Semantic technology for management of data recorded during therapy of cardiac arrest patients

Master's thesis by

Xiaoxi Li

Supervisor:

Trygve Eftestøl

Co supervisors:

Kari Anne Haaland Thorsen, Erlend Tøssebro and Chunming Rong

University of Stavanger, N-4036 Stavanger, Norway

www.uis.no

Faculty of Science and Technology

Department of Electrical and Computer Engineering

2010

Abstract

The topic of this thesis is the integration of data recorded during cardiopulmonary resuscitation (CPR). In the process of CPR many important data should be recorded, such as: time of the cardiac arrest witnessed, number of shocks given to the patient and parameters for each shock. These data will be used for the coming analysis and producing the report of the therapy. The analysis and comparison of the CPR data and reports may improve the survival rate. However the formats and terminologies of CPR data vary greatly between EMS (Emergency Medical Services) systems in different places. So it is inconvenient to access, compare and transfer data between different EMS systems.

In this thesis we use semantic way to solve the heterogeneity problem. An ontology is defined to provide a standardized representation of the heterogeneous data from different EMSs. In this ontology we defined standardized CPR terminologies and report style, and the "dialect" from different EMS systems can be mapped onto this ontology.

An interface is defined to map the different databases from different EMS systems onto this ontology. And we provide a uniform way based on the ontology to query all of the databases as if they are one huge database.

Acknowledgements

I would like to show my acknowledgements to my supervisor professor Trygve Eftestøl for his supervision and support during my master's work, and also my gratitude to my co-supervisor Kari Anne Haaland Thorsen, Erlend Tøssebro and Chunming Rong for their helpful suggestions and guide. During the discussions with them in the meetings I learnt a lot both about how to solve the problems in the project and also how to work as a member in a team.

I also would like to thank my partner Chao Li. He gave me many help and suggestions in the project as well as courage to overcome difficulties.

At last I should show my thanks to my family and my friends for their support in my living and study in the past two years. And I should thank the University of Stavanger and Norwegian government for the opportunity for my study and living here which is a valuable experience for me.

Xiaoxi Li

June 14, 2010, Stavanger

Contents

List of Figures

1. Introduction

Cardiac arrest [1] is the cessation of cardiac mechanical activity (in other words, the heart stops pumping blood) as confirmed by the absence of signs of circulation. In recent years cardiac arrest is a leading cause of death in First World countries. In the United States there are about 265,000 out of hospital cardiac arrests every year [2]. And in Europe, more than 250,000 people suffer from sudden out-of-hospital cardiac arrest annually [3]. Ventricular fibrillation (VF) is the most common cause of cardiac arrest. When VF happens, cardiopulmonary resuscitation (CPR) can be given to the patient to delay the tissue death, thus improve the possibility of survival. The operations of CPR include a series of chest compressions and ventilations. Finally defibrillations (electric shocks) can be delivered to the patient to restore the spontaneous circulation.

The data recorded in the process of CPR and other resuscitation operations like defibrillation and intravenous drugs injection (we generally call them "CPR data" in this thesis) are very important. Here are some examples of such kind of data: "time of the cardiac arrest witnessed", "number of shocks given to the patient" and "heart rhythm after each shock". These data can help researchers to do analysis and make reports of the therapy of cardiac arrest cases. And these analysis and reports may contribute to improving the survival rate.

Survival rates from out-of-hospital ventricular fibrillation cardiac arrest vary dramatically throughout the United States with high survival rates of 46% in Seattle and King County and low survival rates of 3% in Chicago, 5% in New York, and 7% in Los Angeles.[4] Such differences indicate a potential improvement for the places with low survival rate. To close the gap, the hospitals and Emergency Medical Services (EMS) systems in the cities with high survival rate should share their local strategies for success; and those in the cities with low survival rate should study and compare the CPR data and the final reports of the cardiac arrest cases from the cities like Seattle and King County. By doing this, the hospitals and EMS systems in the cities with low survival rate can find their disadvantages and make the direction of their future work clear.

However studying and comparing the CPR data from different places has always been difficult due to the lack of standardization which includes two aspects: the nonuniform terminologies used in the data sources and the different report styles.

The log files from the defibrillators are important data sources of the CPR data. These log files are electronic reports with parameters and time stamps of important events happening in the process of CPR, such as “power on”, “charge”, “shock delivered”, and “heart rhythm”. However the formats and terminologies in the log files vary a lot if the models or manufactures of the defibrillators are different. For example some defibrillators use XML (Extensible Markup Language) format to save the log files, while some others use their own specific formats; and some devices use the term “PowerOn” to represent the power on event, while some others may use the term “DeviceStarted”.

The report is a review of a cardiac arrest case. Most of the important information is recorded in the report, like the key events, key dates and times and the outcome. However different hospitals and EMS systems often have their own report styles, which brings difficulties for studying and comparing the information in the reports.

1.1. Key problem and earlier research

The difficulty about studying and comparing the CPR data between different EMS systems is just like that we have different dialects in different places. To solve this problem, firstly standardized terminologies and a uniform report style should be established, and then the relationships between the standardized concepts and the dialects should be given.

In June 1990 an international resuscitation conference concerned with out-of-hospital cardiac arrest research, held at the Utstein Abbey in Norway. Experts in that conference discussed the lack of standardization and uniform report style in the research of cardiac arrest. And in 1991 a standardized report style “Utstein Style” [5] (updated in 20004) was proposed. Utstein style defines the key information elements like patient identifier, cause of arrest (aetiology), key date and times and treatment factors and outcomes.

Nichol [6] shows the possibility to collect CPR data in different international sites through an internet-based international registry. They collect the original CPR data from EMS personnel and manually fill them into a data file. This data file was then imported into a study database which was based on the Utstein style. This method can only

collect CPR data from the hospitals which use the Utstein style report. However there are still plenty of hospitals which don't build their data systems based on the Utstein style.

As the complexity and amount of data generated during resuscitation efforts are ever increasing, T.Eftestøl and K.A.H.Thorsen proposed to use semantic technologies to provide infrastructure for efficient resuscitation data integration and analysis [7, 8, 9]. In their theory they proposed to build an ontology to describe the knowledge in the domain of CPR. This ontology acts like an information container. All of the standardized concepts (e.g. the definitions in the Utstein style and the standardized event names from the log files of different defibrillators) are defined in this ontology. Then hospitals and EMS systems can map their own dialects onto the standardized concepts defined in this ontology, and queries can be made to retrieve information from it. Finally researchers and EMS personnel can study and compare CPR data from different places based on the standardized concepts defined in the ontology, and the mapping information will help them collect data from different hospitals and EMS systems.

1.2. Contributions of this work

In this work we followed T.Eftestøl and K.A.H.Thorsen's theory. An ontology is developed to represent the concepts and relationships in the domain of CPR (we call it CPR ontology in this thesis). The ontology provides a standardized representation of the heterogeneous CPR data from different EMS systems. All of the different data sources (databases) and report styles can be mapped onto this ontology. Users like researchers and EMS personnel should only get to know the standardized definitions (terminologies), and then they can make queries against the CPR ontology to collect the resuscitation data from all of the data sources which have been mapped onto this ontology.

My job in this project is to build the CPR ontology and map the heterogeneous data sources and different report styles onto the standardized concepts in this ontology. The data sources are two local databases saving the information in the log files from different defibrillators. And at present we only have two report styles: the Utstein style which serves as the standard concepts, and the Public Health - Seattle & King County (PHSKC) report style (we will call it Seattle style in this thesis) which serves as a dialect.

The two local databases are built by my partner Chao Li. He also developed an application to import the data from the log files into the databases automatically. The purpose of these two databases is to facilitate maintaining and accessing the CPR data, and they are also the important parts of this data integration system. A graphic user interface (GUI) was built by Chao to help the users make queries and display the results.

1.3. Thesis outline

The following chapters constitute the thesis:

Introductory part

Chapter 1 introduces the problem of analyzing the resuscitation data from different EMS systems and reviews the earlier work. The scope and contributions of this thesis are summarized.

Chapter 2 gives the background of CPR and the semantic technologies. Some CPR data samples are shown in this chapter. And the concept of ontology is described.

Materials and methods

Chapter 3 describes the structure of the data integration system. Details about the content of the ontology will be introduced. And a scenario is shown to demonstrate how this ontology works.

Chapter 4 shows different methods to map databases onto an ontology. The advantages and disadvantages of these methods are told about. The details about our method are discussed.

Experiments

Chapter 5 introduces the graphic user interface in this data integration system. Some test queries are made to show and how it works.

Conclusive part

Chapter 6 summarizes the major contributions and conclusions of this work, and suggests the problems for further research.

2. Background

In this chapter we first describe some nomenclature basic knowledge about cardiopulmonary resuscitation (CPR). We continue by talking about the main issues of semantic technologies.

2.1. Medical background about CPR

In 1991 an international consensus workshop gave the definition of cardiac arrest: "Cardiac arrest is the cessation of cardiac mechanical activity as confirmed by the absence of signs of circulation" [1]. Cardiac arrest is often caused by an abnormal heart rhythm called ventricular fibrillation (VF). When VF happens, the heart quivers and stops pumping blood. The patient in the VF status needs cardiopulmonary resuscitation (CPR) and defibrillation to restore the spontaneous circulation.

CPR is an emergency procedure to create artificial circulation by performing rhythmic chest compressions with or without ventilations. The main purpose of CPR is not to restart the heart but to maintain an artificial flow of small amount of oxygenated blood to the brain and heart. Effective CPR can delay the tissue death and increase of opportunity for a successful resuscitation without permanent brain damage. The data recorded in this process can be: "heart rhythm", "times of chest compressions", "any ventilations", etc.

Defibrillation is a sequence of electric shocks delivered to the heart. It eliminates the abnormal VF heart rhythm and allows the normal rhythm to resume. Defibrillation is the effective treatment for the cardiac arrest caused by VF. The data recorded in the process defibrillation can be: "heart rhythm", "number of electric shocks", "energy of the shock", etc.

Note: The process of giving treatment to a cardiac arrest patient includes continuous CPR, defibrillation and some other assistance like intravenous drugs. The data information in this whole process is what we focus on in this thesis. And for convenience

Figure 2.1.: (a) Philips HeartStart MRx defibrillator. (b) Philips HeartStart FR2 defibrillator. (c) LifePak 12 Defibrillator.

purpose we use the term "CPR" to stand for the whole treatment and "CPR data" to denote the data information in the whole treatment. More medical background information can be found in [1, 10].

2.1.1. Defibrillators and log files

Defibrillation is delivering a therapeutic dose of electrical energy to the affected heart in VF status with a device called a defibrillator. Typically a defibrillator consists of a pair of pads to deliver the shocks, sensors to collect the information from the patient, and a monitor to display the electrocardiograph (ECG) wave and other information. Figure 2.1 show three different models of the defibrillators.

The defibrillator records the resuscitation events including medical personnel's operation and patient's response in a log file. Figure 2.2, 2.3 and 2.4 show the samples of the defibrillator log files. In these log files we can see the event names, the timestamp and the parameters. And all of the events are listed in chronological order. As mentioned in the introduction chapter, the terms (event names and parameter names) and the structures (formats) vary a lot between the log files from different defibrillator models. For example, the event "Shock Delivered" has three different dialects in different files: "shockDelivered" in Philips HeartStart MRx, "Shock 1 delivered" in Philips HeartStart FR2 and "Shock" in LifePak 12; and the file formats are also different: the first two files are in XML (Extensible Markup Language, will be introduced in section 2.2.1) format, while the third file is not in XML. These heterogeneities bring a lot of trouble for comparing and analyzing the data in the log files.

```
<evt type="charge"
     msecs="235567"
     caption="Charging to 150J">
   <waveFormRef id="0"/>
   <param type="energy" value="150" units="joules" />
   <param type="hr" value="-?-" units="bpm"/>
</evt>
<evt type="shockDelivered"
     msecs="242002"
     caption="Shock # 1, 155.6 J, 101.1 ohms, 16.9 A">
   <waveFormRef id="0"/>
   <param type="shockIndex" value="1" />
   <param type="energy" value="155.6"  units="joules"/>
   <param type="ptImpedance" value="101.1"  units="ohms"/>
   <param type="peakCurrent" value="16.9"  units="amps"/>
   <param type="hr" value="0" units="bpm"/>
</evt>
```

Figure 2.2.: Log file sample of defibrillator Philips HeartStart MRx. This sample contains two events: event “charge” with “energy” and “hr” as its parameters and event “shockDelivered” with parameters “shockIndex”, “energy”, “ptImpendance”, etc.

```
<QDeviceEvents>
    <caption>Armed</caption>
    <description />
    <actualTime>2006-01-13T20:55:55-08:00</actualTime>
    <elapsedTime>57</elapsedTime>
    <caseID>ZF06000003</caseID>
    <eventType>19</eventType>
</QDeviceEvents>
    ... ...
<QDeviceEvents>
    <caption>Shock 1 delivered</caption>
    <description>Impedance: 111 Ohms</description>
    <actualTime>2006-01-13T20:55:58-08:00</actualTime>
    <elapsedTime>60</elapsedTime>
    <caseID>ZF06000003</caseID>
    <eventType>1</eventType>
</QDeviceEvents>
```

Figure 2.3.: Log file sample of defibrillator Philips HeartStart FR2. This sample has a similar structure as the first one, as they are both in XML format. However the terminologies are different. The event “Armed” has the same meaning as the event “charge” in the first sample (MRx log in figure 2.2). And the event “Shock 1 delivered” refers to “shockDelivered”.

```
Total shocks: 6
Lead: Impedance, SampleRate (HZ): 61.038, Counts/Unit: 0.819
Lead: Paddles (Generic), SampleRate (HZ): 125, Counts/Unit: 204.918
**********************************************
EventType,EventName,DeviceSerialNumber,Date/Time,SampleNumber,MiscParams
29| Power On| 10.07.2009 10:59:19| 0
39|VF|10.07.2009 10:59:25|10:57:47.74404
99| Initial Rhythm| 10.07.2009 10:59:26| 3339
84| Check Patient| 10.07.2009 10:59:34| 7519
39|startCC|10.07.2009 10:59:38|10:58:00.32802
39|stopCC|10.07.2009 10:59:40|10:58:02.90404
70| Analysis 1| 10.07.2009 11:01:54| 77634| | | | | | | | | | |
62| Shockable| 10.07.2009 11:01:57|778974|1|286|72|36|65|19||9||9||-4096
54| Charge Complete| 10.07.2009 11:02:05| 83024
12| Shock 1, 200 J| 10.07.2009 11:02:11| 85949
  ... ...
39|ROSC|10.07.2009 11:30:25|11:28:47.39197
13| Power Off| 10.07.2009 11:51:36| 1568500
```

Figure 2.4.: Log file sample of defibrillator LifePak 12. This sample has different structure from the first two, as it isn't in XML format. And the two events metioned in the first two samples have their dialects in this sample: "Charge Complete" and "Shock 1".

2.1.2. Utstein style report

In June 1990 an international resuscitation conference concerned with out-of-hospital cardiac arrest research, held at the Utstein Abbey in Norway. Experts in that conference discussed the lack of standardization and uniform report style in the research of cardiac arrest. And in 1991 a standardized report style "Utstein Style" [5] (updated in 2004) was proposed. Utstein style defines the key information elements like patient identifier, cause of arrest (aetiology), key date and times, treatment factors and outcomes. These standard definitions can be used as "common language" for the study and comparison of the CPR data between different EMS systems. The details about the terminologies and definitions in the Utstein style can be found in the specification [1, 5].

Utstein style is the standard report style and is recommended to be used globally. However plenty of EMS systems at present still use their own report styles, like the style used in Seattle & King County (Seattle style). Many of the items in the Seattle style can be found in Utstein style but with different names, such as: the Seattle style item"Initial rhythm" is called "First monitored rhythm" in Utstein style. In this data integration system Utstein style provides a set of standard definitions, and other report styles can map their items onto these standard items in the Utstein style. The mapping details will be told about in chapter 3.

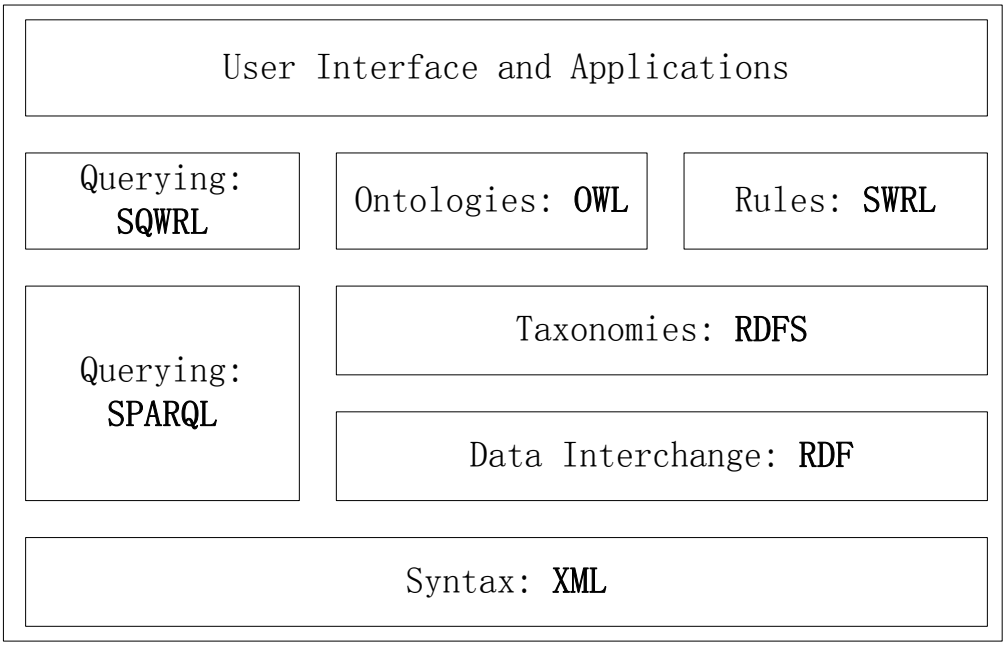


Figure 2.5.: The Semantic Web Stack, also known as Semantic Web Cake or Semantic Web Layer Cake, illustrates the relationship of technologies used in the Semantic Web.

2.2. Semantic web technologies

The Semantic Web [11] is an evolving development of the World Wide Web in which the semantics of data and services on the web is defined, making it possible for the content of web to be understood both by people and by machines. The semantic web consists of the standards and tools, such as: XML (Extensible Markup Language), RDF (Resource Description Framework), RDF Schema OWL (Web Ontology Language) and query languages like SPARQL and SQWRL. The relationships between these technologies are presented in the Semantic Web Stack [12]. Figure 2.5 shows the Semantic Web Stack (a little modified from the original one). In this stack the upper technologies are built based on the lower technologies.

In our application the semantic web technologies can make the CPR data sources, the standardized definitions and the mapping information machine accessible. Most of the technologies listed in the Semantic Web Stack will be used in this ontology based data integration system. For example, XML is the format of some of the log files from defibrillators; OWL is the language used to describe the CPR ontology; and SQWRL is a query language which can extract information from the CPR ontology. The details of the members in the Semantic Web Stack will be described in the following sections.

2.2.1. XML: Extensible Markup Language

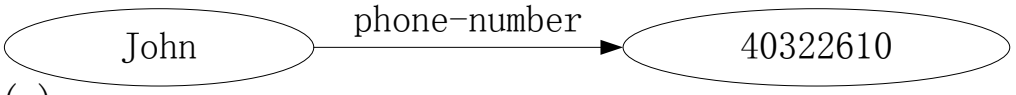
Extensible Markup Language (XML) [13] is a markup language much like HTML (Hyper Text Markup Language). Compared with HTML, which is used for display data, XML is designed for carrying data. It is well formed and based on element tags, whichs make the content clear and accessible for both human and machine. Thus XML usually serves as a uniform data exchange format between applications. With all of these advantages, XML is the choice by many defibrillator manufacturers to carry the data in the log files. In addition there are plenty of application program interfaces (API) which can parse and extract the content of XML files, e.g. JDOM (Java Document Object Model) implemented with Java programming language [14]. JDOM is just the tool we used to import the data from the log files to the databases. Figure 2.2 and 2.3 show the samples of the log files which are both written in XML.

In the family of semantic web technologies XML has another important role. As can be seen in the Semantic Web Stack (figure 2.5), XML is on the bottom of this stack, so it is the fundamental of the semantic web technologies. All of the other semantic web description languages shown in the Semantic Web Stack are based on XML and developed using XML as their basic syntax.

2.2.2. RDF, RDF Schema and SPARQL

RDF (Resource Description Framework) [15], although often called a language, is essentially a data-model. Its basic building elements are subject-predicate-object triples, called statements. The subject denotes the resource, predicate denotes the properties which expresses the relationship between the subject and the object, and the object denotes the value. Just like XML, RDF is also an information container. Usually the RDF triples can be expressed with a RDF model graph or with the XML-based syntax. Figure 2.6 shows an example to express the RDF triple (John, phone number, 40322610) in the two ways.

RDF Schema (RDFS) [16] is the RDF vocabulary description language. In RDFS we can define the vocabulary used in RDF, specify which properties should be applied to which kinds of objects and what values they can take, and describe the relationships between objects. RDFS is the original language to describe an ontology. It defines the classes in the ontology, and RDF triples are the individuals of the classes. RDFS gives the semantic to the RDF triples.



(a)

```
<?xml version="1.0" encoding="UTF-16"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:mydomain="http://www.mydomain.org/my-rdf-ns">
    <rdf:Description rdf:about="John">
        <mydomain:phone-number>
          40322610
        </mydomain:phone-number>
    </rdf:Description>
</rdf:RDF>
```

(b)

Figure 2.6.: Two ways to describe a RDF triple. (a) Using a graph to express the triple; (b) Using a XML-based syntax to express the triple.

SPARQL (SPARQL Protocol and RDF Query Language, pronounced "sparkle") [17] is an RDF query language. Much like a database system, RDFS can be thought as the schema of the database, RDF should be the data stored in the database, and SPARQL is the query language just like SQL (Structured Query Language, for querying the relational databases). With SPARQL we can query the information contained in the RDF triples. RDF, RDFS and SPARQL together provide a whole solution to build and access an ontology, however in this project we choose another set of more powerful tools: OWL and SQWRL.

2.2.3. Ontology and OWL

The term ontology has its origin in philosophy, and has been applied in many different domains. Tom Gruber proposed it in computer and information science in 1992 [18, 19], and Isabel F.Cruz gave a more clear definition: "An ontology is a formal, explicit specification of a shared conceptualization of a domain of interest." [19] This definition describes the following features of the ontology:

- An ontology is an abstract model of a particular domain of knowledge in the world;
- The knowledge defined in an ontology should be widely accepted;
- Concepts and the constraints on them in an ontology are explicitly defined;

- The structure and syntax of an ontology should be well formed and machine accessible.

An ontology describes the concepts (classes) and relationships (properties) in a particular domain, providing a vocabulary for that domain as well as a specification of the meanings of terms used in the vocabulary. In recent years, ontologies have been adopted by business companies and scientific researchers as a way to share, reuse and process domain of knowledge. In many applications like a data management and integration systems, an ontology is usually used as an intermediary to provide semantic for the shared data. The CPR ontology contains the standardized concepts and relationships in the domain of CPR. In this system it acts as a standard and an intermediary; all of the EMS systems should map their dialects to the standardized concepts in the CPR ontology.

The OWL Web Ontology Language (OWL) [20] is a language developed for defining and instantiating Web ontologies. In the Semantic Web Stack (figure 2.5) we can see OWL is on top of XML, RDF, and RDFS, because it has more facilities for expressing concepts and semantics than the others. OWL use XML as its syntax, and adds more predefined vocabulary for describing classes and properties. The OWL language has three increasingly expressive sub-languages: OWL-Lite, OWL-DL and OWL-Full. OWL-Lite is the least expressive sub-language which doesn't support enumerated classes and disjointness statements. OWL-Full is the most expressive one which has highly expressive modeling facilities such as meta-classes (classes of classes), and it has the full upward compatibility with RDF and RDFS, i.e. any valid RDF document is also a valid OWL Full document, and any valid RDF Schema definition is also a valid OWL Full definition. However OWL-Full is undecidable due to its powerful expressiveness, making it have no efficient reasoning support. The expressiveness of OWL-DL falls between that of OWL-Lite and OWL-Full. It is much more expressive than OWL-Lite and is based on Description Logics [21] (hence the suffix DL). It is therefore has complete support of reasoning, such as: automatically computing the classification hierarchy and check for inconsistencies. So in this project we chose OWL-DL to build the CPR ontology.

2.2.4. SQWRL: OWL query language

SQWRL (Semantic Query-enhanced Web Rule Language, pronounced "squirrel") [22] is an ontology query language built on the rule language SWRL (Semantic Web Rule

Language) [23]. It can query knowledge from ontologies developed with OWL. SQWRL has its syntax much like SWRL, and it is defined using a library (key words) of SWRL built-ins. Both SPARQL and SQWRL are developed to query the information in the ontologies; SPARQL is used for the ontologies built with the language RDFS, and SQWRL is used for the ontologies developed with OWL.

2.2.5. Protégé: an ontology development tool

There are many development tools to help developers build ontologies, and we chose Protégé 3.4.4 (see figure 2.7) to develop the ontology. Protégé [24, 25] is an extensible, platform-independent environment for creating and editing ontologies and knowledge bases. Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. And Protégé-OWL editor is an extension of Protégé that supports the Web Ontology Language (OWL). With the help of Protégé, developers never need to edit the OWL code, instead they can create classes, properties and individuals visually. And plenty of useful plug-ins are provided, like "SWRL Rules" which helps the user make SQWRL query sentences and display the results of the queries.

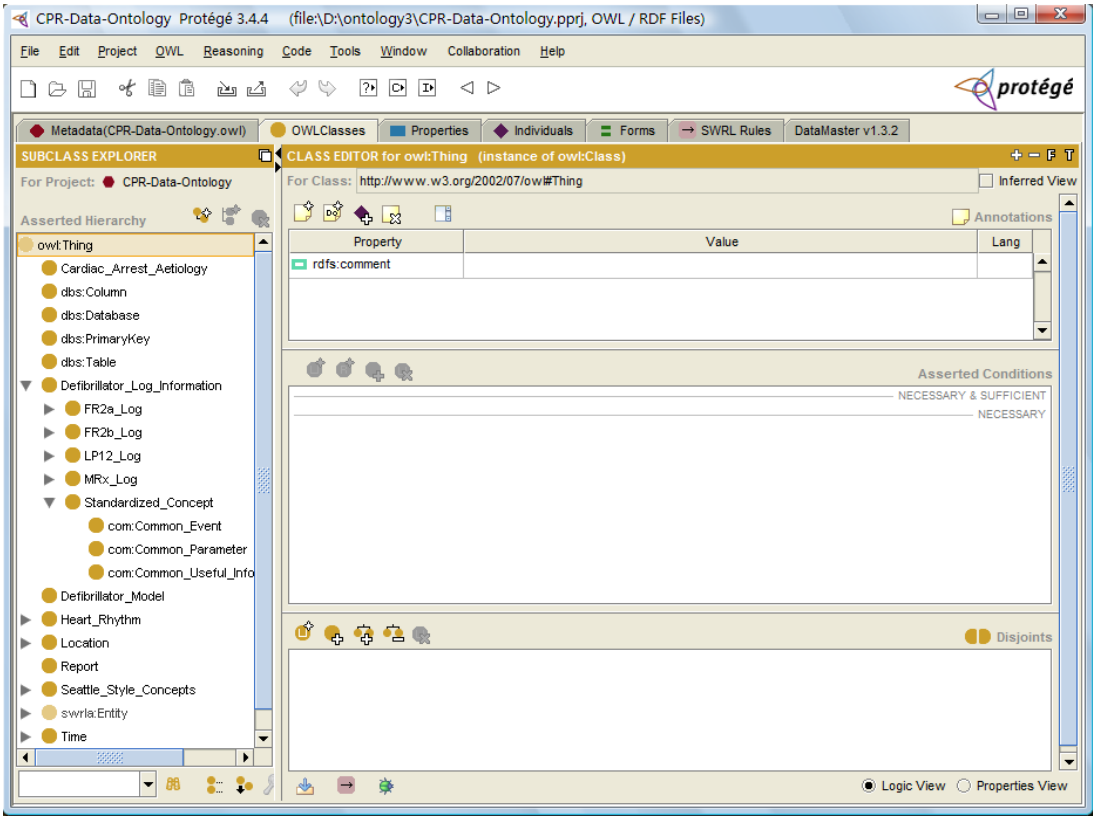


Figure 2.7.: The graphic user interface of Protégé

3. CPR Ontology

In this chapter we will first introduce the structure of this data integration system and the role of the CPR ontology. Then we will describe the content of this ontology. In this part an overview of the CPR ontology will be given, and we will talk about the details in three parts: the concepts of the defibrillator log files, the report part and the database mapping part.

After describing the content of the ontology we will introduce the query language SQWRL which is used to extract information from ontologies. And a scenario will be given to demonstrate how to query information with the CPR ontology. Finally we will discuss the extensibility of the CPR ontology.

3.1. System structure and the role of CPR ontology

This data integration system consists of data sources (databases), CPR ontology and a GUI (Graphic User Interface). Figure 3.1 is an overview of the structure of the system.

The CPR ontology has two important functions: (1) It acts as a container which contains the standardized concepts and the relationships between them. These standardized concepts should cover as much the domain of CPR as possible. The users should just concern about the standardized concepts when studying and comparing the CPR data. (2) It provides information to access the local databases. It contains the local terminologies (dialects) which are the key words to query the databases. And it also contains the schema (structure) information of the databases in different EMS systems. Finally mappings between the dialects and the standardized concepts are defined in the CPR ontology. See the ontology part in figure 3.1.

In this data integration system if a user wants to find some CPR data in an unfamiliar/remote EMS system, he/she should just query the GUI with the standardized con-

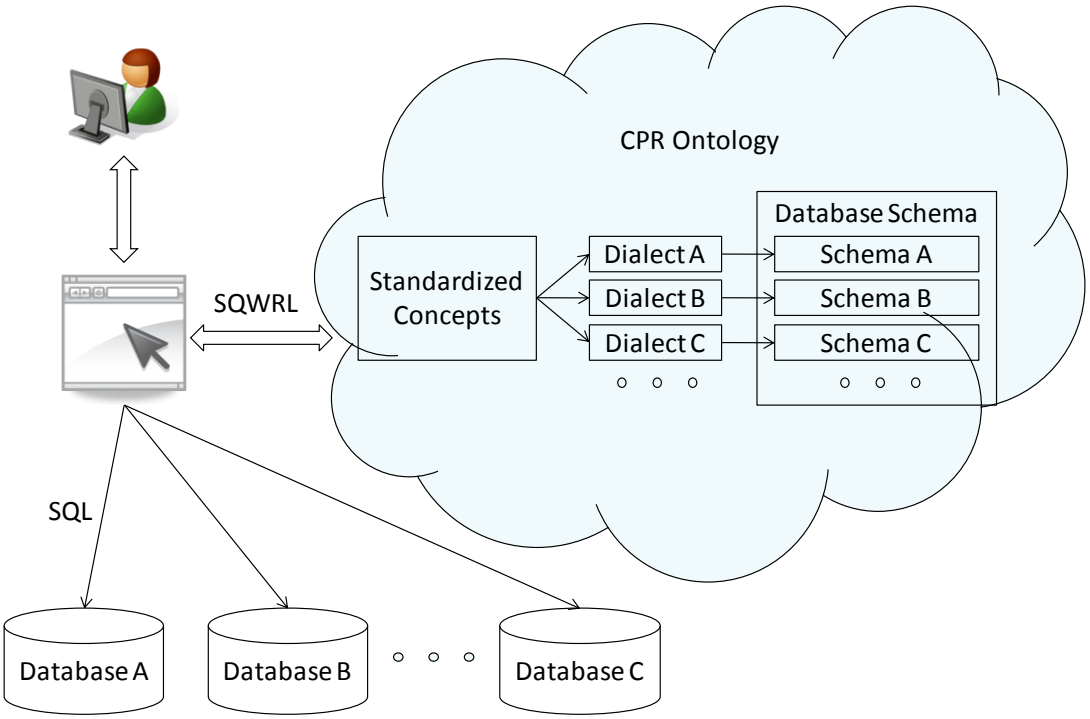


Figure 3.1.: Structure of this data integration system

cept. Then the GUI makes SQWRL queries against the CPR ontology to find the dialects and the schema information of the database in that EMS system. With this information the GUI can in turn query the database in that EMS system and return the CPR data to the user. So the role of CPR ontology in this system is like a standardized concepts container and a mapping information consultant.

3.2. Content of CPR ontology

An ontology consists of a set of classes, properties, and individuals. Classes are the definitions of concepts which place constraints on the individuals. Individuals are the instances of classes. Properties describe the relationships between classes and relate individuals to each other. In the CPR data ontology we defined 46 classes, 19 properties, and 332 individuals. And the OWL file of this ontology can be found here: `http://sourceforge.net/projects/cpr-ontology/files/`.

3.2.1. An overview of the ontology

Figure 3.2 is an overview of the class tree of this ontology in three levels (Thing → subclass of Thing → subclass of subclass of Thing). There are three most important

parts in the ontology: “Defibrillator_Log_Information” part, “Utstein_Style_Concept” part and database mapping part.

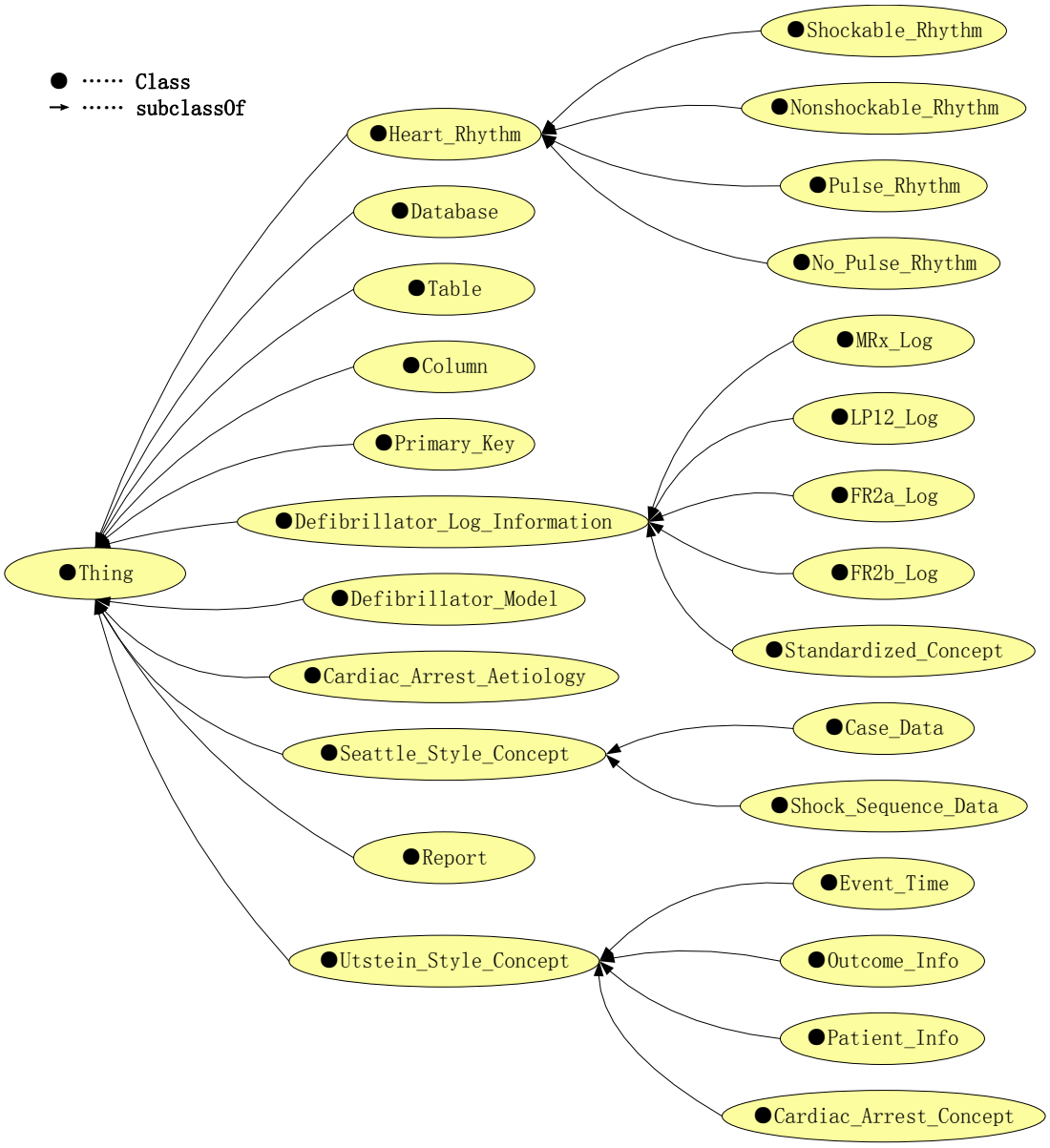


Figure 3.2.: An overview of the classes of the ontology

3.2.2. Standardized definition of the events in the defibrillator log files

In the CPR ontology we defined standardized concepts of the event and parameter names in the log files from different defibrillators. Figure 3.3 is extracted from figure 3.2 with additional information. It shows the details about the standardized concepts and their

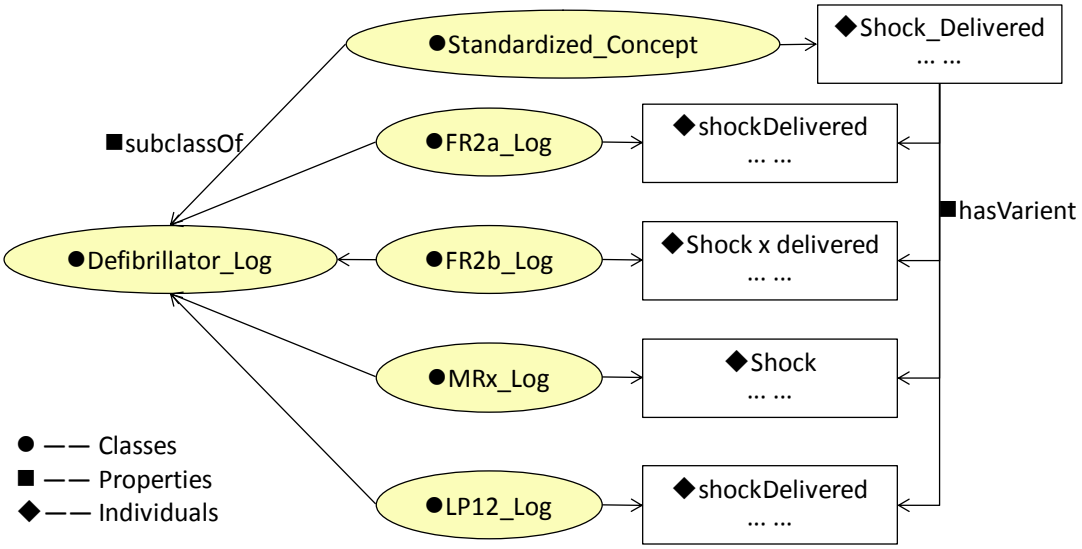


Figure 3.3.: Details about the "Defibrillator_Log" class.

individuals and how they relate to each other. In this figure there are six classes. Class "Defibrillator_Log" denotes all of the events in the log files from all kinds of the defibrillators, and it has five sub-classes. "Standardized_Concept", as its literal meaning, is a class of standardized event and parameter names. The other four classes "FR2a_Log", "FR2b_Log", "MRx_Log" and "LP12_Log" contain the individuals (event and parameter names) from the four different defibrillator models. The heterogeneity of the event names between different log files can be seen in this figure. Finally we mapped the different event names to the standardized event name "Shock_Delivered" with the property "hasVariant".

Figure 3.4 gives us a more intuitive view of the relation between the standardized concept and the concepts in different log files.

3.2.3. Utstein style definitions in the CPR ontology

As mentioned in the "Background" chapter, Utstein style is a set of guidelines for uniform reporting of cardiac arrest. Data produced in the process of resuscitation like key events, key times, patient information, and outcomes are uniformly defined in Utstein style report. These uniform CPR data difinitions can facilitate better communication and data management within hospitals and emergency medical services (EMS) systems.

In the project we added all the terminologies defined in the Utstein style specification [1, 26] into the CPR ontology. Figure 3.5 describes how the terminologies in the Utstein

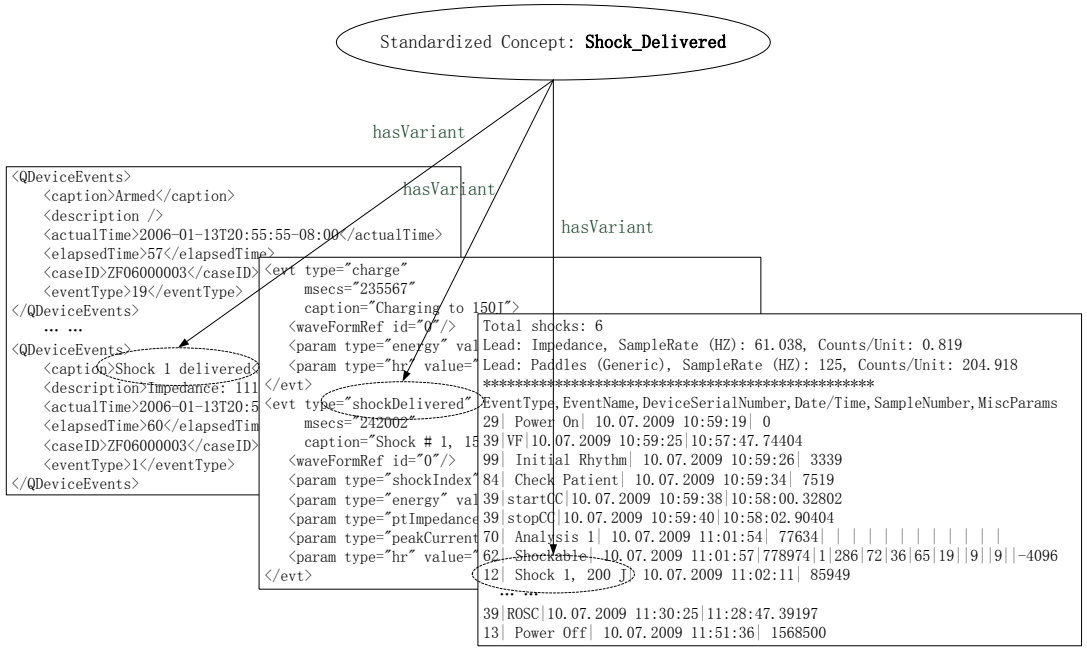


Figure 3.4.: Mapping various event names onto standardized event names.

style report template can be mapped onto the ontology. On the right side of figure 3.5 is a sample of Utstein cardiac arrest data collection form [1], and the contents in this form are all defined in the CPR ontology as individuals of class “Utstein_Style_Concept”.

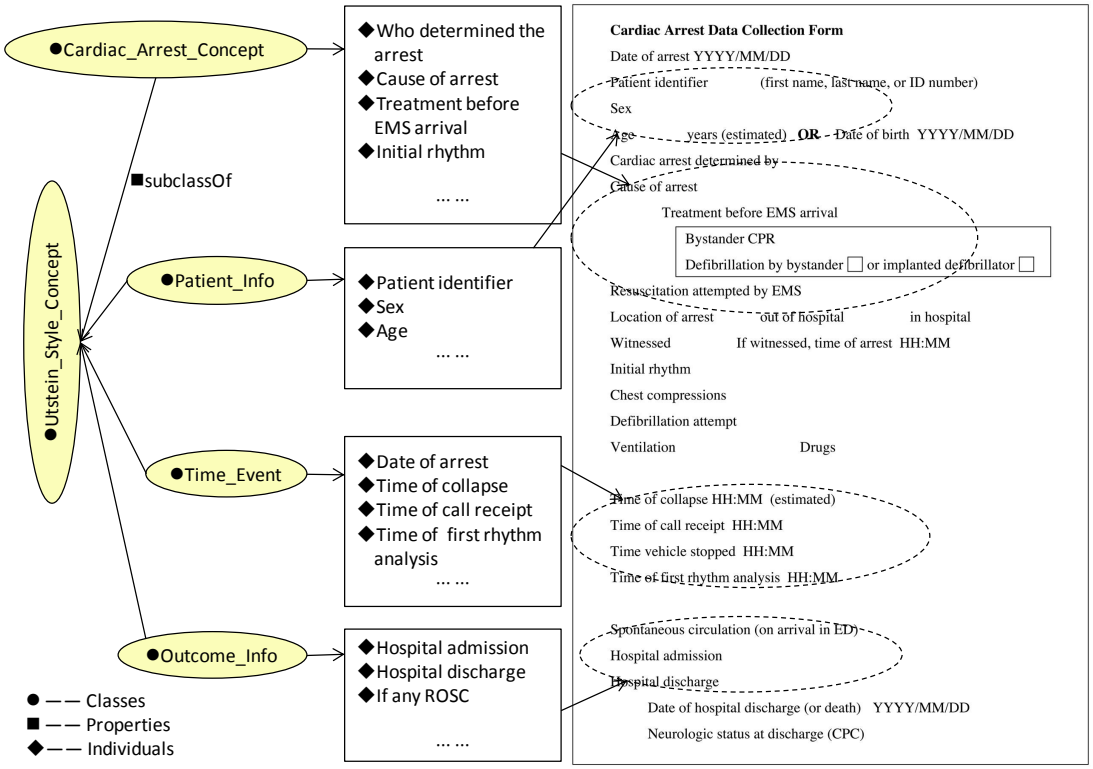


Figure 3.5.: Mapping the items of Utstein style report onto the ontology.

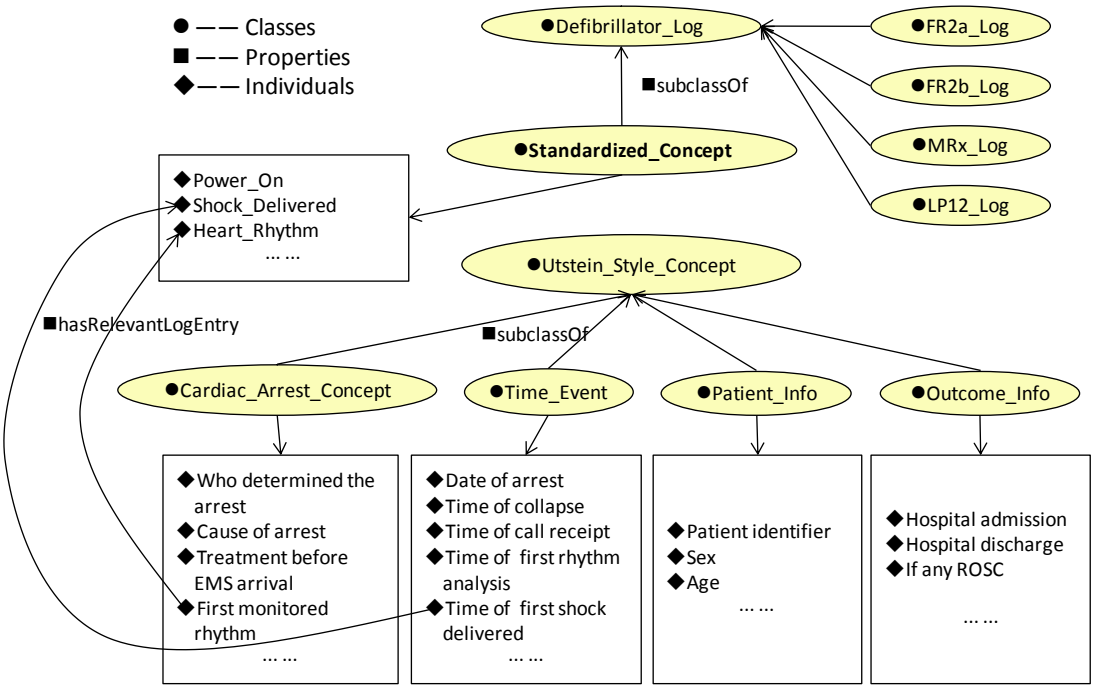


Figure 3.6.: Using the property "hasRelevantLogEntry" to relate the Utstein style items to the standardized defibrillator log events

Relationship between the Utstein style and the standardized definition of events in the defibrillator log files:

In practice some of the events and their parameters in the defibrillator log files can be used to help us finish an Utstein report. Now we want to make use of the ontology to collect data from the log files to fulfill some items in a Utstein report. To achieve this, we have to define the mapping between items in the Utstein style report and the standardized definition of events in the defibrillator log files. Figure 3.6 gives an example of this mapping. In this figure we can see property "hasRelevantLogentry" is the bridge between items in the Utstein style report and events in the log files. For example, if we want to find the information about "First monitored rhythm" in the Utstein style report we can check the event "Heart Rhythm" in the defibrillator log files; and information about "Time of first shock delivered" can be found in the "Index" parameter of the event "Shock Delivered".

Relationship between the Utstein style and Seattle style report:

Although the Utstein style report is defined by experts and researchers to be the standard report style, most of the hospital and EMS systems are still using their own report style at present, like "Seattle style". So for comparing and studying CPR data between hospitals, the transformation between other report styles and the Utstein style is needed. In

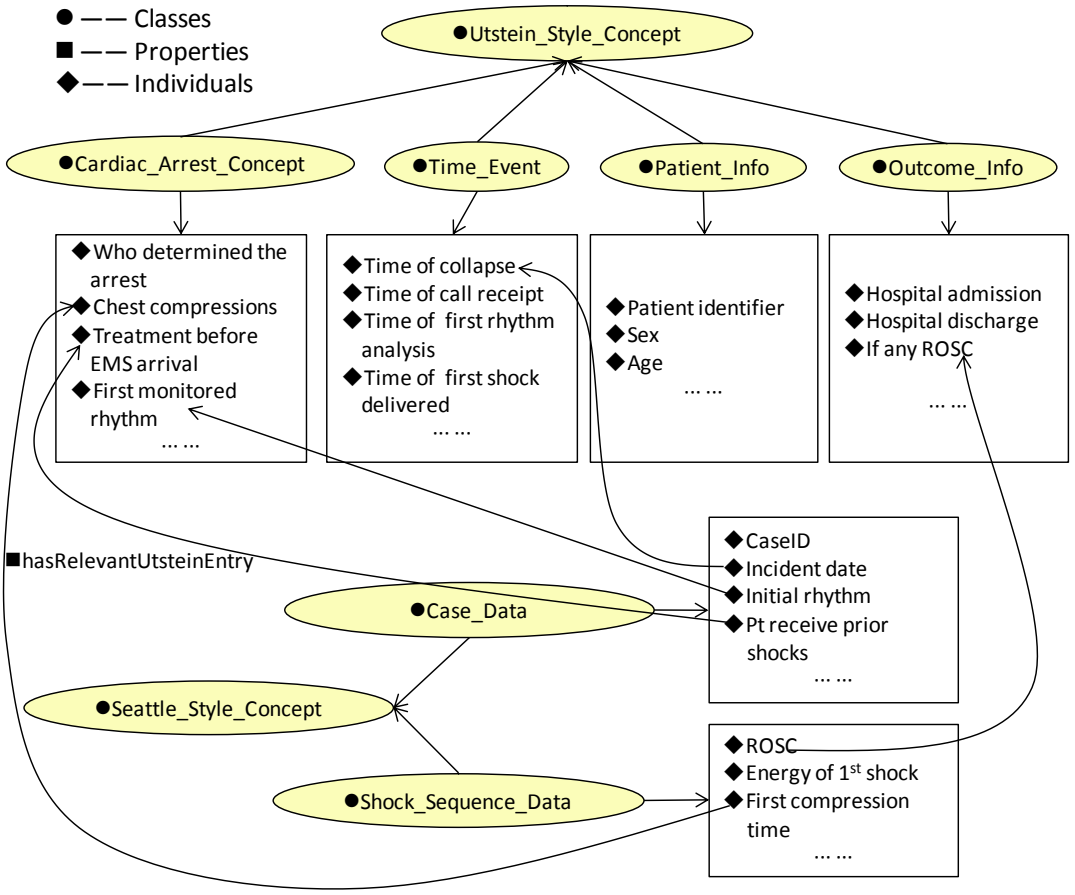


Figure 3.7.: Using the property "hasRelevantUtsteinEntry" to relate the Seattle style items with the Utstein style items

this ontology we defined the items in the Seattle style report, and the property "hasRelevantUtsteinEntry" are used to map the Seattle style items with the Utstein style items. From figure 3.7 we can see the mapping between the two report styles. For example, the item "Initial rhythm" in Seattle style report has the same meaning as the item "first monitored rhythm" defined in Utstein style report, and the item "ROSC" refers to "If any ROSC".

3.2.4. Database mapping part

As there are huge quantity of data produced in the process of CPR, databases are the first choice to save the CPR data. In this data integration system, a function of the CPR ontology is to facilitate accessing data which reside in different databases. So an interface between the ontology and databases are needed. Figure 3.8 shows this interface defined in the CPR ontology. It comes from the "Relational.OWL" ontology proposed in [27]. It defines the essential classes of a database: "Database", "Table", "Column"

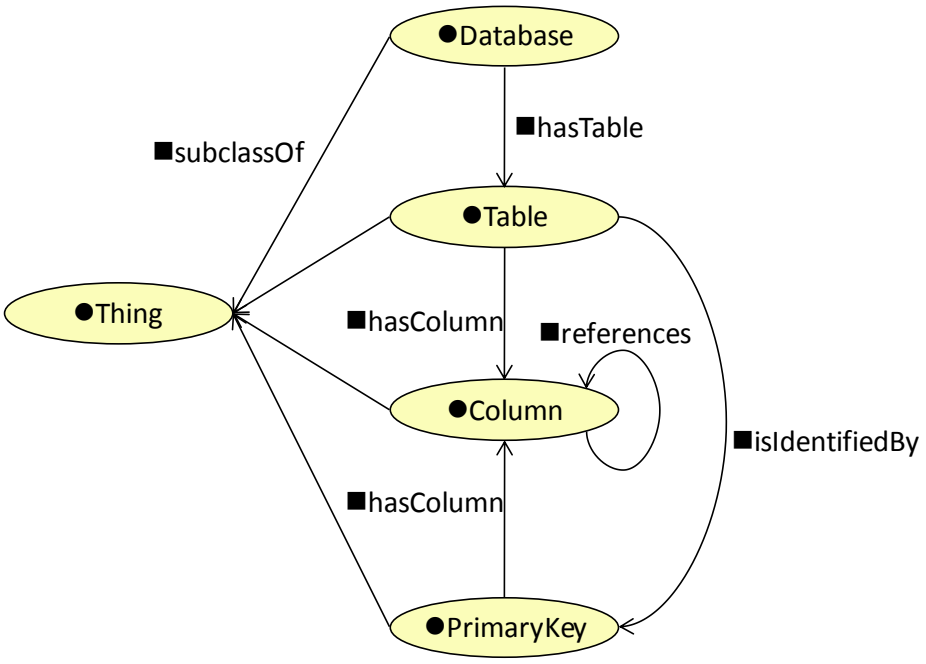


Figure 3.8.: Databases mapping part in the ontology.

and "PrimaryKey". And properties: "isIdentifiedBy", "references", "hasColumn" and "hasTable" are defined to give the relationships between the classes. With these definitions a schema (structure) of a database can be mapped onto the ontology. More details of database mapping will be given in chapter 4.

3.3. Query the ontology

The ontology is like a container collecting all kinds of information (e.g. standardized definitions, mappings etc). If we want to retrieve this information we need ontology query languages. A number of ontology query languages have been developed such as SPARQL [17] and SQWRL [22], and we chose SQWRL in our system.

3.3.1. SPARQL: a RDF (Resource Description Framework) query language

SPARQL (SPARQL Protocol and RDF Query Language) is a W3C Candidate Recommendation [17]. It is currently the de facto standard RDF query language. SPARQL has SQL like syntax, which makes it very easy to use for developers. However SPARQL is essentially a RDF query Language, and it has no native understanding of OWL [28] (Web Ontology Language, the standard recommended by W3C for developing ontolo-

?Event	?Param
mrx:shockDelivered	mrx:energy
lp12:Shock	lp12:Energy

Table 3.1.: The result returned by the example query

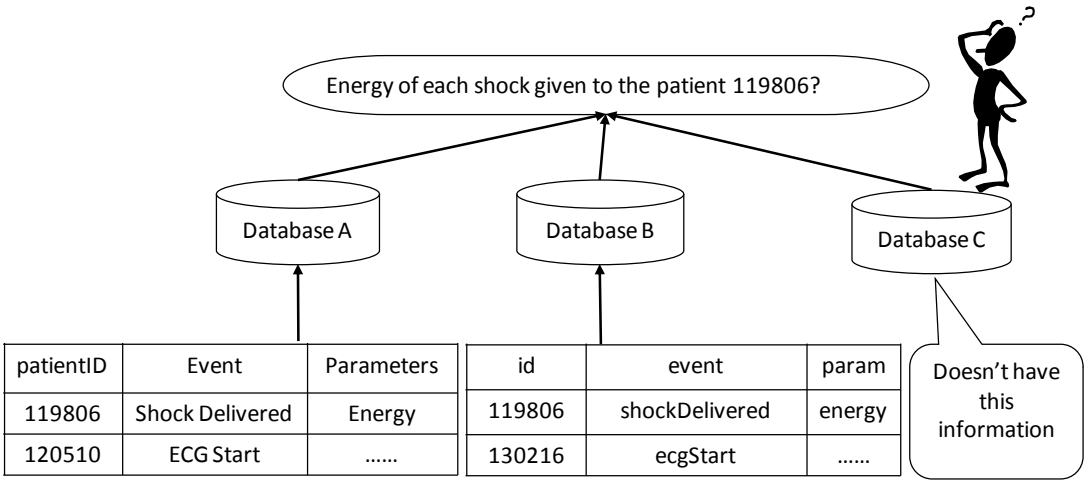
gies). In practice SPARQL has sometimes been pressed to service as an OWL query language since OWL files can be serialized into RDF. But it is in nature not suitable for querying the ontology developed with OWL, as information may be changed or lost in the serialization.

3.3.2. SQWRL: a Query Language for OWL

SQWRL (Semantic Query-Enhanced Web Rule Language) is a SWRL-based [23] language for querying OWL ontologies. Different from SPARQL, SQWRL has native understanding of OWL. So in our system we chose SQWRL to query the mapping information from the CPR ontology. The following, for example, is a simple query using SQWRL to find the variants (dialects) of the standardized event and parameter names "Shock_Delivered" and “Energy”.

$$\text{hasVariant(com:Shock\_Delivered, ?Event)} \wedge \text{hasVariant(com:Energy, ?Param)} \wedge \text{hasParameter(?Event, ?Param)} \rightarrow \text{sqwrl:select(?Event, ?Param)}$$

This query is based on the relationships in figure 3.3, and the results are the different event and parameter names mapped onto the standardized definition “Shock_Delivered” and “Energy”, see table 3.1. This example provides a glance of the syntax of SQWRL. In query sentences above we have three constraints: (1) Event “com:Shock_Delivered” has variant “?Event” (where, “com” is the namespace to avoid reduplicated naming, and “?” stands for variables); (2) Parameter “com:Energy” has variant “?Param”; (3) Event “?Event” has parameter “?Param”. Then we select the variables “?Event” and “?Param” which satisfy all of the constraints.
In applications SQWRL queries are made through a set of Java APIs (Application Program Interface). The introduction and example usages of these APIs can be found on the website: `http://protege.cim3.net/cgi-bin/wiki.pl?SQWRLQueryAPI`.
We also write a Java class using SQWRL APIs to implement the example queries appeared in this thesis. And this Java class is also used in the GUI (Graphic User Interface), which will be introduced in chapter 5, of this data integration system. The source code of this Java class can be found in Appendix A.



patientID	Event	Parameters
119806	Shock Delivered	Energy
120510	ECG Start	.....

id	event	param
119806	shockDelivered	energy
130216	ecgStart	.....

Figure 3.9.: The problem in the scenario.

3.4. Using CPR ontology, a scenario

In this section we will give a scenario to demonstrate how the CPR ontology can be used to integrate CPR data.

Problem:
A user wants to know the energy of each shock given to the patient with patient ID 119806. (See figure 3.9) To answer this question the doctor should know:

- Which database to query, database of hospital A, hospital B or hospital C?
- What is the schema of the database, tables, and columns?
- What is the key word used to query the database, "Shock Delivered", "shockDelivered" or "Shock"?

Solution:
The solution for this scenario is also a perspective of the main contribution we have done:

- Define an ontology to provide standardized terminologies, and map each variants onto them. Thus the user should only get to know the standardized key words. (See figure 3.3 and 3.4.)
- Map the databases onto the ontology. (See figure 3.8.) The ontology should indicate that the information "Shock_Delivered" is only saved in database A and B, so the user don't have to waste time on database C.

- Provide a user interface to help the user be out of the trouble from writing the query sentences (Queries both against the ontology and against the databases are needed).

The query statements for the CPR ontology in this scenario are shown below:

hasVariant(com:Shock_Delivered, ?Event) ∧ hasVariant(com:Energy, ?Param) ∧
hasParameter(?Event, ?Param) ∧ dbs:hasTable(?db, ?tab) ∧
dbs:hasColumn(?tab, ?eCol) ∧ dbs:hasColumn(?tab, ?pCol) ∧
hasDBColumn(?Event, ?eCol) ∧ hasDBColumn(?Param, ?pCol)
→ sqwrl:select(?Event, ?Param, ?db, ?tab, ?eCol, ?pCol)

The sentence “sqwrl:select” decides which variables should be returned as the result of this query. Like the query example in section 3.3.2, the meaning of the symbols is as follows:

- “?Event” stands for different event names in the log files which have been mapped onto the standardized definition “com:Shock_Delivered”;
- “?Param” stands for the paramater for each “?Event”;
- “?db” are the name of the databases containing the data wanted;
- “?tab” are the relevant tables in the databases;
- “?eCol” is the “event” column in the table where users can query the key word “?Event”
- “?pCol” is the “parameter” column in the table where users can query the key word “?Param”
- “∧” symbol collects the statements together, and it means all of the statements shoud be fulfilled.

Figure 3.10 shows the result of the query in this scenario. It is produced by the “SWRL Rules” tab in Protégé platform. The results in figure 3.10 tell us: to get the information of the standard event “Shock_Delivered”, we can query the key words “shockDelivered” and “Shock” in column “eventName” in database “TestDB1”, and query the key word “Shock” in column “eventName” in database “TestDB2”. The same method can be used to query the information of the standard parameter “Energy”.

SQWRLQueryTab | Rule-10

?vEvent	?vParam	?db	?tab	?eCol	?pCol
lp12:Shock	lp12:Energy	uisdb:TestDB2	uisdb2:event	tevent2:eventName	tevent2:eventType
mrx:shockDelivered	mrx:energy	uisdb:TestDB1	uisdb:eventPara	tevent:eventName	tpara:type
lp12:Shock	lp12:Energy	uisdb:TestDB1	uisdb:eventPara	tevent:eventName	tpara:type

Save as CSV... | Rerun | Close

Figure 3.10.: The result of the query in the scenario.

An overview of the system is shown in figure 3.11. Before the user queries the databases he/she first queries the CPR ontology, and the ontology returns the information: which databases have the data the user wants, and how to query the database (the schema and key words). With this information the user can query the databases and get the data. And a graphic user interface is provided to produce all the query sentences for the user. This will help if the user isn't familiar with the SQWRL and SQL query languages.

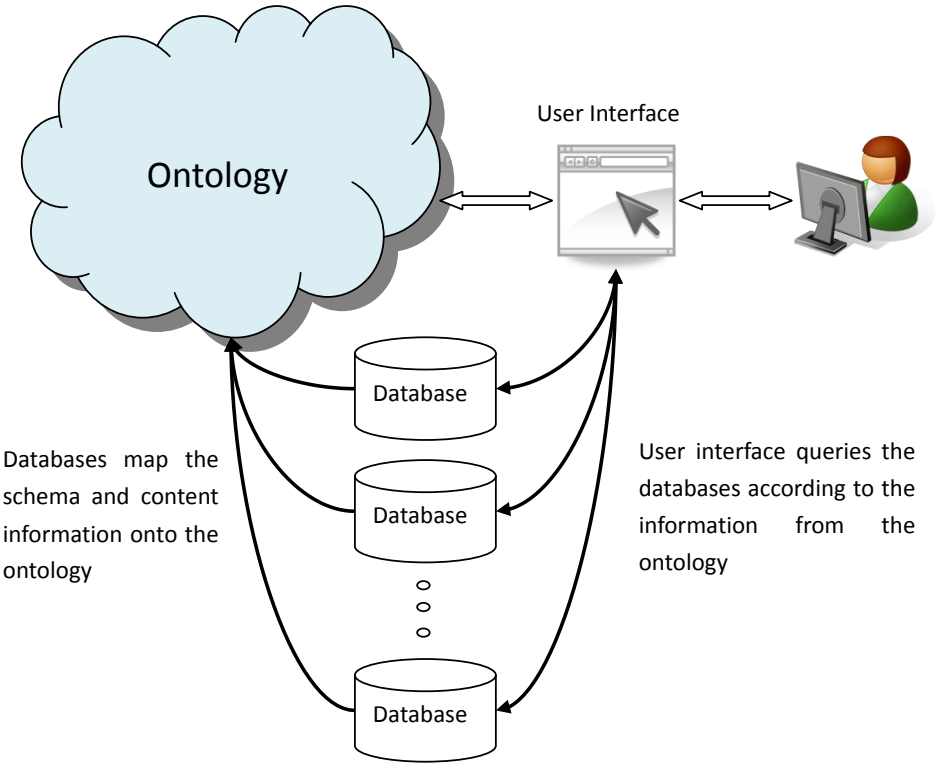


Figure 3.11.: An overview of the system

Conclusion:

With the help of the ontology the user should just query the user interface with the standardized key word “Shock_Delivered”and “Energy”. Data will be collected from databases and returned to the user.

3.5. Extensibility of the CPR ontology

An ontology is a represention of a domain of interest. As the world changes every day, some concepts in the domain of interest change. So it isn’t possible for an ontology to stay the same all the time. It needs to be updated frequently. At present the CPR ontology is too small. A lot of other definitions, terminologies and relationships in the domain of CPR should be added into this ontology. The final target is to make the CPR ontology cover most of the knowledge in the domain of CPR.

In section 3.2 we have discussed the three most important parts of the CPR ontology. However some more classes defined in the CPR ontology haven’t been introduced. These classes are also important concepts in the domain of CPR and may be used in the future. In figure 3.2 we can have a view of these classes. “Heart_Rhythm” class defines the terminologies of heart rhythm like “VF” (Ventricular fibrillation, a condition in which there is uncoordinated contraction of the cardiac muscle of the ventricles in the heart, making them quiver rather than contract properly). “Cardiac_Arrest_Aetiology” class defines the concepts of the aetiology of the cardiac arrest like “Submersion”.

3.5.1. Extensible nature of OWL

In programming we use “include” in C and “import” in Java to add other source code into the current file. In the ontology description language OWL “owl:imports” elements can be used to add the content of other ontologies to be part of the current ontology. And “owl:imports” is a transitive property: if ontology A imports ontology B, and ontology B imports ontology C, then ontology A also imports ontology C. In Protégé platform there is no difference in the view of the “class tree” if we import an ontology or we just define the ontology by ourselves. The extensible nature of OWL makes it quite convenient to extend and update the ontologies.

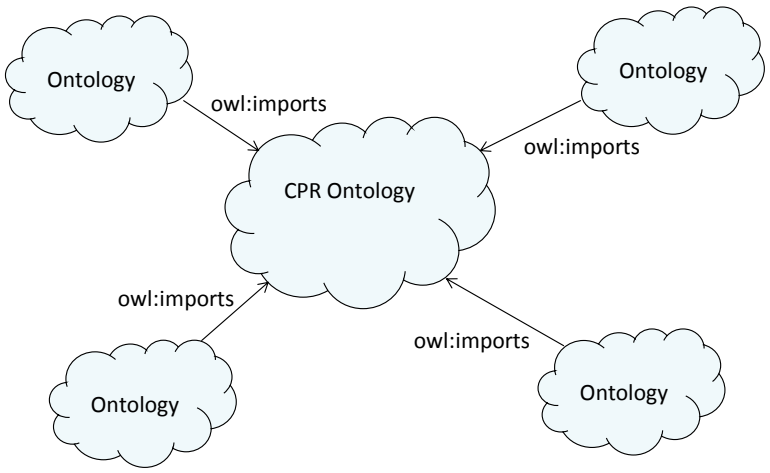


Figure 3.12.: Structure of the extensible CPR ontology.

3.5.2. Extensibility of the CPR ontology

With the extensible nature of OWL the usage of the CPR ontology can be very flexible. This CPR ontology can be considered as a module and imported into some bigger ontologies like ontology describing a medical system. Or it can act like a central ontology. EMS systems from different sites can build their own ontologies containing definitions of their own defibrillator models and report styles. These ontologies can be then imported to this CPR ontology. (See figure 3.12)

4. Mapping Databases onto Ontology

In section 3.2.4 we have briefly introduced the database mapping part in the CPR ontology. In this chapter we will first review the related work has done to relate the databases to the ontologies and introduce our method used in this data integration system. Then we will give the details how to map databases onto the CPR ontology.

4.1. Related work

Methods to build the relationship between database and ontology will be represented in this section. All of these methods aim at accessing databases through ontology.

Bizer introduced a mapping language D2RQ to map legacy relational databases schema to RDFS/OWL ontologies [29]. The D2RQ platform uses these mappings to enable applications to access a relational database through the Jena APIs [30]. In other words, with D2RQ we can query a relational using the SPARQL/RDQL query language (instead of SQL). To achieve this, SPARQL/RDQL queries are first rewritten into SQL queries, and the result sets of these SQL queries are transformed into RDF triples which are then delivered to the higher layers of the Jena framework. D2RQ provides a semantic way to access data stored in relational databases.

Cristian Pérez de Laborda and colleagues developed an ontology "Relational.OWL" [27] to describe the relational database schema. In [31] they represented their method to map a database to semantic web in two steps: First, export the schema of the relational database to RDFS as an instance of the Relational.OWL ontology. Then export the data entries in the database to RDF model which becomes the instances of the RDFS just created. Second, use the "CONSTRUCT" clause of SPARQL query language to query the RDF model, and the resulting data entries can be inserted into an arbitrary RDF skeleton (the target ontology). Finally the target ontology can be processed by

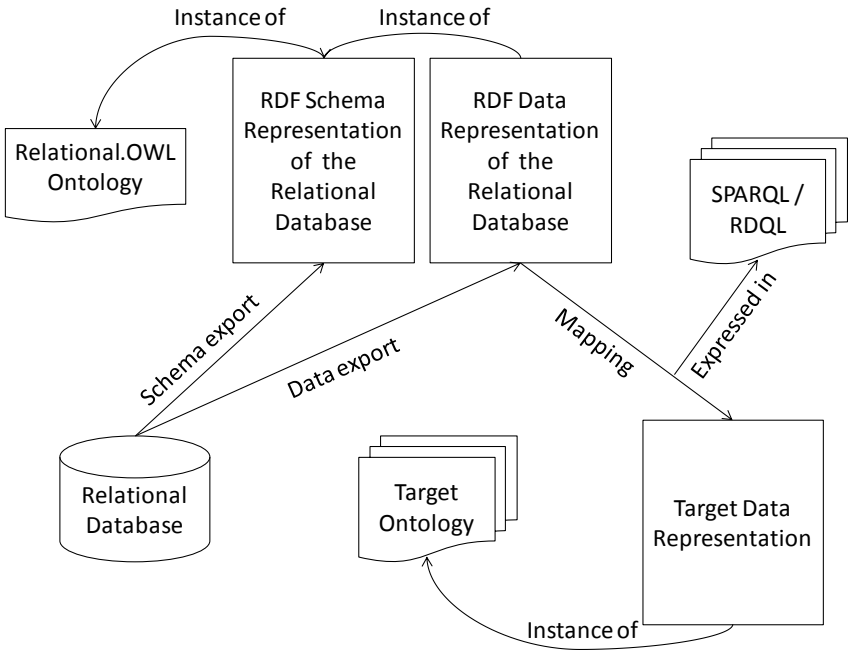


Figure 4.1.: The mapping process in Cristian's method.

any Semantic Web application as usual. In the first step a tool RDQuery [32], which automatically translates SPARQL or RDQL queries into SQL, can be used to export the whole or part of the database to RDF model. Figure 4.1 [31] shows steps in Cristian's method.

DataMaster [33] developed by Csongor Nyulas and colleagues in Stanford University is a plug-in of Protégé for importing schemas and data from relational databases into ontologies. This plug-in also makes use of the Relation.OWL ontology. It defines classes to map the columns of the database and entries in the database can be exported as individuals of the classes.

Huajun Chen [34] and colleagues developed "Dartgrid" which is an application development platform together with a semantic toolkit to help integrate heterogeneous relational databases using semantic web technologies. In their project a set of tools are developed: DartMapping visually maps heterogeneous relational schemas to ontologies; DartQuery helps users to construct semantic queries, and rewrites SPARQL semantic queries to SQL queries; DartSearch, an ontology-based search engine, makes full-text search in all databases and can navigate across the search results semantically. In practice they have used this set of tools to integrate the data in the legacy relational databases about traditional Chinese medicine.

Here is a review of the four methods mentioned above. Bizer firstly gives us an approach to access relational data with semantic query languages like SPARQL. Cristian first exports relational data to RDF models and then constructs the target ontology with the "CONSTRUCT" clause of SPARQL. Both of the methods based on RDFS, RDF and SPARQL, however in our application we use OWL and SQWRL to describe and query the ontology. In Csongor's method relational data can be imported into the ontology as individuals with DataMaster, and then it can be queried by SQWRL. However if the amount of data is huge and the database is updated frequently, it's not a good idea to transfer all the data from database to the ontology as individuals. At last Huajun's toolkit seems to be able to solve all of the problems but it's not available and we cannot get the details of the implementation. So we didn't completely use either of the methods. In the next section we will present our method. Maybe it is not the best solution, but it works in our application.

4.2. Using Relational.OWL to mapping databases onto ontology

In our application we defined a series of standard terminologies about CPR data and mapped the "dialects" in different EMS systems to the standard terminologies. If a user wants to query some information he/she should first have the idea where he/she can get the information (which databases to query) and then he/she needs to know the key words used to query. The key words are the "dialects" which have already been mapped onto the standard terminologies. To answer the question which databases to query, the databases should be mapped on to the "dialects" respectively. So firstly the user makes SQWRL queries with the standard terminology in the ontology to find the "dialects" (key words) and their related databases, and then with this knowledge SQL queries can be made to get the data he/she wants to know.

4.2.1. Using the modified Relational.OWL to describe the schema of databases

Relational.OWL [27] is used to describe the schema of databases. Cristian defined Relational.OWL with OWL-Full, so that it can represent three layers: Relational.OWL itself (meta-classes / classes of classes), schema of the relational database (classes), and the concrete data in the databases (individuals of classes). In our application we just

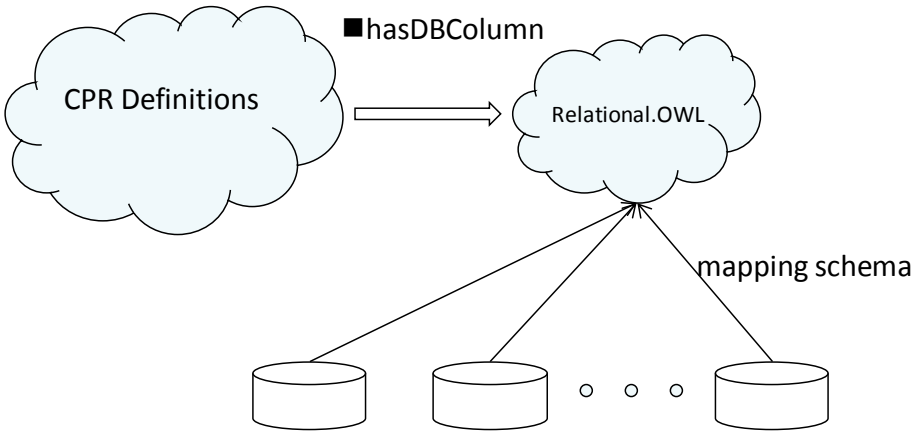


Figure 4.2.: The role of Relational.OWL in the CPR ontology.

need to map the relational schema onto the CPR ontology, in addition SQWRL doesn't support OWL-Full, so we simplified Relational.OWL with OWL-DL. The structure of the modified Relational.OWL is shown in figure 3.8, section 3.2.4. In our application Relational.OWL is imported as a part of the CPR ontology and acts as an interface to relate databases to the CPR concepts. See figure 4.2, the property "hasDBColumn" collects the CPR definitions and Relational.OWL. The next section will show the details about the mapping.

4.2.2. Mapping details

In the project we built two databases which store the events from different defibrillator log files. Now we want to relate the event names in the log files to the corresponding columns in the databases. Figure 4.3 is from part of the CPR ontology and shows the details of this mapping. On the lower side of this figure are the definitions of the log file events. On the upper side of the figure is definition of the schema of the two databases. (Here with the help of the Relational.OWL ontology, schema of heterogeneous databases can be easily described in the CPR ontology.) The bridge to relate the upper and the lower part is the property "hasDBColumn", the two bold arrows in figure 4.3. The semantic is: events names in the log files "FR2a", "FR2b" and "MRx" can be found in the column "eventName" in table "event" in database "TestDB1"; and the events names from "LP12" are stored in the column "eventName" in table "event" in database "TestDB2".

In practice different EMS systems can map their databases onto the CPR ontology. In this chapter we just made an example to demonstrate the mapping on the events in the

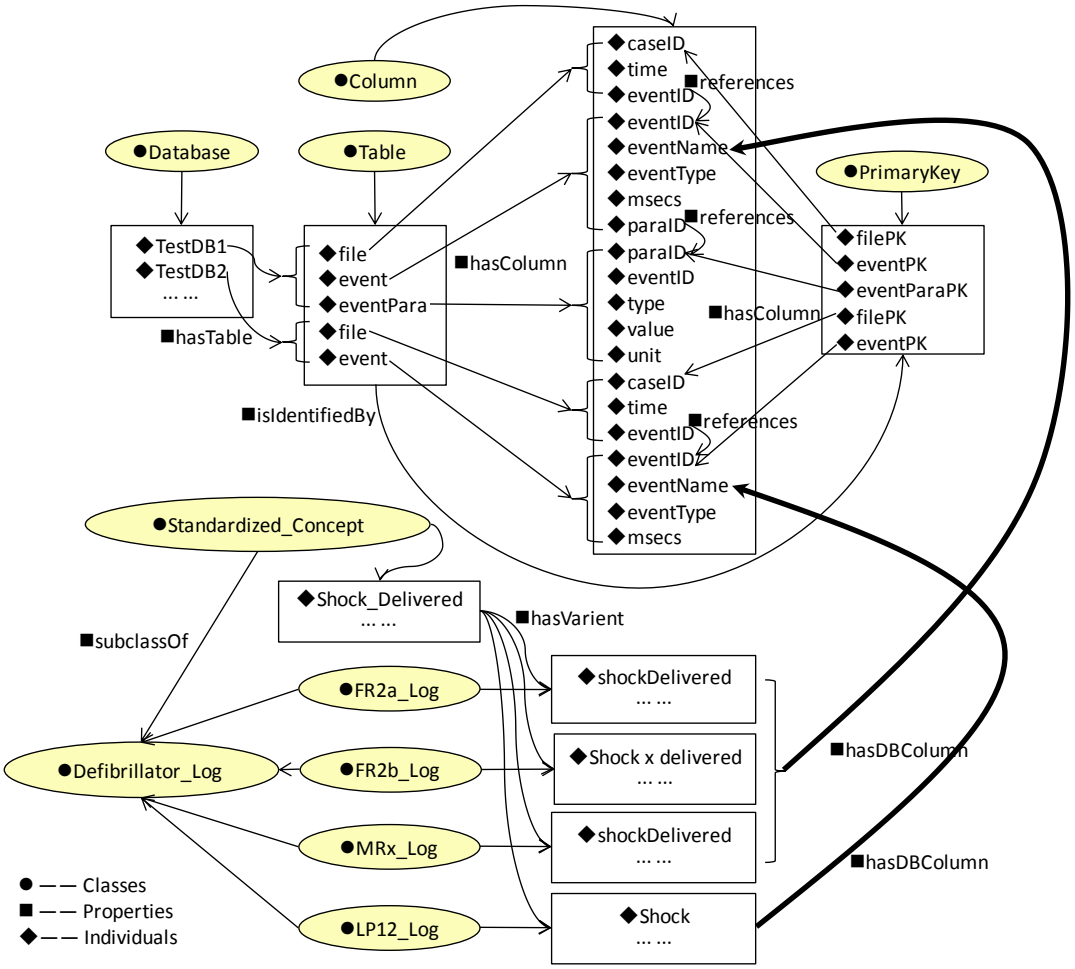


Figure 4.3.: Mapping the databases onto log file events defined in the CPR ontology.

defibrillator log files. In addition databases which store the items of the CPR reports (e.g. Seattle report) can be mapped onto the report definition part of the CPR ontology as individuals of classes “Utstein_Style_Concept” and “Seattle_Style_Concept”.

5. User Interface and Tests

In the last two chapters we defined the CPR ontology and mapped two local databases onto the ontology. In this chapter a graphic user interface (GUI) will be introduced and some query examples are made to show whether each part of the system works as expected. This part of work is done by my partner Chao Li.

5.1. Graphic user interface

To test the performance of this ontology-based data integration system and facilitate users to query the CPR data, a graphic user interface is built. The main functions of this GUI are: receiving the parameters input by the user, query the ontology and then query the databases with the information returned from the ontology, and finally return the results (entries returned from the databases) to the user. This process can be seen in figure 5.1.

This GUI is developed with Java programming language. It runs on any platform which has a Java virtual machine. Figure 5.2 gives us an overview of this GUI. On the top part of the panel is the area for the user to input parameters. The text field below is the area for displaying the results of the queries.

5.2. Test queries and the results

In the test we want to find the energy of each shock given to the patient with a specified case ID. The user must input the case ID, the event name and the parameter name. The case ID in this test is randomly chosen, the event name can be “Shock_Delivered”, and the parameter name is “Energy”. Here the event name and parameter name are both the standardized concepts defined in the CPR ontology.

We made two queries with two different case IDs. Figure 5.3 and 5.4 shows the results of the queries from two different databases. The results in figure 5.3 are from defibril-

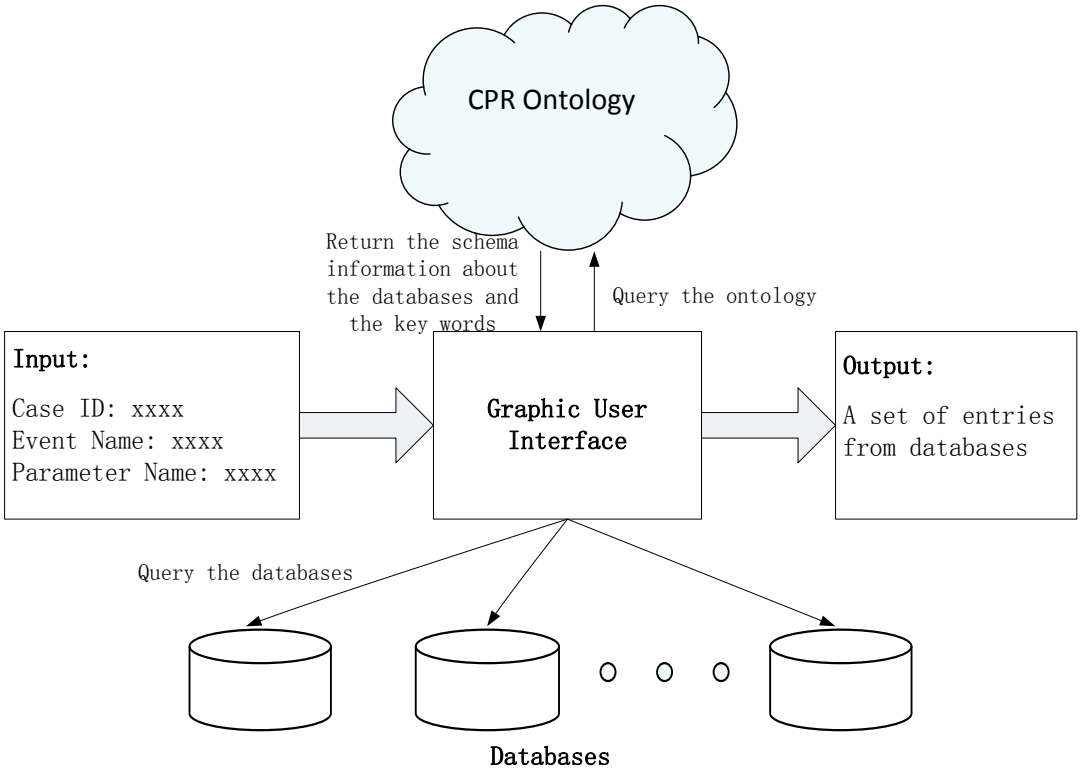


Figure 5.1.: The function of the GUI.

Figure 5.2.: A view of the GUI.

lator model MRx stored in database “TestDB1”, and the results in figure 5.4 are from defibrillator model LP12 stored in the database “TestDB2”. So the user should just query the standard key words “Shock_Delivered” and “Energy”, and the related CPR data can be found in different databases and presented to the user.

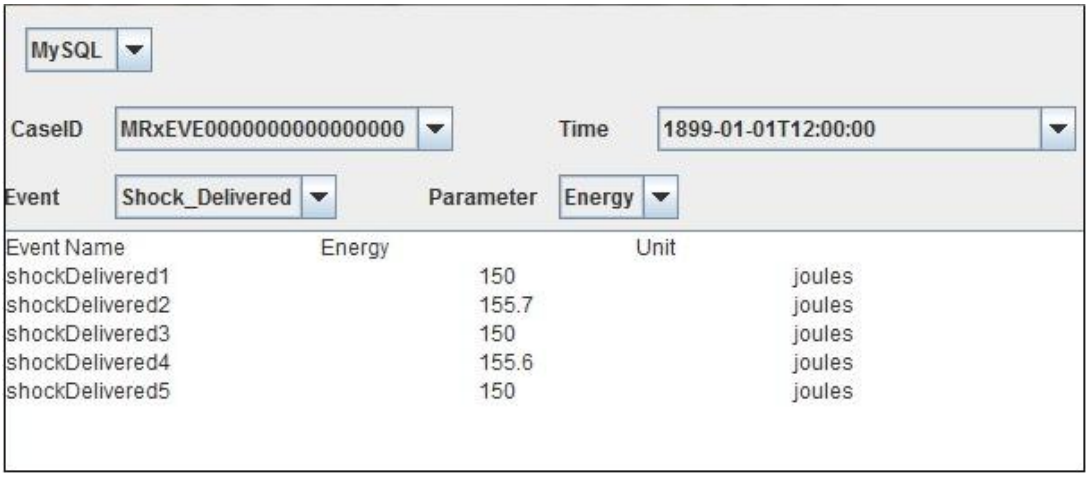


Figure 5.3.: A query example with the GUI. The results give the following information: The MRx dialect of the standard concept “Shock_Delivered” is “shockDelivered”. In this case there are five shocks delivered to the patient, and the energy of each shock is 150J, 155.7J, 150J, 155.6J and 150J.

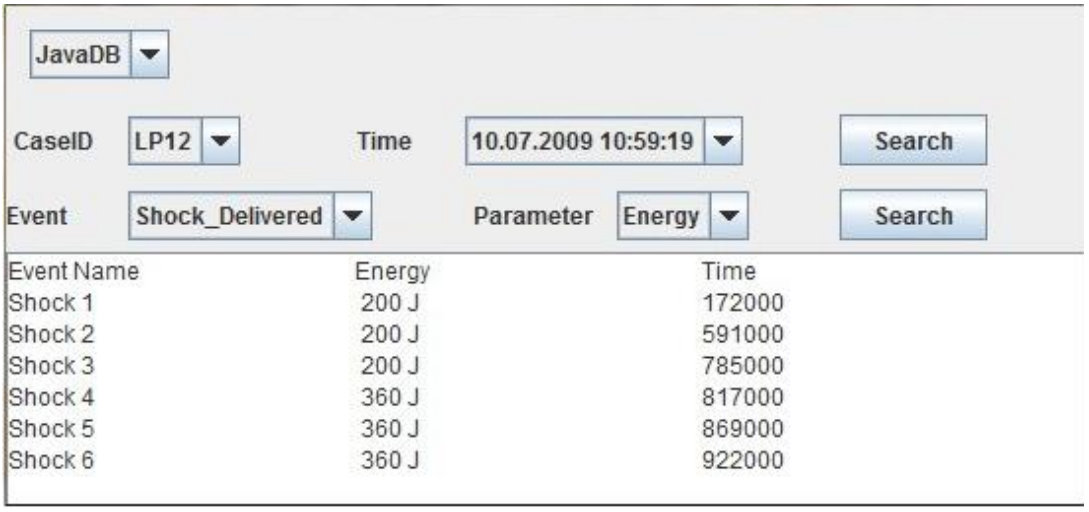


Figure 5.4.: Another query example with the GUI. The results give the following information: The LP12 dialect of the standard concept “Shock_Delivered” is “Shock”. In this case there are six shocks delivered to the patient. The energy and the timestamps are attached to each shock.

6. Conclusion and Future Work

In this chapter we will first make a conclusion of this work. Then we will discuss the disadvantages with this data integration system at present, and suggest the direction for further study.

6.1. Conclusion

In this thesis we built a CPR (cardiopulmonary resuscitation) data integration system. This system standardizes the concepts of the different CPR data in different EMS systems, provides an interface to map different databases, and built a user interface to query the CPR data information. The users can query the standard concepts then all the data in different databases will be collected and delivered to the user.

An important part of this data integration system is the CPR ontology . The CPR ontology contains three parts: the definitions of the standardized concepts of variant event names in the log files from different defibrillators the items defined in the Utstein style report and a report style used in Seattle, and the modified Relational.OWL ontology used for describing the schema of relational databases. In the system the role of the CPR ontology is like a standard concepts container and a mapping information consultant.

We built two databases saving the events in the log files from different defibrillators and mapped them onto the CPR ontology, and we also built a graphic user interface (GUI). Finally the user can just query the standardized concepts against the GUI, the later will query the CPR ontology first and then query the database with the information return from the ontology, and the relevant entries will be collected from the databases and returned to the user.

6.2. Future work

At present the CPR data integration system has not been used in practice, and there is a lot of future work to do. First of all the CPR ontology should be extended all the time. Events and their parameters of new defibrillator models can be added in the ontology and mapped to the standardized event concepts. And now we only have the standardized and recommended report style Utstein and the Seattle report style defined in the ontology, so the definitions of other report styles used in other hospitals or EMS systems can be added in the CPR ontology. The general target is to make the CPR ontology covers as much knowledge in the domain of cardiopulmonary resuscitation as possible, and make the CPR ontology a standard and consultant for users and applications.

Developing a more full-fledged graphic user interface is also an important part in the future work. The user interface used currently can only make very limited queries. Moreover it lacks of extensibility, i.e. if a new database is added in the system, additional Java code should be written to deal with this specific newly added database. So we need to write some program for general use which is capable for any newly added the relational databases.

Finally, a drawback of this system is that every time we map a new database onto the CPR ontology we need to add the relevant dialects (local terminologies) and database schema information into the CPR ontology and map them onto the standardized concepts. That means we must modify the content of the ontology, which brings a lot of trouble for maintaining the ontology. In addition modifying an ontology is a professional work which needs knowledge of computer science. To improve it, we can develop a user interface to facilitate updating the CPR ontology, or we can try an alternative strategy which maintains the mapping information locally.

Figure 6.1 shows the system structure of the alternative strategy for this CPR data integration system. In this strategy the CPR ontology only contains the standardized concepts and the relationships between them. It doesn't contain the mapping information anymore. Instead, the mapping information is defined and maintained at local EMS systems. (Comparing figure 3.1.) With this strategy the user should query the standard key words directly against the local EMS systems, the later can find the dialects according to their mapping sheets respectively, and finally return the results to the user. So in this system we don't need to update the CPR ontology if a database is newly added. However all of the local EMS systems should adopt the definitions in the CPR ontol-

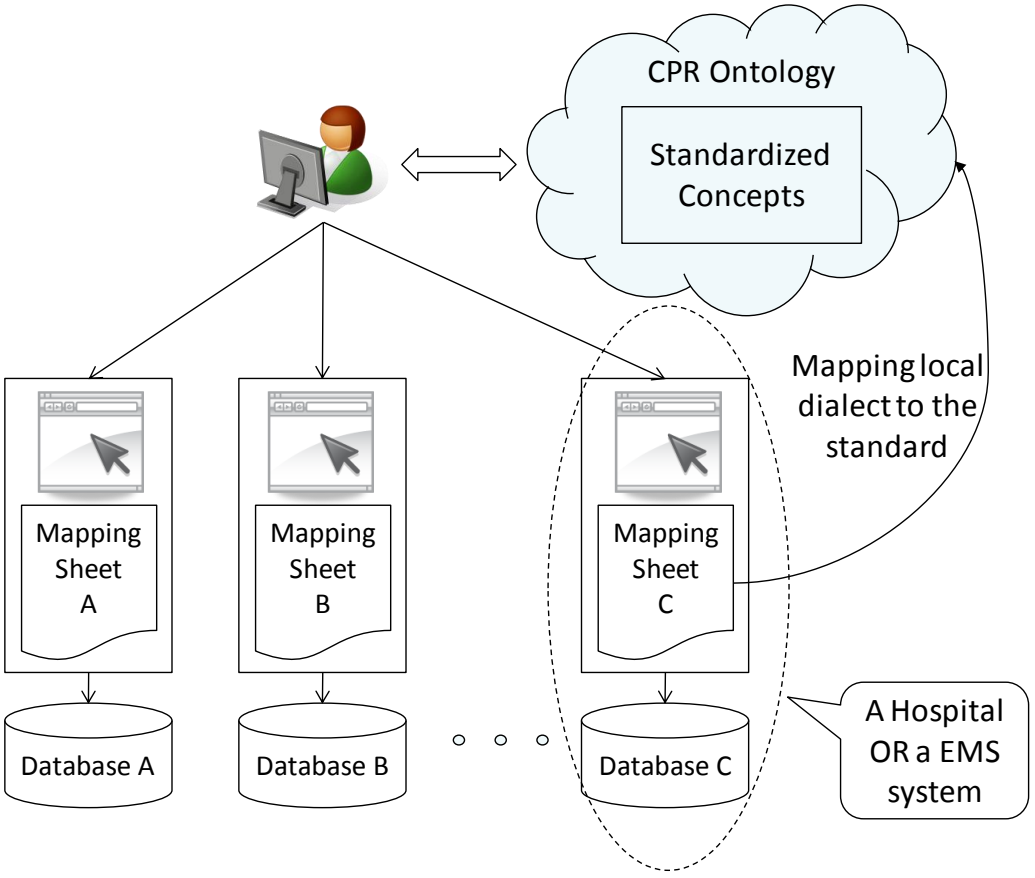


Figure 6.1.: The system structure of an alternative strategy.

ogy as their standard, make a mapping sheet, and develop a user interface based on the standardized concepts. (In our present system local EMS systems needn't to do these things.)

Bibliography

[1] I. Jacobs, V. Nadkarni, J. Bahr, R.A. Berg, J.E. Billi, L. Bossaert, P. Cassan, A. Coovadia, K. D'Este, J. Finn, et al. Cardiac arrest and cardiopulmonary resuscitation outcome reports: update and simplification of the Utstein templates for resuscitation registries.:: A statement for healthcare professionals from a task force of the international liaison committee on resuscitation (American Heart Association, European Resuscitation Council, Australian Resuscitation Council, New Zealand Resuscitation Council, Heart and Stroke Foundation of Canada, Inter-American Heart Foundation, Resuscitation Council of Southern Africa). *Resuscitation*, 63(3):233–249, 2004.

[2] D. Lloyd-Jones, R. Adams, M. Carnethon, G. De Simone, T.B. Ferguson, K. Flegal, E. Ford, K. Furie, A. Go, K. Greenlund, et al. Heart disease and stroke statistics–2009 update: a report from the American Heart Association Statistics Committee and Stroke Statistics Subcommittee. *Circulation*, 119(3):e21, 2009.

[3] C. Atwood, M.S. Eisenberg, J. Herlitz, and T.D. Rea. Incidence of EMS-treated out-of-hospital cardiac arrest in Europe. *Resuscitation*, 67(1):75–80, 2005.

[4] Michele Plorde. Division of Emergency Medical Services 2009 Annual Report to the King County Council, 2009.

[5] A.H. Idris, L.B. Becker, J.P. Ornato, J.R. Hedges, N.G. Bircher, N.C. Chandra, R.O. Cummins, W. Dick, U. Ebmeyer, H.R. Halperin, et al. Utstein-style guidelines for uniform reporting of laboratory CPR research: a statement for healthcare professionals from a task force of the American Heart Association, the American College of Emergency Physicians, the American College of Cardiology, the European Resuscitation Council, the Heart and Stroke Foundation of Canada, the Institute of Critical Care Medicine, the Safar Center for Resuscitation Research, and the Society for Academic Emergency Medicine. *Circulation*, 94(9):2324, 1996.

[6] G. Nichol, P. Steen, J. Herlitz, LJ Morrison, I. Jacobs, JP Ornato, R. O'Connor, and V. Nadkarni. International Resuscitation Network Registry: design, rationale and preliminary results. *Resuscitation*, 65(3):265–277, 2005.

[7] T. Eftestøl, K.A.H. Thorsen, E. Tøssebro, C. Rong, and P.A. Steen. Representing resuscitation data–Considerations on efficient analysis of quality of cardiopulmonary resuscitation. *Resuscitation*, 80(3):311–317, 2009.

[8] K.A.H. Thorsen, T. Eftestøl, E. Tøssebro, C. Rong, and P.A. Steen. Using ontologies to integrate and share resuscitation data from diverse medical devices. *Resuscitation*, 80(5):511–516, 2009.

[9] K.A.H. Thorsen, T. Eftestol, C. Rong, and P.A. Steen. An Integrated Information Sharing Structure for Resuscitation Data. In *Proceedings of the 2009 International Conference on Advanced Information Networking and Applications Workshops*, pages 7–12. IEEE Computer Society, 2009.

[10] C.V.P. Infusions, A. Rhythms, A.P.A. Algorithm, V. Fibrillation, and P.V. Tachycardia. 2005 American Heart Association Guidelines for Cardiopulmonary Resuscitation and Emergency Cardiovascular Care.

[11] T.B. Lee, J. Hendler, O. Lassila, et al. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[12] T. Berners-Lee. Semantic Web-XML2000. *W3C Website*, 2000.

[13] T. Bray, J. Paoli, CM Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible Markup Language (XML) 1.0, 5th Edn. W3C Recommendation 26 November 2008, 2008.

[14] E.R. Harold. *Processing XML with Java*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2002.

[15] D. Beckett and B. McBride. RDF/XML syntax specification (revised). *W3C Recommendation*, 10, 2004.

[16] T. Version, L. Version, P. Version, and B. McBride. RDF Vocabulary Description Language 1.0: RDF Schema. *Changes*, 2004.

[17] A. Seaborne and E. Prud¡¯hommeaux. SPARQL query language for RDF. *W3C recommendation, W3C (January 2008) http://www. w3. org/TR/2008/REC-rdf-sparql-query-20080115*.

[18] T.R. Gruber et al. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5:199–199, 1993.

[19] I.F. Cruz and H. Xiao. The role of ontologies in data integration. *Engineering intelligent systems for electrical engineering and communications*, 13(4):245, 2005.

[20] M.K. Smith, C. Welty, and D.L. McGuinness. Owl web ontology language guide. *W3C recommendation*, 10, 2004.

[21] D. Nardi, J. Brachman, et al. An introduction to description logics. 2009.

[22] MJ O'Connor and AK Das. SQWRL: a query language for OWL. In *OWL: Experiences and Directions (OWLED), Fifth International Workshop*, 2009.

[23] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, 21, 2004.

[24] N.F. Noy, M. Sintek, S. Decker, M. Crubézy, R.W. Fergerson, and M.A. Musen. Creating semantic web contents with protege-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.

[25] M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe. A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools. The University of Manchester.

[26] RO Cummins, DA Chamberlain, NS Abramson, M. Allen, PJ Baskett, L. Becker, L. Bossaert, HH Delooz, WF Dick, and MS Eisenberg. Recommended guidelines for uniform reporting of data from out-of-hospital cardiac arrest: the Utstein Style. A statement for health professionals from a task force of the American Heart Association, the European Resuscitation Council, the Heart and Stroke Foundation of Canada, and the Australian Resuscitation Council. *Circulation*, 84(2):960, 1991.

[27] C.P. de Laborda and S. Conrad. Relational. OWL: a data and schema representation format based on OWL. In *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling-Volume 43*, page 96. Australian Computer Society, Inc., 2005.

[28] D.L. McGuinness, F. Van Harmelen, et al. OWL web ontology language overview. *W3C recommendation*, 10:2004–03, 2004.

[29] C. Bizer and A. Seaborne. D2RQ-treating non-RDF databases as virtual RDF graphs. In *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*. Citeseer, 2004.

[30] B. McBride, D. Boothby, and C. Dollin. An Introduction to RDF and the Jena RDF API. *Retrieved December*, 2005.

[31] C. de Laborda and S. Conrad. Database to semantic web mapping using rdf query languages. *Conceptual Modeling-ER 2006*, pages 241–254, 2006.

[32] C.P. de Laborda, M. Zloch, and S. Conrad. RDQuery-Querying Relational Databases on-the-fly with RDF-QL. *POSTER AND DEMO PROCEEDINGS*, page 19.

[33] C. Nyulas, M. OConnor, and S. Tu. DataMaster–a plug-in for importing schemas and data from relational databases into Protege. In *Proceedings of the 10th International Protege Conference*, 2007.

[34] H. Chen, Y. Wang, H. Wang, Y. Mao, J. Tang, C. Zhou, A. Yin, and Z. Wu. Towards a semantic web of relational databases: a practical semantic toolkit and an in-use case from traditional chinese medicine. *Lecture notes in computer science*, 4273:750, 2006.

A. Implementation of the SQWRL queries with SQWRL API in Java

```
/*
 * Implementation of the SQWRL queries with SQWRL API in Java
 */
package query;

import edu.stanford.smi.protegex.owl.ProtegeOWL;
import edu.stanford.smi.protegex.owl.model.OWLModel;
import edu.stanford.smi.protegex.owl.swrl.sqwrl.SQWRLQueryEngine;
import edu.stanford.smi.protegex.owl.swrl.sqwrl.SQWRLResult;
import java.io.FileInputStream;

public class SQWRLQuery {

private SQWRLResult result;
private SQWRLQueryEngine queryEngine ;

public SQWRLQuery()
{
    initQueryEngine();
}

private void initQueryEngine()
{
    try{
    FileInputStream fis = null;
    fis = new FileInputStream("CPR-Data-Ontology.owl");
    // file can be downloaded here:
    // https://sourceforge.net/projects/cpr-ontology/files/
    // or you can use the following code as an alternative
    //String uri =
    //"http://master.dl.sourceforge.net/project/cpr-ontology/" +
    //"CPR-Data-Ontology.owl";
```

```
    //OWLModel owlModel = ProtegeOWL.createJenaOWLModelFromURI(uri);
    OWLModel owlModel =
            ProtegeOWL.createJenaOWLModelFromInputStream(fis);
    queryEngine = SQWRLQueryEngineFactory.create(owlModel);
    } catch(Exception e) {
        e.printStackTrace();
    }
}

/**
 * This method gives variants of the standardized concept of logs
 * @param comEvent: the standardized event name
 * @return SQWRLResult: event, db, table, column of event
 */
public SQWRLResult logQuery(String comEvent)
{
    comEvent = "com:" + comEvent;  // add namespace
    String strQuery = "hasVariant(" + comEvent +
        ", ?vEvent) ^ dbs:hasTable(?db, ?tab) ^ " +
        "dbs:hasColumn(?tab, ?eCol) ^ hasDBColumn(?vEvent, ?eCol)"
        + " -> sqwrl:select(?vEvent, ?db, ?tab, ?eCol)";
    try {
        result = queryEngine.runSQWRLQuery("Query-1", strQuery);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return result;
}

/**
 * This method gives variants of the standardized concept of logs
 * @param comEvent: the standardized event name
 * @param comParam: the standardized parameter name
 * @return SQWRLResult: event, db, table, column of event and param
 */
public SQWRLResult logQuery(String comEvent, String comParam)
{
    comEvent = "com:" + comEvent;  // add namespace
    comParam = "com:" + comParam;
    // construct the query string ...
    String strQuery = "hasVariant(" + comEvent + ", ?vEvent) ^ " +
    "hasParameter(?vEvent, ?vParam) ^ " +
```

```
    "hasVariant(" + comParam + ", ?vParam) ^ " +
    "dbs:hasTable(?db, ?tab) ^ " +
    "dbs:hasColumn(?tab, ?eCol) ^ dbs:hasColumn(?tab, ?pCol) ^ " +
    "hasDBColumn(?vEvent, ?eCol) ^ hasDBColumn(?vParam, ?pCol) ^ " +
    " -> sqwrl:select(?vEvent, ?vParam, ?db, ?tab, ?eCol, ?pCol)";
    try {
        result = queryEngine.runSQWRLQuery("Query-2", strQuery);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return result;
}

/**
 * This method gives the relevant log entry of the report concept
 * @param entry
 * @return SQWRLResult
 */
public SQWRLResult seattleReportQuery(String entry)
{
    entry = "seattle:" + entry;
    // construct the query string ...
    String strQuery="hasRelevantLogEntry("+entry+", ?comEvent) ^ "+
    "hasVariant(?comEvent, ?vEvent) ^ " +
    "hasParameter(?vEvent, ?vParam) ^ " +
    "hasRelevantLogParam(" + entry + ", ?comParam)  ^ " +
    "hasVariant(?comParam, ?vParam) ^ " +
    "dbs:hasTable(?db, ?tab) ^ " +
    "dbs:hasColumn(?tab, ?eCol) ^ dbs:hasColumn(?tab, ?pCol) ^ " +
    "hasDBColumn(?vEvent, ?eCol) ^ hasDBColumn(?vParam, ?pCol) ^ " +
    " -> sqwrl:select(?vEvent, ?vParam, ?db, ?tab, ?eCol, ?pCol)";
    try {
        result = queryEngine.runSQWRLQuery("Query-3", strQuery);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return result;
}
}
```