Florian Salbrechter

Decision support for diagnosis on base of ultrasound images

Master's Thesis

Graz University of Technology

Institute for Information Systems and Computer Media (IICM)
Head: Univ.-Prof. Dipl-Ing. Dr.techn. Frank Kappe

Supervisor: Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Nikolai Scerbakov

St.Veit an der Glan, September 2014

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz,	______________________	______________________
	Date	Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am	______________________	______________________
	Datum	Unterschrift

[1]Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Dedication

This Master thesis is lovingly dedicated to my family.

Without their love and continuous support throughout my studies this thesis would not have been possible and I would not be where I am today.

Acknowledgement

This thesis would not have been possible without the help of several individuals whom I would like to thank for their constant support and help.

First, I would like to thank my supervisor, Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Nikolai Scerbakov for his guidance, support, and patience throughout the project.

I would like to thank my family for all the support I received from them through my whole studies.

Personally, I would like to thank all my friends for their great support and friendship during this thesis.

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

Abstract

This work shows how ontologies can be used in ultrasound diagnosis decision support systems. With the developed application, ultrasound images can be annotated with observations defined by an ontology. This developed ontology links observations on ultrasound images to potential diseases in the example of the gallbladder. Based on the ontology and the dynamically added image annotations a list of potential diseases is shown. Additionally, the application shows an explanation for each suggested diagnosis. The ontology is stored in a triple store, and SPARQL is used for queries. There are many advantages of enabling a wider group of people to perform simple ultrasound exams. For example, in poor countries where proper ultrasound training can not be provided for cost reasons, it can save lives. Even in developed countries, it can save costs by cutting down unnecessary referrals and improve patient satisfaction by streamlining ultrasound examination workflows. The application can be extended to develop an educational application to bring ultrasound training to a wider audience.

Keywords: Ultrasound, Ontology, OWL, Decision Support Systems

Kurzfassung

Diese Arbeit zeigt wie Ontologien in Entscheidungshilfesystemen für Ultraschalldiagnosen verwendet werden können. Mit der entwickelten Anwendung können Ultraschallbilder mit Beobachtungen, welche in einer Ontologie definiert sind, annotiert werden. Die entwickelte Ontologie verknüpft Beobachtungen auf Ultraschallbildern mit möglichen Krankheiten am Beispiel der Gallenblase. Basierend auf der Ontologie und dem annotierten Bild, wird eine Liste von möglichen Erkrankungen vorgeschlagen. Zusätzlich wird eine kurze Erklärung für jeden Vorschlag gezeigt. Die Ontologie ist in einem Triplestore gespeichert. Für Abfragen wird SPARQL verwendet. Es gibt zahlreiche Vorteile einer größeren Anzahl von Personen einfache Ultraschalluntersuchungen beizubringen. In armen Länder, in denen es aus Kostengründen keine Ultraschallausbildung gibt, kann dieses Wissen Leben retten. In entwickelten Ländern kann die Anwendung durch eine Vermeidung von unnötigen Überweisungen an andere Fachrichtungen Kosten sparen. Die Anwendung ist auch für Ultraschalltrainings erweiterbar.

Schlüsselwörter: Ultrasound, Ontology, OWL, Decision Support Systems

Epigraph

"The only way you can predict the future is to build it."

-Alan Kay

Contents

List of Figures

List of Tables

Listings

Glossary

AI Artificial Intelligence.

ASCII American Standard Code for Information Interchange.

CDSS Clinical Decision Support System.

DTD Document Type Definition.

GP General Practitioner.

HTML HyperText Markup Language.

IRI Internationalised Resource Identifier.

IT Information Technology.

JSON JavaScript Object Notation.

OWL Web Ontology Language.

RDF Resource Description Framework.

RDFS Resource Description Framework Schema.

RFC Request for Comments.

SPARQL SPARQL Protocol and RDF Query Language.

SQL Structured Query Language.

UCS Universal Character Set.

URI Uniform Resource Identifiers.

URL Uniform Resource Locator.

URN Uniform Resource Name.

US Ultrasound.

UTF-8 UCS Transformation Format-8-bit.

XML Extensible Markup Language.

1. Introduction

The combination of Information Technology (IT) and medicine is an extremely challenging task in many different ways. Traditionally, Information Technology is very fast paced. Technologies move fast and changes happen frequently. On the contrary, medicine changes very slowly and uses a more conservative strategy. Changes are preceded by exhaustive testing, studies and reviewing of new technologies or methods before they are actually incorporated in the conventional medical art. Ultrasound is a simple but very powerful technology. Different countries have different healthcare systems, and depending on the country not every physician is trained to perform simple ultrasound scans. In some countries' healthcare systems in the world if a patient needs an ultrasound exam he/she is referred to a specialist. Many patients are referred unnecessarily, which could have been avoided by simple checks at the first physician. This yields a high volume of ultrasound scans and a high number of negative test results. There is a multitude of reasons why physicians are reluctant to get training for ultrasound and include simple ultrasound exams in their daily examination process including:

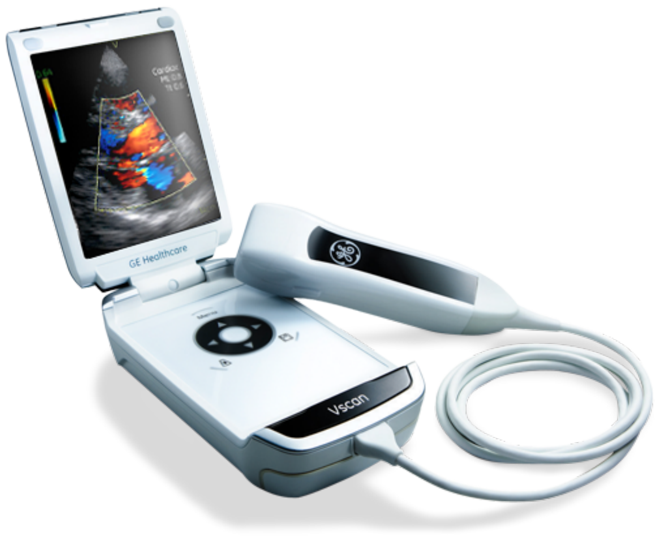


Figure 1.1.: Vscan portable ultrasound device (taken from vscan.gehealthcare.com)

- lack of knowledge,
- lack of competence and
- lack of time

Additional conventional ultrasound equipment is quite expensive and can be a big financial burden, and requires some physical space; both of which a physician might not be able or willing to invest. Depending on the country, the process of getting ultrasound training and a certification usually requires a fair amount of time - and time is a very scarce resource for physicians. A new generation of extremely small, portable ultrasound devices might change that by lowering the threshold of using ultrasound. Vscan is one example of a portable ultrasound device developed by General Electric (GE)

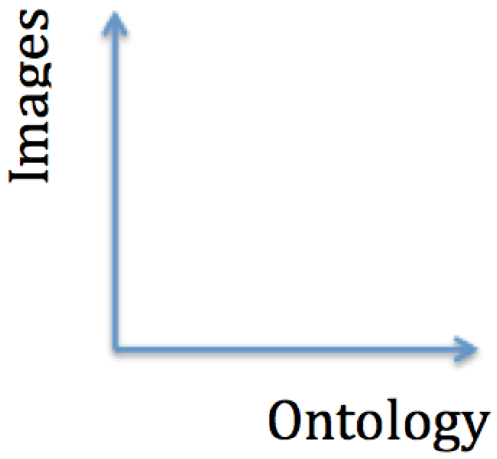


Figure 1.2.: Dimensions

and is shown in Figure 1.1. This new generation of medical devices will challenge the current ultrasound workflows.

There are different ways to tackle the shortcoming and problems described above and give every physician the ability to perform simple ultrasound scans for a better patient care and a more cost effective healthcare system, see Figure 1.2.

Focus on images One idea worth pursuing is to show the doctor ultrasound reference images and let him/her compare them to actual ultrasound images. The reference image that is most relevant can be chosen and used to determine the diagnosis. This approach focuses on reference images and is proposed and analysed in Salbrechter, 2013. The aim of this project was to develop an integrated application that can be used on mobile devices (e.g.

tablets) to assist general practitioners (GP) and nurses with little previous experience in ultrasound scanning. The developed web application is user friendly, helps users to perform simple ultrasound scans using portable ultrasound devices and acts as a triage tool to cut down the number of unnecessary referrals. The application offers guidance, training, and test possibilities for self-study and self-assessment including educational videos. An augmented reality application assists the user in taking the proper picture, which then can be compared to an extensive reference image library.

Focus on ontology Another approach is to aid the doctor through the diagnosis with a decision support tool using ontologies as knowledge base. For each organ, a detailed ontology is created where the user can classify and annotate each ultrasound image by selecting numbers of features from an ontology. The ontology is developed by domain experts and ontology engineers who are subject matter experts for the domain of discourse. Based on the selected features and the values of the features, e.g. size of an organ (gallbladder, heart, ...), colour and shape of a specific part on the ultrasound image and many other organ characteristics, the decision support tool shows the most likely diseases or asks for more specific information and annotations.

This thesis will further evaluate the ontology based approach and will develop an application prototype to better illustrate and demonstrate this idea.

The remainder of this thesis is organised as follows. First, Section 2 provides a detailed literature review. Section 3 covers the methodology of this thesis. Section 4 provides an overview of the Application, its development and its functionality. The thesis finishes with a Conclusion (Section 5) and Future Work (Section 6).

2. Literature Review

This literature review explores and assesses three different areas on how these technologies could be potentially useful if utilised:

- **Ultrasound**: This part will give the reader an introduction to ultrasound and the underlying physics involved. Ultrasound is a simple, yet powerful diagnostic tool. Interpreting and diagnosing ultrasound images is difficult and there are numerous systems that aid the physician. The developed system will operate on ultrasound images and assist the user to understand and annotate certain features.
- **Artificial Intelligence (AI)**: The process of diagnosing is very complex and requires a lot of joined-up thinking. Current IT systems are very good at completing simple, straightforward tasks with few exceptions, e.g. prescription management and dose suggestions based on weight. There are many attempts to apply AI for medical diagnosis tasks but most of them only have moderate success. The biggest success is achieved when the AI system focuses on very specific topics or aspects of diagnosis, e.g. diagnosis of myocardial infarction (Baxt and

Skora, 1996) (Hedén et al., 1997) and the detection of breast cancer on screening mammograms. (Petrick et al., 2002) (Bates et al., 2001) (Salbrechter, 2013)

- **Ontology**: Ontologies are used to share and exchange formalised knowledge. In the context of this thesis they can be used to create a vocabulary for describing physical and medical properties of ultrasound images as well as describing diseases using this vocabulary. Ontologies can be created from scratch or by modifying and reusing parts of existing ones. Ontologies will be used to represent the knowledge structure for the application.

2.1. Ultrasound

In this chapter, a basic introduction into ultrasound and the underlying physics is given. Ultrasound waves are sound waves above a certain frequency (>20kHz). These sounds can not be heard by humans. Ultrasound has many applications both in nature and science. Examples are:

- Bats use ultrasound to navigate.
- Sonar uses sound waves to detect obstacles underwater.
- Quality control by checking flaws in metal.
- Taking two dimensional images of tissue in medicine.

(Lieu, 2010)

2.1.1. Sound

There are two different types of waves:

- **Longitudinal**: In longitudinal waves the vibrations in the medium are in the same direction as the wave motion. These waves cyclically compress and rarefy the medium they are traveling through. Examples are sound waves and seismic waves created by an earthquake.
- **Transverse waves**: In transverse waves the vibrations in the medium are perpendicular to the direction of the motion. Examples are waves created by a stone that was thrown in a quiet lake or waves that travel in a rope from one end to another.

Sound is a mechanical longitudinal wave. Since ultrasound is sound with a special frequency, it uses longitudinal waves to travel through mediums. In this thesis, only the medical use of ultrasound is covered.

Sound waves can be defined by frequency f (measured in hertz [Hz]), wavelength λ (measured in millimetres) and amplitude A. Figure 2.1 shows how the medium is compressed and decompressed, and how the corresponding sound wave looks like.

The speed v of a wave is the product of wavelength and frequency:

$$v = \lambda f$$

(Lieu, 2010) (Palmer, 1995) (Narouze, 2011)

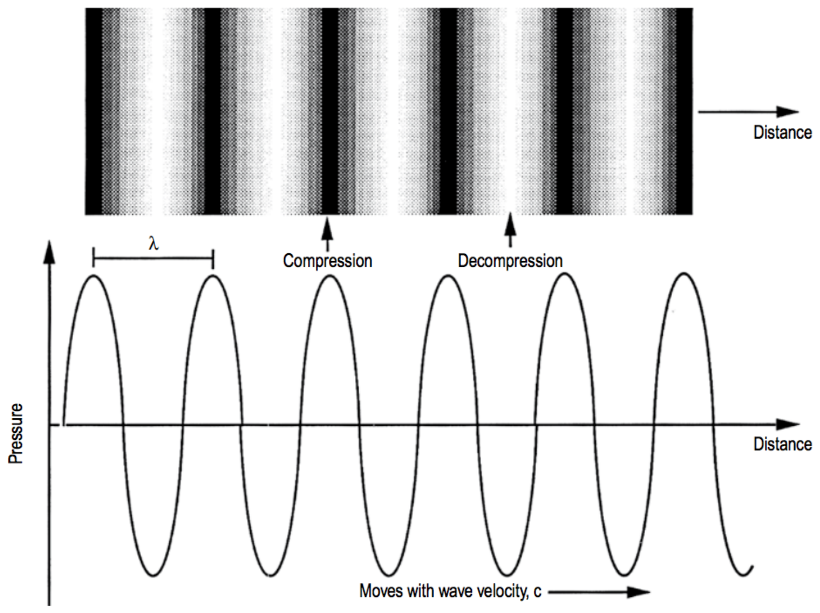


Figure 2.1.: Sound waves (taken from Eik-Nes and Wladimiroff, 2009)

The higher the frequency, the better the resolution of the resulting ultrasound image, but high frequency waves are more attenuated than low frequency waves in a given distance. High frequency ultrasound can be used for high resolution images of surface tissue. Low frequency ultrasound produces lower resolution images, but can be used to scan deeper body tissues.

There are different modes of ultrasound, but B-mode (brightness mode) is the most widely used one. Ultrasound can be compared to a speaker and microphone system. The basic principle is to emit ultrasound waves using a transducer (also called probe). As the ultrasound beam hits tissues with different impedances, it is partially reflected. The transducer registers these reflected echoes and creates an image out of the received signals using the

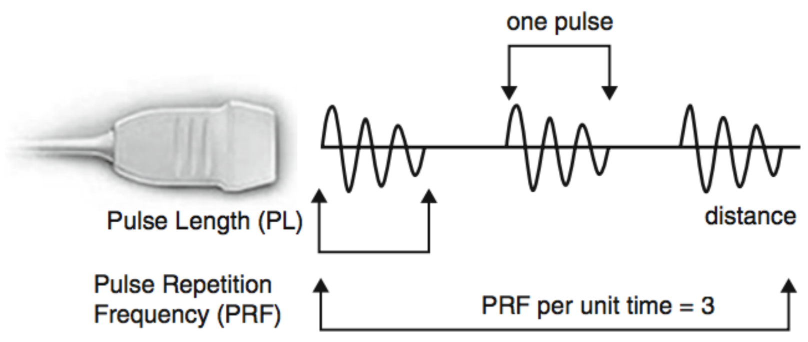


Figure 2.2.: Probe generating ultrasound pulses (taken from Narouze, 2011)

timing, phase and amplitude of the response.

Ultrasound transducers (probes) make use of the piezoelectric effect which enables the transduction (conversion) between mechanical energy and electrical energy.

(Eik-Nes and Wladimiroff, 2009) (Narouze, 2011)

Figure 2.2 shows how the probe generates impulses.

As the ultrasound waves travel through tissue, they are partly transmitted to deeper structures/tissues and partly reflected. Some of the reflected ultrasound will be received by the probe, and the timing and amplitude information is used to generate an image.

The reflection of ultrasound happens between the interface of two tissues with different acoustic impedance, which is an intrinsic physical property of a medium. The greater the difference in acoustic impedance, the more

Table 2.1.: Acoustic impedances of different tissue (taken from Eik-Nes and Wladimiroff, 2009)

Impedances of different tissue	
Body tissue	Acoustic impedance (10^6 Rayls)
Air	0.0004
Lung	0.18
Fat	1.34
Liver	1.65
Blood	1.65
Kidney	1.63
Muscle	1.71
Bone	7.8

ultrasound waves are reflected, resulting in less ultrasound waves being transmitted to deeper body tissues.

Table 2.1 shows acoustic impedances for different tissues.

(Eik-Nes and Wladimiroff, 2009) (Palmer, 1995) (Narouze, 2011)

2.2. Artificial Intelligence

Although artificial intelligence (AI) is a very common word and we are surrounded by intelligent machines (so called agents), it is not easy to define it. Russell and Norvig, 2009 give an overview of the most widely used definitions.

(Salbrechter, 2013)

2.2.1. Definition

Acting humanly One way to define AI is by the ability of acting humanly, as reflected in:

“The art of creating machines that perform functions that require intelligence when performed by people.” (Kurzweil, 1990)

“The study of how to make computers do things at which, at the moment, people are better.” (Rich and Knight, 1991)

To measure the process in this field Alan Turing (1950) developed the Turing Test. The test begins with a human subject asking a series of questions. A computer passes the test if the human subject can not tell the answers of a computer and human apart.

Thinking humanly This approach tries to teach a computer how humans think in addition to just acting humanly:

“The exciting new effort to make computers think ... machines with minds, in the full and literal sense.” (Haugeland, 1985)

"[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ..." (Bellman, 1978)

Thinking rationally This approach is based on logic and inference. This definition considers AI doing the "right thing" given the knowledge available to the machine:

"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1978)

"The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)

Acting rationally AI is to develop rational agents (something that acts) which interact with the environment and try to achieve the best possible outcome given a certain task:

"Computational Intelligence is the study of the design of intelligent agents." (Poole, Mackworth, and Goebel, 1998)

"AI ...is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

(Salbrechter, 2013)

2.2.2. Basics

Artificial Intelligence (AI) is a very abstract concept and can be applied to a multitude of fields. AI can be used to assist or replace humans with certain tasks and is the basis for all "intelligent" tools. What intelligence is depends on the application and needs to be defined on a context or domain basis. One of the biggest use cases of AI is to automate and unify decisions which are based on the same set of rules. AI usually consist of two phases:

1. **Training phase**: The AI system is presented with vast amounts of data, which is usually too big to be processed by humans in a reasonable period of time. In this system the AI tries to "learn" the underlying structure of the data and extract patterns.
2. **Test phase**: In this phase the AI is evaluated on new data to see how effectively it performs. After this phase the AI can be either used or trained again using more and/or different data.

One of the biggest advantages of AI is that it is objective and unbiased, which means that it only operates on facts and enables a far more objective judging and diagnosing power compared to humans. AI is a very broad field and there are many possible applications. Referencing the stated problem in the Introduction (Section 1) of the thesis, there are many potentially beneficial applications:

- **Administration**: AI is very good at optimisation problems and can help to facilitate and streamline workflows. Other examples include

Electronic Health Records which store all the necessary medical information about patients and provide systems to exchange them.

- **Clinical decision support system**: As medicine advances, there is more and more information available for diagnosis. AI aids in organising and structuring this information by simplifying the complex decision making, which is essential for a correct diagnosis. It facilitates the diagnosis process by assisting and giving guidance. The clinical decision support system guides the user throughout the process, from identifying symptoms to a diagnosis.
- **Image Analysis**: Analysing a large amount of images can be very tedious and error-prone. Analysts can get tired by this repetitive work. As a potential solution, AI can be trained to detect certain predefined patterns in images as well as direct physicians' attention to targeted images. The throughput of such systems is much higher than human systems can ever achieve. Additionally to drawing the physician's attention to certain images it can also suggest diagnoses. Besides the advantages in mass screenings, AI can also be used to help physicians to gain a more in-depth understanding of complex pictures, e.g. ultrasound images. The underlying principle of ultrasound images is that they are 2D images of complex shaped 3D objects. Usually physicians work on these 2D images but AI can be used to create a 3D representation of a series of 2D ultrasound images which ultimately helps the physician to get a full overview of the scanned tissues.
- **Sound Analysis**: Sound and images can both be seen as signals, and similar AI techniques can be applied. One application of sound analy-

sis could be the automatic examination of heart beats. Heart beats are a complex combination of different valve beat sounds. AI can be used to decompose the signal and diagnose any problems.

(V. Patel et al., 2009) (Kushniruk and V. L. Patel, 2004) (Salbrechter, 2013)

2.2.3. Development

In the beginning of the AI era scientists were highly motivated and very confident about developing an AI which is equivalent to the human brain in a short period of time. Expectations were very high and there was enormous pressure on the scientists. Although AI had some initial success, the progress flattened and the researchers were not able to fulfil their high expectations. It turned out that the underlying problem was way more complex than previously estimated. With the growing field of AI, researchers decided to focus on subsets of the original problem. One of the division of AI is to divide it between "weak" and "strong" methods. "Weak" methods focus on one specific task. Expert systems are a good example of weak methods. They solve a sub problem and try to understand, mimic and optimise the human decision making process. The problem of weak methods is that they do not scale up to general problems. "Strong" methods on the other hand try to teach machines to think and to be conscious.

(Russell and Norvig, 2009)

There were two big expert systems for medical purposes which marked the beginning of a new era:

- **MYCIN** (Shortliffe and Buchanan, 1975) was used to identify bacteria which cause severe infections, and
- **CADEUS** (Banks, 1986) extended MYCIN and was able to diagnose 1000 diseases in internal medicine.

Expert systems store their knowledge in a knowledge base. During the development of early expert systems, a new challenge arose: acquiring, representing and handling all the necessary knowledge. The discipline of knowledge engineering was born. It is concerned with the collection and representation of knowledge.

(Hayes-Roth, 1984)

As “strong” methods turned out to be unsolvable during that time, the research focus has since shifted to decision support systems. A good example for this is IBM’s supercomputer Watson. With its algorithms, it is able to analyse a vast amount of unstructured data and save it in its knowledge base. Watson was able to demonstrate its abilities in some high public profile applications, including the television game show Jeopardy, which it won. After having success in this game, the plan was to bring Watson to new domains. There has been an announcement that one branch of the Watson program will specialise on lung cancer.

(Feldman, 2012) (Salbrechter, 2013)

SonaRes is an excellent example for a diagnostic decision support system (DDDS), which is designed for a specific area (ultrasound). It focuses on assisting doctors for ultrasonographic examinations in the abdominal zone. This project also creates a knowledge base in order to structure the knowledge about organs. For each fact, questions are formulated which the user needs to answer and thereby annotating the observation, e.g. "Is the volume of the gallbladder enlarged?". The project also considers two dimensions or layers of structure of information in an image:

1. **Graphical features**: all the information stored in the image itself. Systems that focus on this features are called image-based systems.
2. **Medical features**: the textual description in medical terms (annotations). Systems that focus on this features are called knowledge-based systems.

Since ultrasound is based on images and their interpretation, there is a major dependency on the operator to construe its significance.

(Burtseva et al., 2011) (Gaindric, 2009)

2.3. Ontology

Ontology is a well known term in philosophy concerning the metaphysics of being. In computer science, the same term is also used but with a slightly different meaning:

"Computational ontologies are a means to formally model the structure of a system, i.e., the relevant entities and relations that emerge from its observation, and which are useful to our purposes." (Guarino, Oberle, and Staab, 2009b)

Ontologies consist of two parts:

- Vocabularies that define terms and concepts in a region of discourse, and
- Relationships between the defined terms/concepts/entities vocabularies.

Ontologies play a crucial role in the vision of the semantic web. In order to better understand the role and importance of ontologies, the idea of the semantic web is described in the next section.

2.3.1. Semantic Web

The current web as we know it is primarily designed for human users and is very difficult to be processed automatically by machines. The idea of the semantic web and linked data is to enable machines to have better access to information on the web.

The basic principle behind search machines, e.g. Google, Yahoo, and Bing is to find web pages that match the search keywords. They rarely consider the meaning, i.e. semantic information of the search keywords or the semantic information of the indexed websites. In the semantic web, so called agents will be able to communicate and interpret information provided by different websites directly.

A good example to show how different the semantic web works from the current web is a price comparison engine for flights. Imagine a website that compares multiple airline sites to get you the cheapest flight. The price information has to be “crawled” from every airline. This means that for each airline, a piece of software has to be written in order to retrieve the price information. Any changes in the layout can possibly break this piece of software, resulting in the need for it to be rewritten.

In the semantic web, the information about flights and prices is available in a machine-readable format and can be retrieved directly. The structure of the data and the meaning is encoded in a machine-accessible way using semantic web technologies.

Layers of the Semantic Web

The semantic web is organised in layers and each layer builds upon the layer below. This chapter shortly explains each layer in order to better understand the concept of an ontology. Figure 2.3 shows the layer model originally proposed by Tim Berners-Lee during the XML2000 conference.

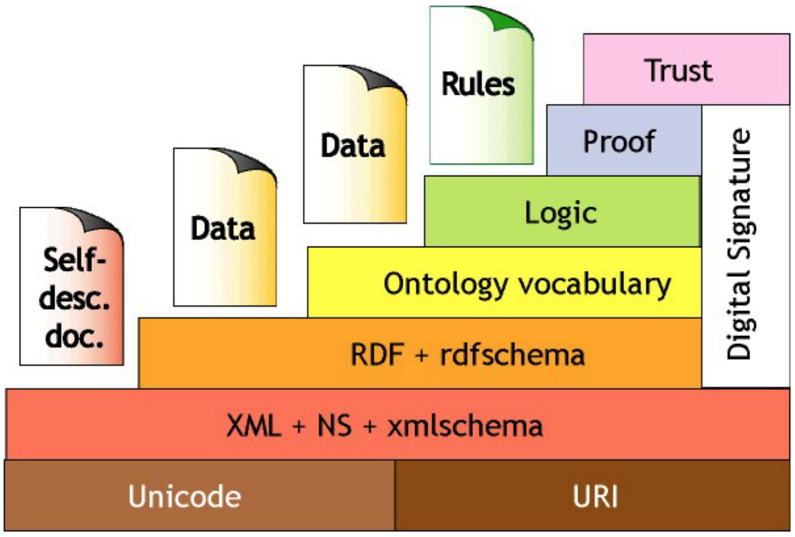


Figure 2.3.: Semantic web layer model (Antoniou and Harmelen, 2008)

Unicode Unicode is an encoding scheme that enables a display of a larger variety of characters than ASCII. Unicode uses the concept of code points to represent characters and uses different encodings, e.g. UTF-8 which uses between 8 and 32 bit for each character or UCS-2 which uses 2 bytes for every character.

(Korpela, 2006)

URI URIs (Uniform Resource Identifiers) are used to uniquely identify resources in the web and on the real world. A resource can be any object that is of interest in the current domain, e.g. people, articles, shoes, books. URI are defined in RFC 2396[1]. URI can be either a URL (Uniform Resource

[1]http://www.ietf.org/rfc/rfc2396.txt

Locator), a URN (Uniform Resource Name), or both.

(Antoniou and Harmelen, 2008) (Allemang and Hendler, 2008)

XML, namespaces, and schema XML (EXtensible Markup Language) is similar to HTML but focuses more on the representation of information in a machine-accessible way. It is a uniform data exchange format between applications. The following listing shows an example of a simple XML file describing a book.

Listing 2.1: XML example for a book

```
<book type="thesis">
  <author>Florian Salbrechter</author>
  <title>Master's␣thesis</title>
␣␣<publicationDate>1984</publicationDate>
</book>
```

XML works with tags which are enclosed by angular brackets. Every start tag has an end tag, in the above example <book> and </book>. Elements are formed by a start tag, content and end tag.

XML attributes provide additional information about an element, e.g. the type attribute of the book tag in the above example.

With XML a custom data format can be created that fits the application best. XML documents can be structured in many different ways. In order to define some limitations or structure for XML documents, two standards exist:

- **DTD (Document Type Definition)**: DTD is one of the earlier, and older approaches and provides a basic language to define some limitations on the structure.
- **XML schema**: XML schema is a newer approach, and will replace DTD. It provides a richer language to define the structure. XML schema is an XML document itself.

The concept of namespaces is used to avoid name clashes between elements defined in different XML documents. Imagine you want to combine two different XML documents that have different structures and XML schemata, but the elements have the same name, e.g. book in the example above. Namespaces provide a way to properly address the two definitions by adding a prefix.

XML has a tree-based structure and can be queried using XQL, XML-QL, and XQuery.

RDF and RDFS "The Resource Description Framework (RDF) is a framework for representing information in the Web." (Cyganiak, Wood, and Lanthaler, 2014)

The atomic unit of information in RDF are triples. Each triple consists of a subject, a predicate (also called property), and an object, see Figure 2.4. A set

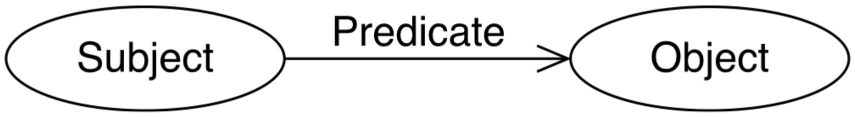


Figure 2.4.: RDF triple (Cyganiak, Wood, and Lanthaler, 2014)

of triples is called an RDF graph. RDF is a data modelling language and can be represented in many different file formats including XML (RDF/XML), N3 and Turtle. N3 and Turtle are file formats especially designed for human readability.

(Pan, 2009)

The fundamental concepts of RDF are resources, predicates, and statements. There are three kinds of nodes in an RDF graph:

- **Resource nodes**: Resources (also entities or subjects) are things that are described more detailed. Resources are named by URIs.
- **Literal nodes**: Are actual values, e.g., strings, integers, floats...
- **Blank nodes**: Are placeholder nodes (resources without URIrefs).

Consider a simple example:

```
("David Billington", http://www.mydomain.org/site-owner,
http://www.cit.gu.edu.au/db).
```

In this example the triple consists of:

Subject: **David Billington**

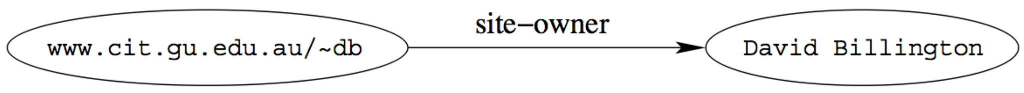


Figure 2.5.: RDF graph example (Antoniou and Harmelen, 2008)

Predicate: **http://www.mydomain.org/site-owner**

Object: **http://www.cit.gu.edu.au/db**

The RDF graph of this triple is shown in Figure 2.5.

(Antoniou and Harmelen, 2008)

RDF schema (RDFS) is used to define classes, resources, and properties which can then be used in RDF. RDFS is the first attempt to define a language which can be used to define an ontology. RDF defines the actual data and RDFS describes the structure and relationship of the data on a meta level. Similar to namespaces in XML, prefixes are used to avoid name clashes in RDF schema.

Consider a simple RDF schema that defines classes for animal, habitat and elephant.

Listing 2.2: Simple RDFS example

```
1. @prefix rdf:  <http://www.w3.org/1999/02/22-rdf-
   ↪ syntax-ns#>
2. @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#
   ↪ >
```

```
3. @prefix elp:  <http://example.org/Animal#>

4. elp:Animal rdf:type rdfs:Class .
5. elp:Habitat rdf:type rdfs:Class .
6. elp:Elephant rdf:type rdfs:Class ; rdfs:subClassOf
   ↪ elp:Animal .

7. elp:liveIn rdf:type rdf:Property ;
8. rdfs:domain elp:Animal ; rdfs:range elp:Habitat .
```

Line 1 to 3 define the prefixes. Line 4 defines a new class called "Animal", line 5 defines a new class called "Habitat" and line 6 defines a class "Elephant" which is a subclass of animal. Line 7 defines a property called "liveIn". Additionally, a domain and range is specified which can be used by the reasoning engine to classify subjects and objects that this property is used for. In this case subjects which use the "liveIn" property are classified as "Animal" ("rdfs:domain") and objects are classified as "Habitat" ("rdfs:range").

(Pan, 2009)

Ontologies As described at the beginning of this chapter, ontologies are a way to make knowledge explicit. Gruber, 1993 defines ontologies by:

“A specification of a representational vocabulary for a shared domain of discourse — definitions of classes, relations, functions, and other objects”

Ontologies are an important pillar of the semantic web as described by Berners-Lee, Hendler, and Lassila, 2001. Ontologies will be discussed further in the remainder of this chapter.

Logic, Proof, Trust, and Digital Signature These layers are summed up here because they are still being researched and additionally they are not directly relevant to this thesis. The Logic layer allows to encode application specific knowledge in the form of rules. The Proof layer executes these rules and together with the Trust layer, it is decided if these proofs or information can be trusted. Digital Signature is a tool to demonstrate the authenticity of digital information.

2.3.2. Web Ontology Language: OWL

Since RDFS turned out to be not expressive enough, OWL was invented. It can be used to describe far more complex ontologies. The current version of OWL is version 2, which differs to OWL 1 by further enhancements including new syntax, new profiles, some syntactic sugar, and various language enhancements.

Depending on the application scenario, OWL comes in several profiles each differing in complexity and expressivity:

- **OWL 2 Full**: This profile is the full version and every construct can be used as it allows even the most expressive constructs. This profile is undecidable and should be used carefully when using an automated reasoner.
- **OWL 2 EL**: This profile is very useful for a large amount of properties and/or classes. The reasoning problems can be performed in polynomial time with respect to the size of the ontology.
- **OWL 2 QL**: This profile is geared toward efficiently processing a large amount of instance data. Conjunctive query answering can be implemented on the top of relational database and a query written in OWL 2 QL can be fully rewritten as SQL query.
- **OWL 2 RL**: This profile is also optimised to run efficiently on the top of a relational database. OWL 2 RL reasoning systems can be implemented using rule-based reasoning engines.

(Motik et al., 2014 and Tester, 2014)

Appendix A shows how an example ontology can be created.

2.3.3. SPARQL

SPARQL is also one of the major components of the semantic web. SPARQL is a recursive acronym and stands for SPARQL Protocol and RDF Query Language. It is used to query data stored in RDF or compatible formats, e.g.

there exist converters for JSON, XML, ... It can be seen as the equivalent of SQL in relational databases.

As described in earlier chapters, RDF is not a data format but a data model where you express facts as triples. Out of these facts a RDF graph can be constructed. SPARQL is based on matching graph patterns.

Similar to SQL, SPARQL queries consist of three parts:

- **SELECT**: Specifies the projection, i.e. which data should be displayed and the order.
- **FROM**: Specifies the data source which should be queried (optional).
- **WHERE**: Imposes constraints using graph patterns to the data that should be displayed.

(Guarino, Oberle, and Staab, 2009a)

Consider a simple RDF graph (Turtle format):

Listing 2.3: Simple RDF graph

```
@prefix ad: <http://floriansalbrechter.com/addressbook#
    ↪ > .

ad:florian ad:tel "0650␣28␣52" .
ad:florian ad:email "flo@flo.com" .

ad:monica ad:tel "0123455" .
ad:monica ad:email "pupu@pupu.com" .
```

```

ad:bettina ad:tel "23343443" .
ad:bettina ad:email "pizza@bettina.com" .

ad:morten ad:tel "0815" .
ad:morten ad:email "morten@just.com" .
ad:morten ad:email "just@morten.com" .
```

Now simple queries can be executed against the RDF graph. The following listing shows an example of a SPARQL query which retrieves the email address of one individual. Identifiers starting with "?" are variables. In this example, email is a variable.

Listing 2.4: Simple SPARQL query

```
PREFIX ad: <http://floriansalbrechter.com/addressbook#>

SELECT ?email
WHERE
{
  ad:florian ad:email ?email .
}
```

The output of the query is as expected:

Listing 2.5: Output of the sample query

```
  ------------------------
```

```
| email          |
=================
| "flo@flo.com"  |
-----------------
```

(DuCharme, 2011)

2.3.4. Editors

Ontology editors are used to create, manipulate, and maintain ontologies. In this chapter, different ontology editors are described to give the reader an overview (descriptions taken from corresponding websites).

Protégé Is the most popular editor and de facto standard. It is a free and Open Source ontology editor developed by Stanford Medical Informatics Department at the Stanford University School of Medicine. There is a desktop and a web version available. The editor can be extended through a plugin architecture. It is very mature and well documented.

Webpage: http://protege.stanford.edu

TopBraid Composer TopBraid Composer by TopQuadrant is implemented as Eclipse plugin. There are three different licenses available: free, standard, and Maestro Edition. TopBraid Composer also includes a localhost Jetty web server for testing web services and applications.

Webpage: http://www.topquadrant.com/

OntoStudio OntoStudio supports comprehensive functions and intuitive ontology modelling. It also offers a graphical tool to easily connect and map databases and knowledge bases. Multiple collaborators can simultaneously develop using the OntoBroker Enhancement Collaboration server.

Webpage: http://www.semafora-systems.com/en/products/ontostudio/

3. Methodology

This chapter describes the process, steps, and iterations in which the application has been developed.

3.1. Stages

Initially, the research phases were defined. Figure 3.1 shows the stages of the thesis. Each stage will be described in further detail.

3.1.1. Literature review

In order to better understand the problem and its domain, an exhaustive literature review was carried out. The literature review was focusing on ultrasound, artificial intelligence, and ontology. More than 50 papers were reviewed and the most relevant were incorporated in the literature review

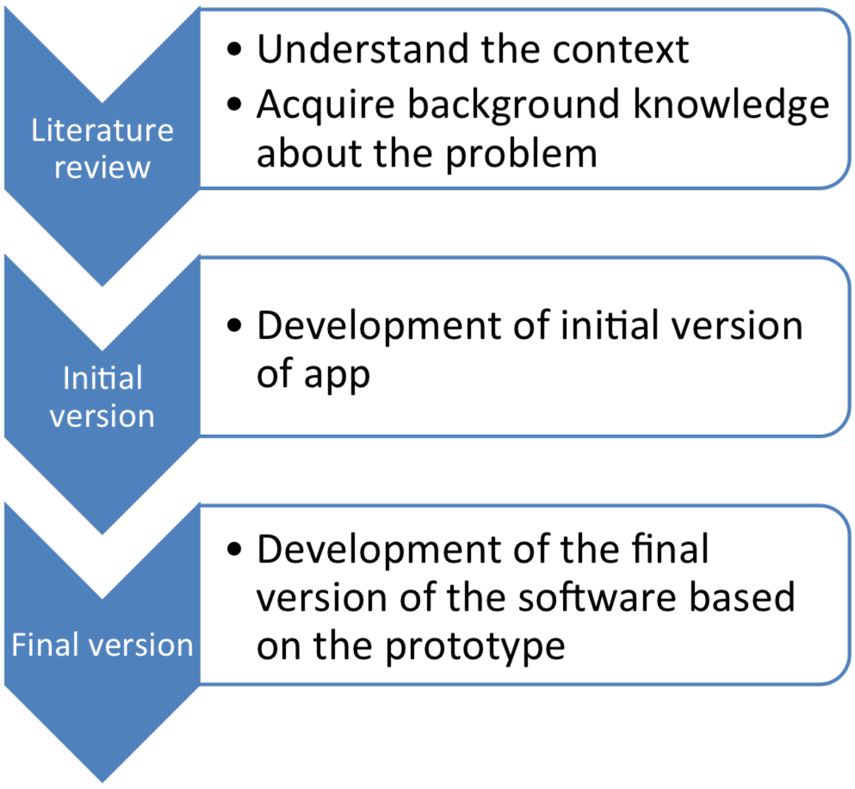

Literature review
Understand the context
Acquire background knowledge about the problem
Initial version
Development of initial version of app
Final version
Development of the final version of the software based on the prototype

Figure 3.1.: Stages

in order to understand and summarise the opinions and developments in the respective field. The following keywords have been used:

- **Ultrasound**: ultrasound, ultrasound introduction, ultrasound basic, ultrasound decision support system, ultrasound physics, and ultrasound images
- **Artificial Intelligence**: AI, clinical decision support system (CDSS), image analysis, AI definition, expert system and AIME artificial intelligence in medicine
- **Ontology**: ontology introduction, ontology examples, OWL, RDF, RDFS, Semantic Web, XML, namespaces, prefixes, SPARQL, Ontology editors, and ontology tutorial

The references of each interesting and relevant paper were followed in order to find more relevant papers. The main search engine used was Google (www.google.com) and Google scholar (scholar.google.com). Additionally scopus (www.scopus.com) and IEEE Xplore (ieeexplore.ieee.org) were used to search relevant papers.

Since this project also builds on my previous work (Salbrechter, 2013) it was used as a starting point and is referred multiple times throughout this thesis.

3.1.2. Initial version

The initial version of the example application is created. The details of the architecture and the technologies are figured out in this stage. Also the ontology is designed and implemented.

3.1.3. Final version

Building upon the experiences made with the initial version, the final version is created.

4. Application

In this chapter, we discuss the developed application, its architecture, its functionality, and all technologies used.

Disclaimer The created ontology is just a proof of concept, since the author has no professional background knowledge in medicine or ultrasound. If this application is created for a real life test, a proper ontology with domain experts needs to be created. Nevertheless, the simple ontology is good enough to demonstrate the idea and can be exchanged with a more sophisticated ontology easily at any time.

4.1. Features

This chapter describes the functionality of the developed application. The most important features of the final application are:

Web app In order to maximise portability, this application is created as a web application.

Ontology The application relies heavily on the quality of the ontology. The application was designed to enable the easy replacement or update of the underlying ontology.

Guidance The application is designed to guide the doctor through the diagnosis process.

Interactive The system is interactive and reacts immediately to user input.

Explanation The system gives feedback on how it came up with the suggested diagnoses.

4.2. Ontology

The ontology is a very crucial part of the application. The whole application revolves around the ontology and displays the knowledge encoded in the ontology. Additionally, the application lets the user query the ontology using a graphical user interface in a very intuitive way.

4.2.1. Design process

There are different opinions about the best approach to define an ontology. Two major discussion points are:

- **Bottom up vs top down**: Do you start by designing the most abstract levels of concepts first (top down) or do you start at a very low abstract level and build upon it?
- **Reuse existing ontologies vs create a new one**: Do you create a new ontology from scratch, extend or reuse existing ones?

Some of the already existing medical related (anatomy, diseases, ...) ontologies that influenced the design of the demo ontology include (descriptions taken from corresponding websites):

- **Human Phenotype Ontology (HPO)**: Aims to provide a standardised vocabulary of phenotypic abnormalities encountered in human disease. (http://www.human-phenotype-ontology.org/)
- **Ontobee**: A linked data server designed for ontologies. Ontobee is aimed to facilitate ontology data sharing, visualization, query, integration, and analysis. (http://www.ontobee.org/)
- **SUPERFAMILY**: is a database of structural and functional annotation for all proteins and genomes. (supfam.cs.bris.ac.uk/)

Since many ontologies are very specific they are very hard to read and grasp for non-experts.

The purpose of this thesis is not to create a perfect, ready to use ontology. Since the author is not a trained medical professional, it is impossible to create a final ontology. Instead, a demo ontology is created to illustrate how the application can be used in conjunction with an ontology. The ontology can later be replaced by a more elaborate one. Information and knowledge about gallbladders and diseases were taken from Salbrechter, 2013.

The ontology was created using Protégé Version 5.0.0 (Build beta-12) running on Mac OS X 10.9.3.

4.2.2. Ontology

This section describes the created ontology. The target is to create an ontology that is simple to understand on one side, yet at the same time illustrates the features of the developed application. The ontology can be replaced easily since it is loaded dynamically at run time.

The ontology IRI[1] is http://floriansalbrechter.com/ns.

Classes

In a high level view the ontology consists of two classes called Gallbladder-Observation and Disease, as illustrated in Figure 4.1.

[1]Internationalised Resource Identifier

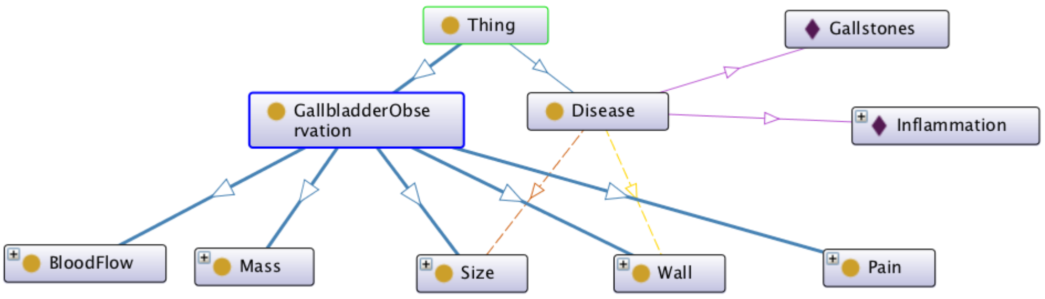


Figure 4.1.: Ontology overview

GallbladderObservation This class defines different kinds of observation that can be made on the gallbladder. Possible observations are:

- **BloodFlow**: This observation defines the blood flow through the organ which could help to narrow down the number of possible diseases.

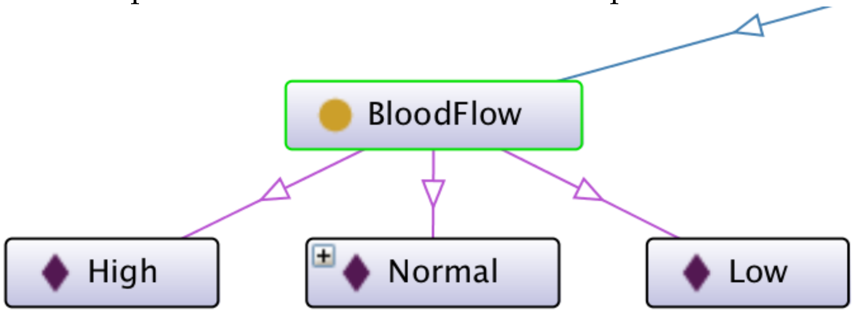


Figure 4.2.: Blood flow instances

  Instances of blood flow are:

  - High
  - Normal
  - Low

- **Mass**: This observation lets the user specify that there is a mass in the gallbladder which is visible on the ultrasound image.

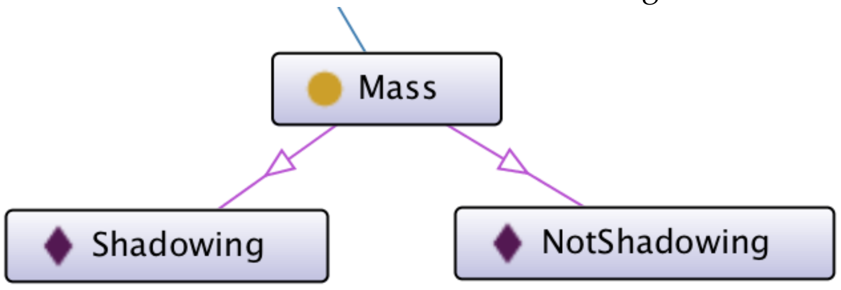


Figure 4.3.: Mass instances

Instances of mass are:

- Shadowing
- Not shadowing

- **Size**: This observation lets the user specify the size of the gallbladder based on the ultrasound image.

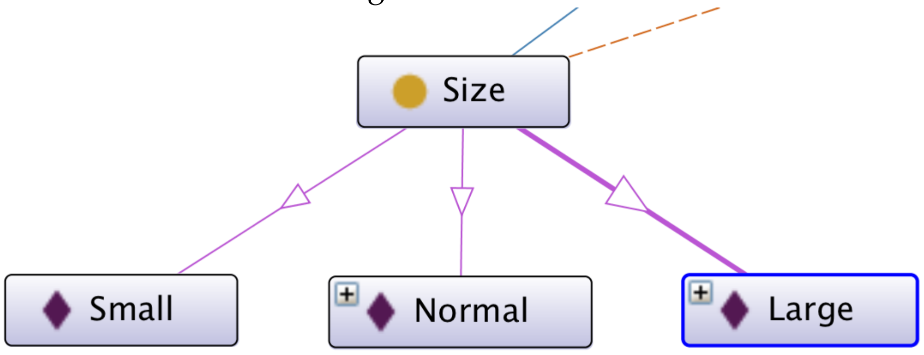


Figure 4.4.: Size instances

Instances of size are:

- Small
- Normal
- Large

- **Wall**: This observation lets the user specify the size of the gallbladder wall based on the ultrasound image.

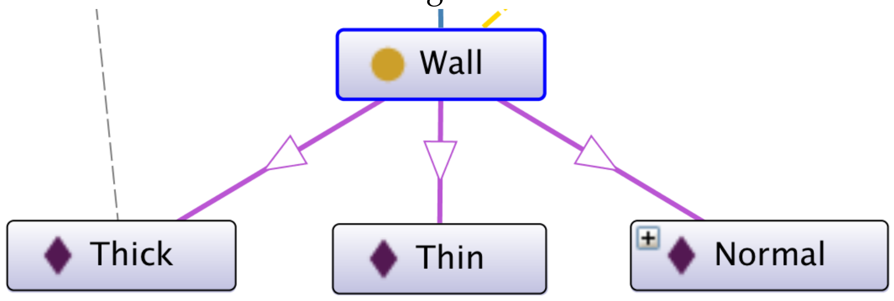


Figure 4.5.: Wall instances

  Instances of wall size are:

  – Thin
  – Normal
  – Thick

- **Pain**: This observation lets the user specify if and where pain is perceived.

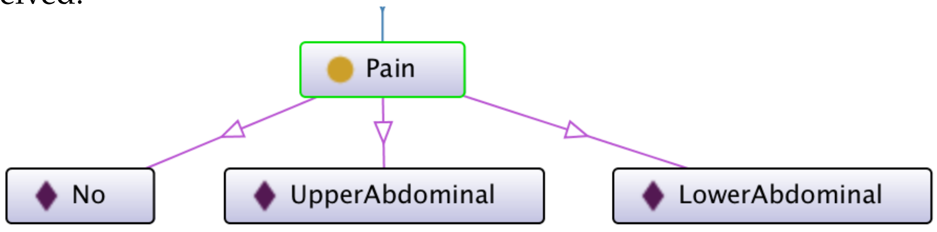


Figure 4.6.: Pain instances

  Instances of pain are:

  – No
  – Lower Abdominal
  – Upper Abdominal

Disease This class defines possible diseases using object properties. Object properties are:

- **hasMass**: This is used to link a disease to mass observations.
  - **Domain**: Disease
  - **Range**: Mass
- **hasSize**: This is used to link a disease to size observations.
  - **Domain**: Disease
  - **Range**: Size
- **hasWall**: This is used to link a disease to wall observations.
  - **Domain**: Disease
  - **Range**: Wall

Instances For demonstration purposes three diseases have been defined. As described earlier these are just based on the simplified view of a non-professional. In a real ontology more object properties might be needed but for the demonstration this is enough.

Disclaimer To create meaningful examples for this thesis, the following relationships between observations and diseases are extremely simplified to provide a better understanding.

- **Cancer**: Cancer is the simplest disease in this ontology having only one object property.

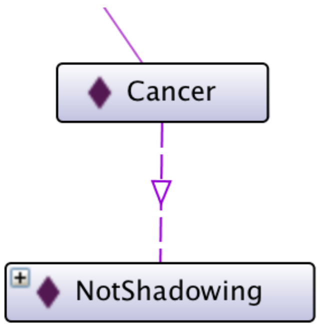


Figure 4.7.: Cancer instance

- **hasMass NotShadowing**: A cancer is defined by having a non-shadowing mass.

- **Inflammation**: Inflammation of the gallbladder is a bit more complex having two object properties.

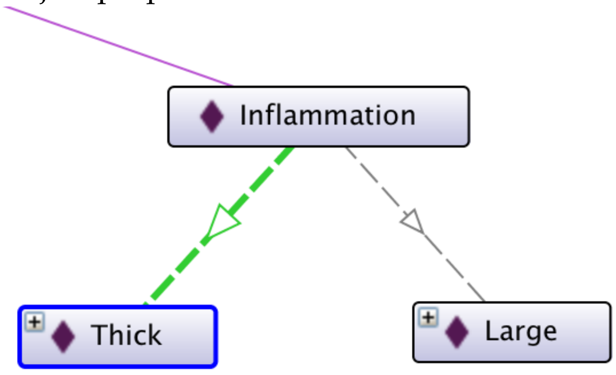


Figure 4.8.: Inflammation instance

  - **hasSize Large**: A large size can be an indicator for an inflammation of the gallbladder.
  - **hasWall Thick**: A thick gallbladder wall can be an indicator for an inflammation of the gallbladder.

- **Gallstones**: Gallstones in the gallbladder have the most object properties.

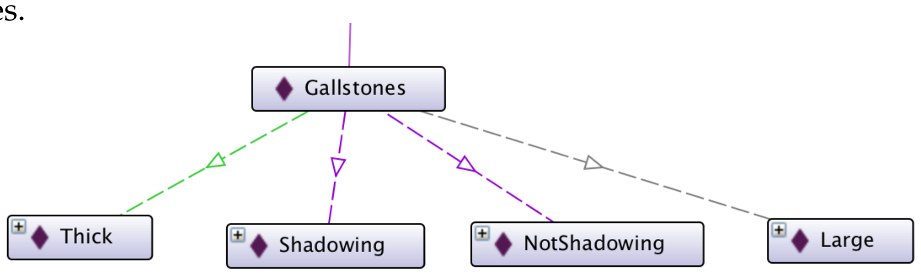


Figure 4.9.: Gallstone instance

  - **hasMass Shadowing**: A gallstone can usually be seen on the ultrasound image as a shadowing mass.
  - **hasMass NotShadowing**: In rare cases gallstones can also be non-shadowing. This example was picked to illustrate that instances can have multiple object properties of the same type.
  - **hasWall Thick**: A thick gallbladder wall can be an indicator for a gallstone.
  - **hasSize Large**: A large gallbladder can be an indicator for a gallstone.

Data Properties

No data properties are used in this ontology because all relations reference resources or instances.

The full ontology is available in Appendix B.

4.3. Web Application

In this chapter the web application is described.

4.3.1. Architecture

The architecture of the developed application is shown in Figure 4.10.

Once the ultrasound image is taken it is uploaded to the webserver where the user can then interact with it. jQuery is used to facilitate JavaScript development.

Webserver The webserver plays a central role in the architecture. It serves websites written in HTML, CSS and JavaScript to the user which can then interact with the websites.

Client side ontology libraries were considered but were not mature enough. The two ontology javascript libraries considered are:

- **owlreasoner**: Very basic functionality and the SPARQL query processing is implemented only very basic and at the time of writing, namespace support is buggy.
- **jOWL**: This library only supports OWL 1 and there is no recent activity on this project. SPARQL-DL is used as a query language.

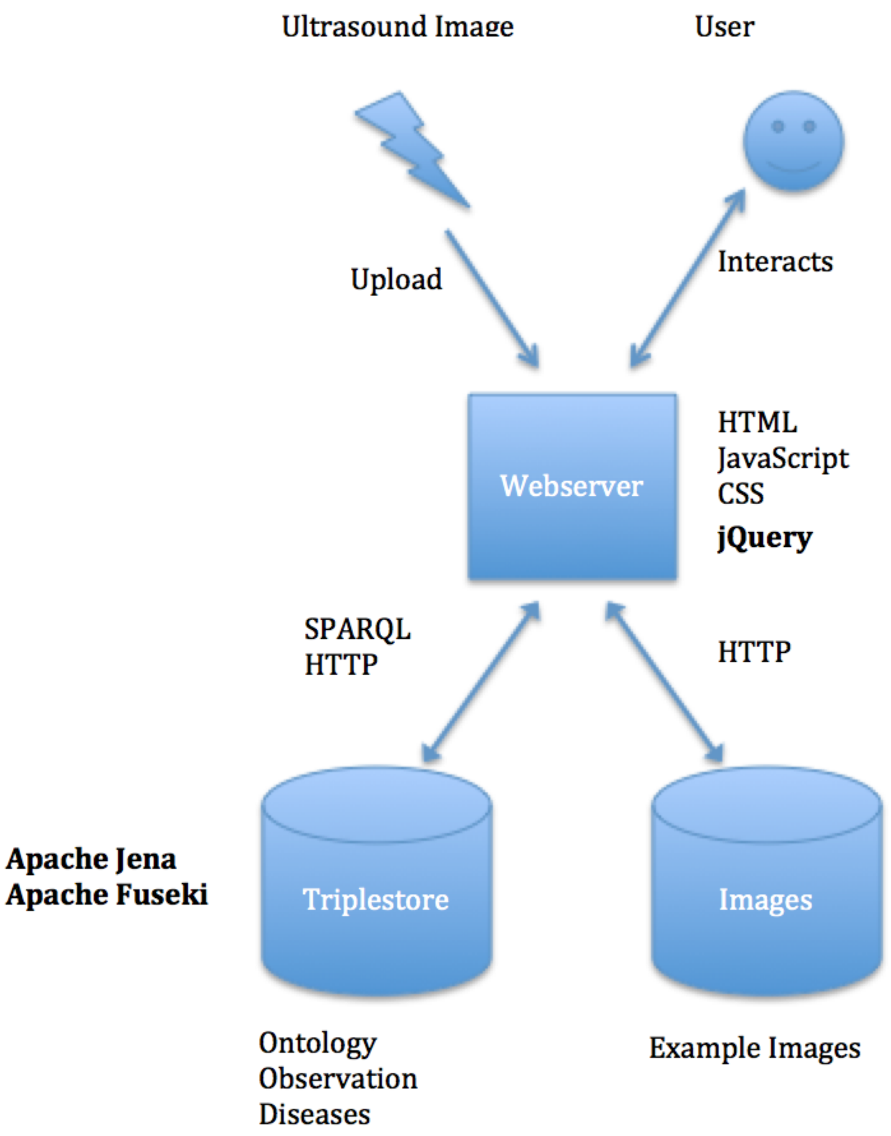


Figure 4.10.: Architecture overview

The ontologies reside in a Triplestore and are queried with JavaScript and SPARQL over HTTP using AJAX requests. The application UI is dynamically loaded based on the ontology.

Triplestore Apache Jena is an open source framework that provides a multitude of APIs to work with RDF data. Fuseki builds on top of Apache Jena and provides a REST-style interface that also supports SPARQL Queries using the SPARQL protocol over HTTP.

Images Images that help the user to make a decision are stored in a standard filesystem.

4.3.2. Usage

This section provides a walkthrough of the components, shows how this application can be used, and the number and order of necessary steps.

Step 1: Healthcare professional performs ultrasound examination The healthcare professional chooses to scan a specific organ using his/her ultrasound equipment.

Step 2: Upload ultrasound image The healthcare professional uploads the image to the server and starts the application.

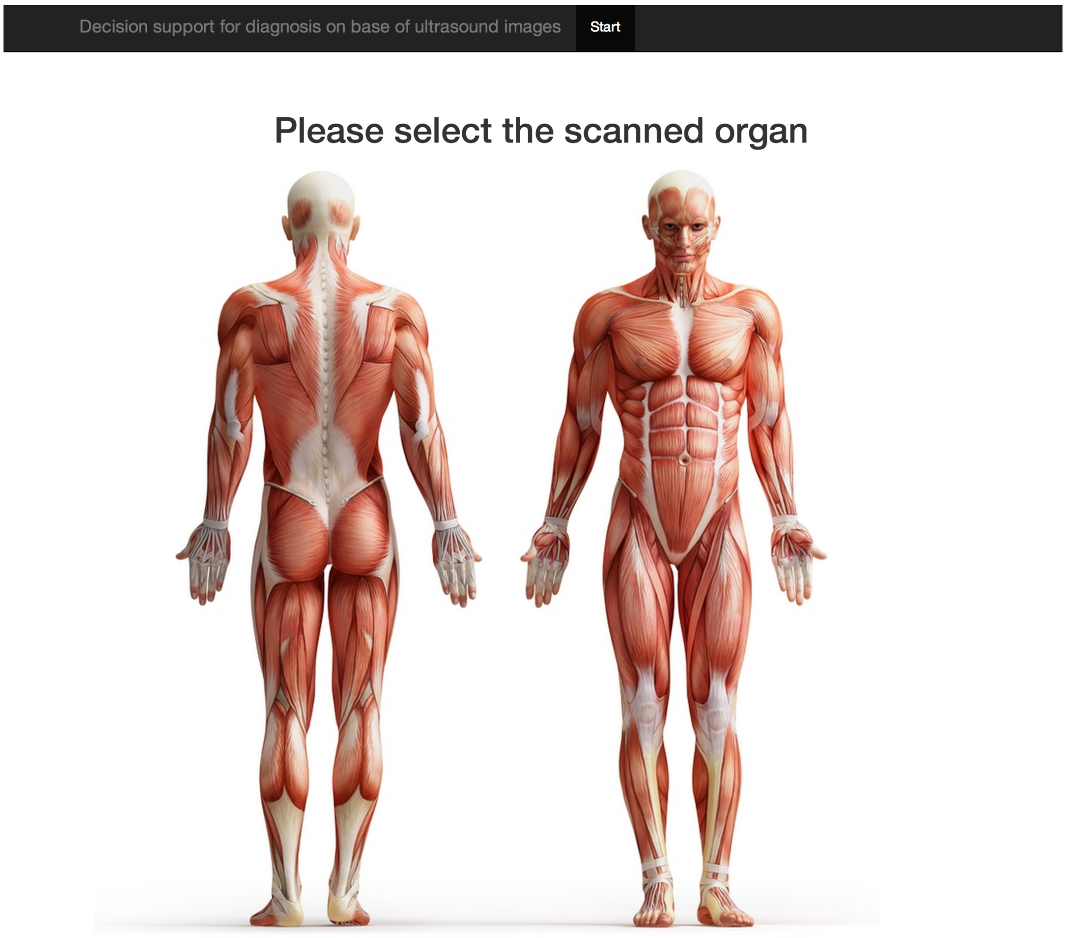

Decision support for diagnosis on base of ultrasound images
Start
Please select the scanned organ

Figure 4.11.: Start screen

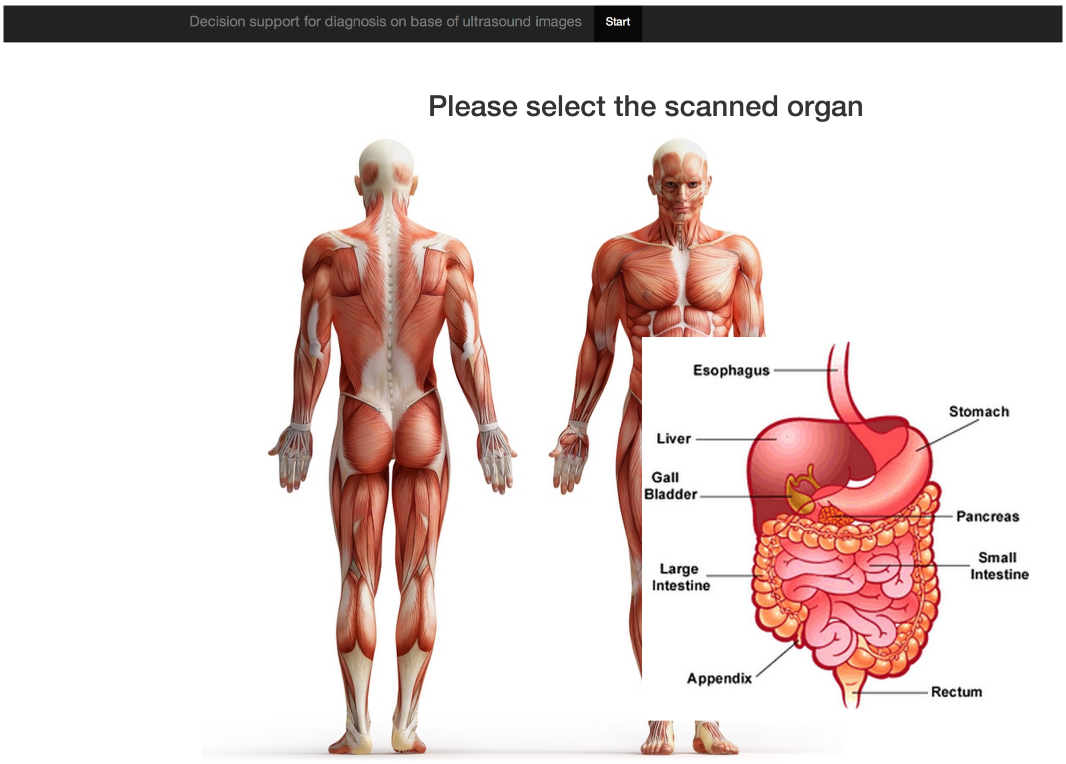
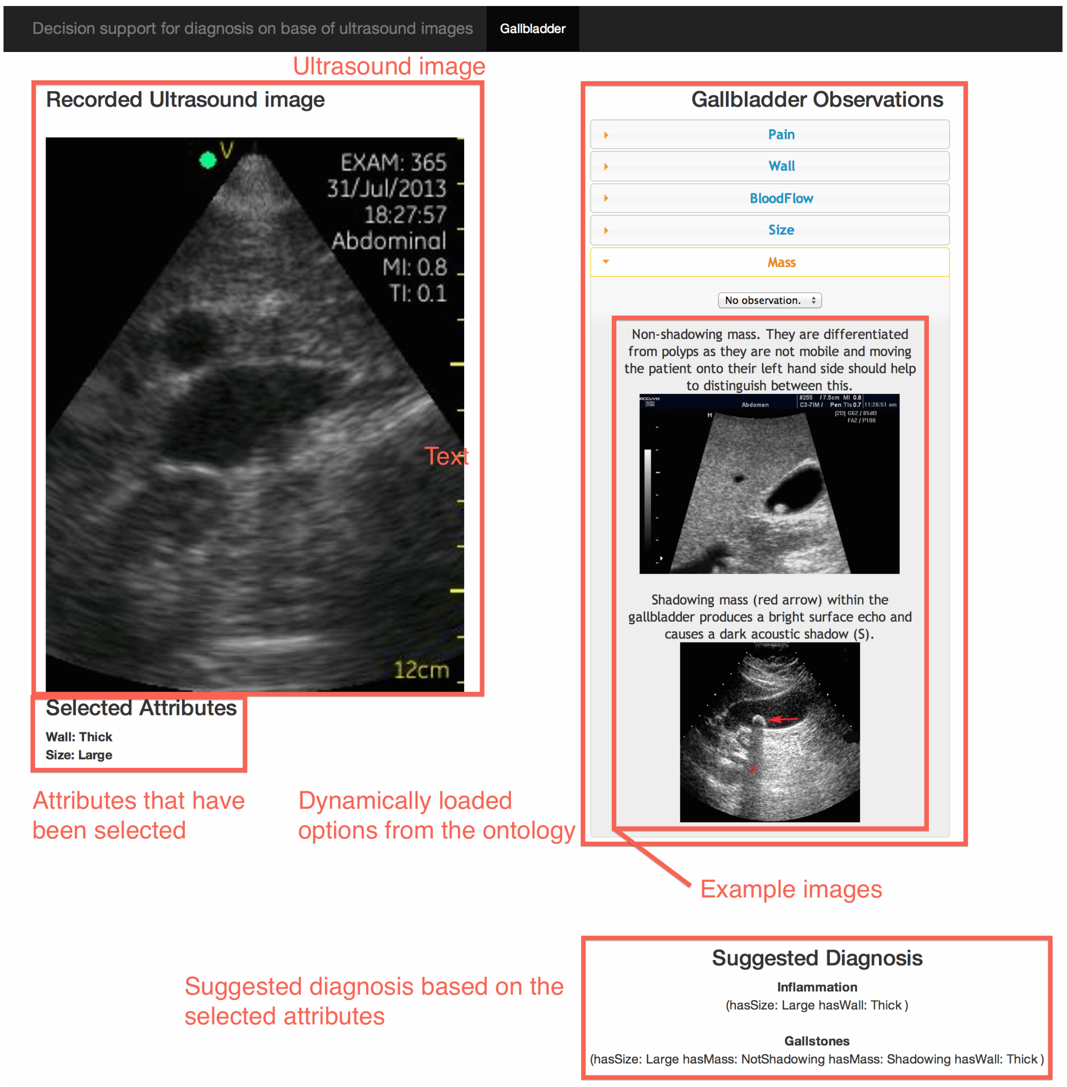

Decision support for diagnosis on base of ultrasound images
Start
Please select the scanned organ
Esophagus
Stomach
Liver
Gall Bladder
Pancreas
Large Intestine
Small Intestine
Appendix
Rectum

Figure 4.12.: Detailed organ screen



Non-shadowing mass. They are differentiated from polyps as they are not mobile and moving the patient onto their left hand side should help to distinguish between this.

Shadowing mass (red arrow) within the gallbladder produces a bright surface echo and causes a dark acoustic shadow (S).

Inflammation
(hasSize: Large hasWall: Thick)

Gallstones
(hasSize: Large hasMass: NotShadowing hasMass: Shadowing hasWall: Thick)

Figure 4.13.: Diagnosis screen

Step 3: Start the application and choose the scanned organ Figure 4.11 shows the start screen. The healthcare professional selects the region of the scanned organ. Once the healthcare professional hovers over areas, a more detailed view of that part is shown, see Figure 4.12.

Step 4: Examine the uploaded ultrasound image and add observations to receive suggestions On this screen, the healthcare professional sees the uploaded ultrasound image (top left) and can add attributes or annotations based on the ontology used, see Figure 4.13. On the right there is a list of possible observations which is loaded dynamically from the ontology (top right). Example images can be added to help the user decide which option to select (right side in the "Mass" tab). All the selected attributes are shown in the "Selected Attributes" section (bottom right). Based on the selection from the user the suggested diagnosis are displayed in "Suggested Diagnosis" (bottom right). In addition to the diagnosis, the reason why it has been selected as a potential diagnosis is displayed in brackets (bottom right inside "Suggested Diagnosis"). The suggested diagnosis is dynamically updated when the observations are changed, and each diagnosis has a justification why it has been added to the list attached to it.

4.3.3. Dynamic UI

As described above, the application UI adapts to the ontology. This section gives a more detailed description how and where this is done. In our exam-

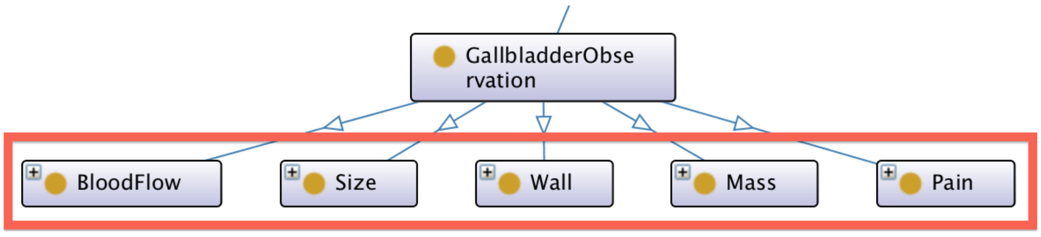


Figure 4.14.: GallbladderObservation

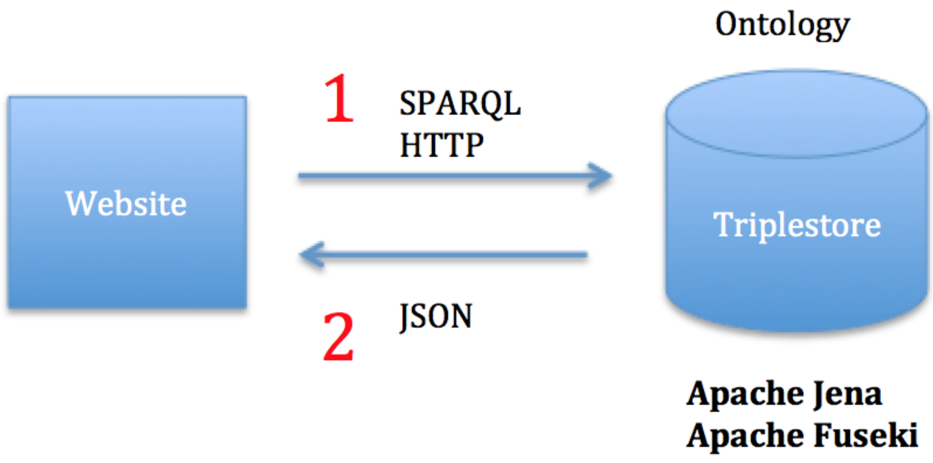


Figure 4.15.: Query architecture

ple, the GallbladderObservations are loaded dynamically. An overview of the GallbladderObservations is shown in Figure 4.14. The different observations are marked by the red rectangle.

The data is queried in SPARQL using AJAX and the response is in the JSON format. Figure 4.15 shows the query architecture. The first step is to query the Triplestore for data (1 in Figure 4.15) which then replies with JSON (2 in Figure 4.15).

The SPARQL query asks for all the subclasses of a GallbladderObservation

Listing 4.1: SPARQL query for GallbladderObservations

```
PREFIX fs: <http://floriansalbrechter.com/ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?subject
  WHERE { ?subject rdfs:subClassOf
     ↪ fs:GallbladderObservation }
```

The answer to the query are all the subclasses of GallbladderObservation and the answer format is JSON:

Listing 4.2: JSON reply for GallbladderObservations

```
{
  "head": {
    "vars": [ "subject" ]
  } ,
  "results": {
    "bindings": [
      {
        "subject": { "type": "uri" , "value": "http:
           ↪ //floriansalbrechter.com/ns#Pain" }
      } ,
      {
        "subject": { "type": "uri" , "value": "http:
           ↪ //floriansalbrechter.com/ns#Wall" }
      } ,
      {
        "subject": { "type": "uri" , "value": "http:
           ↪ //floriansalbrechter.com/ns#BloodFlow"
           ↪ }
      } ,
      {
        "subject": { "type": "uri" , "value": "http:
           ↪ //floriansalbrechter.com/ns#Size" }
      } ,
      {
        "subject": { "type": "uri" , "value": "http:
           ↪ //floriansalbrechter.com/ns#Mass" }
      }
    ]
  }
}
```

After removing the namespace the web application knows that it needs to display UI controls for Pain, Wall, BloodFlow, Size, and Mass. Figure 4.16 shows the conversion process between the ontology on the web server (top) to the HTML UI elements on the web page (bottom).

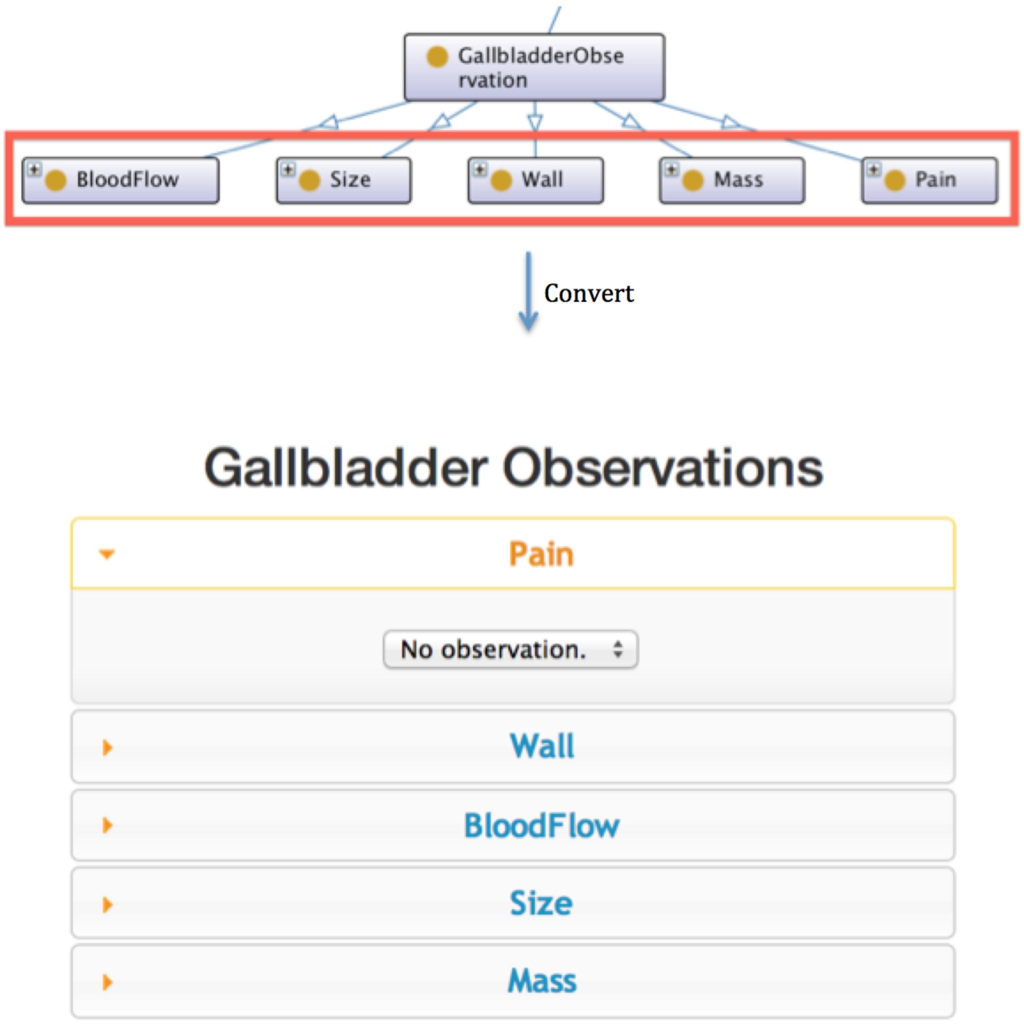


Figure 4.16.: Conversion process

Once the observations are loaded, the next queries aim to find the possible values for each observation which correspond to instances of the classes shown above (Pain, Wall, BloodFlow, Size, and Mass). The web application queries the triplestore for each observation to find the possible selections. The following listing shows an example of such a query for the "Pain" class:

Listing 4.3: SPARQL query for Pain

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns
    ↪ #>
PREFIX fs: <http://floriansalbrechter.com/ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?subject
  WHERE { ?subject rdf:type fs:Pain }
```

The JSON reply to this SPARQL query looks like:

Listing 4.4: JSON reply for Pain

```
{
  "head": {
    "vars": [ "subject" ]
  } ,
  "results": {
    "bindings": [
      {
        "subject": { "type": "uri" , "value": "http://
           ↪ floriansalbrechter.com/ns#No" }
      } ,
      {
        "subject": { "type": "uri" , "value": "http://
           ↪ floriansalbrechter.com/ns#LowerAbdominal"
           ↪  }
      } ,
      {
        "subject": { "type": "uri" , "value": "http://
           ↪ floriansalbrechter.com/ns#UpperAbdominal"
           ↪  }
      }
    ]
  }
}
```

Figure 4.17 shows the conversion process.

The supplemental reference images are loaded from the filesystem and can be added to any observation, see Figure 4.18. In a real application, these images need to be carefully selected and annotated by experts.

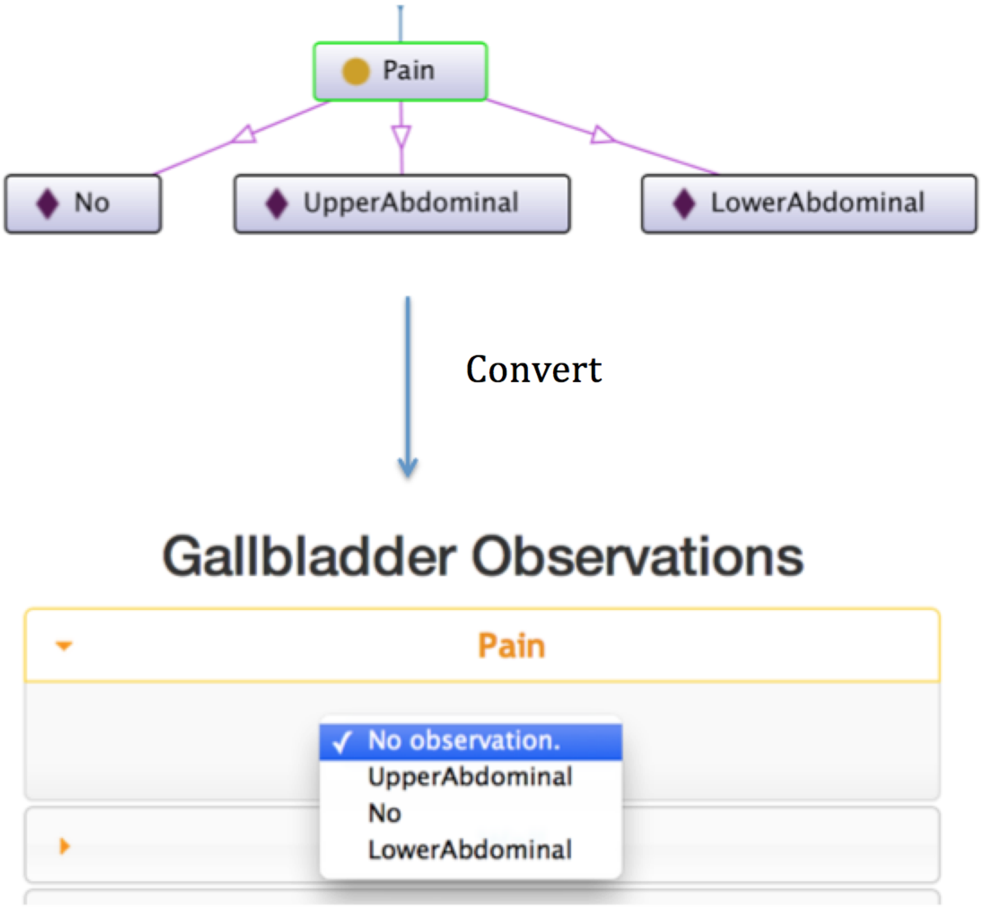
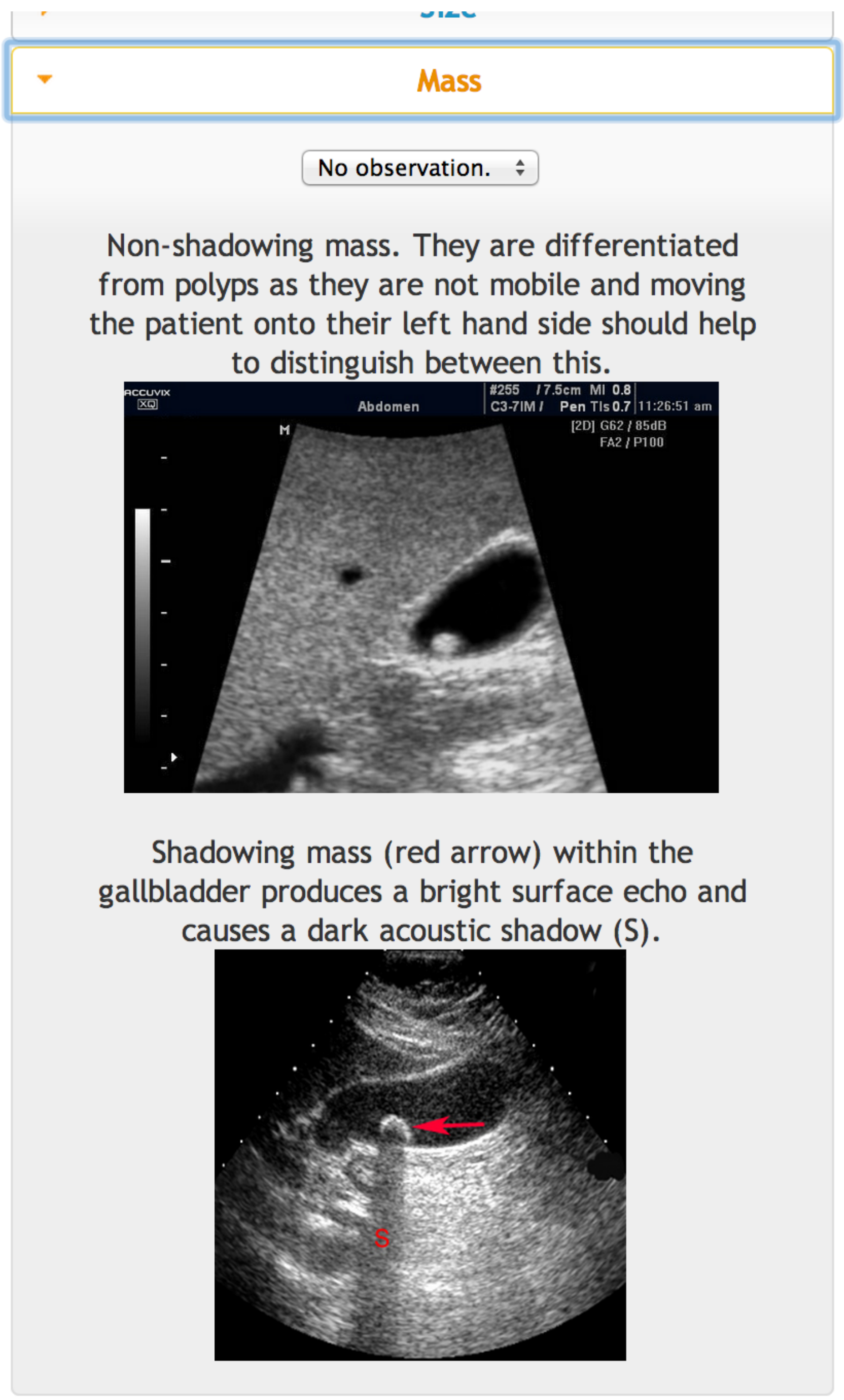


Figure 4.17.: Conversion process for instances



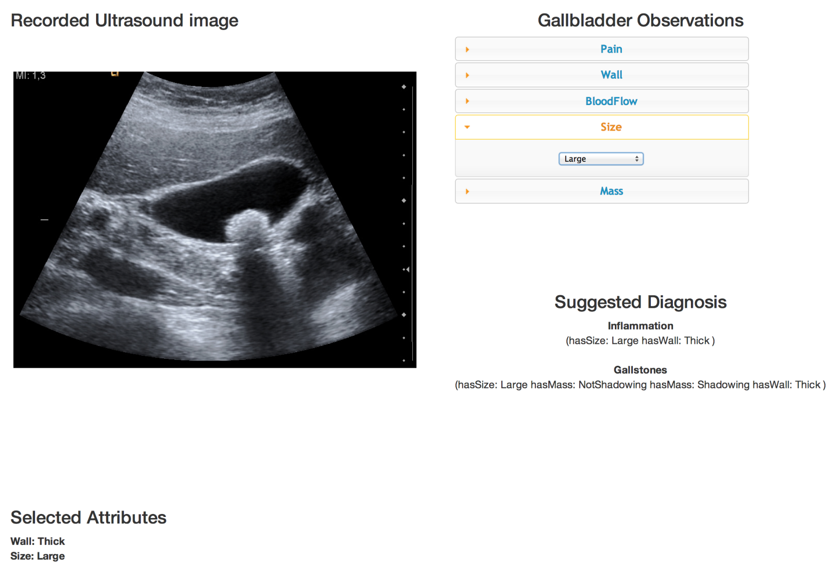
Mass

No observation.

Non-shadowing mass. They are differentiated from polyps as they are not mobile and moving the patient onto their left hand side should help to distinguish between this.

Shadowing mass (red arrow) within the gallbladder produces a bright surface echo and causes a dark acoustic shadow (S).

Figure 4.18.: Help example



Size: Large

Suggested Diagnosis

Inflammation
(hasSize: Large hasWall: Thick)

Gallstones
(hasSize: Large hasMass: NotShadowing hasMass: Shadowing hasWall: Thick)

Selected Attributes

Wall: Thick
Size: Large

Figure 4.19.: Example

4.3.4. Example

This chapter shows an example of how this application can be used. Imagine a patient having pain in his/her abdomen. The physician decides to scan the lower abdomen, especially the gallbladder. Once the image is uploaded, the application is started and the physician enters the observations. The observations are:

- Thick wall
- Large gallbladder

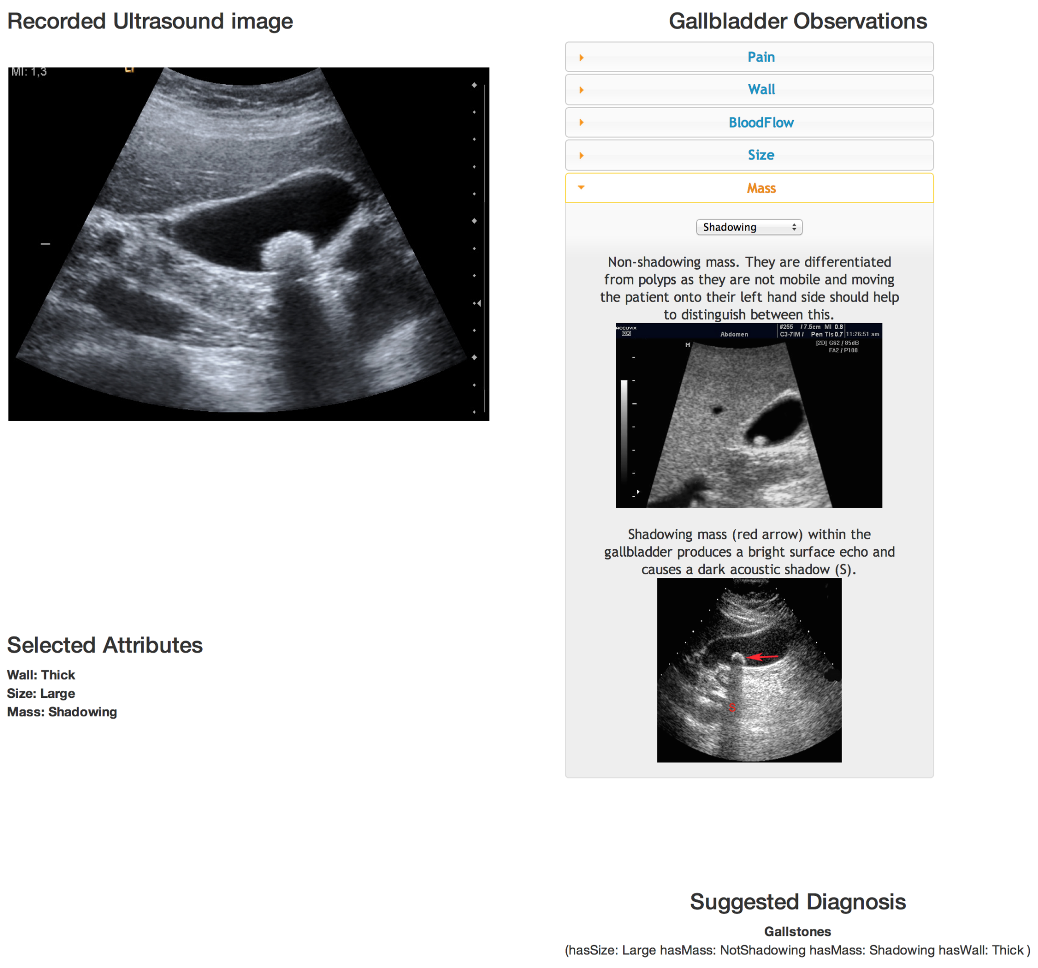
After entering the observations, the application looks like 4.19.

Now the physicians sees that there are two possible diseases: Inflammation and Gallstones. Now there are two possible diseases satisfying the specified symptoms:

- After looking through the observation possibilities, he/she sees the mass option and looks at the help. He/she then recognises a mass in the gallbladder and sees that it is shadowing.
- He/She looks at the suggested diagnosis and sees that he could narrow down the diagnosis by checking if there is a mass present in the gallbladder. He/she sees this information by looking in the explanation part in the brackets. After looking at the ultrasound image again and reading the help for masses, he/she sees a shadowing mass.

After adding an observation about a shadowing mass, the most likely diagnosis is a Gallstone, see Figure 4.20 [2].

[2]Ultrasound image from http://upload.wikimedia.org/wikipedia/commons/



Recorded Ultrasound image

Gallbladder Observations

Mass: Shadowing

Non-shadowing mass. They are differentiated from polyps as they are not mobile and moving the patient onto their left hand side should help to distinguish between this.

Shadowing mass (red arrow) within the gallbladder produces a bright surface echo and causes a dark acoustic shadow (S).

Selected Attributes

Wall: Thick
Size: Large
Mass: Shadowing

Suggested Diagnosis

Gallstones
(hasSize: Large hasMass: NotShadowing hasMass: Shadowing hasWall: Thick)

Figure 4.20.: Example

5. Conclusion

This work shows how ontologies can be used in diagnosis decision support systems in the domain of ultrasound images. An example ontology has been developed which focuses on the gallbladder. The application has been designed to be able to modify or replace the ontology easily by employing a dynamic visualisation process. The ontology provides a link between observations on ultrasound images and potential diseases. Ultrasound images can be annotated with observations. Based on the ontology, a list of potential diseases is shown which can be modified in an interactive way.

The ontology creation is a very hard problem which can only be solved by a team of knowledge engineers, software engineers, and physicians. Medical decision support systems are a good example of interdisciplinary medical IT systems. The knowledge engineers in cooperation with the physicians are needed to create a knowledge base. Software developers are needed to implement the application. The created example ontology works well with simple cases, but will need further improvements if it gets bigger. In real life, with a vast amount of diseases they will not be as easy to distinguish. The current application takes a very simple approach on filtering and ordering the results. This will need to be expanded. The loading of the structure, possible observations and values at run time make the application very dynamic.

The annotation approach, where observations are added to an ultrasound image, is very interesting and enables to tag and store images in a new way. Exploiting these annotations in future applications will make it easier to find example images and similar images.

There are many advantages of enabling a wider group of people to perform simple ultrasound scans. In poor countries where proper ultrasound training can not be provided for cost reasons, it can save lives. Even in developed countries, it can save costs and improve patient satisfaction by streamlining ultrasound examination workflows. The reference image approach can be extended in order to develop an educational application. This application can be used in combination with educational videos to bring ultrasound training to a wider audience in a very effective way.

Clinical decision support systems are by no means designed or intended to replace the physician. Instead they try to assist the physician as much as possible in order to draw his/her attention to special facts or cases.

5.1. Limitations

This thesis focused on one particular condition to showcase the concept of a diagnosis decision support system (gallstones). A medical study has not been conducted and medical experiments have not been performed. This could be subject to future work.

6. Future Work

The author sees two big fields where future work can be conducted:

- **Ontology creation**
- **Automatic feature extraction**

The ontology creation is a very challenging process. The ontology is the base of the rest of the application, so a weakness there has big impacts on every other component. Future work could investigate if it is possible to create a crowd sourced diagnosis network ontology similar to Wikipedia. Instead of using an offline ontology, a central ontology which is constantly updated could be used.

Another research project could examine how the already existing ontologies could be combined. There are some ontologies that focus mainly on anatomical features and some other ontologies that describe diseases. A study could be conducted to see if and how these two ontologies could be combined or mapped to create a new ontology combining anatomical features, observations, and symptoms and link them with diseases.

Ultrasound decision support systems could be extended by automatic processing of images and automatic feature detection, and extraction. Ultrasound images could be analysed automatically for common diseases and other abnormalities. As a first step, regions of interest could then be automatically marked to draw the attention of the healthcare specialist to certain parts.

A research project could evaluate how decision support tools can be used for educational purposes. Applications like the one introduced in this thesis could help to teach ultrasound scanning skills and provide training examples.

Appendix

Appendix A.

Creating a simple ontology

In this chapter, a simple ontology is created using Protégé[1], which describes the concepts of person (pilot and passenger), and plane (jet and turboprop). Additionally, the use of disjoint classes, object properties, data properties, ranges, and domains is illustrated.

Note: This example has been created using Protégé Version 5.0.0 (Build beta-12) running on Mac OS X 10.9.3.

The final model should look like Figure A.1.

In this simple model a person can either be a pilot or a passenger. A plane can either be a jet or turboprop. A pilot can fly planes. A passenger travels by planes.

[1]http://protege.stanford.edu

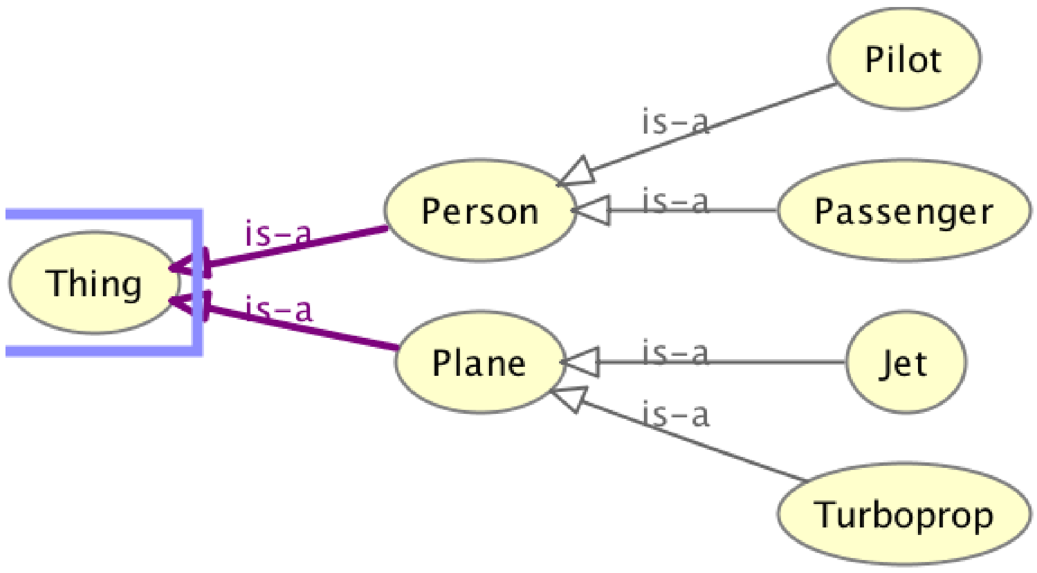


Figure A.1.: Example Ontology

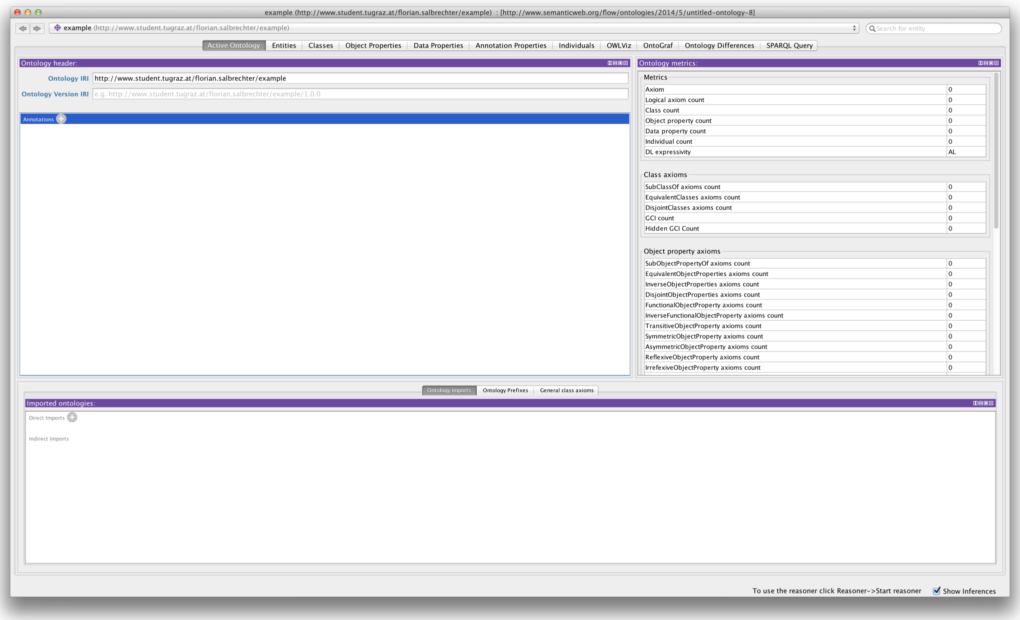
Step 1: Opening the application and creating a new ontology Once the application is started the first step is to create a new ontology. For the Ontology IRI[2] we enter:

http://www.student.tugraz.at/florian.salbrechter/example

We save the ontology under any name, see Figure A.2.

Step 2: Creating the classes The next step is to create the classes mentioned in the beginning. Switch to the **Classes** tab and notice that there is already a class defined. **Thing** is the superclass of all classes. Create a new class for **Person**, see Figure A.3.

[2]An IRI is similar to an URI but using unicode encoding.



Ontology IRI: http://www.student.tugraz.at/florian.salbrechter/example

Ontology Version IRI: e.g. http://www.student.tugraz.at/florian.salbrechter/example/1.0.0

Metrics

Axiom	0
Logical axiom count	0
Class count	0
Object property count	0
Data property count	0
Individual count	0
DL expressivity	AL

Class axioms

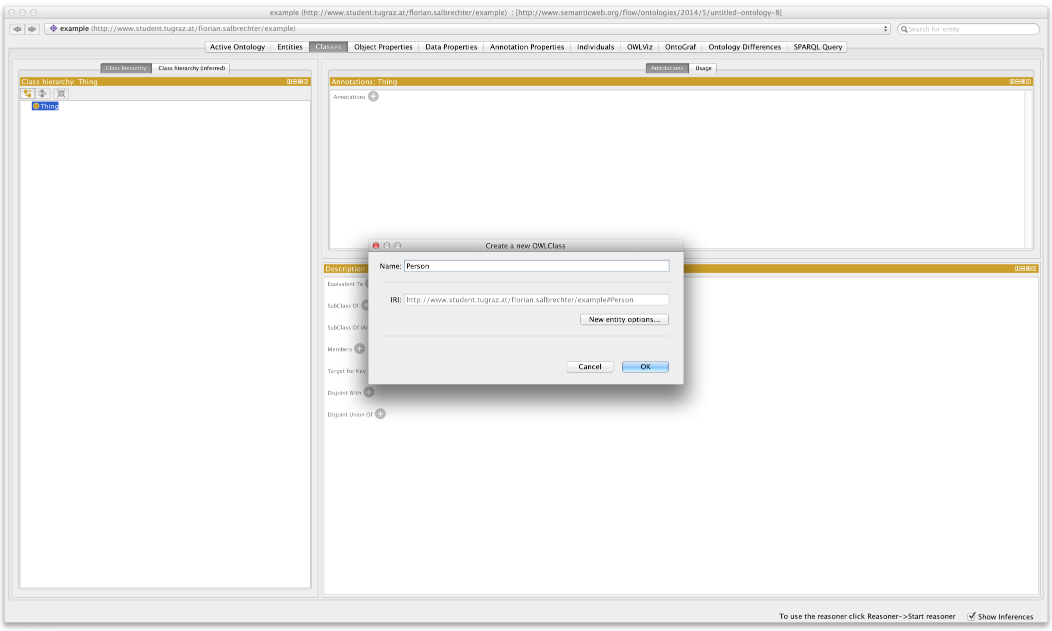
SubClassOf axioms count	0
EquivalentClasses axioms count	0
DisjointClasses axioms count	0
GCI count	0
Hidden GCI Count	0

Object property axioms

SubObjectPropertyOf axioms count	0
EquivalentObjectProperties axioms count	0
InverseObjectProperties axioms count	0
DisjointObjectProperties axioms count	0
FunctionalObjectProperty axioms count	0
InverseFunctionalObjectProperty axioms count	0
TransitiveObjectProperty axioms count	0
SymmetricObjectProperty axioms count	0
AsymmetricObjectProperty axioms count	0
ReflexiveObjectProperty axioms count	0
IrrefexiveObjectProperty axioms count	0

To use the reasoner click Reasoner->Start reasoner

Figure A.2.: Opening screen of Protégé



Name: Person

IRI: http://www.student.tugraz.at/florian.salbrechter/example#Person

Figure A.3.: Creating a new class dialog

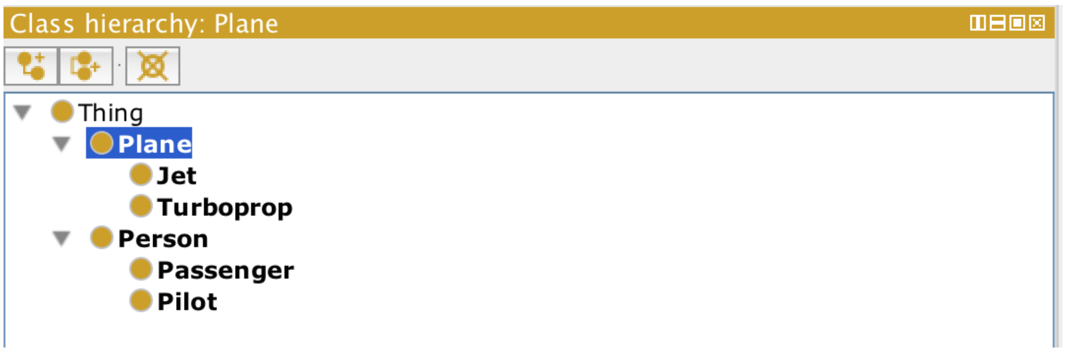


Figure A.4.: Class hierarchy

Additionally, create subclasses of **Person** called **Pilot** and **Passenger**. Create another class called **Plane** and subclasses for **Jet** and **Turboprop**. The final class hierarchy should look like Figure A.4.

Step 3: Making classes disjoint The next step is to make the classes disjoint. This indicates that an individual (also called instance, depending on the terminology used) can only be in one class. This prevents the inference engine from wrong classifications, e.g. classifying a plane as both jet and turboprop. This step adds metadata and additional information about the relationships between entities to the ontology. To mark classes as disjoint, click on *Disjoint With* and select all the classes that should be disjoint. In our example we select all the other classes; see Figure A.5.

Step 4: Defining object properties The next step is to define object properties, which defines relationships between classes. An object property relationship has a domain and a range. The domain defines the type of the

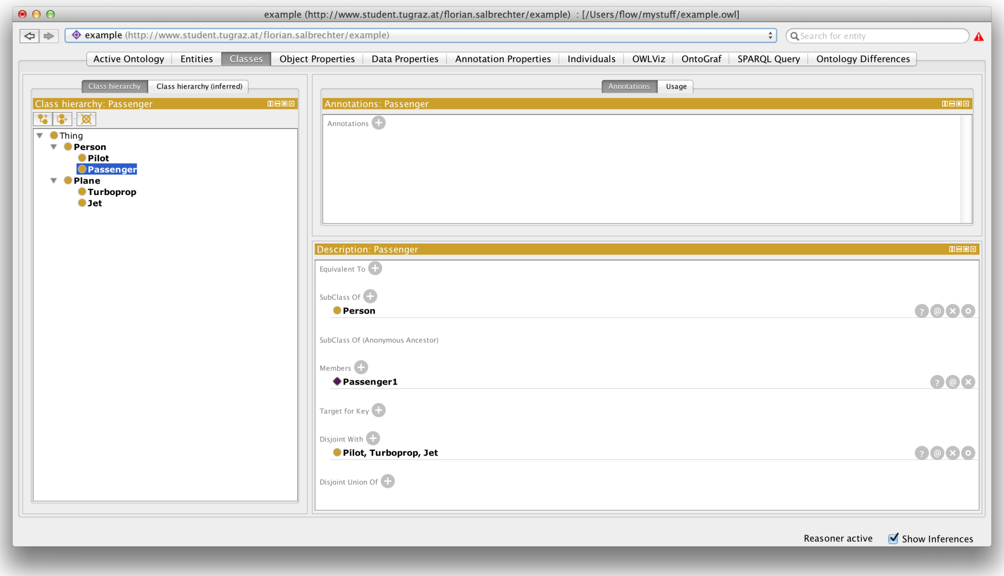


Figure A.5.: Disjoint classes

class where the relationship can “start” and the range defines the type of the class where the relationship “ends”. This information is used by the reasoner. In our example, we have two object properties.

- flies: A pilot flies a plane. Domain: **Pilot**, Range: **Plane**.
- travelsBy: A passenger travels by a plane. Domain: **Passenger**, Range: **Plane**.

Object properties are defined by switching to the **Object Properties** tab and adding a subclass property to *topObjectProperty*. Note that there can be a hierarchy of properties. Figure A.6 shows the defined object properties.

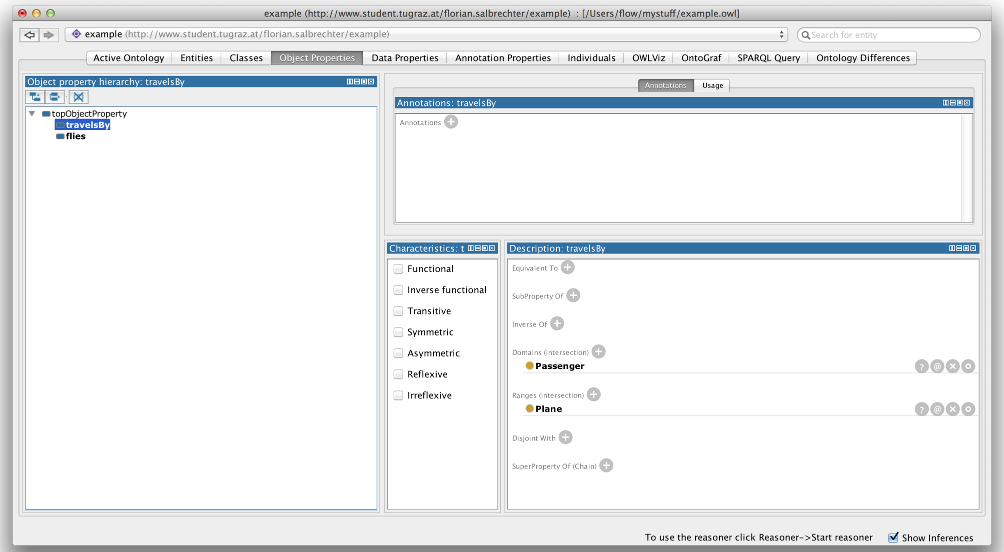


Figure A.6.: Object properties

Step 5: Defining data properties In the next step, a couple of simple data properties are added. In contrast to object properties that define relationships between instances, data properties define relationships between instances and data values, e.g. strings, integers, floats. Data properties are defined by switching to the **Data Properties** tab and adding a subclass property to *topDataProperty*. The concepts of domain and range apply analogously to data properties. Domain in this case is the type of class for which the data property should be defined. Range is the data type of the value of the data property.

We will add the following data properties for illustration:

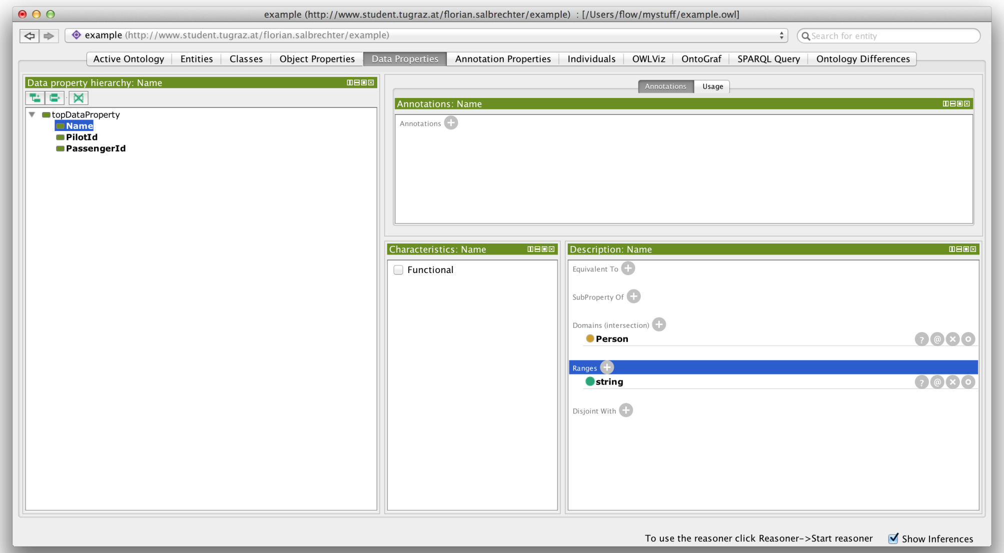


Figure A.7.: Data properties

- Passenger: **PassengerId** as **integer**
- Pilot: **PilotId** as **integer**
- Person: **Name** as **string**

Figure A.7 shows the defined data properties.

Step 6: Adding individuals The next step is to add some individuals or instances. To add individuals, switch to the **Individuals** tab. On the left side ("Class hierarchy") select the class of the new instance. Then click add in the "Members List" tab to add instances. To add data or object properties, select the new individual and click on the "Property assertions" tab on the right side. If there are any inconsistencies in the ontology, the reasoner will

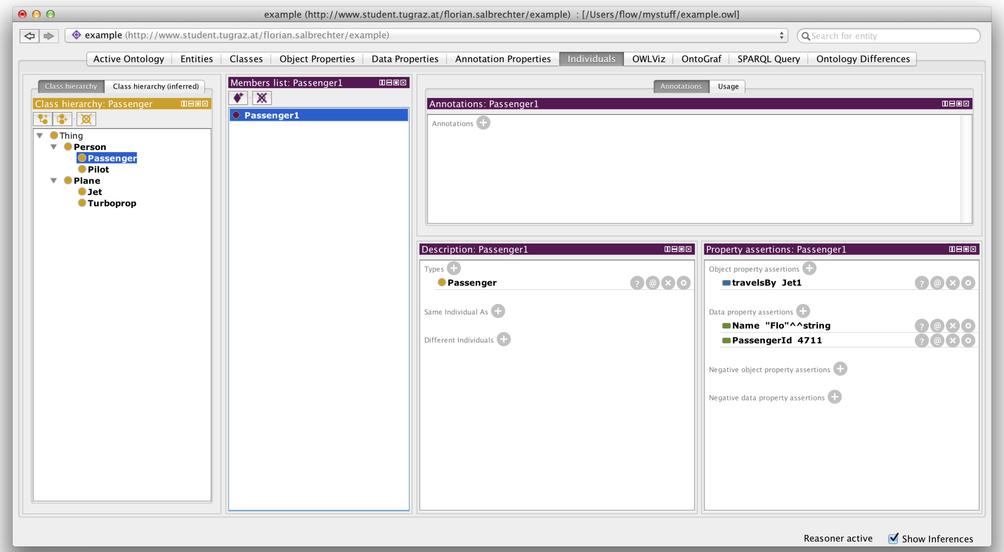


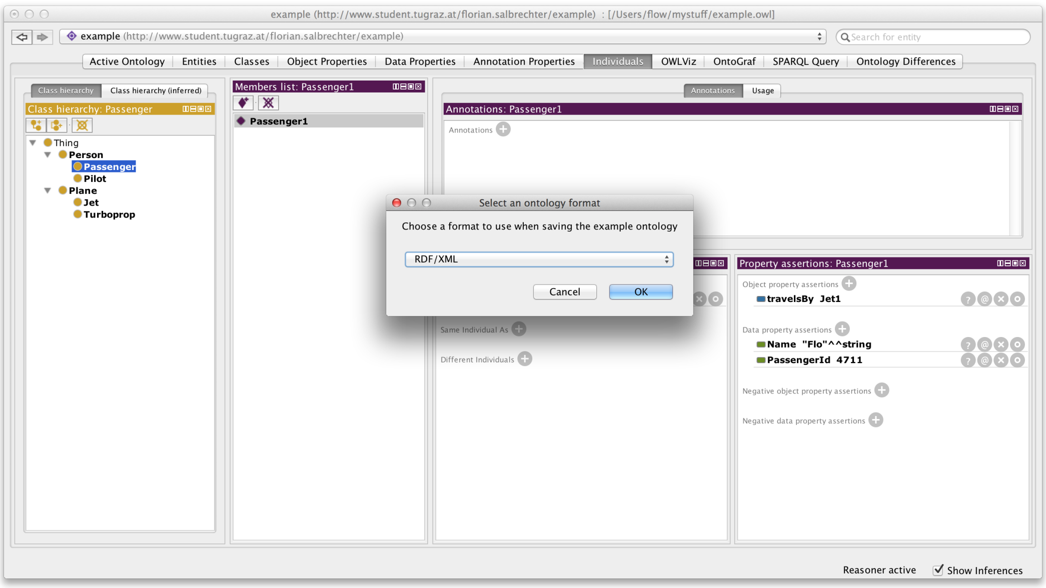
Figure A.8.: Individuals

show and explain them. To start the reasoner click on "Reasoner" and then "Start reasoner". Figure A.8 shows the individuals.

Step 6: Save ontology The last step is to save the ontology. Click "File" and then "Save as" to choose an output folder for your ontology. Use the preferred format. In this example we will use RDF/XML.

Figure A.9 shows the dialog.

Example OWL File The following listing shows the OWL file of the created ontology.


example (http://www.student.tugraz.at/florian.salbrechter/example) : [/Users/flow/mystuff/example.owl]

Select an ontology format

Choose a format to use when saving the example ontology

RDF/XML

Annotations: Passenger1

Property assertions: Passenger1

Object property assertions

travelsBy Jet1

Data property assertions

Name "Flo"^^string

PassengerId 4711

Figure A.9.: Export ontology

Listing A.1: Example Ontology

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#"
        ↪ >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-
        ↪ schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-
        ↪ syntax-ns#" >
    <!ENTITY example "http://www.student.tugraz.at/
```

```
        ↪ florian.salbrechter/example#" >
]>

<rdf:RDF xmlns="http://www.student.tugraz.at/florian.
    ↪ salbrechter/example#"
     xml:base="http://www.student.tugraz.at/florian.
         ↪ salbrechter/example"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
         ↪ syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema
         ↪ #"
     xmlns:example="http://www.student.tugraz.at/
         ↪ florian.salbrechter/example#">
    <owl:Ontology rdf:about="http://www.student.
        ↪ tugraz.at/florian.salbrechter/example"/>



    <!--
    ///////////////////////////////////////////////
    //
    // Object Properties
```

```
//
///////////////////////////////////////////////
 -->

<!-- http://www.student.tugraz.at/florian.
   ↪ salbrechter/example#flies -->

<owl:ObjectProperty rdf:about="&example;flies">
    <rdfs:domain rdf:resource="&example;Pilot"/>
    <rdfs:range rdf:resource="&example;Plane"/>
</owl:ObjectProperty>

<!-- http://www.student.tugraz.at/florian.
   ↪ salbrechter/example#travelsBy -->

<owl:ObjectProperty rdf:about="&example;travelsBy
   ↪ ">
    <rdfs:domain rdf:resource="&example;Passenger
       ↪ "/>
    <rdfs:range rdf:resource="&example;Plane"/>
</owl:ObjectProperty>
```

```
<!--
///////////////////////////////////////////////
//
// Data properties
//
///////////////////////////////////////////////
 -->

<!-- http://www.student.tugraz.at/florian.
   ↪ salbrechter/example#Name -->

<owl:DatatypeProperty rdf:about="&example;Name">
    <rdfs:domain rdf:resource="&example;Person"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<!-- http://www.student.tugraz.at/florian.
   ↪ salbrechter/example#PassengerId -->

<owl:DatatypeProperty rdf:about="&example;
   ↪ PassengerId">
    <rdfs:domain rdf:resource="&example;Passenger
       ↪ "/>
    <rdfs:range rdf:resource="&xsd;integer"/>
```

```
</owl:DatatypeProperty>


<!-- http://www.student.tugraz.at/florian.
    ↪ salbrechter/example#PilotId -->

<owl:DatatypeProperty rdf:about="&example;PilotId
    ↪ ">
    <rdfs:domain rdf:resource="&example;Pilot"/>
    <rdfs:range rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>




<!--
 ///////////////////////////////////////////////
 //
 // Classes
 //
 ///////////////////////////////////////////////
  -->

<!-- http://www.student.tugraz.at/florian.
    ↪ salbrechter/example#Jet -->
```

```
<owl:Class rdf:about="&example;Jet">
    <rdfs:subClassOf rdf:resource="&example;Plane
        ↪ "/>
</owl:Class>

<!-- http://www.student.tugraz.at/florian.
    ↪ salbrechter/example#Passenger -->

<owl:Class rdf:about="&example;Passenger">
    <rdfs:subClassOf rdf:resource="&example;
        ↪ Person"/>
</owl:Class>

<!-- http://www.student.tugraz.at/florian.
    ↪ salbrechter/example#Person -->

<owl:Class rdf:about="&example;Person"/>

<!-- http://www.student.tugraz.at/florian.
    ↪ salbrechter/example#Pilot -->

<owl:Class rdf:about="&example;Pilot">
    <rdfs:subClassOf rdf:resource="&example;
        ↪ Person"/>
```

```
</owl:Class>


<!-- http://www.student.tugraz.at/florian.
    ↪ salbrechter/example#Plane -->

<owl:Class rdf:about="&example;Plane"/>


<!-- http://www.student.tugraz.at/florian.
    ↪ salbrechter/example#Turboprop -->

<owl:Class rdf:about="&example;Turboprop">
    <rdfs:subClassOf rdf:resource="&example;Plane
        ↪ "/>
</owl:Class>




<!--
 ///////////////////////////////////////////////
 //
 // Individuals
 //
 ///////////////////////////////////////////////
 -->
```

```
<!-- http://www.student.tugraz.at/florian.
    ↪ salbrechter/example#Jet1 -->

<owl:NamedIndividual rdf:about="&example;Jet1">
    <rdf:type rdf:resource="&example;Jet"/>
</owl:NamedIndividual>

<!-- http://www.student.tugraz.at/florian.
    ↪ salbrechter/example#Passenger1 -->

<owl:NamedIndividual rdf:about="&example;
    ↪ Passenger1">
    <rdf:type rdf:resource="&example;Passenger"/>
    <PassengerId rdf:datatype="&xsd;integer">4711
        ↪ </PassengerId>
    <Name rdf:datatype="&xsd;string">Flo</Name>
    <travelsBy rdf:resource="&example;Jet1"/>
</owl:NamedIndividual>

<!-- http://www.student.tugraz.at/florian.
    ↪ salbrechter/example#Pilot1 -->

<owl:NamedIndividual rdf:about="&example;Pilot1">
```

```
    <rdf:type rdf:resource="&example;Pilot"/>
    <PilotId rdf:datatype="&xsd;integer">4811</
       ↪ PilotId>
    <Name rdf:datatype="&xsd;string">Horst</Name>
    <flies rdf:resource="&example;Jet1"/>
</owl:NamedIndividual>




<!--
///////////////////////////////////////////////
//
// General axioms
//
///////////////////////////////////////////////
 -->

<rdf:Description>
    <rdf:type rdf:resource="&owl;
       ↪ AllDisjointClasses"/>
    <owl:members rdf:parseType="Collection">
        <rdf:Description rdf:about="&example;Jet"
           ↪ />
        <rdf:Description rdf:about="&example;
```

```
                ↪ Passenger"/>
            <rdf:Description rdf:about="&example;
                ↪ Pilot"/>
            <rdf:Description rdf:about="&example;
                ↪ Turboprop"/>
        </owl:members>
    </rdf:Description>
</rdf:RDF>
<!-- Generated by the OWL API (version 3.5.0) http://
    ↪ owlapi.sourceforge.net -->
```

Appendix B.

Application ontology

This appendix contains the created ontology. The listing shows the OWL file of the created ontology.

Listing B.1: Ontology used for the web application

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
     <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
     <!ENTITY ns "http://floriansalbrechter.com/ns#" >
     <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#"
         ↪ >
     <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-
         ↪ schema#" >
```

```
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-
       ↪ syntax-ns#" >
]>

<rdf:RDF xmlns="http://floriansalbrechter.com/ns#"
     xml:base="http://floriansalbrechter.com/ns"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
        ↪ syntax-ns#"
     xmlns:ns="http://floriansalbrechter.com/ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema
        ↪ #">
    <owl:Ontology rdf:about="http://
       ↪ floriansalbrechter.com/ns"/>



    <!--
    ///////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////
```

```
 -->


<!-- http://floriansalbrechter.com/ns#hasMass -->

<owl:ObjectProperty rdf:about="&ns;hasMass">
    <rdfs:domain rdf:resource="&ns;Disease"/>
    <rdfs:range rdf:resource="&ns;Mass"/>
</owl:ObjectProperty>


<!-- http://floriansalbrechter.com/ns#hasSize -->

<owl:ObjectProperty rdf:about="&ns;hasSize">
    <rdfs:domain rdf:resource="&ns;Disease"/>
    <rdfs:range rdf:resource="&ns;Size"/>
</owl:ObjectProperty>


<!-- http://floriansalbrechter.com/ns#hasWall -->

<owl:ObjectProperty rdf:about="&ns;hasWall">
    <rdfs:domain rdf:resource="&ns;Disease"/>
    <rdfs:range rdf:resource="&ns;Wall"/>
</owl:ObjectProperty>
```

```
<!--
///////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////
 -->

<!-- http://floriansalbrechter.com/ns#BloodFlow
   ↪ -->

<owl:Class rdf:about="&ns;BloodFlow">
    <rdfs:subClassOf rdf:resource="&ns;
        ↪ GallbladderObservation"/>
</owl:Class>

<!-- http://floriansalbrechter.com/ns#Disease -->

<owl:Class rdf:about="&ns;Disease"/>

<!-- http://floriansalbrechter.com/ns#
   ↪ GallbladderObservation -->
```

```
<owl:Class rdf:about="&ns;GallbladderObservation"
    ↪ />

<!-- http://floriansalbrechter.com/ns#Mass -->

<owl:Class rdf:about="&ns;Mass">
    <rdfs:subClassOf rdf:resource="&ns;
        ↪ GallbladderObservation"/>
</owl:Class>

<!-- http://floriansalbrechter.com/ns#Pain -->

<owl:Class rdf:about="&ns;Pain">
    <rdfs:subClassOf rdf:resource="&ns;
        ↪ GallbladderObservation"/>
</owl:Class>

<!-- http://floriansalbrechter.com/ns#Size -->

<owl:Class rdf:about="&ns;Size">
    <rdfs:subClassOf rdf:resource="&ns;
        ↪ GallbladderObservation"/>
</owl:Class>
```

```
<!-- http://floriansalbrechter.com/ns#Wall -->

<owl:Class rdf:about="&ns;Wall">
    <rdfs:subClassOf rdf:resource="&ns;
        ↪ GallbladderObservation"/>
</owl:Class>




<!--
///////////////////////////////////////////////
//
// Individuals
//
///////////////////////////////////////////////
 -->

<!-- http://floriansalbrechter.com/ns#Cancer -->

<owl:NamedIndividual rdf:about="&ns;Cancer">
    <rdf:type rdf:resource="&ns;Disease"/>
    <hasMass rdf:resource="&ns;NotShadowing"/>
</owl:NamedIndividual>
```

```
<!-- http://floriansalbrechter.com/ns#Gallstones
    ↪ -->

<owl:NamedIndividual rdf:about="&ns;Gallstones">
    <rdf:type rdf:resource="&ns;Disease"/>
    <hasSize rdf:resource="&ns;Large"/>
    <hasMass rdf:resource="&ns;NotShadowing"/>
    <hasMass rdf:resource="&ns;Shadowing"/>
    <hasWall rdf:resource="&ns;Thick"/>
</owl:NamedIndividual>

<!-- http://floriansalbrechter.com/ns#High -->

<owl:NamedIndividual rdf:about="&ns;High">
    <rdf:type rdf:resource="&ns;BloodFlow"/>
</owl:NamedIndividual>

<!-- http://floriansalbrechter.com/ns#
    ↪ Inflammation -->

<owl:NamedIndividual rdf:about="&ns;Inflammation"
    ↪ >
    <rdf:type rdf:resource="&ns;Disease"/>
    <hasSize rdf:resource="&ns;Large"/>
    <hasWall rdf:resource="&ns;Thick"/>
</owl:NamedIndividual>

<!-- http://floriansalbrechter.com/ns#Large -->

<owl:NamedIndividual rdf:about="&ns;Large">
    <rdf:type rdf:resource="&ns;Size"/>
</owl:NamedIndividual>

<!-- http://floriansalbrechter.com/ns#Low -->

<owl:NamedIndividual rdf:about="&ns;Low">
    <rdf:type rdf:resource="&ns;BloodFlow"/>
</owl:NamedIndividual>

<!-- http://floriansalbrechter.com/ns#
   ↪ LowerAbdominal -->

<owl:NamedIndividual rdf:about="&ns;
   ↪ LowerAbdominal">
    <rdf:type rdf:resource="&ns;Pain"/>
</owl:NamedIndividual>

<!-- http://floriansalbrechter.com/ns#No -->
```

```
<owl:NamedIndividual rdf:about="&ns;No">
    <rdf:type rdf:resource="&ns;Pain"/>
</owl:NamedIndividual>

<!-- http://floriansalbrechter.com/ns#Normal -->

<owl:NamedIndividual rdf:about="&ns;Normal">
    <rdf:type rdf:resource="&ns;BloodFlow"/>
    <rdf:type rdf:resource="&ns;Size"/>
    <rdf:type rdf:resource="&ns;Wall"/>
</owl:NamedIndividual>

<!-- http://floriansalbrechter.com/ns#
   ↪ NotShadowing -->

<owl:NamedIndividual rdf:about="&ns;NotShadowing"
   ↪ >
    <rdf:type rdf:resource="&ns;Mass"/>
</owl:NamedIndividual>

<!-- http://floriansalbrechter.com/ns#Shadowing
   ↪ -->
```

```
<owl:NamedIndividual rdf:about="&ns;Shadowing">
    <rdf:type rdf:resource="&ns;Mass"/>
</owl:NamedIndividual>

<!-- http://floriansalbrechter.com/ns#Small -->

<owl:NamedIndividual rdf:about="&ns;Small">
    <rdf:type rdf:resource="&ns;Size"/>
</owl:NamedIndividual>

<!-- http://floriansalbrechter.com/ns#Thick -->

<owl:NamedIndividual rdf:about="&ns;Thick">
    <rdf:type rdf:resource="&ns;Wall"/>
</owl:NamedIndividual>

<!-- http://floriansalbrechter.com/ns#Thin -->

<owl:NamedIndividual rdf:about="&ns;Thin">
    <rdf:type rdf:resource="&ns;Wall"/>
</owl:NamedIndividual>

<!-- http://floriansalbrechter.com/ns#
   ↪ UpperAbdominal -->
```

```
    <owl:NamedIndividual rdf:about="&ns;
       ↪ UpperAbdominal">
        <rdf:type rdf:resource="&ns;Pain"/>
    </owl:NamedIndividual>
</rdf:RDF>

<!-- Generated by the OWL API (version 3.5.0) http://
    ↪ owlapi.sourceforge.net -->
```

Bibliography

Allemang, Dean and James Hendler (2008). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN: 0123735564, 9780123735560 (cit. on p. 22).

Antoniou, Grigoris and Frank van Harmelen (2008). *A Semantic Web Primer, 2Nd Edition (Cooperative Information Systems)*. 2nd ed. The MIT Press. ISBN: 0262012421, 9780262012423 (cit. on pp. 21, 22, 25).

Banks, G. (1986). "Artificial intelligence in medical diagnosis: the INTERNIST/-CADUCEUS approach." In: *Critical reviews in medical informatics* 23 (cit. on p. 17).

Bates, D. et al. (2001). "Reducing the frequency of errors in medicine using information technology." In: *Journal of the American Medical Informatics Association* 8(4), pp. 299–308 (cit. on p. 7).

Baxt, W. G. and J. Skora (1996). "Ultrasound Physics and Instrumentation for Pathologists." In: *The Lancet* 347(8993), pp. 12–15 (cit. on p. 6).

Bellman, R. E. (1978). *An Introduction to Artificial Intelligence: Can Computers Think?* Boyd and Fraser Publishing Company (cit. on p. 13).

Berners-Lee, Tim, James Hendler, and Ora Lassila (2001). “The semantic web.” In: *Scientific american* 284.5, pp. 28–37 (cit. on p. 27).

Burtseva, L. et al. (2011). “SonaRes - Diagnostic decision support system for ultrasound examination.” In: *E-Health and Bioengineering Conference (EHB), 2011*, pp. 1–4 (cit. on p. 18).

Charniak, E. and D. McDermott (1978). *Introduction to Artificial Intelligence*. Addison-Wesley (cit. on p. 13).

Cyganiak, Richard, David Wood, and Markus Lanthaler (2014). *RDF 1.1 Concepts and Abstract Syntax* (cit. on pp. 23, 24).

DuCharme, Bob (2011). *Learning SPARQL*. Ed. by Simon St. Laurent and Jasmine Perez (cit. on p. 31).

Eik-Nes, Sturla H and Juriy W. Wladimiroff (2009). *Ultrasound in Obstetrics and Gynaecology*. first edition. Elsevier (cit. on pp. 9–11).

Feldman, Susan (2012). *IBM's Watson: From Winning Games to Saving Lives*. URL: http://www-03.ibm.com/innovation/us/watson/pdf/lcUS23400812.pdf (cit. on p. 17).

Gaindric, C (2009). “Decision support systems in ultrasound diagnostics.” In: *Computer Science* 17.3, p. 51 (cit. on p. 18).

Gruber, Thomas R. (1993). “A Translation Approach to Portable Ontology Specifications.” In: *Knowl. Acquis.* 5.2, pp. 199–220. ISSN: 1042-8143. DOI: 10.1006/knac.1993.1008. URL: http://dx.doi.org/10.1006/knac.1993.1008 (cit. on p. 26).

Guarino, Nicola, Daniel Oberle, and Steffen Staab (2009a). “RDF Storage and Retrieval Systems.” In: *Handbook on ontologies*. Springer, pp. 510–529 (cit. on p. 29).

Guarino, Nicola, Daniel Oberle, and Steffen Staab (2009b). “What is an Ontology?” In: *Handbook on ontologies*. Springer, pp. 1–17 (cit. on p. 19).

Haugeland, J. (1985). *Artificial Intelligence: The Very Idea*. 2nd edition. MIT Press (cit. on p. 12).

Hayes-Roth, Frederick (1984). “The knowledge-based expert system: A tutorial.” In: *Computer* 17.9, pp. 11–28 (cit. on p. 17).

Hedén, B. et al. (1997). “Acute myocardial infarction detected in the 12-lead ECG by artificial neural networks.” In: *Circulation* 96(6), pp. 1798–1802 (cit. on p. 7).

Korpela, Jukka K. (2006). *Unicode explained - internationalize documents, programs, and web sites.* O’Reilly, pp. I–XVII, 1–658. ISBN: 978-0-596-10121-3 (cit. on p. 21).

Kurzweil, R. (1990). *The Age of Intelligent Machines*. 3rd edition. MIT Press (cit. on p. 12).

Kushniruk, Andre W and Vimla L Patel (2004). “Cognitive and usability engineering methods for the evaluation of clinical information systems.” In: *Journal of biomedical informatics* 37, pp. 56–76 (cit. on p. 16).

Lieu, David (2010). “Ultrasound Physics and Instrumentation for Pathologists.” In: *Archives of Pathology and Laboratory Medicine* 134, pp. 1541–1556 (cit. on pp. 7, 8).

Motik, Boris et al. (2014). *OWL 2 Web Ontology Language Profiles (Second Edition)* (cit. on p. 28).

Narouze, Samer N (2011). *Atlas of Ultrasound-Guided Procedures in Interventional Pain Management*. first edition. Springer (cit. on pp. 8, 10, 11).

Nilsson, N. J. (1998). *Artificial Intelligence: A New SynthesisArtificial Intelligence: A New Synthesis*. Morgan Kaufmann (cit. on p. 13).

Palmer, P.E.S. (1995). *Manual of Diagnostic Ultrasound*. first edition. World Health Organization (cit. on pp. 8, 11).

Pan, Jeff Z (2009). "Resource description framework." In: *Handbook on Ontologies*. Springer, pp. 71–90 (cit. on pp. 24, 26).

Patel, VL et al. (2009). "The coming of age of artificial intelligence in medicine." In: *Artificial Intelligence in Medicine* 46, pp. 5–17 (cit. on p. 16).

Petrick, N. et al. (2002). "Breast Cancer Detection: Evaluation of a Mass-Detection Algorithm for Computer-aided Diagnosis—Experience in 263 Patients." In: *Radiology* 224(1), pp. 217–224 (cit. on p. 7).

Poole, D., A. K. Mackworth, and R. Goebel (1998). *Computational intelligence: A logical approach*. Oxford University Press (cit. on p. 13).

Rich, E. and K. Knight (1991). *Artificial Intelligence*. 2nd edition. McGraw-Hill (cit. on p. 12).

Russell, S. J. and P. Norvig (2009). *Artificial intelligence: a modern approach*. 2nd edition. Prentice Hall (cit. on pp. 12, 16).

Salbrechter, Florian (2013). "Building a software platform to guide doctors and nurses with ultrasound scanning." MA thesis. Cranfield, UK: Cranfield University (cit. on pp. 3, 7, 12, 13, 16, 17, 35, 40).

Shortliffe, Edward H. and Bruce G. Buchanan (1975). "A model of inexact reasoning in medicine." In: *Mathematical Biosciences* 23, pp. 351–379 (cit. on p. 17).

Tester, David (2014). *Flavors of OWL* (cit. on p. 28).

Winston, P. H. (1992). *Artificial Intelligence*. 3rd edition. Addison-Wesley (cit. on p. 13).