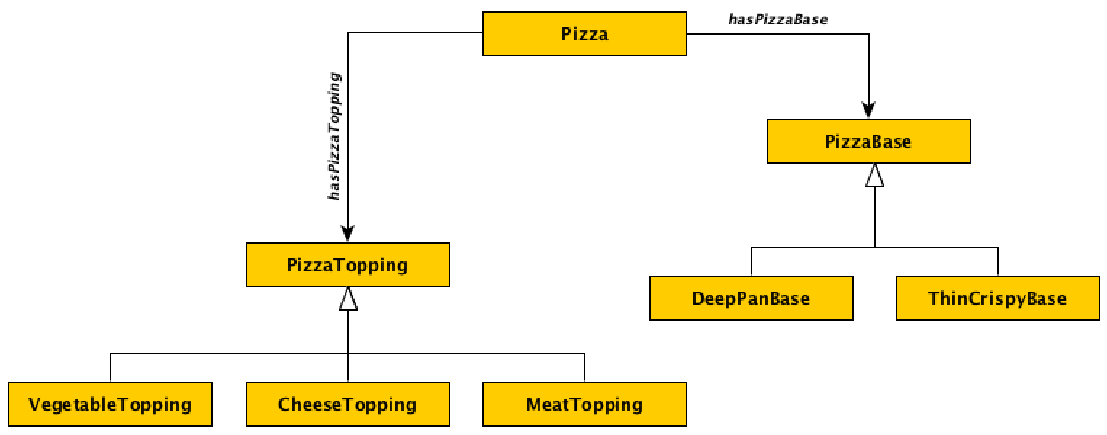

Figure 3.1: Ontology of pizza specific domain inspired by [11].

Classes - Describing the concepts in the specific domain. (e.g. *Pizza*, *PizzaTopping*, *PizzaBase*, etc.)

Individuals - Actual Instances of the classes. (e.g. *MyMargheritaPizza*, *MyDiavolaPizza*, etc.)

Object Properties - Relations between classes and/or individuals. (e.g. *hasTopping*, *hasBase*, etc.)

Data Types - Data types of primitive form values (e.g *short*, *byte*, *integer*, etc.)

Data Properties - Relations between classes or individuals and values in their primitive form. (e.g. *strings*, *literals*, *numbers*, etc.)

Annotation properties - Additional informative data, not affecting ontology in structure and functionality (e.g *labels*, *comments*, *version*, etc.)

3.4 Syntax of ontology

While it is nice and appealing to have graphic form for representing ontology, that is not the most commonly used form for transferring ontological knowledge. To ensure easy transfer over the web, semantic web ontologies typically uses textual formats. There are various textual forms of syntax for expressing ontology knowledge from which two of them are used commonly on the web - *RDF/XML* syntax and *OWL/XML* syntax.

The *RDF*, *RDFS* and *OWL* shortly described[1] in following paragraphs are in fact web standards for expressing knowledge and capturing information over the web. The original versions become a *W3C* recommendations in February 2004 and are generally understood

[1]To describe *RDF/RDFS* and *OWL* thoroughly is simply out of the scope of this diploma thesis. For further reading please refer to [7] and [6]

by the industry and the web community. These standards provide structure for describing identified things and are built on top of one another. They are designed to be read by computers and to provide a common way to process the information on the web. These standards are essential parts and building blocks of what is today known as a semantic web and linked data.

RDF is a framework used for expressing information about resources. With *RDF* almost everything is a resource - classes, individuals, properties, etc. A resource in a real world is basically anything - people, images, documents, physical objects, abstract concepts, etc. *RDF* uses a specific small set of terms and structure in a form of *triples* which allows to make statements about the resources. The form of a statement is as follows:

```
<subject> <predicate> <object>
```

With such structure, the *RDF* is really powerful to express almost any information about some resources, for example[2]:

```
:Dominic  :owns  :Computer
```

The downside of pure *RDF* is that it allows every part of the statement to be any resource, the statements which are not meaningful are also allowed. The following example represents such situation:

```
:Orange  :Apple  :Mango
```

Such statement is clearly meaningless, but in fact it is a valid *RDF* statement.

However, *RDFS* deals with such meaningless statements. In fact *RDFS* is a semantic extension on *RDF* and adds vocabulary for the schema and data modeling. It provides vocabulary and mechanisms for groups of resources (related resources), relations between resources and their characteristics. *RDFS* in an example:

```
:Horse  rdfs:subClassOf  :Animal
:hasLeg  rdfs:domain  :Horse
:hasLeg  rdfs:range  :Leg
:Horse  :hasLeg  :Leg
```

As presented in this simple example, *RDFS* added some vocabulary to define hierarchies and constraint properties to be applied only to some of the resources and vice-versa.

Thinking that *RDFS* might be enough (in fact for some ontologies which serve as a simple taxonomies it is enough), there are actually many things missing in comparison with all what *OWL* is capable of. *OWL*, and especially *OWL2* which is an evolution of original *OWL* used and referred to in this diploma thesis, is yet another extension of *RDF/RDFS* which adds even more expressiveness, so it is a stronger language with greater interpretability than *RDF/RDFS*. *OWL2* became a *W3C* recommendation in December 2012.

OWL2, adds means to:

[2]The *prefix:* is just a label for namespace for the terms to know which vocabulary are they coming from, similar to classic xmlns used in *XML*. Blank namespace means that the term comes from a global namespace (or a specific domain)

- state that two things are exactly the same,
- define cardinality of relations,
- say that something is a complement of something else,
- say that two classes are disjoint,
- state that a property is transitive, functional or inverse to some other property,
- and many more ...

The following is an example of *OWL2* used in practice:

```
:Car  :hasPart  :Wheel
:hasPart  owl:inverseOf  :isPartOf
:Wheel  rdfs:subClassOf  :CarPart
:Mirror  rfds:subClassOf  :CarPart
:Mirror  owl:disjointWith  :Wheel
```

As it is stated in the example, there is a simple hierarchy of concepts defined by *RDF/RDFS*. But with *OWL2* also the information that some classes are disjoint can also be read. *OWL2* introduced yet additional vocabulary, so the information can be interpreted even more precise than before with just *RDF* and *RDFS*.

3.5 SPARQL

SPARQL is a standardized query language for *RDF*. It is also a web standard and original version became a *W3C* recommendation in January 2008. Nowadays, *SPARQL 1.1* is used which is an evolution of original version and became a *W3C* recommendation in March 2013. From now on, the term *SPARQL* will be used in reference to the actual recommendation which is *SPARQL 1.1*. *SPARQL* is yet another of the key technologies of the semantic web.

SPARQL is in fact a query language for a database (similar to *SQL* in relational databases), which stores data in *RDF* format. However it can work on any *RDF* data format and also on classic relational databases that can me mapped to *RDF* format. The database is then a set of *triples* as described in a previous section.

Similar to *SQL*, *SPARQL* also provides full set of analytic functions for sorting, joining and aggregating the data. The key difference of *SPARQL* is that it uses a specific feature of *RDF* that the schema of the data is intrinsically a part of data itself, thus the requirement of data structure is not needed or it is provided externally (e.g by the ontology).

The simple query syntax is shown here:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX usr: <http://example.com/UserOntology#>
SELECT ?email ?phone
WHERE {
 ?user rdf:type usr:User .
 ?user usr:email ?email .
 ?user usr:phone ?phone .
}
```

This query simply returns a set of tuples of email and phone for every user which have been associated with one or more name and phone resource. As it can be seen, the *SPARQL* syntax and behavior of query is somehow similar to *SQL* with just little deviations.

There are few different types of queries in *SPARQL*:

SELECT - query returning raw data in a form of table

ASK - query resulting in simple *true/false* value

EXTRACT - query returning an *RDF* representation of the data

DESCRIBE - query returning a description of resource and any resources directly related to it

The *SPARQL* provides very powerful means to ask almost any kind of question on a particular data set and retrieve a suitable answer.

There is also a versions of *SPARQL* specifically tailored for use with *OWL2* called *SPARQL-DL*. It's a subset of *SPARQL* query language used for ontology specific questions. However, within the scope of this diploma thesis, the use of *SPARQL* will be enough and *SPARQL-DL* won't be described thoroughly.

3.6 Purpose of ontology

The main purpose of ontology is to enable sharing and reuse of the knowledge. They are built, because of the need for consistent and coherent communication of information between agents. Ontology provides means for sophisticated organization of knowledge.

The following example originally presented in [8] would be a typical use-case where ontologies can help.

There are two people that have programmed an algorithm to solve some real world problem, each one on their own. They both have embedded their own conceptualization of the specific domain into their application. When they start to communicate to each other about the solution they immediately realize, that their implicit understanding of the specific domain world is fundamentally different. That they have thought of the problem in a different ways. There may be some different meanings behind some concepts and even differences like missing concepts between the two specific conceptualizations of each other programs.

When their programs would have to communicate with each other, some problems in the communication might arise. There would be few possible solutions to solve the communication problem. The worst solution is to rewrite one of the programs to commit to the conceptualization of the other one. Other solution is to create some adapter for mapping from one conceptualization to another. But the conceptualization of the programs would have to be described explicitly by some formal knowledge - ontology. The latter solution would be more suitable in reality, but such explicit description is often really time consuming and expensive.

However, if there was an ontology in the first place to commit to, the communication between the two programs would be solved much easier.

From this described example we can say that an ontology is a description of essential understandings of the agent on the world of interest and is useful to come to mutual understanding through explicitly specifying of what was left implicit. The reason of mutual communication and understanding is why it is important to study the ontologies and semantic technologies in general.

3.7 Types of ontology

Some people say that ontology is a domain specific knowledge base system. Others say that ontology is very generic, and thus it can be applied and shared in wide range of domains. While these opinions are against each other, both of them are right in some way. Both categories of people are right, because they are describing different types of ontology.

Basically, according to [8] there is a number of different types of ontology:

- Upper ontology
- Task ontology
- Domain ontology
- Heavy-weight ontology
- Light-weight ontology

In the following sections these types of ontology are presented together with the discussion in which situations they are applied.

3.7.1 Upper ontology

Upper ontology is used when there is a need of very high reuseability. Usually, It defines higher level categories, which are trying to describe for example - what is being, what exists, objects and relations in the world.

One of the typical examples of upper ontology would be Pierce's and Sowa's [15] three higher level categories which are:

firstness describes the form, here belongs objects, beings, etc., which can be defined without assuming other objects in the world, according to this ontology, such objects are for example - human, iron, wood, etc.

secondness describes the roles of objects, relations, etc. These have to be defined necessarily depending on other objects, such objects are for example - father, brother, doctor, etc.

thirdness describes mediation - which provides context or environment for the secondness, typical examples would be family, school, hospital, etc.

There are many upper ontologies. Therefore many groups of researchers are trying to come up with one universal or standard upper ontology. One of the most comprehensive standard upper ontologies is *SUMO* which is being developed at *IEEE P1600.1* [13]. It describes what is identity, how are *3D* and *4D* space modeled and how they are compatible or different.

As described, upper ontology tries to apply in every situation, it tries to describe everything from the point of higher level categories. While this aspect is good for reuseability and shareability of the ontology, there is a big problem on agreeing what should be the best or main standard upper ontology. The development process of such generic ontology takes years of research and collaboration.

3.7.2 Task ontology and Domain ontology

Description of task ontology and domain ontology is presented together, while they usually come hand in hand.

Typical ontology would in most cases consist of a task ontology and a domain ontology. Task ontology would describe the architecture of the knowledge-based system which performs a task and domain ontology would describe the knowledge where the task is performed.

To give an example of the contrast between them, domain ontology will characterize and describe the vocabulary related to the specific domain (e.g. medicine, automobile, etc.), while the task ontology will characterize and describe the vocabulary related to the specific task or activity (e.g. diagnosing, driving).

There are however task and domain ontologies which do not come together. In such cases, for example the task ontology is aiming to describe the vocabulary for problem solving structure of the existing tasks domain-independently. Furthermore, the ultimate goal of such task ontology would include the theory of all the vocabulary and concepts for building a model of human problem solving processes.

When trying to determine how specific or generic such task ontology should be, deep consideration of the granularity and generality of the unit of problem solving action is needed. According to [8] and [9] such observations suggests four kinds of concepts:

Task roles are the roles played by the domain objects

Task actions are the activities appearing in the process

States are the states of the objects

Others are other concepts which are specific to the task

When viewing on a problem solving process based on search as a sentence of natural language as presented in [9], the description of task ontology would be a system of semantic vocabulary for representing the meaning of the sentence. Then, by applying previously mentioned 4 kinds of concepts the ontology would consists of four concepts:

Generic nouns which represents objects reflecting their roles

Generic verbs which represents the unit activities

Generic adjectives which represents the modifications of the objects

Others concepts specific to the task

The following example of task ontology used for scheduling tasks could be defined:

Nouns: "Schedule recipient", "Schedule", "Constraints", "Priority", "Due date", etc.

Verbs: "Assign", "Pick up", "Select", "Classify", etc.

Adjectives: "Assigned", "Unassigned", "Pending", "Last", etc.

Others: "Attribute", "Strong constraint", "Constraint satisfaction", etc.

Before task ontology, people believed in little reuseability of ontology, due to it's domain-specific task. The reason was that the task ontology mixes task ontology and domain ontology. If the task ontology describes the roles of the domain objects, the domain ontology developed after the development of the task ontology will become independent at least of the particular task. Because all the task-specific concepts are detached from domain concepts to form task-specific roles in task ontology. Thus developing a task ontology first will help with the development of more neutral domain ontology and raise the reuseability and shareability of the domain ontology.

3.7.3 Heavy-weight and light-weight ontology

Light-weight ontology is usually very use-dependent. Typical example of light-weight ontology could be a search engine ontology like Yahoo ontology. This ontology includes hierarchy of topics with very little considerations of concept definitions. It also does not pay much attention to principle of concept organization or distinction of concept and word. It usually have tree-like structure where nodes consists of propositional description logic formulas which are encoding the meaning of the nodes.

Taking this into consideration, each formula encoded node is subsumed by the formula encoded node of the above node. Following this principle - path from the root node, the backbone structure of light-weight ontology is represented by subsumption of relations between formula encoded nodes.

A little example is a node labeled "bicycle" under node labeled "mountain". Therefore it is a mountain bicycle. The logical formula would be "mountain AND bicycle".

Heavy-weight ontology can essentially be based on light-weight ontology, but in contrast to light-weight ontology, the heavy weight-ontology pays much more attention to rigorous meaning of each concept, semantic relations between concepts and principles developed in philosophy. Heavy-weight ontology requires careful conceptualization of the target world or specific domain.

The typical examples of heavy-weight ontologies are upper ontologies.

3.8 Ontology analysis

There are many types of ontologies as it was presented in section 3.7. However they all have one thing in common and that is the specification of conceptualization. In other words, ontologies are trying to be standardized and reuseable.

Ontologies are often used, when formalization of the knowledge is needed. The process of normalization helps identifying the important concepts in the specific domain - classes, object properties, etc. Formalized concepts are then often hierarchically structured, i.e. forms a taxonomy of the concepts, which usually is a stable backbone structure of the whole ontology.

This analysis is related more to the use of ontologies in mind maps, which benefits to the topic of this diploma thesis but most of the knowledge can be applied to a more generic ontology analysis.

One of the outcomes from this formalization process is a vocabulary which is used in the ontology. This vocabulary represents common set of words which can be used when capturing knowledge according to the ontology. This common usage of vocabulary helps with identification of the concepts and overall reuseability of the ontology.

The concepts in the ontology are defined precisely within the world of interest, which helps with the interpretation of the knowledge captured. By assigning a concept to a piece of information, the precise meaning of the information is set. Furthermore the relationships in the ontology are also precisely defined. This way, the pieces of information can be connected only in a way defined by the ontology and the final structure of the information captured is known in advance.

On the other hand, when the formalization of knowledge is introduced, the whole knowledge capture process is much more verbose. Everything needs to be captured in the way that ontology dictates. This may lead to capturing also some information, which is not important at the moment.

The ontology also dictates the structure of the knowledge captured. It has to conform and obey the axioms introduced by the ontology. While the restructuring of ontology is possible, it is a time consuming task and requires a deep analysis. Such restructuring cannot be done easily when trying to capture concrete data based on ontology.

Creating an ontology is a long process, the analysis of the important concepts, their generalizations and subsequent specializations is very time consuming. Furthermore the knowledge capture process also suffers from it, because during the process the concepts needs to be assigned to the actual information. The knowledge needs to be captured the right way, and this way is, as stated before, more verbose, more structured, more thought through.

Ontology is not able to capture the knowledge which is not defined. The new concepts for the new knowledge has to be created and properly accommodated into the existing ontology. This process of creating new concepts is very time consuming and it should be done by experts on the specific domain and not just anyone.

3.9 Ontology pros and cons

From the analysis of ontologies in previous section the pros are summarized:

Conceptualization - Provides precise definition of the ontology concepts and relations within the world of interest.

Clarity - Knowledge captured by good ontology should not be misinterpreted.

Standardization - Standardized way of capturing knowledge according to the ontology.

Structure - The structure of the knowledge captured according to the ontology is known in advance.

Vocabulary - Common vocabulary used for concepts and relations within the ontology.

And the most obvious cons are summarized:

Restrictiveness - The defined structure of knowledge captured by the ontology is restricting when capturing the information.

Adaptability - Ontologies are not good when capturing new knowledge which is not formalized in advance.

Time consumption - Time spent on creating the ontology and capturing the knowledge according to the ontology is substantially higher.

Verbosity - Knowledge captured according to the ontology is much more verbose than ordinary knowledge capture techniques.

In contrast to the mind maps, ontologies pros and cons are somehow opposite. The pros aims at high reuseability and cons are about time consumption and adaptation to different use-cases.

3.10 Ontologies summary

In the first part of this chapter ontologies as semantic web technologies were presented. Their short history, definition, purpose in todays world together with the structure and syntax they use was described. Following was the discussion about various types of ontologies used in different use-cases. In the second part a successful analysis of ontologies has been made together with the identification of their pros and cons.

The outcome from the final analysis is that ontologies are excellent when it comes to reuseability and shareability because of their specification nature and use of common vocabulary. On the other hand, ontologies lack adaptability, creativity and information captured by the ontology is usually more verbose. Furthermore the time spent on developing an ontology or just capturing information according to the ontology is much bigger.

Chapter 4

Ontomind

This chapter is devoted to the discussion about Ontomind. Ontomind is a tool developed at CTU in Prague, particularly department of cybernetics. As it is stated in [22] it is a part of MONDIS project which is aimed at efficient knowledge management of cultural heritage objects, which is used by civil engineering experts for collaborative authoring of structural damage records.

In the first part of this chapter the Ontomind application is described from the conceptual point of view and some of the main requirements for the application are mentioned. Then the short overview of the system architecture containing a brief discussion of technologies used is presented. Following is the purpose and main use case of Ontomind application for which it was developed.

In the second part starting with section 4.4, Ontominds utility for different use-cases of capturing information is analyzed together with the discussion about ontology driven mind mapping technique. At the end the pros and cons of Ontomind are summarized from a practical point of view.

4.1 Concept features

In the conceptual phase of the project, there were several functional features that were required for the system to provide to the users of Ontomind.

While the system should be able to author the information about cultural heritage objects, it should do it in an intuitive and easy to use form for domain experts. Authors of Ontomind have chosen mind maps, because of their ability to author any kind of information as discussed in chapter 2, and use rather fresh and intuitive form. Mind maps provide Ontomind with easy to use, visually attractive and generally accepted and understood way of such authoring process.

Another requirement was, that Ontomind should guide the user throughout the information capture process using some kind of formal knowledge. For this feature, authors of Ontomind have chosen ontologies, because they provide standardization of formal knowledge of a specific domain as discussed in chapter 3. During the first phases of the project, an extensive ontology for representing the concepts and formal knowledge to capture what was needed by the experts on cultural and historical objects was developed. With the power of such domain specific ontology, the users of Ontomind are provided with the guidance and verification during the information capture process according to formal knowledge described by the ontology.

Ontomind should also allow collaboration, thus it has to provide means for multiple users to have the access to the mind map creation process. That's why the authors chose Ontomind to be a web-based application. With such architecture, it allows multiple users to connect to the same workspace and work on the shared mind maps stored on a server.

4.2 System architecture

From the system architecture point of view, Ontomind is enormous piece of software. It combines several open-source projects for mind mapping, *APIs* for manipulating ontological data and adapters to connect to different underlying storage systems.

Ontomind is a web application based on a free open-source project named *wisemapping*, which is a tool developed specially for collaborative creation of mind maps. Ontomind extends *wisemappings* functionality and adds semantic web technologies in forms of ontology.

Ontomind has client-server architecture. A figure (Fig. 4.1) from [22] depicts how Ontomind is working behind the scene.

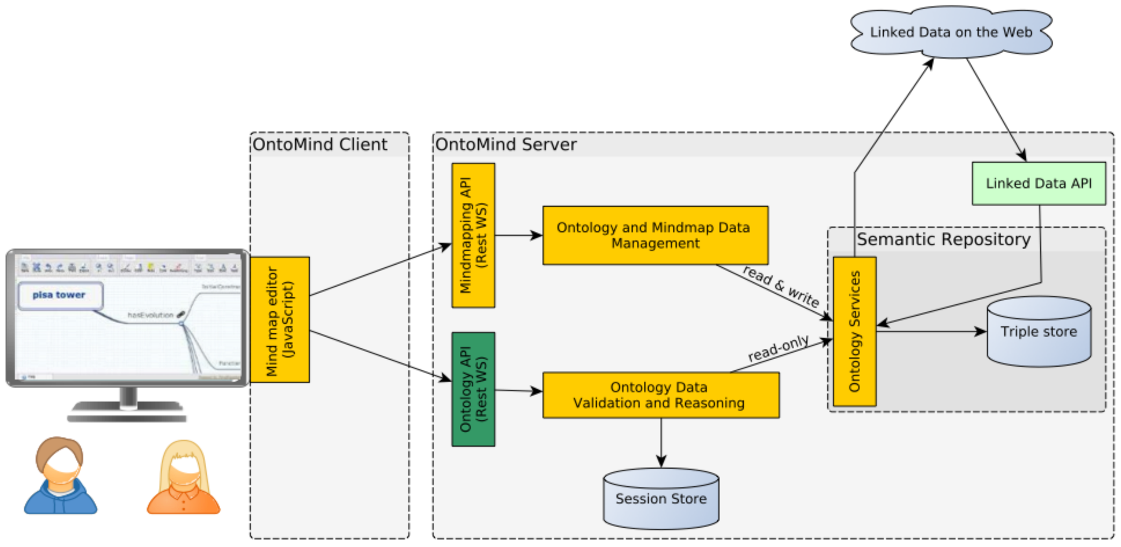

Figure 4.1: Ontominds system architecture taken from [22].

As depicted on figure (Fig. 4.1), there is a lot of going on behind the scene. The following sections are devoted to brief overview of the technologies from Ontominds technology stack worth mentioning.

4.2.1 Client

The client side is *wisemappings* rich and interactive tool developed specially for collaborative creation of the mind maps. *Wisemappings* client part itself is a project that stacks several other open-source projects to deliver rich interaction environment for manipulating *2D* graphical mind maps. For such interactive collaboration, it combines technologies like *HTML 5* and *Javascript* for the interactivity and *SVG* for *2D* graphics. *Wisemapping* consists of projects called *web2d* and *mindplot*.

Web2d is open-source project which is in fact a *Javascript* library for manipulation with defined *SVG* vector elements. The elements consists of geometrical objects like points, lines, poly lines, curves, ellipses, rectangles, and many more. *Mindplot* is also written in *Javascript*, but in contrast to *web2d* it is not a library, but rather a whole application with helper objects that allow the representation of the mind map as in-memory object. *Mindplot* allows:

- creating topics and branches of a mind map,
- creating links between topics,
- moving and restructuring of a mind map,
- serializing a mind map,
- and many more ...

In addition to mind map building blocks *mindplot* defines helper objects for collaboration of groups of users and means for serialization of mind maps over the HTTP protocol which allows communication with the server-side *REST API* of the Ontomind.

Client side of Ontomind uses another project called *AutocompleteTree*. This project is also developed by authors of Ontomind. *AutocompleteTree* is a component of Ontomind used for visualization, choosing and assigning the actual ontology concepts and relations for each mind map node. It is based on *Javascript* open-source plugin called *jsTree*, which is a plugin that represents the hierarchy of ontological concepts and relations in a from of a structured tree using *Javascript* and *HTML*. It is one of the key components of Ontomind.

4.2.2 Server

Server side of Ontomind consist also of few different projects. *Wisemappings* application server side used for manipulation of mind map data, Ontominds custom programmed services for manipulating ontological data and few other custom based adapters for connecting to different ontology providers. The server side is built on *Java* platform using various standardized frameworks for building web, enterprise and semantic applications. To name some of the most commonly used:

Spring framework - used for the web container bean management, *MVC* architecture, security , web services, and many more.

Hibernate - used for communication with databases, *ORM* mappings and transaction management.

Apache Jena - used for representation and manipulation of ontology models.

OpenRDF Sesame - used for representation, manipulation and storing of *RDF* data.

Ontomind uses two data storages. The first one, *Sesame*, is used for storing the formal knowledge. According to [21] *Sesame* is a standardized framework which aims at processing of *RDF* data. It provides means for manipulation, reasoning querying and storing of the *RDF* data in so called *triple stores*. Furthermore *Sesame* provides various implementations of such *triple stores* as *in-memory triple store*, *file-based triple-store*, *RDBMS triple store*, etc. *Sesame* also provides various connection types (local and remote) for connecting to *triple store*. With such features provided by *Sesame*, the ontology data can be retrieved in form of *RDF* triples from virtually any kind of storage system, which could be located on a local computer, or on a remote server. The second storage represents a more of a standardized approach in form of a classic relational database which is used by the server part of Ontomind and also *wisemapping* application to store actual data about the mind map and users of the application.

Apache Jena is according to [17] a free and open-source *Java* framework for building applications with the use of semantic web technologies. *Apache Jena* brings many different technologies and *APIs* to the table, from which the most important for the Ontomind is the *Ontology API*. *Ontology API* provides the means of the in-memory representation of the ontology, as well as the manipulation with the ontology model and making assertions and retrieving inferences. The in-memory model of ontology is stored in session based storage.

The server side also has two spring-based *REST* controllers which serves as a communication endpoints with underlying technologies. The communication is provided via *XML* or *JSON* serialization of the data transferred. First controller is an *Ontology API* which manipulates with ontology data, does reasoning, validation according to the rules and restrictions of the ontology, etc. The other *Mindmapping API* is a controller used for manipulation with the layout of the mind map.

4.3 Purpose of Ontomind

As mentioned in the beginning of this chapter Ontomind is used to effectively manage the captured knowledge about cultural heritage objects. However this is not the most important purpose. The current and main use of Ontomind is to validate the underlying ontology which was developed in the beginning of MONDIS project and is evolving over the time.

The civil engineering experts on cultural heritage are creating examples by capturing the information according to their needs and the underlying ontology. If they are able to capture the information needed in the specific scenario example with the underlying ontology then the validation of this example is complete. On the other hand, when the ontology is unable to capture the information, it needs to be altered so it will be able to capture such information in the similar future scenarios.

4.4 Ontology driven mind mapping

Ontomind is of course a mind mapping application, so it can be used in various use-cases where ordinary mind maps are used. In this section the impact of ontology on mind mapping techniques within Ontomind is presented. Then in the following section the pros and cons of such ontology driven mind mapping in Ontomind are analyzed and discussed according to the use-cases.

Authors of Ontomind have chosen to use mind maps together with ontologies. This approach brings together two fundamentally different approaches of information capture, because their usage and aim are, as it was analyzed in sections 2.7 and 3.9, exactly opposite.

As presented in section 2.3.3 mind maps are great for brainstorming sessions. The expressiveness and freedom of ordinary mind maps together with slice and dice approach is great for creation of ideas. According to the pros in section 2.7 mind maps are easy to restructure and the branches and nodes can be moved around with ease. However this is not easy when ontology is introduced into the structure of the mind map. As analyzed in section 3.8 the structure needs to commit to the conceptualization. Ontomind mind map nodes representing concepts can only be connected through mind map nodes representing ontology relations. The restructuring of such ontology driven mind map is possible only to the level which the ontology permits. In fact Ontomind does not allow moving of the mind map branches at all.

Mind maps are able to capture any kind of information and this benefit makes them highly adaptable (pros of section 2.7) to almost any use-case. However Ontomind, with its underlying ontology, is able to capture the information according to the ontology only to the extent of what is the ontology capable of capturing. Thus, Ontomind only allows users to make assertions, which means stating that this information belongs to this concept or relation. Like ontologies (cons of section 3.9) Ontomind also struggles when new concepts or relations needs to be introduced or previously unknown knowledge needs to be captured according to the ontology. This process of assessing new terms into the ontology is complex and time consuming as discussed in section 3.8. and should be properly separated. Furthermore, Ontomind only deals with the class assertions and object property assertions within the mind maps. When the node of the mind map is labeled with some text and the class is assigned through the *AutocompleteTree* component, Ontomind has just made an assertion that the individual with the label is of class which has been assigned. Same goes for object properties, but instead of individual, Ontomind knows that this node represents object property assertion. Therefore, Ontomind does not know how to handle the rest of ontology basic building blocks mentioned in section 3.3 like data properties and data types. Of course that Ontomind is able to capture the information which is not described by the onotlogy in advance, but it doesn't address these nodes as important as the ones with assigned concepts or relations.

Mind maps are also used for making decisions, which was discussed in section 2.3.1. The use of colors and also a slice & dice approach helps to distinguish different aspects of the decision or point out the importance of the information. Furthermore, use of visual means like images, icons, tags and different fonts aids the overall interpretation of the mind map. However, when Ontomind and its ontology driven mind mapping is used in such scenario, the use of colors or importance of concepts is somehow not important. Ontology describes only the structure and hierarchy of the mind map and not other visual features. The information is just captured according to the ontology, there is no level of importance (from visual point of view), no visual stimuli which can stand out apart from other information captured.

As presented in section 2.6 mind maps are usually easy to create. Choosing a central topic, making branches with associations, capturing anything that comes to mind. The process is fairly quick and straightforward. This is also one of the key benefits of using mind maps when taking notes, or just capturing information. This quick and easy process is not the case when the information needs to be captured according to the ontology. Ontologies clearly states how the concepts are associated with one another through relations and thus the structure of such captured data is known even before it is captured (cons of section 3.9). Such process takes more time when creating such ontology driven mind map and many times it results into creating much more nodes of the mind map because everything has to be clear and brain associations between mind map nodes are just left out. Therefore, Ontominds mind maps are more verbose than classic mind maps.

However this standardized process of mind map creation and use of a common vocabulary helps with mind map reuseability. According to the pros in section 3.8 the highly normative description of the specific domain described by the ontology helps to understand the information better, because the vocabulary and the meaning behind each concept or word used in the mind map is specifically described by the ontology. It eliminates the ambiguity of what is meant by each word or phrase used. It also helps to capture information about similar topics in a standardized and also similar way and thus allowing readers of the mind map who are familiar with the ontology to understand it better. Standardization of information capture is good for the data interpretation. When there is a meaningful concept behind each word or phrase used in a mind map, it actually means that the data has some informational value and it is not just an idea or word from top of someones head.

The figures (Fig. 4.2, Fig. 4.3) addresses the problem of verbosity and advantage of added informational value of underlying ontology when capturing the information about a car.

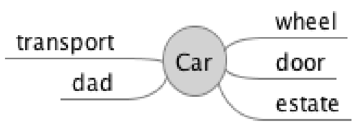

Figure 4.2: Classic mind map with some information captured about a specific topic.

The mind map depicted in figure (Fig. 4.2) is definitely less verbose than a mind map in figure (Fig. 4.3). There is a noticeable difference in the number of nodes used in the actual mind maps rendering the ontology enhanced mind map much more verbose and thus taking longer to create.

On the other hand when there is an ontology which describes the terms used in the mind map, such ontology enhanced mind map cannot be misinterpreted. The relations between concepts clearly states what the captured information means. Ontology enhanced mind map

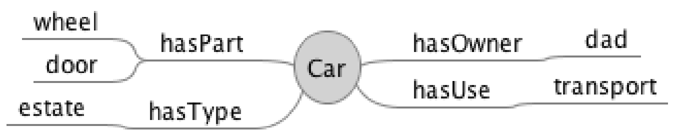

Figure 4.3: Ontology enhanced mind map with some information captured about a specific topic with underlying ontology which defines the concepts and relations.

says that the *car* has some *parts* like *wheel* and *door*. Furthermore it is clear that the *car* is actually an *estate car* and so on. Of course that some things may be magically understood by the reader of the classical mind map, but there is no clarification if the *car* is an *estate car*, or is just parked at the *estate* or if it is *car* given to someone from *dad* or it actually is a *dad's car*.

Ontomind is great when creating mind maps which are highly reuseable because of the underlying ontology. Application uses a component called *AutocompleteTree* which is handy companion when creating mind maps according to the ontology concepts and relations. It serves as a central proxy point of assigning a concept or a relation behind an actual mind map node. The *AutocompleteTree* component is very clever, because it reads information from the parent node and thus it provides users only with concepts or relations allowed for a specific node according to the underlying ontology. The *AutocompleteTree* component goes even further by allowing users to filter the allowed concepts and relations by typing in a name of the concept or relation. The figure (Fig. 4.4) shows the *AutocompleteTree* component in action. The assignment of the actual concept or relation is done just by clicking on a chosen label.

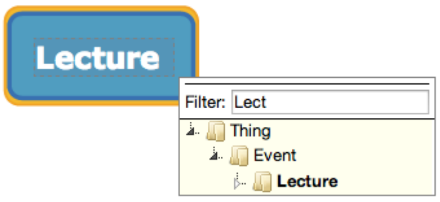

Figure 4.4: Ontominds *AutocompleteTree* component with filtered concepts.

Another advantage of Ontomind is the process of verification of each node of the mind map if it corresponds to some concept or relation of the underlying ontology or not. This is handy, when identifying the nodes which have not been assigned to some concepts or relations, thus giving a quick way to find and correct the mind map errors pointed out by the underlying ontology. Each node which is not corresponding to some concept or relation is marked by a gentle exclamation mark icon, as depicted on (Fig. 4.5).

According to section 3.3, in ontology the concepts can be connected only by the relationships and only the valid relationships. Ontomind marks every node which represents an ontological relationship with a tiny chain icon next to the mind map node. This helps to immediate identification of what node represents relationship and what node represents concept shown on (Fig. 4.5).

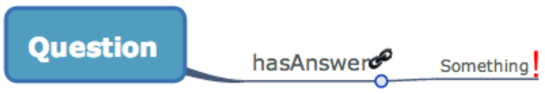

Figure 4.5: Marker icons for relationship and node with unassigned concept.

However, from a different perspective, such marking of each node might become disturbing, when users are just trying to quickly capture some information and don't want to spent too much time on assigning concepts or relations to every mind map node. It is possible to capture the information in such way, but some might say that Ontomind is too ontological because it will complain about each node without assigned concept or relation. So the creation of non-ontological mind map nodes in Ontomind is not that easy and user-friendly.

4.5 Ontominds pros

The issue of ontology driven mind mapping was discussed in previous chapter. By mixing mind maps and ontologies together, Ontomind has made some improvements of the information capture process. Here is the summary of pros from the analysis of Ontomind use-cases, what has Ontomind done right by introducing ontology into mind mapping.

Clarification - When mind maps are created according to the underlying ontology and every node has some concept or relation assigned, the added information helps the clarification of what information has actually been captured. This helps the reuseability of such mind maps.

Vocabulary - The use of common vocabulary prescribed by the ontology helps the reuseability even further.

Standardization - When capturing the information according to the ontology, users will develop a standardized way of capturing similar pieces of information.

Guidance - The *AutocompleteTree* is great feature of Ontomind when it comes to assigning a concept or a relation to a mind map node.

Verification - The verification and marking of the nodes with unassigned concept or relation helps the identification of the purpose of each mind map node.

Most of these pros of Ontomind are aimed at raising the reuseability of the mind maps created, which is a great thing, because the reuseability is the weakest point of them. Other pros are aimed at the guidance of users and user friendliness when it comes to reading and creating such ontology driven mind maps.

4.6 Ontominds cons

There are however some cons to the ontology driven mind mapping with Ontomind. Ontomind has introduced some restrictions to mind map creation process and also some annoyances when it comes to user friendliness. Here is the summary of the cons of the Ontomind from the analyzed use-cases and what has Ontomind done wrong by introducing ontology into mind mapping.

Associations - The biggest disadvantage of using ontology driven mind maps in Ontomind is the association. Mind maps created in such way are fairly restricted when it comes to brainstorming because the underlying ontology dictates what information can be captured. Although Ontomind is able to capture the previously not defined knowledge, these nodes are not treated with the same importance as the ones with the assigned concepts or relations.

Verbosity - Ontomind is definitely much more verbose, because the structure of underlying ontology dictates also the structure of the mind map.

Time consumption - It takes much more time to capture the information according to the ontology. Finding the right concepts or relations and assigning them is time consuming. Also the information needs to be captured in a standardized fashion.

Restructurization - Ontology driven mind maps are limited in case of making modifications of the mind map structure to the point of what is allowed according to the ontology. Furthermore Ontomind does not allow any movement and restructuring of branches and nodes of the mind map.

Appearance - Ontomind does not use any predefined colors for different concepts or relations and the use of images, icons and different fonts is not automated. Users have to do the styling manually like in a normal mind maps.

Ontological - Ontomind is too ontological about the mind maps created. The mind map with tens of nodes with exclamation marks next to each node might look frustrating.

Only assertional - Ontomind is capable of only making assertions, thus the capture of information previously unknown and not defined by the ontology is limited.

Not complete - Ontomind is capable of handling only class assertions and object property assertions within the mind maps. Data properties and data types functionality is simply missing.

The introduction of ontology into mind mapping process fairly crippled one of the key advantages of the mind maps which is the freedom of idea creation and capture. Furthermore, such ontology driven mind maps take more time to create because there is a need of standardized approach and it usually introduces the creation of more nodes. Such mind maps don't focus on the visual aspects of the mind map, because the information is captured in a prescribed form.

4.7 Ontomind summary

In this chapter the Ontomind application was described from few different points of view. First the actual concept of what the application should be able to do. Then the architecture of Ontomind application from the technological point of view was presented. In the last part of this chapter, the issue of ontology driven mind mapping from a practical point of view was analyzed. In the end the summary of the pros and cons of the Ontomind application were described and pointed out.

Chapter 5

Proposed improvements & Methodology

Chapters 2 and 3 were devoted to discussion of mind maps and ontologies. In each of these chapters the pros and cons of corresponding technologies were summarized in sections 2.7 and 3.9. Then in chapter 4 an ontology driven mind mapping tool was briefly analyzed for the utility of knowledge management in different use-cases where previously mentioned technologies excels. The analysis of Ontominds pros and cons was presented according to what has Ontomind done right and wrong (or left out) when mixing those two approaches together.

Based on the knowledge gained in the previous chapters, this chapter discuss the methodology of creating mind maps within the Ontomind with help of the ontologies. The focus is on the improvements which can help with information management when creating typical mind maps and ontology driven mind maps within Ontomind.

5.1 Guidelines revisited

The methodology for creating the mind maps within Ontomind make use of the guidelines for creating mind maps from section 2.8. Users of Ontomind should also create their mind maps following these guidelines.

The rule of just one topic is not problematic and it can easily be done also in Ontomind. Users just have to stick to create a mind map for a specific purpose and if there is a need to capture information about something else, they should create another map. This rule also supports the other rule of being concise, which is yet another key thing to look out for when creating a mind map which will be easy to read. In the following sections of this chapter, there are improvements of Ontomind proposed to support the problem of concision and managing bigger amounts of information in a mind map.

However, the rules about using visual means and creating own personal techniques should be carefully thought through. Because of the aim of the mind map to be more reuseable, the visual means and specific techniques of capturing information should be used in a standardized way. If a group of users wants to use specific icon or a color for a specific purpose, the new users invited to this group should do it the same way. In the following sections the improvements of Ontomind are proposed to support such standardized way of using visual means when creating a mind map. The standardized way of capturing information is already present in Ontomind because of the use of the underlying ontology, but the whole process can be improved, which is also discussed in the upcoming sections.

5.2 Two modes of Ontomind

As mentioned in section 4.6, Ontomind is too ontological when it comes to the scenario, when there is not enough time to create a mind map according to the ontology. The other scenario might be if users just want to create a more personal mind map, which doesn't have to be made according to the ontology and will be used just for private purposes. Ontomind doesn't differentiate between shareable mind map and private mind map. The only option is shareable. Otherwise, Ontomind is always informing the users about the inconsistency between the assigned concepts or relations with mind map nodes. Furthermore the *AutocompleteTree* component is always popping out and requiring the users to assign concepts or relations to nodes.

Ontomind can be improved in a way that it would allow users to create their mind maps quick and easy as normal mind maps were primarily intended to be created. The ability to enable and disable the ontological functionality of Ontomind whenever users wants to divides the knowledge capture process into two optional phases. First phase is a non-formal quick and easy way of capturing the knowledge which renders the mind map more personal as any other classic mind map. Then the second phase is aimed at formalization of the previously captured data, restructuring of the knowledge according to the ontology and making it more shareable.

In a non-ontological mode the speed in which such mind map can be created is much higher. Furthermore the nodes of the mind maps can be moved around because there is no verification to the structure of the knowledge captured in this mode. There is no need of assignment of the concepts or relations to each node of the mind map. In addition, the mind map is able to capture any knowledge, because it doesn't use any ontological data to verify if the concept or relation was previously defined or not.

In ontological mode however, Ontomind behaves as usual and all of it's added functionality of underlying ontology is present.

With such improvement there is a mode, in which no cons that Ontomind introduced into classic mind map creation, are introduced by the addition of the underlying ontology.

5.3 Automatic colors

One of the key benefits of mind maps is the usage of colors as it was stated in pros of the section 2.7. Also, according to guidelines in section 2.8 the use of colors is recommended and very welcome. Colors help with the associations, distinction of branches, identifying important information, capturing similar information and many more situations.

Ontology driven mind maps created in Ontomind should also use color to their advantage. While Ontomind can use colors, the whole process of changing the color of a mind map node is rather unfortunate. This process is not easy as it was meant to be in the first place. Furthermore the time consumption and the steps which have to be taken into consideration may irritate users of Ontomind over time and discourage them from using the colors which is a great benefit of any mind map.

Ontomind can be improved to use colors in an automated way and the users are spared of this manual process. With the help of ontology, specific concepts or relations can have a unique color assigned through the annotation properties of another ontology. This helps their identification in the mind map and overall readability of the mind map. Such improvement also helps shareability of the mind map, because the assignment of unique colors to specific concepts or relations is done in one place and every user is using the unique color the same standardized way.

With such improvement of automated color assignment , Ontomind makes use of colors within the mind maps, helps the readability, shareability, and furthermore, makes the process quick and easy.

5.4 Automatic icons

The same guideline about icons mentioned in section 2.8 applies for ontology driven mind maps created within Ontomind, but the problem of icons in Ontomind is somehow similar to the problem of colors. Ontomind does have the ability to assign icons, but the whole process of such icon assignment is cumbersome and time consuming as it was mentioned in section 5.3. Additionally, these icons are not assigned in the standardized way, so they can have a special meaning only to the users which assigned them, which is not helping the shareability of such mind maps.

With the help of underlying ontology, Ontomind can be improved to support automated icon creation according to some rules. These rules may be really simple or complex.

An example of simple rule is an assignment of a specific icon to a particular node. With such improvement, these particular nodes may be identified within the mind map immediately, which helps readability and navigation within the mind map when looking for a specific piece of information.

A more complex rule might be assigning an icon with special meaning to individuals based on its associations. As it was mentioned in section 3.8, with the underlying ontology, the structure of the captured information is known in advance. With this feature Ontomind can verify, if for example a specific individual has all of its properties set and assigned to their instances, and mark the node of the mind map with "checkmark" icon to state that the information captured is complete and nothing else is missing.

Another example could be differentiating two mind map nodes representing instances based on whether the specific associations are made. When an instance of *Task* concept has a *DueDate* associated with it, it can be marked with different icon than other instances which don't have *DueDate* associated. This way, the user of Ontomind immediately recognizes the instances of *Tasks* which have their *DueDates* associated.

These rules may be assigned in a similar way as it was mentioned in previous section concerned with colors. Through the annotation properties of another ontology. This way it has a central place where such behavior can be configured and thus every user is using this feature the same way which also boost shareability of the mind map.

5.5 Verbosity

As it was presented in section 4.6, one of Ontominds cons is verbosity. According to section 3.8 when information is captured according to ontology, everything has its predefined structure and it usually involves creation of additional nodes because such mind maps does not use associations which is yet another key advantage of mind maps (pros of section 2.7). According to the guidelines in section 2.8 mind maps should be simple and kept small to really exploit the advantage of understanding and outlining the big picture of the information captured. That is why another proposed improvement of Ontomind is concerned with making the mind map more concise.

There are situations where the structure of the ontology driven mind map is redundant in number of nodes and the actual information captured cannot be interpreted ambiguously.

Figure (Fig. 5.1) depicts such example excerpt from a mind map, where exam has some questions and these questions have answers. The presented mind map is really verbose and it contains only information about three questions. In case if there were 50 questions the mind map can take too much space and with so much nodes it can have confusing effect on users.

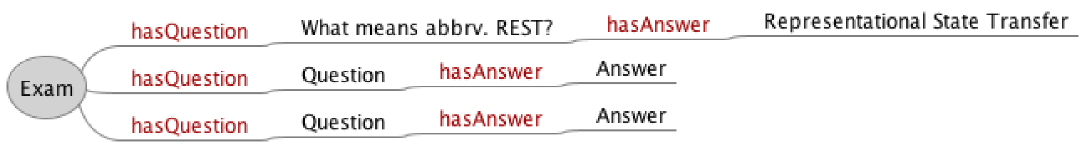

Figure 5.1: Verbose ontology driven mind map.

The structure of the captured information is defined by the underlying ontology and in this particular case the range of *hasQuestion* relationship has only one valid class which is *Question*. The same principle applies to *hasAnswer* and *Answer*. In such situation, the information captured cannot be interpreted ambiguously and Ontomind can help users with automatic concision of the mind map. With combination of another improvements mentioned in sections 5.3 and 5.4 Ontomind can mark appropriate nodes with specific icons or colors and hiding unnecessary nodes which are not needed for the clear interpretation of the information captured.

Figure (Fig. 5.2) depicts the same situation as figure (Fig. 5.1), but in the more concise form with additional icons (randomly chosen for the purpose of demonstration) automatically assigned by Ontomind. Such mind map is easier to read and it is not much more verbose than the actual information which was needed to be captured.

Such improvement can make mind map, more readable, concise and the reuseability doesn't suffer from the concision. However the situations to which such improvement can

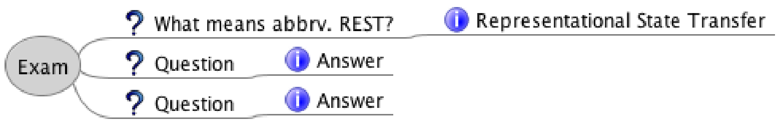

Figure 5.2: Concise ontology driven mind map with additional icons.

be applied have to be specifically stated because with improper specification such mind map may suffer from ambiguity in the meaning.

5.6 Filtering

There are plenty of situations when there is a lot of information captured within the mind map and users struggle to find the particular information within it. Classic mind maps are not good at managing big amounts of information as presented in section 2.6.

Even though Ontomind uses ontology to keep the information structured and organized in a standardized form, it also suffers from problem of concision and thus hard identification of particular information. Ontomind even may add the problem of verbosity (see section 4.6).

However, Ontomind can be improved to help users to find the specific information. With the power of the underlying ontology the structure of the information captured is known in advance (see sections 3.8 and 4.5) and thus it can be filtered according to the underlying ontology.

The idea behind such improvement is similar to what databases or spreadsheet processors use. In these programs the information is entered in a specific column of a specific table. This added information adds context which is allowing these programs to filter the results according to this context and the resulting information is as specific as it needs to be.

With the use of underlying ontology by Ontomind, the added information in form of assigned concepts and relations can be read from the mind map and identified nodes of the mind map can be filtered out in a similar fashion as databases or spreadsheet processors do. The assignment of the concept or a relation to a mind map node is similar to entering the value into a particular column in the database table. For example - putting a text *Dominik* into a column named *firstname* of the table named *Person* is the same thing as assigning an ontological object property *hasFirstName* and an actual string value to some individual which is of class *Person*.

With such structural scheme of data captured in the mind map and ontology, Ontomind is able to filter the information captured according to classes, object properties, data values, restrictions and every added information provided by the ontology. If the ontology extends in number of classes, then Ontomind will be able to filtrate the information according to new added classes. The same goes with object properties, restrictions, data properties and every aspect or feature that ontologies provide. Such filtration ability boosts the readability of mind maps which store big amount of data. Furthermore the filtration helps with the verbosity and reuseability of the mind map.

A nice example (but maybe repeated, see section 5.4) would be if a student wants to see only *Tasks* which are associated with *DueDate*. All other mind map nodes can be filtered out (made invisible) and a student can focus on the important tasks in front of him.

5.7 Autocomplete

Capturing the information within the mind map according to the underlying ontology in Ontomind is time consuming as it was summarized in cons in section 4.6. By addition of the ontological data in Ontomind, one of the advantages of classic mind maps - easy and quick way to create mind maps (see section 2.7) was compromised. Although Ontomind is trying to ease the time consumption by its *AutocompleteTree* component, it still suffers from longer creation times.

The Ontominds *AutocompleteTree* component can be improved even more and further improve the time it takes to create nodes and assign concepts to them.

The structure and thorough definition of concepts and relations described by the ontology is an advantage as summarized in pros of section 3.9. The ability of ontologies to declare the domain and range of a particular object property is a key thing behind the improvement of the *AutocompleteTree*. Ontomind can analyze the range of the object property defined by the ontology and take actions if the answer is clear. The best way to explain such feature is through an example.

In the ontology developed for the purpose of demonstration in chapter 7 from students life there is a class named *Course* which represents concept of a particular course the student attends and wish to take some notes on. There is also an object property called *hasEvent* through which a relation between *Course* and *Event* class can be set. The *hasEvent* object property has its domain and range defined. The range of this object property is *Event*. The problem here is that *Event* class has a few subclasses - *LectureEvent*, *ExcersiseEvent*, etc. In ontology, when the range of some object property is a class which has some subclasses, then all of its subclasses can be assigned to that particular range. The hierarchy described is depicted on the figure (Fig. 5.3). This is the case where the Ontomind can't take actions and make a decision for the user, because more than one class can be assigned to a particular node and the final structure of the mind map is unclear.

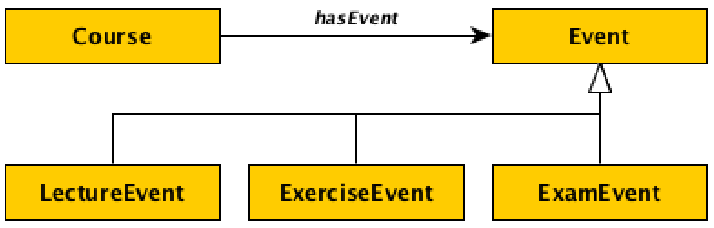

Figure 5.3: Hierarchy of *Event* classes and relation to the *Course* class.

However in the same ontology there is an object property *hasTopic* of class *Event*, which also has its domain and range declared. The range of *hasTopic* object property is different though. The range is specified to be a class of *Topic* and class *Topic* has no known subclasses. Hierarchy described is depicted on figure (Fig. 5.4). Unless there is a way of adding new classes to the range of this object property, there exists only one option which can be assigned as a class to a particular child node of this object property. This is the case where Ontomind can make a decision and take some actions. Because there is no other class to be assigned to particular node than just one, after creating a node with the relation of *hasTopic*, the Ontomind creates a child node with already assigned class of *Topic* for the user automatically.

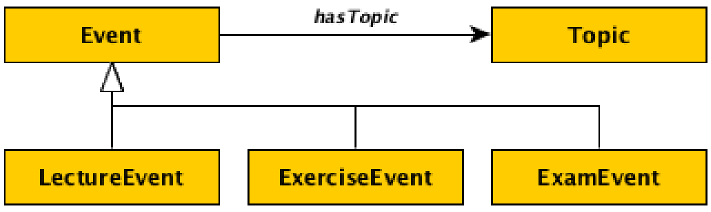

Figure 5.4: Hierarchy of *Topic* classes and relation to the *Event* class.

Such improvement is applicable not in just the case of *hasTopic* object property but in all cases where the range of the object property consists of just one class. With this feature mind maps created according to the ontology in Ontomind are done faster and users don't have to do repetitive tasks over and over which helps boosting user-friendliness.

5.8 Terminology

As summarized in section 3.9 one of the pros of an ontology is terminology. Ontomind brought this terminology to mind maps with its ontology driven mind mapping approach which is yet another pro of Ontomind as analyzed in pros of section 4.5. With a comprehensive set of terms, precisely described by the ontology, users of Ontomind application don't have to think of their own words when trying to capture some specific domain knowledge. This terminology conformity is great for the reuseability of the whole mind map. While this is a good thing, there are situations, when a new term should be created, described and added to the ontology. However Ontomind is lacking the ability to easily do such things as summarized in section 4.6. Ontomind is only capable of making assertions[1].

Mind maps are great when capturing new information. To make Ontomind more new information friendly another possible improvement of Ontomind is an ability to extend the terminology of the underlying ontology together with management of some small subset of

[1]In the past however, Ontomind was capable of adding new concepts into the ontology, but the addition process was very simple and it didn't allow all the possibilities of configuration. Just a simple addition of concept into a hierarchy. Furthermore, the new created terms were only visible in personal scope and they needed to be approved through process of standardization to be accepted into the shared ontology. Because of this complex process of standardization, Ontomind was prohibited the addition of new terms for now and a completely new application was developed for the purpose of extending Ontominds underlying terminology.

ontology features. For example the creation of new concepts and management of annotations would be enough. There is no need to make Ontomind a full featured Ontology editor, there are already many good ones.

With such feature at hand, Ontomind is much more powerful in terms of what can be achieved and what data can be captured. While Ontomind users already have the ability to capture any information in the mind map, the problem was that it didn't committed to the underlying ontology. With the ability to extend the underlying ontology, it is a matter of creating the needed terminology. Then Ontomind is able to make use of its assertion functionality and assign the new created concepts or relations to the actual information. The new knowledge is captured according to the extended ontology and all other features of Ontomind might use this additional information for their own needs.

The impact on reuseability in many use-cases greatly improves with such feature of extending Ontomind to be also terminological, because mind maps are able to capture anything with the right extension of the specific domain. Furthermore, with many extensions of the underlying ontology, the domain which was originally described may not just be extended but changed completely.

5.9 Data verification

Although not really connected to the mind mapping techniques, data verification might come handy in many situations. As it was stated in section 4.6 Ontomind is not able to handle data types and data properties of the underlying ontology. These are however another key building blocks of ontologies as it was described in 3.3. With Ontominds support of these features, Ontomind is able to do some data verification automatically, and guide the user further if the information captured is correct.

Data properties are similar to object properties, but the difference is that the relation is made rather between a class and a data type and not between two classes. Therefore, ontology data properties have a list of primitive value data types that can be assigned as a range of the data property.

The need of verification of actual data values comes from many use-cases. For example, what is the purpose of stating that some data represent the price of some product if actual data captured is something completely different and unrelated like for example a simple string "tomato".

With the improvement of Ontomind to provide means of data properties and stating that some values don't represent some concepts or relationship, but that they are rather ordinary literals, strings, numbers, abilities of data verification and actual guidance of the users greatly improves. By knowing the data type of the actual primitive value, Ontomind can automatically verify if the data entered is correct, otherwise take some actions.

For example if the data property *hasPrice* has range of floating point number, the verification if the actual value entered is in form of floating point number is undergone. If the data entered is correct, everything is fine, but if the data is incorrect, Ontomind can inform the user that the value entered is malformed and should be corrected.

With the ability to state annotations on classes and object properties, the idea of data verification could be pushed to extremes. With the use of annotations the verification can go as far as regular expressions can go. For example, the data property *hasEmail* has a range of string, which is pretty simple and straightforward. But, through the annotation, the configuration of how the string should be verified according to the regular expression could be stated. Therefore, with the right regular expression pattern Ontomind would be able to verify, if the data entered is in the form of email.

However, while the ontologies posses the features of data properties, Ontomind application lacks in the ability to represent them. So to enhance Ontomind of such features, the representation of data properties should be implemented first[2],

5.10 Improvements & methodology summary

In this chapter, the methodology of creating mind maps was revisited. The focus was on proposed improvements of Ontomind application with examples in their usage. The theoretical functionality of the improvements was drafted and the impact of the everyday usage of the improvements was discussed. All of the proposed improvements have a significant impact on mind map creation and manipulation process, readability and reuseability of the mind map. Improvements were aimed at bringing back what was good about classic mind maps and improving further what has Ontomind done right with the ontology driven mind mapping approach. More important thing to stress out is a fact that most of these improvements are only possible in such ontology driven approach and environment.

[2]There may exist some exploits and workarounds through the annotations and special classes representing primitive data types which will not be discussed.

Chapter 6

Implementation

This chapter is devoted to the implementation of some of the improvements proposed in chapter 5. In the first part of this chapter, Ontominds setup process is discussed. Then in the second part, some of the implementation specifics of particular chosen improvements are described.

Since Ontomind is already developed application running in production environment, there is no need to decide which technologies will be used for the actual implementation of the improvements. By sticking to the technologies already used and not introducing new ones, the whole deployment of the improvements to production will be much easier.

6.1 Ontominds setup

Throughout the long life of the MONDIS project Ontomind as an application went through many phases which affected how the application works and thus affecting the process of setting up the original Ontomind application on a local machine.

First it was primarily a web application accessed like a normal web page. It was secured by the classic web security components of used frameworks. It had functionality which included the ability to extend the underlying ontology with own vocabulary.

Then, Ontomind was changed to support portal technology which resulted into elimination of Ontominds security system. The ability of extending the underlying ontology and having multiple vocabularies was prohibited and a completely new portal application for such terminology extension was developed.

Nowadays, the *AutocompleteTree* component of Ontomind is undergoing the process of migration to a mobile environment, so Ontomind can be also used on mobile and tablet devices.

All of these decisions and life phases affected the codebase of the original application by a great margin and made some portion of the codebase non-functioning. Following the original read me file for setting up the application, many problems were encountered which had to be fixed during the setup process. These problems included:

- Fixing dependency management when packaging the application for deployment,

- fixing logging problems when built in users are used when application is not run inside a portal,
- fixing communication problems with triple store database over *HTTP* protocol when mind map contained assigned concepts,
- fixing the behavior of how the *AutocompleteTree* guiding component filters and expands nodes representing ontological concepts.

These problems resulted into many hours dedicated to debugging and fixing the application in an early implementation phases for the purpose of implementing the proposed improvements of this thesis. In the end, bugs were fixed and the application was able to run on the local machine, which enabled the further implementation and testing.

6.2 Choosing improvements

During the process of choosing which of the improvements will be implemented, following criteria were taken into consideration:

Fundamentals - The focus was on not changing the already fundamental functionality of Ontomind. Some of the improvements proposed in previous chapter like data verification in section 5.9 may require such changes and implementing the handling of data properties and data types by Ontomind in advance.

Codebase - The focus was on the functional code base of Ontomind and technologies used to successfully implement such feature. Ontomind has a huge codebase and uses a lot of technologies and this should be taken into consideration.

Time - This criterion is closely related to two previous criteria, because fundamental changes of the Ontominds functionality or Ontominds codebase are time consuming and such changes are simply out of the scope of this diploma thesis.

After the consideration of the mentioned criteria, the following improvements were chosen:

- Ontological mode
- Improved autocomplete
- Automatic icons

Some of the technologies Ontomind uses and how they are connected together were already described in sections 4.2, 4.2.1 and 4.2.2. In the following sections the implementation of chosen improvements is described together with the little deeper look on some of the mentioned technologies used by Ontomind.

6.3 Ontological mode

Ontomind restricts some of the functionality of manipulation with mind maps, which *wisemapping* supports. Furthermore it introduces the *AutocompleteTree* component and verification of mind map nodes if they are assigned with concepts or relations. However, these features shouldn't be present in non-ontological mode.

Most of the restrictions done to the mind mapping techniques are considering *mindplot* which is one of the core projects *wisemapping* uses as was presented in section 4.2.1. These restrictions are made without touching the code base of the original *mindplot* project, but rather with refactoring of the original codebase. *Mindplot* is written in *Javascript* and the refactoring of its classes is done via another *Javascript* framework called *mootools*. *Mootools* is a *Javascript* framework which provides means for easy implementation of new functions of *Javascript* objects as well as refactoring already defined functions and many more. For further reading please refer to [20].

To disable these features a mechanism of reverting back to original codebase of *mindplot* was implemented. This way all of the restrictions that Ontomind introduced to *mindplots* functionality are gone. But, the users of Ontomind should be able to activate and deactivate such mode at any time at will. Therefore, refactoring of the *mindplots* functionality back to Ontominds version should be done again.

This feature lead to implementing a mechanism of enabling and disabling the ontological mode and thus refactoring of the *mindplots* functionality back and forth to original *mindplots* and Ontominds version.

However, disabling and enabling the functionality of the *mindplot* is not enough. In non-ontological mode also the *AutocompleteTree* component together with the Onotminds verification should be disabled.

Therefore, the mechanism of enabling and disabling ontological mode was extended to enable and disable also the features of *AutocompleteTree* and node verification.

A short summary of what need to be done is described:

- Implementation of visual means for activation and deactivation into Ontominds mind map editor,
- implementation of the mechanism for toggling ontological and non-ontological mode into Ontominds mind map editor,
- implementation of the distinction between ontological and non-ontological mode and different behavior into *AutocompleteTree* component and verification component of Ontomind,
- analysis and implementation of refactoring back and forth to original *mindplots* mind mapping functionality and Ontominds restricted mind mapping functionality,
- customizing the saving mechanism of Ontomind mind maps.

6.4 Improved autocomplete

The ordinary *AutocompleteTree* works in a way that it checks the class of parent mind map node and then with the use of *AJAX* technology it asks the backend of the Ontomind application via *REST API* for appropriate concepts or relations. The resulting options for appropriate node are shown in a tree structure with use of *jsTree Javascript* plugin. Then, after choosing an option from the tree an appropriate concept or relation is assigned to the mind map node.

The *ImprovedAutocompleteTree* should work in a way, that after assigning a relations, the range of this relation should be examined and if there is only one option that could be assigned, Ontomind will automatically create node for the user and assign the appropriate concept to the mind map node.

There are two ways how can this be accomplished.

First one is by adding the information about the range of the relation into the serialized form of the mind map. Then analyzing this range from the serialized form and taking actions. The disadvantage of this approach is that it is not usable on already created mind maps, which don't have this information serialized. Another disadvantage is that the serialized form of the mind map is bigger in data and thus with mind map becoming bigger and bigger it is harder to transfer over the internet with every update.

The second and a chosen approach is via creating a hidden child node after assigning a relation to mind map node. Then making another *AJAX* call for the appropriate concepts for the actually hidden mind map node. If the result of the *AJAX* call is only one concept, then the concept is assigned to the mind map node and the node is made visible. If the result is more than one concept, the answer is unclear and the hidden mind map node is deleted. This approach is good for reuseability, because it can be used even on older mind maps when the *ImprovedAutocompleteTree* functionality was not present. Mind map is not cluttered with the information about ranges, which may be extensive when really generic relations with many concepts in their range are assigned. Although it may look that there is twice the communication, that's not true. It only affects the mind map nodes representing relations and the results are stored in cache and server-side session, so the repeating request are done much quicker and don't always need to connect to the database.

As the ontological mode improvement described in section 6.3 also the *ImprovedAutocompleteTree* needs to be enabled and disabled at any time and at users will. Therefore, the mechanism of enabling and disabling this feature was also implemented.

A short summary what needed to be done is described:

- Implementation of visual means for activation and deactivation of *ImprovedAutocompleteTree* into Ontominds mind map editor,
- implementation of the mechanism for toggling *ImprovedAutocompleteTree* mode into Ontominds mind map editor,
- implementation of making another modified *AJAX* call for examining the range of the relation mind map node,

- implementation of creating hidden node and making it visible or deleting it after analyzing the result of the additional *AJAX* call,
- customizing the saving mechanism of Ontomind mind maps.

6.5 Automatic icons

As proposed in section 5.4 the improvement of Ontomind to automatically assign icons to mind map nodes according to the external configuration is implemented.

Ontomind uses the *triple store* not just for the underlying ontology, but also for each mind map. Asserted ontological data through Ontomind then have its equivalent in a form of *RDF* graph. Based on the information stored in this *RDF* graph, Ontomind can retrieve information about particular mind map node and check what concept or relation has been associated with it. Ontomind will use *SPARQL* for querying the *RDF* graph.

The *SPARQL* queries and appropriate icon filenames are stored in a separate ontology used for a specific purpose of configuring such feature which imports the underlying ontology to refer to the concepts and relations. By applying such mechanism, underlying ontology stays clean of icons configuration and configuration itself is done in one place.

This *SPARQL* query is configured through appropriate annotation property called *"query"*[1]. This way, a specific *SPARQL* query can be assigned to any concept or relation of the underlying ontology.

With such approach, Ontomind can mark mind map nodes with icons based on the *SPARQL* query itself. This means that if the query is simple

```
?subject  rdf:type  :Topic .
```

it will return every mind map node which have been associated with a concept of *Topic*. Then these identified mind map nodes can have appropriate icon appended. But if the query was more complex

```
?subject  rdf:type  :Person .
?subject  :hasEmail  ?object1 .
?subject  :hasTelephone  ?obect2 .
```

it will return only the mind map nodes, which have been associated with the concept of *Person* and additionally have been associated with other concepts through relations of *hasEmail* and *hasTelephone*. Combining such query for example with an icon of "checkmark" Ontomind can automatically mark such mind map nodes to inform a user, that for example all the usual information about a person has been already captured.

Icons, as queries are also configured through annotation property, but the name of the annotation property is *"icon"*[2] in this case. Also same as with queries, by applying such

[1]The *IRI* of this *query* annotation property can be configured through configuration file so it can refer to any annotation ontology.

[2]The *IRI* of this *icon* annotation property can be configured through configuration file so it can refer to any annotation ontology.

mechanism the specific icon can be assigned to any concept or relation of the underlying ontology.

The only requirement is that the *icon* and the *query* annotation properties comes together when assigned to a concept or a relationship. For example, a concept of *Topic* should have both the *icon* and the *query* annotation properties set in order to function properly.

The custom icons should be placed inside a folder *"icons/custom"* and it is recommended for them to have a square format and dimensions of 16 pixels.

The configuration of these icons is read from the underlying *triple store* only when the whole underlying ontology needs to be reloaded. On the other hand, the evaluation of the nodes which need to be appended with icons is done after every mind map node addition or removal, because the *SPARQL* queries might return different results.

Once again this improvement should be enabled and disabled at users will at any time. Therefore, also the mechanism of enabling and disabling this feature was also implemented.

A short summary of what needed to be implemented is described:

- Implementation of visual means for activation and deactivation of *automatic icons* into Ontominds mind map editor,
- implementation of the mechanism for toggling *automatic icons* mode into Ontominds mind map editor,
- implementation of reading the configuration of annotation ontology from properties file,
- implementation of reading the queries and icons configuration from *triple store*,
- implementation of querying the *RDF* graph of specific mind map with configured queries,
- implementation of evaluating each mind map node whether the icon needs to be appended to mind map node,

6.6 Activation of improvements

One of the required feature of every new implemented improvement is the activation and deactivation at any time at users will. To minimize the impact of the improvements on Ontominds original behavior, the improvement of *ontological-mode* is enabled by default when the new mind map is created. The other improvements need to be activated first to enhance the user experience.

The activation and deactivation is done through toggle buttons on Ontominds toolbar with the icons depicted on figure (Fig. 6.1). Enumerating from left to right the icons represent the improvements of *ontological/non-ontological mode* (icon with letter 'm'), *improved autocomplete* (icon with letter 'a') and *automatic icons* (icon with letter 'i').

Figure 6.1: Toggle buttons for activating and deactivating improvements.

6.7 Implementation summary

In this chapter the implementation of new features to Ontomind was shortly described. The focus was on used technologies and summary of what was needed to analyze and implement. The newly implemented features help with day to day usage of Ontomind which are trying to minimize Ontominds cons introduced to mind mapping techniques.

The information about what was implemented and how the codebase changed can be found in appendix D. The installation guide for prepackaged Ontomind application can also be found in appendix C.

Chapter 7

Demonstration and testing

This chapter is devoted to the demonstration and testing of the implemented improvements of the Ontomind application. In the first part of this chapter an ontology for the purposes of the demonstration is developed and described. In the second part the improvements implemented in chapter 6 are tested and demonstrated on the specific domain described by the mentioned ontology in an upfront designed use-cases and test scenarios.

7.1 Demonstration ontology

For the purpose of demonstration, a testing ontology needs to be created. The ontology is describing a specific domain from a students life when taking notes during the attendance on various courses throughout the semester.

There are various types of ontologies as it was presented in section 3.7. The appropriate one for the purpose of demonstration is students life domain ontology. A students specific domain ontology describes the concepts and relations which are representing various events during the semester, various objects or persons with which students can come in contact with or abstract information that needs to be captured during note taking process.

If such ontology is developed, and notes about various courses are taken according to this ontology, a standardized structure is introduced to the notes and such notes in form of a mind map are more reuseable. Furthermore the information captured according to the ontology can then be easily extracted from the mind maps and processed further by other applications.

The key thing to focus on is what this conceptualization adds to the information and which information in fact needs to be conceptualized and captured in a standardized way. The ontology which is presented in the following sections may not be the right one and the only one, but it serves for the purpose of testing and demonstration of the methodology and improvements.

7.1.1 Main concepts

Figure (Fig. 7.1) depicts the main and most notable concepts and relations in the demonstration ontology.

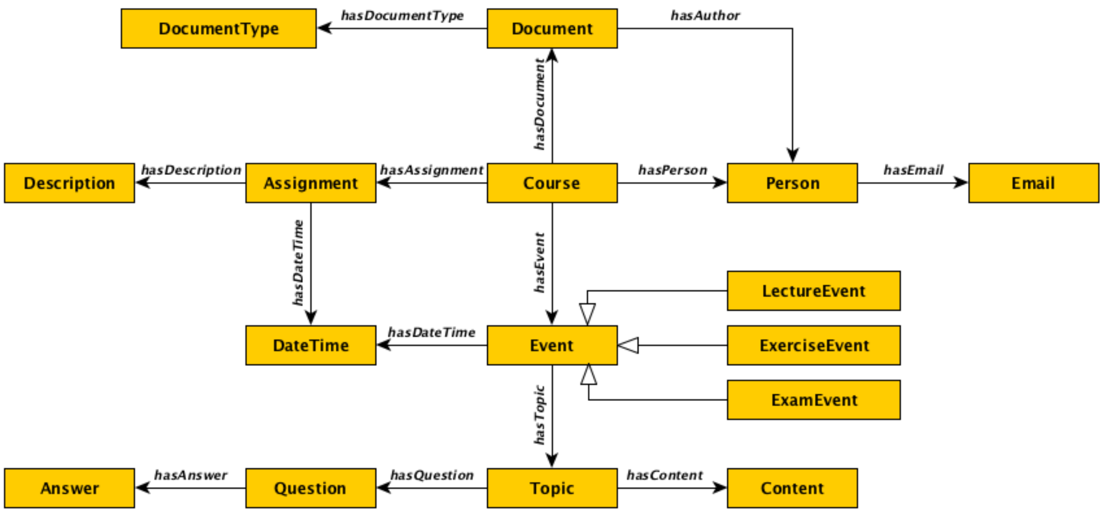

Figure 7.1: Hierarchy of concepts and relations between them in demonstration onotlogy.

Most of the concepts and relations are rather self-explanatory, but some of them need more explanation why they were chosen. The demo ontology is not used just for the demonstration of the improvements to the Ontomind, but also as the demonstration that capturing information in the mind map according to the ontology has some additional value.

If the right concepts or relations are assigned to the appropriate pieces of information, the needed information can be extracted from the mind map by third-party applications. With use of the *Person* and *Email* concepts an application can process students mind maps and extract a contact list based on this information. If a student wants to know something about a particular topic, with the help of *Topic* concept, his mind maps can be processed and the information extracted. Again with concepts of *Assignment* and *DateTime* a to-do list with due-dates can be created.

The concept of *Content* represents any piece of information which can not be categorized or specified further. Some might say that there should be more concepts introduced for titles, aside notes, etc. They might be right, but as it was stated in section 3.7.2 this conceptualization process is time consuming and needs a careful consideration on the granularity and generality of the concepts and the knowledge which needs to be captured by the domain ontology. For the purpose of the students needs it is enough.

Concepts *Question* and *Answer* are however not generalized into *Content* concept. It happens so many times, that teachers are pointing out that a particular information will be a question on a final exam, that the need of pointing out such information might come handy during the note taking process and consequent processing and reading the mind map.

7.2 Testing scenarios

This section is devoted to the demonstration as well as testing of the new implemented features of Ontomind. With the help of the ontology developed for the demonstration purposes in section 7.1 three test scenarios were developed to test these features.

7.2.1 Testing ontological mode

The new feature of enabling and disabling ontological mode in Ontomind was implemented in chapter 6. This feature tries to bring back one of the most important advantages of mind maps which is the ease of creation and manipulation with nodes.

In students life there are situations during lectures, when the flow of new information which needs to be captured is just too quick and there is no time left for the proper structuring of the information. This structuring can be done later. This is exactly the use-case for the new implemented feature of enabling and disabling ontological-mode. This scenario is set to exactly such situation when after few information captured in the mind map, student deactivates ontological mode and is able to capture information more quickly and restructure it freely.

The actions taken by the student and corresponding responses of the application are stated in table (Tab. 7.1).

The responses of the Ontomind application are clearly showing that the newly implemented feature of non-ontological mode is functional and that this mode can be activated and deactivated at users will. This testing scenario also demonstrated and tested the methodology and new implemented feature on a specific domain.

7.2.2 Testing improved autocomplete

ImprovedAutocompleteTree feature was also implemented in chapter 6. This feature however tries to ease the amount of actions which are needed to capture some information according to the ontology. This improvement helps reducing time consumption when creating such mind maps.

When a student tries to capture the notes during the lecture with the help of the demo ontology developed in section 7.1, he will definitely come across a situation where the range of the particular relationship is just one class. In fact most of the relationships in the demo ontology have range of just one class. Therefore this ontology is suitable for demonstration and testing purposes of this *ImprovedAutocompleteTree* feature.

The second testing scenario is divided into two parts. The first part of the scenario is with the feature of *ImprovedAutocompleteTree* not enabled. Also the steps 1-8 of this first part are left out from the table (Tab. 7.2) because they are already described in the first testing scenario and Ontomind responds are the same. The second part is with the use of the *ImprovedAutocompleteTree* feature enabled. Similar to the first testing scenario, the actions taken by the student and corresponding responses of the application are stated in table (Tab. 7.2).

Ontomind application responses in table (Tab. 7.2) show that the new implemented *ImprovedAutocompleteTree* feature is functional and that it can be activated and deactivated at any time. This testing scenario also demonstrated the use-case in which such feature can be used.

7.2.3 Testing automatic icons

Ontomind was improved with the feature of *automatic icons* in section 6.5. Such feature comes handy when specific concepts or relations in the mind map need to stand out to identify them immediately. Furthermore it helps with time consumption when creating such mind map and with the help of icons the mind map will look more stimulating to the brain.

For the testing purposes, some basic queries and one complex query for assigning an icon to a mind map node were prepared and inserted into annotation configuration ontology of Ontomind. A set of assigned icons together with the respected query descriptions is presented:

1. *Star* icon for a concept of *Topic* together with the query to append the icon to every mind map node which has been assigned with *Topic* concept.
2. *Hour* icon for a concept of *DateTime* together with the query to append the icon to every mind map node which has been assigned with *DateTime* concept.
3. *Person shape* icon for a concept of *Person* together with the query to append the icon to every mind map node which has been assigned with *Person* concept.
4. *Question mark* icon for a concept of *Question* together with the query to append the icon to every mind map node which has been assigned with *Question* concept.
5. *Info* icon for a concept of *Answer* together with the query to append the icon to every mind map node which has been assigned with *Answer* concept.
6. *Checkmark* icon for a concept of generic *Event* together with the query to append the icon to every mind map node which has been assigned with *Event* concept and furthermore that the particular mind map node has been associated through relations with another mind map nodes with concepts of *DateTime* and *Topic* to mark the event mind map node that the additional information has been captured.

The third testing scenario is devoted to the testing of the automatic icons. The student activates the *automatic icons* improvement in the beginning of the test and then captures some information. The table (Tab. 7.3) shows the students actions and Ontominds responses in a similar way as previous testing scenarios. However, the steps 1-3 are left out because they are similar to the first test scenario. Furthermore some steps in advanced phases are also left out, for the sake of concision.

Ontomind application responses in table (Tab. 7.3) show that the new implemented *automatic icons* feature is functional and that it can be activated and deactivated at any time. This testing scenario also demonstrated the use-case in which such feature can be used.

7.3 Testing and demonstration summary

In this chapter first a demonstration ontology from students life was developed. The demonstration ontology might come useful for students in various use-cases. Then in the second part of this chapter, the new implemented features of Ontomind were tested with prepared test scenarios. These testing scenarios represented an average situations in which a student can find himself every day on a lecture. The result of these testing scenarios showed, that the new improvements help with the ease of creation of the mind maps and are trying to erase the restrictions which Ontomind introduced into mind mapping in a first place.

It is hard to capture the essence of the implemented improvements in a figure, because they deal with real-time manipulation with nodes. However, the improvement of *automatic icons* impacts the visual aspects of the mind map. The figures (Fig. 7.2 and Fig. 7.3) shows the same mind map with the automatic icons turned on and turned off. Furthermore, with the *ImprovedAutocompleteTree* enabled, these mind maps can be created even quicker.

#	**Students action**	**Ontominds response**
1.	Student creates new mind map.	Ontomind creates new mind map and takes student to the mind map editor, showing there is no assigned concept to the central node by exclamation mark icon.
2.	Student types in the name of the central topic.	Ontomind shows the *AutocompleteTree* component to assign a concept for the node.
3.	Student chooses a Course concept.	Ontomind assigns chosen concept and validate all nodes of the mind map if they are assigned to some concepts. (None marked).
4.	Student wants to capture the name of the teacher and creates a new mind map node.	Ontomind creates new mind map node.
5.	Student types in the name of the teacher.	Ontomind shows the allowed relations for the *Course* concept.
6.	Student chooses the relation of *hasPerson*.	Ontomind renames the node to *hasPerson*, assigns appropriate relation, prohibits the change of the text in the mind map node and again verifies all mind map nodes.
7.	Student creates a child node of the *hasPerson* node and enters the name of the teacher again.	Ontomind now shows the allowed concepts for the range of the relation.
8.	Student chooses the concept of *Person* for the appropriate node.	Ontomind assigns chosen concept and validate all nodes of the mind map.
9.	Student wants to capture information about another teacher, and repeats the steps 4-8.	Ontomind responds appropriately and in the same way to each of the students actions.
10.	Student wants to restructure the mind map by dragging and dropping the second person as a child of the one and only *hasPerson* relation making mind map less cluttered	Ontomind prohibits the restructuring of the mind map and allows only the original relation.
11.	Student deactivates ontological mode of Ontomind by clicking on the appropriate button on the toolbar.	Ontomind switches to non-ontological mode, informs student about the deactivation by flash message and all validation icons disappear.
12.	Student now restructures the mind map by dragging and dropping the second person to appropriate relationship and deleting the hanging relationship.	Ontomind restructures the mind map appropriately.
13.	Student now wants to capture the information in a sped up manner without assigning concepts or relations and the ability to restructure the mind map freely in non-ontological mode.	Ontomind does not show the *AutocompleteTree*. component in non-ontological and allows user to restructure the mind map according to his will.
14.	After the information capture is done, student switches back the Ontomind into ontological mode.	Ontomind informs user about activating ontological mode by flash message, prohibits mind map restructuring, activates *AutocompleteTree* component, and check all the mind map nodes if they are assigned with concepts or relations marking them with appropriate icons.

Table 7.1: Students steps and Ontominds responses in test scenario for ontological mode.

#	**Students action**	**Ontominds response**
9.	Student activates the *ImprovedAutocompleteTree* feature by clicking the appropriate button in the toolbar.	Ontomind turns on the feature and informs the student by flash message.
10.	Student wants to capture information about another teacher. Students creates another mind map node.	Ontomind creates new mind map node.
11.	Student assigns *hasPerson* relation to the new node.	Ontomind assigns the relation to the mind map node and analyzes the range property of the relation. There is only one option of *Person*. Ontomind automatically creates new child node for the relation and assigns a *Person* concept to it.
12.	Student wants to capture information about a lecture, thus student creates new mind map node.	Ontomind creates new mind map node.
13.	Student assigns *hasEvent* relation to the new node.	Ontomind assigns the relation to the mind map node and analyzes the range property of the relation. There are more options to be assigned. Ontomind does not create any new child nodes.
14.	Student now adds new child node to the *hasEvent* relation node.	Ontomind creates new child node.
15.	Student chooses *LectureEvent* concept for the mind map node.	Ontomind assigns appropriate concept to the mind map node and verify each node if they have concepts assigned. (Marking nodes with appropriate icons).
16.	Student wants to create topic for the *LectureEvent* so he creates new child node for the appropriate node.	Ontomind creates new child node for the *LectureEvent* node.
17.	Student chooses *hasTopic* relation for the node.	Ontomind assigns the relation to the mind map node and analyzes the range property of the relation. There is only one option of *Topic*. Ontomind automatically creates new child node for the relation and assigns a *Topic* concept to it.

Table 7.2: Students steps and Ontominds responses in test scenario for improved autocomplete.

#	**Students action**	**Ontominds response**
4.	Student activates automatic icons of Ontomind by clicking on the appropriate button on the toolbar.	Ontomind switches the automatic icons mode on, informs student about the activation by flash message and appends icons to appropriate nodes of the mind maps (if any - in this case none).
5.	Student wants to capture the name of the teacher and creates a new mind map node.	Ontomind creates new mind map node.
6.	Student chooses relation of *hasPerson*.	Ontomind assigns the appropriate relation to the mind map node and renames it.
7.	Student creates new mind map node, enters the name of the person and assigns a concept of *Person*.	Ontomind assigns the concept to the mind map node and automatically appends the node with icon of *person shape*.
⋮	Student repeats the steps 5-7 in a similar fashion trying to capture the information needed.	Ontomind responds appropriately and with expected results.
8.	Student wants to capture information about some unspecified event. He creates a new mind map node and assigns a concept of *Event* to it.	Ontomind assigns the appropriate concept to the mind map node.
⋮	In a similar fashion of steps 5-7, student adds child nodes to this event representing the *hasDateTime* and *hasTopic* relations. He then creates child nodes to these relations and assign appropriate concepts the the children mind map nodes.	Ontomind responds as expected and additionally, after the assignment of appropriate concept to the last child node it appends the icon of *checkmark* to a specific parent node with *Event* concept assigned to it.

Table 7.3: Students steps and Ontominds responses in test scenario for automatic icons.

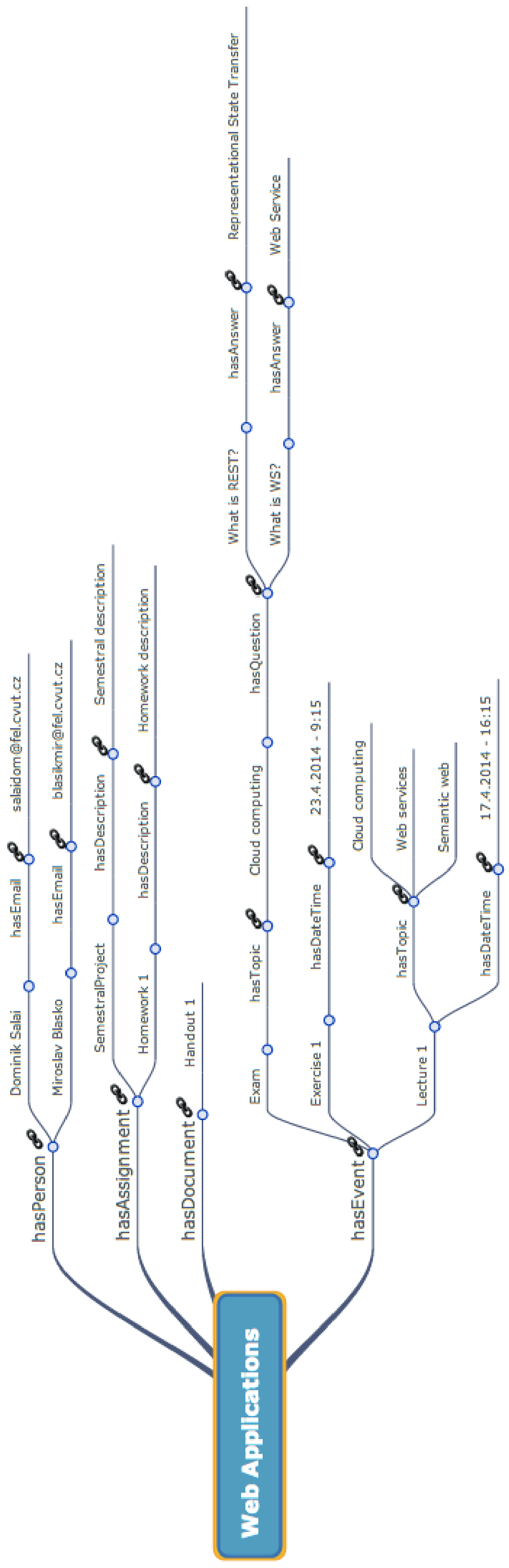

Figure 7.2: Typical Ontomind mind map.

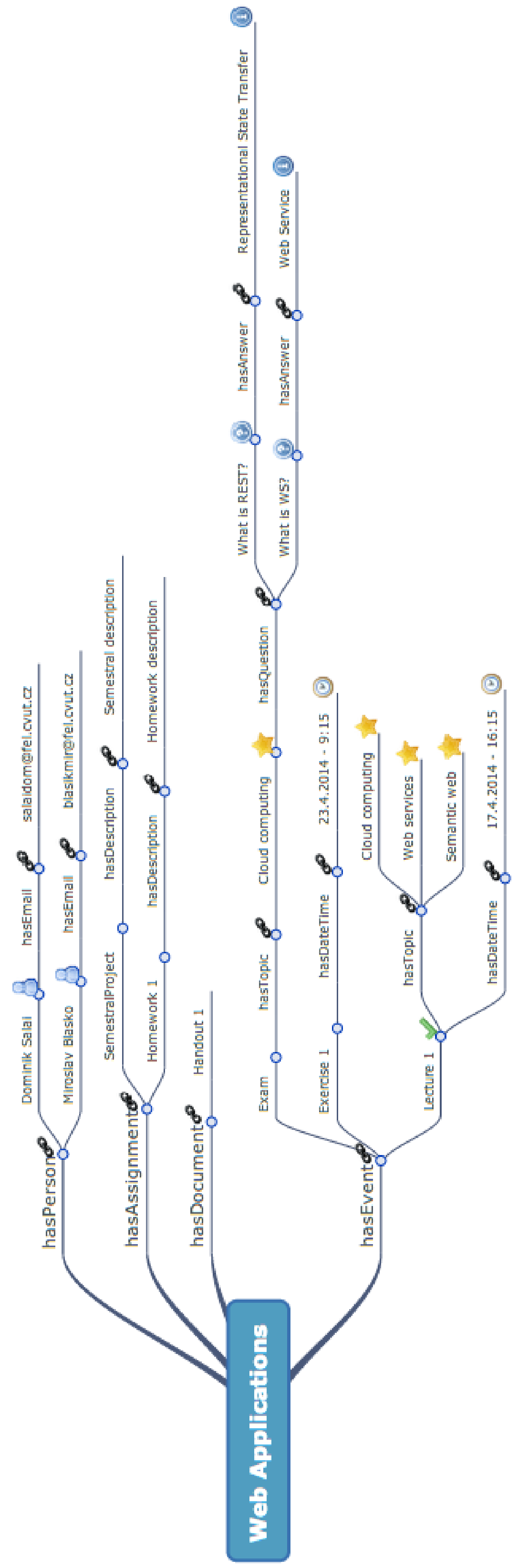

Figure 7.3: Ontomind mind map enhanced with automatic icons feature.

Chapter 8

Conclusion

This diploma thesis was devoted to the topic of mind mapping using semantic technologies. According to the goals presented in section 1.1 it was divided into few chapters to focus the discussion towards these goals.

Chapter 2 was devoted to mind maps analysis. This analysis showed that mind maps are quick and easy to made, highly adaptable to any use-case and great when it comes to creativity. On the other hand, mind maps suffer from reuseability and shareability, due to the personal nature of information captured.

Chapter 3 was dedicated to the analysis of ontologies. The final outcome of the analysis was that ontologies excel in reuseability and shareability because of the use of common vocabulary and thorough specification of the domain of interest. On the contrary, ontologies are difficult to develop and the development process is time consuming. They lack creativity and adaptability, and information captured according to ontology is usually more verbose.

The analysis of the knowledge management within Ontomind together with the discussion of ontology driven mind mapping was accomplished in chapter 4. The analysis showed that Ontomind has restricted some of the advantages of mind mapping techniques like freedom and easiness of mind map creation. On the other hand, the added value of ontological concepts and relations assigned to the information captured helps with the interpretation and overall reuseability and shareability of the mind map.

Methodology to manage the knowledge within Ontomind was first mentioned in section 2.8 where guidelines of creating good mind maps were presented. These guidelines were revisited in section 5.1 and together with the pros and cons of the mind maps presented in section 2.7, they were used and elaborated on throughout the whole chapter 5. Also the improvements of the knowledge management within Ontomind according to this methodology were proposed and discussed further in chapter 5 with the focus on maximizing the advantages of both the mind maps and ontologies.

The implementation of the chosen improvements was successfully accomplished in chapter 6 where a short overview of the implementation specifics was described. Then, the demonstration and testing of the methodology and the improvements of Ontomind on a specific domain was done in chapter 7. This demonstration and testing showed that with the improvement of non-ontological mode, Ontomind is able to handle more traditional use-cases where mind maps excel. Furthermore with the improvements of autocomplete and automatic icons the mind map creation process and visuals of the mind map have greatly improved which helps user friendliness and overall reusability and shareability.

Taking all the information presented throughout this diploma thesis, all of the goals were successfully accomplished.

This diploma thesis shown that the two fundamentally different techniques of information capture can be brought together to work even better. The biggest problem of reuseability of mind maps can be solved by introducing the common vocabulary and standardized way of capturing information which is one of the advantages of ontology. Furthermore the additional information of ontological concepts and relations brought to the mind mapping helps with mind map creation process, separation of valuable information from invaluable, and further processing and manipulation with the information captured in mind map is much easier.

8.1 Further research

If all of the proposed improvements were implemented, the overall experience of Ontomind would improve further by a great margin. The information in Ontomind mind maps will be captured and managed much easier than before. Such mind maps would be even more reuseable and shareable, and will enhance development of personal or group mind mapping techniques.

Additionally, the topic of ontology driven mind mapping is an interesting one from the point of what can be achieved by the added ontological information. Third-party applications can work with data easier and in a much more effective way. Furthermore, every ontological mind map can contribute to creation of really big knowledge archive with the help of linked data, thus making ontology driven mind mapping a definite candidate for further research.

Bibliography

[1] Maria Birbili. Mapping knowledge: Concept maps in early childhood education. `http://ecrp.uiuc.edu/v8n2/birbili.html` [Online; from 1-Apr-2014].

[2] Tony Buzan & Barry Buzan. *The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential.* First Plume Printing, 1996.

[3] Tony Buzan & Barry Buzan. *Myšlenkové mapy.* Biz Books Brno, 2012.

[4] Joseph D. Novak & Alberto J. Cañas. The theory underlying concept maps and how to construct and use them. `http://cmap.ihmc.us/Publications/ResearchPapers/TheoryCmaps/TheoryUnderlyingConceptMaps.htm|` [Online; from 1-Apr-2014].

[5] Lee Feigenbaum. Semantic web vs. semantic technologies. `http://www.cambridgesemantics.com/cs/semantic-university/semantic-web-vs-semantic-technologies` [Online; from 22-Jan-2014].

[6] W3C Working Group. Owl 2 web ontology language primer (second edition). `http://www.w3.org/TR/2012/REC-owl2-primer-20121211` [Online; from 9-Mar-2014].

[7] W3C Working Group. Rdf 1.1 primer. `http://www.w3.org/TR/rdf11-primer` [Online; from 9-Mar-2014].

[8] Riichiro Mizoguchi. Tutorial on ontological engineering. `http://www.ei.sanken.osaka-u.ac.jp/pub/miz/Part1-pdf2.pdf` [Online; from 10-Feb-2014].

[9] Riichiro Mizoguchi. Task ontology for reuse of problem solving knowledge. In *KB & KS95*. Enshede, The Netherland, 1995.

[10] Marek Obitko. Ontologies. `http://www.obitko.com/tutorials/ontologies-semantic-web/specification-of-conceptualization.html` [Online; from 4-Feb-2014].

[11] Matthew Horridge & others. A practical guide to building owl ontologies using protege 4 and co-ode tools, 2011.

[12] Vasilis Siochos & Christos Papatheodorou. Developing a formal model for mind maps.

[13] James Schoening. Standard upper ontology. `http://suo.ieee.org` [Online; from 9-Mar-2014].

[14] Word Smiths. Mind mapping for decision making. `http://wordsmithsuk.wordpress.com/2011/03/02/mind-mapping-for-decision-making/` [Online; from 25-Dec-2013].

[15] J. Sowa. Distinction, combination and constraints. In *Proc. of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[16] Thinkbuzan - imindmap mind mapping software. `http://thinkbuzan.com/products/imindmap/` [Online; from 24-Apr-2014].

[17] Apache jena. `https://jena.apache.org` [Online; from 11-Mar-2014].

[18] Tony buzan - inventor of mind mapping - what is a mind map? `http://www.tonybuzan.com/about/mind-mapping` [Online; from 25-Dec-2013].

[19] How imindmap can make your revision a success. `http://blog.thinkbuzan.com/imindmap/how-imindmap-can-make-your-revision-a-success` [Online; from 25-Dec-2013].

[20] mootools. `http://mootools.net/` [Online; from 11-Apr-2014].

[21] openrdf.org ... home of sesame. `http://www.openrdf.org/index.jsp` [Online; from 11-Mar-2014].

[22] Křemen & Mička & Šmíd & Blaško. Ontology-driven mindmapping. In *I-SEMANTICS 2012, 7th Int. Conf. on Semantic Systems*, 2012.

Appendix A

Personal contribution

The structure of this diploma thesis is not ordinary and the personal contribution to this topic is scattered into different chapters, the figure (Fig. A.1) identifies the personal contribution to this topic in a from of a mind map. The mind map mirrors the structure of this diploma thesis and the personal contribution is marked with the light bulb icon next to the section name.The branches of the mind map representing each chapter are also colored for better orientation.

Also the key sections and chapters of personal contribution are summarized:

- Mind maps analysis in section 2.6.
- Mind maps pros and cons in section 2.7.
- Ontologies analysis in section 3.8.
- Ontologies pros and cons in section 3.9.
- Ontominds ontology driven mind mapping approach in section 4.4.
- Ontominds pros and cons in sections 4.5 and 4.6.
- Proposed Ontominds improvements and methodology in chapter 5.
- Implementation of chosen improvements in chapter 6.
- Development of demonstration ontology in section 7.1.
- Demonstration and testing of the new improvements in chapter 7.

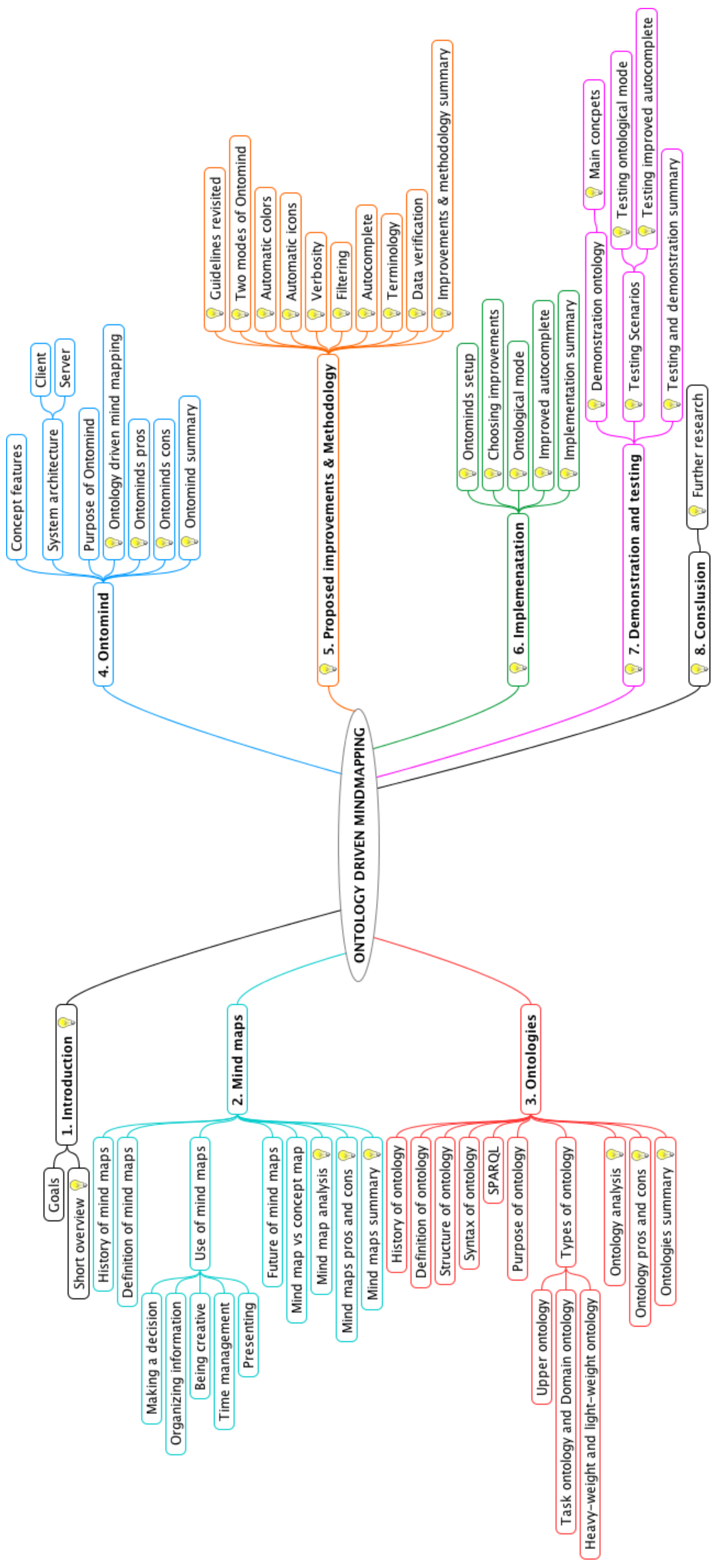

Figure A.1: Mind map with personal contribution to the topic.

Appendix B

Abbreviations

2D - Two-Dimensional space

3D - Three-Dimensional space

4D - Four-Dimensional space

AJAX - Asynchronous Javascript And XML

API - Application Program Interface

CTU Prague - Czech Technical University in Prague

HTML - Hypertext Markup Language

IEEE - Institute of Electrical and Electronics Engineers

IRI - Internationalized Resource Identifier

JRE - Java Runtime Environment

JSON - Javascript Object Notation

JVM - Java Virtual Machine

MVC - Model, View, Controller

OWL - Web Ontology Language

ORM - Object Relational Mapping

PDF - Portable Document Format

RDBMS - Relational Database Management System

RDF - Resource Description Framework

RDFS - Resource Description Framework Schema

REST - Representational State Transfer

SPARQL - SPARQL Protocol And RDF Query Language

SQL - Structured Query Language

SUMO - Suggested Upper Merged Ontology

SVG - Scalable Vector Graphics

URI - Uniform Resource Identifier

W3C - World Wide Web Consortium

XML - Extensible Markup Language

Appendix C

Installation & user guide

This installation guide describes the steps which are needed to be done in the following order to get the prepackaged improved Onotmind application up and running on a local machine together with additional database and back-end software running also on a local machine.

C.1 Prerequisites

The back-end of the Ontomind application is written in *Java* language and needs a web server which implements *Java Servlet* and *Java Server Pages* specification to run the *Java* code. Such web servers are for example *Apache Tomcat*, *jBoss WildFly*, *Oracle Glassfish* or *Jetty*. The installation and correct setup of such web servers is out of the scope of this installation guide and is left on the users. The correct installation may also include the installation of either specific *JDK* or *JRE* on local machine.

Another prerequisite is installation and configuration of database application such as *MySQL* or *PostgreSQL* (used in this installation guide) which is also left on the user. However there may be a need to provide additional library for the previously mentioned Java web server to be able to connect to the database.

The requirements are summarized:

Java Web Server - needs to implement *Java Servlet* and *JSP* specification.

Memory - the *JVM* on which the server will be running needs at least 512 MB of memory for Ontomind application and other supporting applications.

Database - installation and setup of appropriate database system together with provided libraries for communication with the Java web server

C.2 Deploying Ontomind

This tutorial for deploying Ontomind uses *Apache Tomcat 7* as a Java web server and *PostgreSQL* for database system. Using different applications may introduce some problems to the deployment but in general they shouldn't.

1. Create two databases in *PostgreSQL*, one for mind maps called *"mindmaps"* and one for ontological data called *"ontologies"*. Create user named *"salaidom"* with *blank* password and give it privileges to modify these databases.

2. Deploy *"application/openrdf-sesame.war"* and *"application/openrdf-workbench.war"* files on the web server.

3. In browser, navigate to http://localhost:8080/openrdf-workbench.

4. In *openredf-workbench*, create new *"PostgreSQL RDF Store"* and enter required information for the *"ontologies"* database, give the repository a name of *"ontomind_psql"*.

5. In *openredf-workbench*, navigate to *"Contexts"* and in *"Modify"* section of the menu click *"Add"*.

6. Enter *Base URI* and *Context* values to be the same - *"<http://www.semanticweb.org/salaidom/ontologies/course-ontology.owl>"*

7. Choose *RDF Data File* to be *"ontologies/course-ontology.owl"* and click upload.

8. Repeat the steps 5-7 for two additional files *"ontologies/ontomind-annotation-ontology.owl"* and *"ontologies/course-ontology-ontomind-annotations.owl"*. Please, make sure to give them appropriate *Base URIs* and *Contexts* according to the filenames.

9. Deploy *"application/ontomind/ontomind.war"* on the web server.

10. Run script *"appplication/scripts/test-data.sql"* on *"mindmaps"* database to load testing users.

11. In browser, navigate to http://localhost:8080/ontomind.

After these steps, the application should be up and running. If some error occurs, please review the steps again.

C.3 User guide

Ontomind is based on *wisemapping* and the user guide on how to work with *wisemapping* mind map editor can be found here or on wisemappings website.

Working with Ontomind is almost identical as working with *wisemapping*, but Ontomind introduces some restrictions to the *wisemapping*. Ontominds ontology driven mind mapping approach is enabled by default and works automatically, so no special user guide is needed to use Ontomind.

The only relevant information to the user guide is the activation and deactivation of new implemented features of Ontomind. To minimize the impact of the improvements on Ontominds original behavior, the improvement of *ontological-mode* is enabled by default when the new mind map is created. Other improvements need to be activated first to enhance the user experience.

Activation and deactivation is done through toggle buttons on Ontominds toolbar with the icons depicted on figure (Fig. C.1). Enumerating from left to right the icons represent the improvements of *ontological/non-ontological mode* (icon with letter 'm'), *improved autocomplete* (icon with letter 'a') and *automatic icons* (icon with letter 'i').

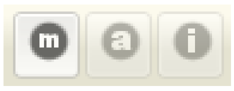

Figure C.1: Toggle buttons for activating and deactivating improvements.

Appendix D

CD content

sourcecode - Folder containing source code of Ontomind application and additional mondis project components which ontomind uses.

text - Folder containing diploma thesis text in TEX and PDF format.

mindmaps - Folder containing personal contribution mind map in *freemind* mind map format.

application - Folder containing Ontomind application and other applications and files needed to setup and run prepackaged Ontomind on a local machine.

ontologies - Folder containing demonstration and annotation ontologies in *RDF/XML* format.

other - Folder containing *GIT* diff files to address the changes in the sourcecode.

readme.txt - File containing the content of the CD folders together with instructions on how to setup prepackaged Ontomind application on a local machine.