Towards Semantic Web: Current XML resources conversion and RFID employment

SON MINH HUYNH

A thesis submitted to Auckland University of Technology (AUT) in partial fulfillment of the requirements for the degree of Master of Computer and Information Sciences (MCIS)

School of Computing and Mathematical Sciences

2014

ATTESTATION OF AUTHORSHIP

ACKNOWLEDGEMENTS

There are a number of people I would like to express my sincere thanks to. Without them, I would not have been able to finish this thesis.

First and foremost, I would like to thank Prof. Alvis Fong, my primary supervisor. Prof. Fong helped me greatly with directing, advising and supporting me in my research. He was quick to spot what I did well and what I did badly, and this helped me keep on the right track and finish the work early. He was so enthusiastic and it was so easy to talk to him. He recommended which journals and conferences to target based on my research topic, the one year time frame I had, my existing knowledge, and my ability. I hope I am able to inherit his great vision in research and his great ability to make the right decisions.

Second, I would like to sincerely thank Assoc. Prof. Dave Parry, my secondary supervisor. Though Assoc. Prof. Parry was my secondary supervisor, which meant I should only seek his help when my primary supervisor was unavailable, he was there when he knew that I needed him. My research involves the employment of RFID (Radio Frequency IDentification), which is Assoc. Prof. Parry's field of expertise, in Semantic Web environments. He was quick to give me valuable support when I consulted him about RFID technology and RFID related research. He was able to transfer to me a great amount of knowledge within the limited time he could give me. His great problem solving skills and research vision have inspired me a lot, especially in building IT systems that are user-friendly and which people like to use.

Finally, I would also like to thank others who supported me during my study. My first year coursework lecturers gave me a solid background knowledge and prepared me for my one year thesis. School of Computer and Information Sciences' staff were always helpful and supportive when I had any queries. My fellow postgraduate students were always helpful, not only in discussing academic matters but also with social support helping me balance between studying and having a social life. Last but not least, Catriona Carruthers provided a high-standard, professional proof-reading service on my thesis report.

ABSTRACT

The Semantic Web, which is a machine understandable Web likely to be the future of the Web, is being developed and capturing much attention in recent years. Many things need to be done to ensure the success of moving from the Syntactic Web (the current Web) to the Semantic Web. The first problem identified is there being considerable amounts of data being stored as XML documents for the current Web. However, as XML was invented for structuring data rather than describing the meaning of data, XML documents cannot be used as they are in the Semantic Web environment. One way to make the XML data usable to the Semantic Web is converting them to RDF formatted data. However, it is labor-intensive to perform the conversion manually. The second problem relates to building a Smart Home system that will operate in the Semantic Web environment and assist the occupier. For example, people sometimes forget where they have left certain things, such as glasses, wallets, keys, etc. and it is often troublesome to find them.

This research aims to address the two identified problems. For the first one, a universal XML to RDF conversion on a large scale algorithmic solution is developed to automatically transform XML documents to RDF documents. Outputs of this solution include a transformation procedure and an actual implementation of the procedure in the form of a Java tool called X2R (XML to RDF). For the second identified problem, this study proposes an RFID solution to localize the easily-lost objects based on three nearest reference points and a recommendation as to how to build the indoor localization system, named HLSM (Home Localization System for Misplaced objects), so that it can be easily integrated into the entire Smart Home system that adheres to the Semantic Web context.

Promising results were obtained for both solutions to the two aforementioned problems. The universal XML to RDF conversion solution can efficiently transform XML documents to RDF with 100% accuracy. One could confidently apply the X2R tool to convert XML documents on a large scale. In regard to the other solution, the HLSM system prototype provided a localization accuracy rate of 87.5%, in the form of user-friendly statements, e.g. “The glasses are at the kitchen table”, which are not given by existing solutions. These encourage further enhancement of the localization technique and the implementation of the complete HLSM system.

Table of Contents

List of Figures

List of Tables

List of Abbreviations and Terms

Abbreviation/Term	**Description**
aka	Also known as
DOM4J	An open source Java library for working with XML documents
DTD	**D**ocument **T**ype **D**efinition, used to define the legal blocks of an XML document.
DTNL	**D**e**T**ailed **N**avigation **L**andmark is an RFID tag used as a reference point.
Fuseki	A SPARQL server that serves RDF data over HTTP
GPS	**G**lobal **P**ositioning **S**ystem
HLNL	**H**igh-**L**evel **N**avigation **L**andmark is an RFID reader at a room door entry.
HLSM	**H**ome **L**ocalization **S**ystem for **M**isplaced objects
Jena	aka Apache Jena, a free and open source Java framework for building Semantic Web and Linked Data applications (http://jena.apache.org/)
JSON	**J**ava**S**cript **O**bject **N**otation is a lightweight data-interchange format.
JSP	**J**ava **S**erver **P**ages is a technology that helps create dynamic web pages.
JVM	**J**ava **V**irtual **M**achine
LO	easily**-L**ost **O**bject is an object need to be tracked and localized.
OWL	**W**eb **O**ntology **L**anguage, builds on RDF and RDF Schema and adds more vocabulary for describing properties and classes.
RDF	**R**esource **D**escription **F**ramework, a standard model for data interchange on the Web. An RDF statement (aka RDF triple) consists of a subject, a predicate, and an object; for example, “A teaches B”.
RDFS	**RDF S**chema adds semantic extension to RDF. It gives mechanisms for describing groups of related resources and the relationships between these resources [1].
RF	**R**adio **F**requency
RFID	**R**adio **F**requency **ID**entification
RSSI	**R**eceived **S**ignal **S**trength **I**ndication
SAX	**S**imple **A**PI (Application Programming Interface) for **X**ML
SAX parser	An event-based sequential access parser API developed by the XML-DEV mailing list for XML documents.

SH	**S**mart **H**ome
SPARQL	A recursive acronym for **S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage, an RDF query language.
UHF	**U**ltra-**H**igh **F**requency
UHF **Gen2** RFID reader	UHF RFID reader that belongs to EPCglobal Class 1 **Generation 2**
XML	e**X**tensible **M**arkup **L**anguage
XML Schema	Describes the structure of an XML document. An XML Schema file has the file extension of .XSD
XQuery	A language used to query XML data
XSD	**X**ML **S**chema **D**efinition
XSL	e**X**tensible **S**tylesheet **L**anguage, a stylesheet language for XML documents
XSLT	XSL Transformations

Chapter 1 Introduction

1.1 Research Background and Motivation

The Semantic Web, a machine understandable Web to hasten information retrieval [2], is likely to be the future of the Web. The main limitation of the Syntactic Web (the current Web) is that the contents displayed on web pages are not comprehensible to computers. What the machine sees is just a bunch of text on the web page and therefore users have to do all the reading, understanding, information extraction and analysis themselves. Even data displayed on the web page that were originally queried from a database, once displayed on the web page, lose the structure and the relationship of the data. One can see the effect of this on information retrieval with the large amount of unwanted information returned by a search engine. In the Syntactic Web, though search engines seem to be successful and indispensable tools for Web users, Antoniou and Van Harmelen [2] point out that search engines are just location finders, not information retrievers. For example, Google displays hundreds or even thousands of links for a search phrase entered by the user and the user has to select the right links, read the documents and find the required information. This is because search engines work by matching key words but they do not understand the meaning (semantics) of them. This limitation of the Syntactic Web can be overcome in the Semantic Web. With the Semantic Web, data are linked together (linked data) and displayed on the web; from this data linking approach, data displayed on the web are structured and automatic retrieval, extraction and integration are enabled [3]. Once successfully implemented, the current keyword-based searching of documents will be replaced by query answering, and querying the data displayed on the web pages can be enabled in a similar way to querying data from a database [2].

The possibility of widespread machine comprehensible data and the myriad opportunities for automatic information processing on the Semantic Web are enabled by ontologies [4]. In philosophy, ontology is the study of things that exist [5]. Artificial-intelligence and Web researchers have restricted “ontology” to mean a document or file that formally defines terms and relationships among terms [6]. Ontologies play a prominent role in the Semantic Web and in other forms of knowledge management [7]. An example of the power of using ontologies is the inference rules in ontologies [6]. Berners-Lee et al. gave an example of the power of inference rules with real-world data below:

> If a city code is associated with a state code, and an address uses that city code, then that address has the associated state code. A program could then readily deduce, for instance, that a Cornell University address, being in Ithaca, must be in New York State, which is in the U.S., and therefore should be formatted to U.S. standards. The computer does not truly "understand" any of this information, but it can now manipulate the terms much more effectively in ways that are useful and meaningful to the human user. [6]

The important technologies for developing the Semantic Web are summarized as follows. Resource Description Framework (RDF) is a model for describing things as triples; each triple has the form of <subject><predicate><object>, for example "Alvis Fong is a professor" [2, 6]. The RDF triples are written using XML (eXtensible Markup Language) syntax and have become a standard data model for describing resources for the Semantic Web [2]. RDFS (RDF Schema) is considered a primitive language for writing ontologies; however, a more powerful language is needed to deal with complex relationships among objects [2]. OWL (Web Ontology Language), built upon RDF and RDFS and having the same kind of syntax [8], was officially released in 2004 by W3C (World Wide Web Consortium) to do the job of writing ontologies [9]. If there is an RDF statement "Dave Parry teaches Autoidentification class", then with OWL, it is also implied that "Autoidentification class is taught by Dave Parry".

Two interesting points arose which motivated this research. The first point is dealing with the current Syntactic Web resources, particularly XML resources, as the Semantic Web would become a reality. Currently, considerable amounts of data are being stored as XML documents, but the Semantic Web works with RDF formatted documents [10]. As the Semantic Web cannot work directly with the current XML resources [10], how are the data being stored under XML formatted documents reused? Manual recreation of all the data to RDF would be enormously labor-intensive. The second point is related to the idea of bridging the physical world to the virtual one using RFID (Radio Frequency IDentification) as a universal entry point [11]. RFID can uniquely identify physical artifacts and thus can be used to build the Internet of Things, a vision in which the Internet extends into people's daily lives through a network of uniquely identifiable objects [12]. To be specific, the second point in this research is about building a Smart Home system that can manage physical artifacts and enable a real-world search for these objects, which is quite different from the virtual-world search of current search engines such as Google [13]. In addition, this proposed system should also ensure the adaptability with the emerging Semantic Web. Smart Home systems that can help track and localize objects within the home have been an important part of research in recent years. In practical terms, these systems can help people find certain things, such as glasses, wallets, keys, etc., and it is often troublesome to look for them. Previous research has shown great achievements in terms of localization accuracy of up to about 10 cm [14-16]. However, no existing solution gives a meaningful description of the location of the misplaced object related to important known areas in the home or other space, which are easily understood by the user.

1.2 Research Objectives

The main objectives of the research are as follows.

1. To develop a universal XML to RDF conversion algorithmic solution that can transform XML documents to RDF documents automatically and correctly. The solution should consist of two parts. The first part is a transformation procedure that expresses conversion rules for interpreting and converting an XML document to RDF and provides an algorithm described as pseudo code that complies with the conversion rules to convert an entire XML document to RDF. The second part is an actual implementation of the first part as a tool whose inputs are XML documents and outputs are RDF documents.

2. To develop an indoor localization system named Home Localization System for Misplaced objects (HLSM) solution to localize easily-lost objects (LOs), such as glasses, wallets, keys, etc., within the home. HLSM should give the localization result in the form a user-friendly statement, e.g. "The wallet is on the tea table inside the living room".

Moreover, it should adhere to the Semantic Web protocols in order to be forward compatible as on the move towards Semantic Web.

1.3 Methodology

As this research aimed to develop a Universal XML to RDF conversion algorithm and an HLSM, which are new and innovative artifacts, Design Science was chosen as it is a suitable methodology [17]. In Design Science, the research activities are twofold: build and evaluate [18]. Build refers to the activity of constructing the artifact, showing that such an artifact can be constructed. Evaluate is the activity of developing criteria and assessing the performance of the created artifact against those criteria. The development process of this research followed a spiral model as depicted in Figure 1-1. The use of a spiral model in developing the two modules of this research was due to ease of management [19]. With a spiral model, problems can be identified early and appropriate actions can be taken quickly in the next iteration/phase.

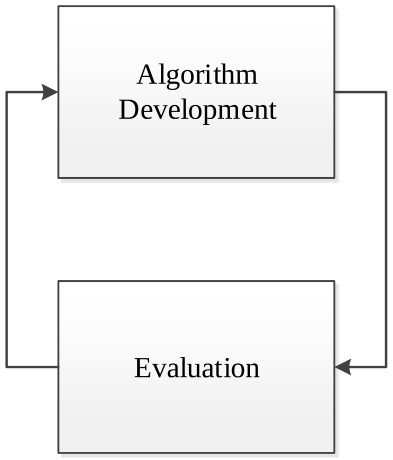


Figure 1-1: Research development process model [18]

As the nature of this research is to improve IT performance by providing an automatic conversion algorithm and a smart home system, it belongs to prescriptive research [18]. Therefore, this research was shaped by the positivist paradigm where it is believed that reality is independent of human factors. As such, logical reasoning and objectivity, rather than subjectivity and intuitive interpretation, were applied in this research [20]. Since this research consists of two research modules, the details of the application of the methodology will be mentioned in each research module separately.

1.4 Thesis Details

1.4.1 Publications

During the course of this research thesis, five papers have been written for publication. Two of them have been published, one of them has been accepted for publication, and two papers are still under review process. Details three papers that have been accepted or published are briefly introduced below.

1. S. M. Huynh, D. Parry, A. Fong, and J. Tang, "Home Localization System for Misplaced Objects," in Proc. *IEEE International Conference on Consumer Electronics*, 2014, pp. 462-463.

2. *"Novel RFID and ontology based Home Localization System for Misplaced Objects" transactions paper*: This paper, an extension of the paper number 1 above, has gone through the first review and has been resubmitted to IEEE Transactions on Consumer Electronics to be published in August 2014 (Accepted).

3. S. M. Huynh, A. Alshubaily, F. M. Mir, O. Smirnov, M. Thomas, J. O. Ogunyebi, D. Parry, and A. Fong, "Pharmacy Drug Administration System," in Proc. *39th Annual Conference of the IEEE on Industrial Electronics Society*, 2013, pp. 8437-8442.

1.4.2 Thesis Contribution

This thesis provides a number of contributions to the existing knowledge. These contributions are summarized below.

1. An extensive literature review on the topic of transforming XML to RDF is provided. The literature review gives the current status of the topic as well as highlighting the drawbacks of the existing techniques from the literature. These can be seen in section 2.2.

2. A transformation procedure that can convert XML documents to RDF is developed. The transformation procedure is followed by the implementation of the X2R tool to do the conversion. These are described in sections 2.4 and 2.5. The solution in this research has a conversion accuracy rate of 100% that is better than the previous solutions.

3. An extensive literature review about indoor localization is presented. The literature review gathers the challenges to indoor localization systems and also provides the limitations of the existing solutions in the reviewed literature based on the identified challenges. These are mentioned in detail in section 3.2.

4. An indoor localization system called HLSM (Home Localization System for Misplaced objects) that uses RFID and works in the Semantic Web environment is designed and prototype implemented. The solution focuses on the usability of the system and the adaptability of the system to being integrated into the whole Smart Home system. The usability of the system is achieved by generating a meaningful description of the location of the misplaced object instead of its coordinates, such as "the key is at the kitchen table". None of the existing literature has the same approach. Details of HLSM are described in section 3.4.

1.4.3 Thesis Organization

The organization of this thesis is structured to hold two modules incorporated in this thesis. The first module; “Universal XML to RDF Conversion” is to achieve the first research objective and the second module; “Home Localization System for Misplaced objects” is for the second research objective. The rest of this thesis report is organized into the following chapters:

Chapter 2: This chapter introduces the first module of this research, Universal XML to RDF Conversion. The chapter consists of seven sections, Introduction in section 2.1, Literature Review in section 2.2, Methodology in section 2.3, Transformation Procedure in section 2.4, Implementation in section 2.5, Evaluation in section 2.6, and Summary in section 2.7.

Chapter 3: This chapter introduces the second module of this research, Home Localization System for Misplaced objects. The chapter consists of seven sections, Introduction in section 3.1, Literature Review in section 3.2, Methodology in section 3.3, Solution in section 3.4, Implementation and Experiments in section 3.5, Results and Discussion in section 3.6, and Summary in section 3.7.

Chapter 4: This chapter concludes the research and includes recommendations for future work for each research module.

Chapter 2 Universal XML to RDF Conversion

2.1 Introduction

An important aspect of the transition process from Syntactic Web (the current Web) to Semantic Web involves making the current Web resources usable on the Semantic Web. This can be done by either transforming current Web resources to Semantic Web resources or building intermediate layers on which the Semantic Web can work with current Web resources. Currently, considerable amounts of data are being stored as XML documents [10]. Unfortunately, XML is for structuring data rather than describing the meaning of data and therefore cannot be directly used by agents or machines in the Semantic Web [21]. [10, 21-28] followed the first approach to convert XML data to RDF, while [29-32] took the second approach of building intermediate layers. For example, [30] created the XSPARQL language, which is a combination of XQuery and SPARQL that allows querying of both XML data and RDF data using the same framework.

This research follows the first approach, developing an algorithm to do the conversion automatically. A transformation procedure and its implementation in Java as the X2R tool (XML-To-RDF tool) are developed to convert any valid XML document into RDF.

Input XML documents can be divided into six partitions illustrated in Figure 2-1. XML documents are classified into three types, XML document supported by DTD (Document Type Definition) document, XML document supported by XML schema, and XML document alone. Therefore, the input XML data in the conversion algorithm can be horizontally divided by the three types of XML document and vertically divided by whether or not there are associated ontologies.

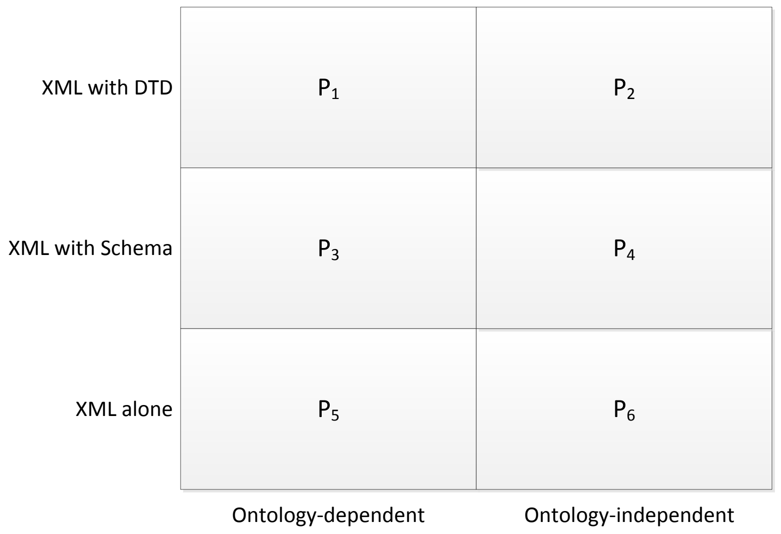


Figure 2-1: Partitioning input XML documents

The rest of this chapter is organized as follows. Section 2.2 reviews related work. Section 2.3 provides details of the methodology used in the research for this module, Universal XML to RDF Conversion. Section 2.4 describes the transformation procedure and section 2.5 explains how the X2R tool was implemented. Section 2.6 evaluates the procedure and the tool. Finally, section 2.7 concludes this chapter.

2.2 Literature Review

A number of related studies have been done since Tim Berners-Lee introduced the concept of the Semantic Web in 2001 [21]. They can be classified into two categories: ontology-dependent and ontology-independent.

2.2.1 Ontology-dependent documents

This category means using existing ontologies of the same domain to interpret XML documents. The general approach entails mapping elements and attributes of an XML document into classes and properties of the corresponding RDF document, checking against the predefined ontologies. One disadvantage to this is that not all XML tags and attributes are transferred to RDF data as not all of them are already defined in the existing ontologies. However, the advantage of this is the reliability of the RDF data provided as they have been checked against existing ontologies. [22-24] have been worked in this category.

Klein's [27] conversion procedure can transform ambiguous XML data to RDF statements. The ontology used specifies which labels of the XML document are to be transformed and also decides whether a label is interpreted as the name of a class, or as the name of a property. One limitation of this procedure is that, the XML elements are transformed to classes or properties based on the user's opinion.

Cruz et al. [22] provided a framework to make two XML documents collaborate at the semantic level while still keeping their nesting structure. For each XML document, a local RDF ontology is created. These two local ontologies are then merged to generate a global ontology. The global ontology has two roles, unifying the query interface to XML sources and serving as the mediation mechanism for accessing the data in XML sources.

Van Deursen et al. [23] proposed a conversion procedure for transforming XML data to an RDF instance. The transformation process is supported by XML Schema, a mapping document and the OWL (Web Ontology Language) ontology. OWL is an enhancement of RDF and RDF Schema, providing vocabularies with associated inference capabilities (For example, if there is a statement "A teaches B" then it is also implied that "B is taught by A"). They illustrated their approach by applying it to the DIG35 specification, which is an XML-based metadata standard for digital image description.

In the framework provided by Yihua et al. [24], local ontologies are extracted from XML documents. Then, a proposed algorithm measures the similarity between the extracted ontologies and the domain ontologies to build mapping files and stores these files in a Business Collaboration Ontology Library. This library is used to provide feasible solutions for inter-enterprise business integration.

2.2.2 Ontology-independent documents

This category transforms XML documents to RDF documents without the aid of existing ontologies in the same domains. The general approach entails converting elements and attributes of XML documents to subjects (aka classes) and objects (aka properties) of RDF statements, and then adding the predicates to the RDF statements. The amendment of predicates can be assisted by RDF Schema vocabularies [10]. Disadvantages of this category include the following. First, DTD and XML Schema can be used to specify restrictions on the contents of XML tags, but cannot specify the meaning of these tags [23]. Second, the converted RDF statements could be unreliable as there could be noise in the source XML documents, such as synonyms and homonyms. Third, the added predicates would be very limited and have only general meanings, for example “rdf: value”, “rdf: hasClass” and “rdf:property”, as they need to be general enough to satisfy all cases of subjects and predicates. One advantage of this category is that there is no need of an existing ontology. Therefore, any XML document can be transformed without the effort of building the ontologies in advance. Another advantage of this category is the possibility of converting all XML data to RDF statements. [10, 21, 25, 26] have been worked in this category.

Ferdinand et al. [28] proposed a mapping rule to map XML to RDF and XML Schema to OWL. However, the solution provides only general rules of mapping. In addition, no testing was performed on specific cases and therefore the solution could be unreliable. A year later, Bohring and Auer [25] presented a more sophisticated mapping between the data elements of XML and OWL. The solution is accompanied by an implemented framework based on a ready-to-use XSLT (eXtensible Stylesheet Language Transformations) framework. Further, the implemented framework was tested on common use cases.

Later, Battle [26] introduced a bidirectional mapping between XML and RDF based on the XML Schema. It resulted in an implemented tool called Gloze that works within the Jena framework. Pham et al. [10] then developed a procedure for converting valid XML documents to RDF statements by using RDF Schema vocabularies. This method can transform all XML elements based on their definition in DTD and RDF schemata. However, the limitation of this method is that it only applies to XML documents with DTD while there are many XML documents that do not have DTD.

In 2010, Yang et al. [21] developed an automatic extraction algorithm that converts XML data to RDF statements through XML Schema. The algorithm can even transform XML data without the availability of XML Schema. In the case of missing XML schema, the algorithm needs to traverse the XML document to get the ontology based on its structure. This algorithm provides an accuracy of 80 – 90% which still leaves room for improvement.

2.2.3 Summary

Table 2-1 summarizes the reviewed literature in terms of the XML document partitions they address.

Table 2-1: LITERATURE AND ADDRESSED PARTITIONS OF XML DOCUMENTS

#	Literature	Partition
1	Klein [27]	P5
2	Cruz et al. [22]	P1, P3, P5
3	Van Deursen et al. [23]	P3
4	Yihua et al. [24]	P3, P5
5	Ferdinand et al.[28]	P4, P6
6	Bohring and Auer [25]	P4, P6
7	Battle [26]	P4
8	Pham et al. [10]	P2
9	Yang et al. [21]	P4, P6

No existing literature has proposed a method to convert all types of XML documents (all partitions) to RDF. There could be an argument to combine these approaches in order to cover all six partitions. However, it would be hard in practice to implement one unified conversion tool which combines these approaches. Therefore, a fresh approach is needed to convert all six XML partitions.

In June 2012, Petruska and Curda [33] developed a Java application named XMLtoRDF which based only on the XML tags and their structures in the input XML files to convert any valid XML document to RDF. However, this application does not work correctly when dealing with reasonably large XML files, as can be seen in Table 2-3 in Section 2.6. Moreover, it does not work properly when converting special XML documents such as *special.xml* file, as seen in Table 2-3.

2.3 Methodology

As laid out in Chapter 2, Design Science methodology was chosen for this research. This section provides details of the application of the methodology when applied to the Universal XML to RDF Conversion module.

2.3.1 Research Framework

The framework described in Figure 2-2 was used in this module's research. It is an adaption of March and Smith's [18] framework for information technology research.

		Research Activities	
		Build	**Evaluate**
Research Outputs	**Transformation procedure**	2.3.4. Transformation Procedure Model	2.3.6.1 Transformation Procedure Evaluation
	X2R tool	2.3.5. Transformation Procedure Implementation	2.3.6.2 X2R Tool Evaluation

Figure 2-2: Research framework for Universal XML to RDF Conversion (adapted from [18])

March and Smith's [18] original research framework has four research activities, Build, Evaluate, Theorize and Justify, for typical natural science research. However, computer science, which includes human intervention, is not a natural science. "Justify" refers to theory proving [18]. In this research, only the transformation procedure and the X2R tool were developed, no theory needed to be created. Therefore, Theorize and Justify activities were removed and this resulted in the pruned framework containing only Build and Evaluate activities in the Research Activities dimension.

Each research output went through Build and Evaluate activities. The first output, the transformation procedure, is an abstract model for transforming XML documents into RDF documents. The second output, the X2R tool is the implementation of the procedure, developed in Java, to carry out the conversion job. Sections 2.3.4 and section 2.3.5 explain the Build activities in detail while sections 2.3.6.1 and 2.3.6.2 describe the Evaluate activities for the research outputs.

2.3.2 Data Collection

Several sources where XML files were obtained are listed below:

- the XML documents of the reviewed literature
- the sample XML files from the installation of Stylus Studio software, http://www.stylusstudio.com/xml_download.html
- the sample XML files from the installation of XML Pro software, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4392679&tag=1
- XML documents from the W3C website, http://www.w3.org/

2.3.3 Data Analysis

As the input XML files were collected from the trusted sources mentioned in section 2.3.2, the input XML data can be assumed valid. However, to make sure the developed algorithm is fairly evaluated based on the quality of the output RDF documents, the input XML documents were validated before hand as there was no error-free guarantee for the collected data. The input XML files were validated in twofold, syntax validation and semantic validation. The first level of validation was syntax validation. This was done by using the following validation services provided by W3C (World Wide Web Consortium), http://www.w3schools.com/xml/xml_validator.asp (XML & DTD validator), and http://xsdvalidation.utilities-online.info/ (XML Schema Validation online). The second level of validation, semantic validation, was a time-consuming process. Checking the meanings of tags, attributes and values of XML elements was done manually. The following XML document provides an example (Figure 2-3).

```xml
<bookstore>
  <book category="COOKING">
    <title lang="en">Superman</title>
    <author>Tra Nguyen</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>-39.95</price>
    <course>Research Method 2</course>
  </book>
</bookstore>
```

Figure 2-3: XML document example for data analysis

It can be seen that the negative value of the price (-39.95) is not correct as a price should not be negative. However, there can be much more complicated cases where it is hard to conclude whether the meanings of tags, attributes and values are right or wrong. For example, in the above XML document, whether the element <*course*> is valid or not would be hard to decide as the book with the tittle "*Learning XML*" could be a textbook for "*Research Method 2*" course. Another ambiguous example in the above XML document is the book with the title "*Superman*" in the "*COOKING*" category. Although it does not seem right, the author may have titled her book "*Superman*".

2.3.4 Transformation Procedure Model

The transformation procedure was modeled to cover all six partitions of input XML documents. It should be detailed enough to cover all cases within each partition. The cases here were the different structure combinations of XML documents, not the combinations of data values as the number of combinations of data values is infinite. In addition, as the transformation procedure is an abstract model and is independent of implementation languages (e.g. Java, C#, Python), it was described in plain English.

2.3.5 Transformation Procedure Implementation

The modeled transformation procedure can be implemented in different programming languages. This research used Java as the programming language for the following reasons:

- Java applications are cross-platform (can run on any platform). It is useful to have a cross-platform tool as different organizations keep their XML datasets on different platforms (Linux, Windows, Mac, Solaris etc).
- Java is fully object oriented.
- Java and many Java supported IDEs (e.g. Eclipse) are free.
- Open-source communities provide support.

2.3.6 Evaluation Protocol

Evaluation covers both transformation procedure evaluation and X2R tool evaluation, depicted in Figure 2-4.

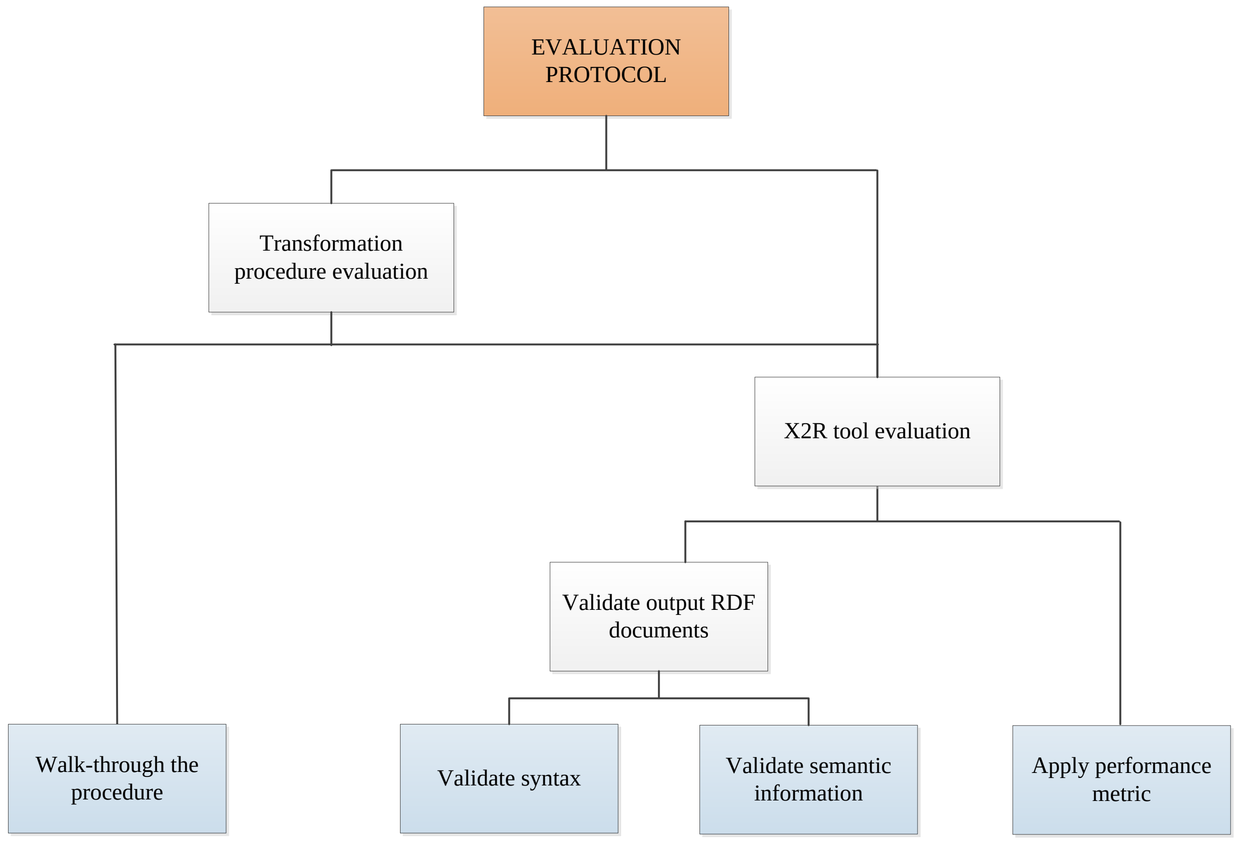

EVALUATION PROTOCOL
Transformation procedure evaluation
X2R tool evaluation
Validate output RDF documents
Walk-through the procedure
Validate syntax
Validate semantic information
Apply performance metric

Figure 2-4: Evaluation protocol for Universal XML to RDF Conversion

It can be seen that there is no input XML file validation step in the evaluation protocol because the input XML file should be checked (section 2.3.3) before entering the conversion step. Once entering the conversion process, it is assumed that the input XML file is valid and it is not necessary for the X2R tool to validate the input XML file.

2.3.6.1 Transformation Procedure Evaluation

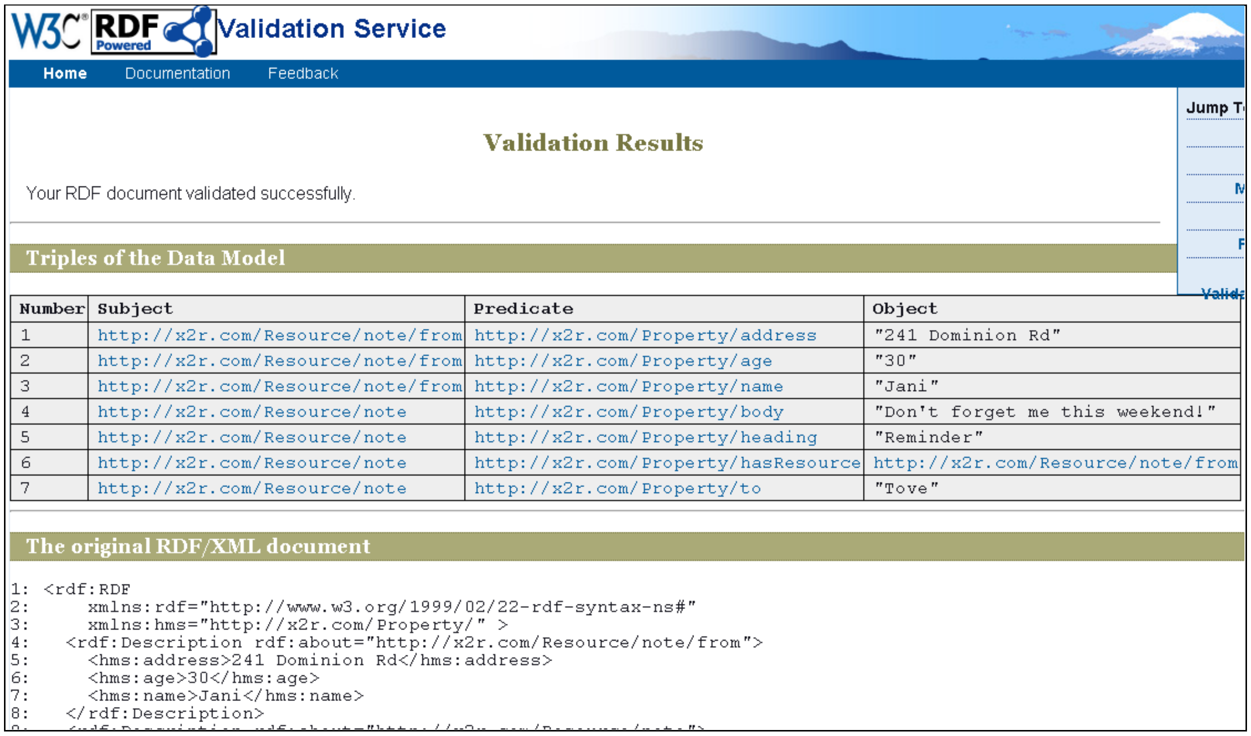
The transformation procedure evaluation included a "walk-through" [34] activity and X2R tool evaluation. Walk-through means going through the transformation procedure line by line to convert the input XML documents to RDF documents. This process was time consuming but was a useful first step to test the model to identify basic errors related to the main functionalities of the model. The second step in the transformation procedure evaluation was the evaluation of its actual implementation, the X2R tool. Following the spiral model of build and evaluate, results from the X2R tool evaluation were inputs for decision making in updating the transformation procedure to achieve a better design as well as a better implementation in the next iteration of the development process [19].

2.3.6.2 X2R Tool Evaluation

The X2R tool evaluation involved output RDF document validation and performance metric application. To evaluate the output RDF documents, both quantitative and qualitative approaches were used. Quantitative approach was using W3C's RDF Validation Service, http://www.w3.org/RDF/Validator/, to validate the syntax of the output RDF documents. World Wide Web Consortium (W3C), which was founded and is directed by Tim Berners-Lee, the father of the World Wide Web, provides standards for the Semantic Web and therefore its services should be reliable [35]. The qualitative approach was used to validate the semantic information of the RDF statements in the output RDF documents. Details of the three procedures, validating syntax, validating semantic information, and applying the performance metric, will be explained in the following sub-sections.

2.3.6.2.1 Syntax Validation

W3C provides the RDF validation service, http://www.w3.org/RDF/Validator/, to evaluate whether an RDF document is correct syntactically. Figure 2-5 illustrates an example of the use of the RDF Validation Service.



Validation Results

Your RDF document validated successfully.

Triples of the Data Model

Number	Subject	Predicate	Object
1	http://x2r.com/Resource/note/from	http://x2r.com/Property/address	"241 Dominion Rd"
2	http://x2r.com/Resource/note/from	http://x2r.com/Property/age	"30"
3	http://x2r.com/Resource/note/from	http://x2r.com/Property/name	"Jani"
4	http://x2r.com/Resource/note	http://x2r.com/Property/body	"Don't forget me this weekend!"
5	http://x2r.com/Resource/note	http://x2r.com/Property/heading	"Reminder"
6	http://x2r.com/Resource/note	http://x2r.com/Property/hasResource	http://x2r.com/Resource/note/from
7	http://x2r.com/Resource/note	http://x2r.com/Property/to	"Tove"

The original RDF/XML document

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:hms="http://x2r.com/Property/" >
  <rdf:Description rdf:about="http://x2r.com/Resource/note/from">
    <hms:address>241 Dominion Rd</hms:address>
    <hms:age>30</hms:age>
    <hms:name>Jani</hms:name>
  </rdf:Description>
```

Figure 2-5: Example of the use of RDF Validation Service

The validation result is also accompanied by an RDF graph (Figure 2-6):

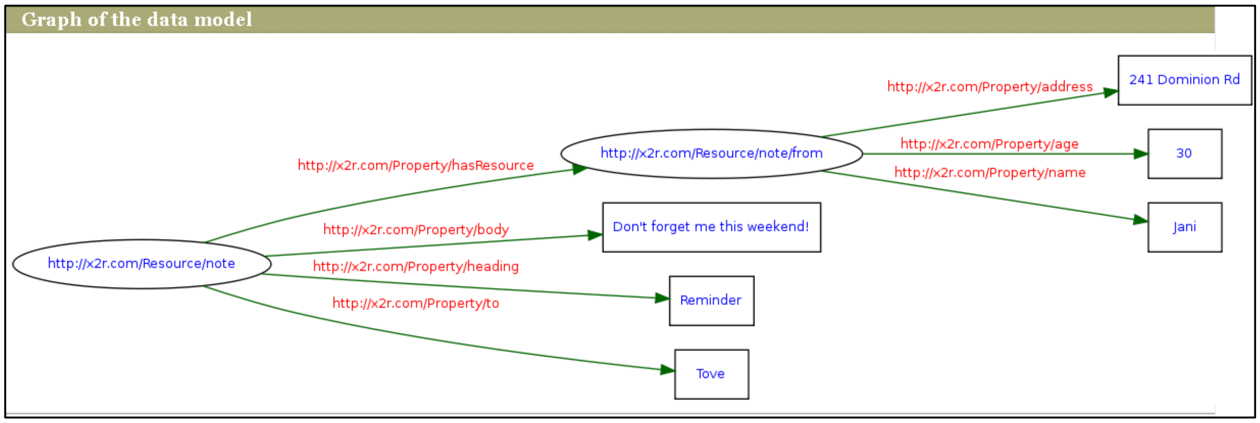

Graph of the data model
http://x2r.com/Resource/note
http://x2r.com/Property/hasResource
http://x2r.com/Resource/note/from
http://x2r.com/Property/address
241 Dominion Rd
http://x2r.com/Property/age
30
http://x2r.com/Property/name
Jani
http://x2r.com/Property/body
Don't forget me this weekend!
http://x2r.com/Property/heading
Reminder
http://x2r.com/Property/to
Tove

Figure 2-6: RDF Graph from RDF Validation Service

2.3.6.2.2 Semantic Information Validation

This was a time consuming validation process. An output RDF statement has the form of *<subject><predicate><object>*, for example *<bookstore><hasClass><book>*,

<book><hasProperty><category>, *<book><hasInstance><book1>* or *<book1><hasTitle><Harry Potter>*. The subject and object were checked against the input XML documents. The predicate was checked to see whether it connected the subject and the object correctly.

2.3.6.2.3 Performance Metric

According to March and Smith [18], the performance of an artifact needs to be justified by a performance metric (e.g. "Is usability more important than maintainability?"). To evaluate the performance of the X2R tool, the performance metric below consisting of the following factors in priority order, was used.

- **Extraction accuracy:** This is the most important factor of the performance metric as the goal of developing this X2R tool was to transform XML documents to RDF documents so that they can be used by the Semantic Web. Accuracy should be the first and foremost criterion to be considered. The accuracy rate was measured by how many valid RDF documents transformed per the number of input XML documents. To better understand which partitions of XML documents the tool operates on well, the accuracy rates for all partitions were also considered. Table 2-2 below formulates the accuracy measurements.

Table 2-2: ACCURACY MEASUREMENT OF X2R TOOL

	Partition 1	Partition 2	Partition 3	Partition 4	Partition 5	Partition 6	All
No. of input XML documents	x_1	x_2	x_3	x_4	x_5	x_6	$\sum_1^6 x_i$
No. of correct output RDF documents	r_1	r_2	r_3	r_4	r_5	r_6	$\sum_1^6 r_i$
Accuracy rate	r_1/x_1	r_2/x_2	r_3/x_3	r_4/x_4	r_5/x_5	r_6/x_6	$\frac{\sum_1^6 ri}{\sum_1^6 x_i}$

- **Time**: Running time is not so critical when the number of input XML documents and the document sizes are small. However, when number and size are large, the running time of the X2R tool becomes crucial. One often with a Java program is that the program can hang because of running out of memory from dealing with large input dataset. This should not happen in this implemented X2R tool.

- **Usability**: Whether or not it is easy to use the X2R tool. Usability is always a factor when evaluating software. The program should give descriptive prompts to instruct, hint or warn users when they are using the application.

- **Cross-platform**: How many platforms the X2R tool can run on. Imagine a case where all of client's existing XML documents are stored on Linux machines while the tool can only run on Windows.

2.4 Transformation Procedure

After becoming aware of the limitations of all existing transformation procedures, a different transformation procedure was constructed in this study to convert any XML document into RDF. The procedure does not rely on existing ontologies, DTDs or XML schemata but rather only relies on the XML tags and their structures in the input XML files. A valid XML file itself (valid in both syntax and semantics) already organizes data in a way that people can understand. Starting from the root element, XML elements are nested to form a hierarchy which shows the relationships among them. Meanwhile, a leaf XML element (an XML element that has no child element) describes its data using the tag name while an XML attribute describes its attribute value using the attribute name. The point here is that they are not written in RDF.

The task here is to reorganize the XML elements into RDF triples. An RDF triple (aka RDF statement) has the form of <*subject*><*predicate*><*object*> where <*subject*> is always a *resource* and <*object*> can be either a *resource* or a *literal*. A *literal* is an atomic value (string) [2]. It is reasonable to turn data stored as leaf XML elements and XML attribute values into literals of RDF triples. For the former, the leaf XML tag names can be used as predicates and the parent XML tag names can be used as subjects. Element A is called the parent of element B when B is contained within A and A is exactly one level above B. For the latter, attribute names can be used as predicates and XML tag names can be used as subjects. In addition, the relationship between a parent XML tag and a child XML tag that is not a leaf element that has no attribute can be turned into an RDF triple of <*parent tag name*><*hasResouce*><*child tag name*> where *"hasResource"* is the predicate created in this research to show relationships between parent tags and child tags. These rules were formed based on the way an XML document is interpreted.

2.4.1 Conversion Rules

As, in an XML document, there are three and only three cases (attributes, data stored in leaf elements and relationships between parent tags and child tags) that can be turned into RDF triples, the transformation procedure can be described in three mapping rules which are sufficient to convert the entire XML document to RDF. The three rules are defined as follows:

1. If an XML element *<S>* has attribute *att1="value1"*, create an RDF statement *<S><att1><value1>*.
2. If an XML element *<O>*, a child element of element *<S>*, has child element(s) and/or attribute(s), create an RDF statement *<S><hasResource><O>*. Further, if element *<S>* has more than one *<O>* child element, set a counter to element *<O>* to have *<S><hasResource><O>*, *<S><hasResource><O$_1$>, <S><hasResource><O$_2$> etc.*
3. If an XML element *<P>* has no child element, aka a leaf element, and has parent element *<S>*, there are two cases. First, if it has no attribute *(<P>data</P>)*, create an RDF statement *<S><P><data>*. Second, if it has attribute(s) *(<P att1="value1"...>data</P>)*, create an RDF statement *<P><P><data>*.

2.4.2 Algorithm

From the three rules above, an algorithm was developed to traverse the XML document from the opening root tag to the closing root tag and convert the XML document to RDF statements. While traversing, four different components, *opening tags*, *element attributes*, *texts*, and *closing tags*, are alternately met and obtained. *Text* is the text that lies between two consecutive tags (either *opening tag – opening tag*; *opening tag – closing tag*; *closing tag – opening tag*; or *closing tag – closing tag*). However, the algorithm only cares about *texts* which lie between *opening tag – closing tag* tags (leaf elements) and calls them *element texts*. It discards other *texts* although they do not make the XML document not-well-formed. For example, in the following XML document named *special.xml*, only "text1_1", "text1_2", and "text2" are considered, "aaa" and "bbb" should be discarded because these are not the genuine ways to store data in an XML document (Figure 2-7).

```
<root>
  <tag1 att1="value1">aaa
    <tag1_1>text1_1</tag1_1>
   <tag1_2>text1_2</tag1_2>
   bbb
  </tag1>
  <tag2>text2</tag2>
  <tag3 att3_1="value3_1" att3_2="value3_2" />
</root>
```

Figure 2-7: *special.xml* document

After meeting and obtaining the four components, three types of events can be introduced, *startElement*, *characters*, and *endElement* events. These events are named in the same way as event names in the SAX (Simple API for XML) parser so that the developed code, which uses the SAX parser, is well aligned with the algorithm. In a *startElement* event, the *opening tag* and the *element attributes* can be captured. In a *characters* event, the *text* can be obtained. In an *endElement* event, the *closing tag* is retrievable.

Definition 1: An XML element has the properties (e, n, A, d) where e is the XML element name (aka tag name), n is the number of attributes of the element, A is the collection of the attributes, and d is the data (aka *text*) of the XML element. Empty *text* is denoted by λ

Definition 2: The collection of all (attribute name, attribute value) pairs of an XML element is denoted by $A = \{(a_i, v_i) : i \in [0, n) \land n \in \mathbb{N}\}$ where n is the number of attributes of the XML element.

The algorithm (pseudo code) to create all the RDF triples is below (Algorithm 1).

Algorithm 1 X2R algorithm

```
while not end of document do
    if $startElement$ then
        $counter \leftarrow$ number of same name siblings have been traversed
        if $counter = 0$ then
            $uri \leftarrow rsPrefix + path + e$ /* rsPrefix (resource prefix): e.g. http://x2r.com/Resource/; path: e.g. bookstore/book1/ */
        else
            $uri \leftarrow rsPrefix + path + e + counter$
        end if
        $n \leftarrow$ number of attributes
        for $i = 0$ to $(n-1)$ do
            create: $<uri><a_i><v_i>$
        end for
        set the leaf identifier of the first position stored element in the storage to 1 /* means not a leaf element */
        $leafIdentifier \leftarrow 0$
        store the current element's resources ($e$, $uri$, $n$, $leafIdentifier$) by adding to the first position in the storage
    end if
    if $characters$ then
        if $d \neq \lambda$ then
            store $d$ to the current element's resources
        end if
    end if
    if $endElement$ then
        if not root element then
            if $leafIdentifier = 0$ AND $d \neq \lambda$ then
                if $n > 0$ then
                    create: $<uri><e><d>$
                else
                    create: $<parent\_uri><e><d>$
                end if
            end if
            if $leafIdentifier = 1$ OR $n > 0$ then
                create: $<parent\_uri>$<hasResource>$<uri>$
            end if
        end if
        remove the current element's resources ($e$, $uri$, $n$, $leafIdentifier$, $d$) from the storage
    end if
    go to next event
end while
```

2.4.3 Proof

This section mathematically proves that the developed algorithm (Algorithm 1) is correct. The XML document is encoded as mathematical notations and loop invariant technique is used to prove that the algorithm works correctly. A loop invariant is a statement of the conditions that should be true on entry into the loop and remains true (holds) on every iteration of the loop. In this case, "*correct RDF statements are generated*" is the loop invariant. The three properties need to hold (be true) in loop invariant technique are initialization, maintenance, and termination [36].

An XML document can be graphically represented as a tree (Figure 2-8). The elements of an XML document form a tree starting from the root element and branching out to leaf elements [37]. In this research, the XML tree's orientation is horizontal rather than vertical, as in [37], in order to easily map with XML elements written in the XML document. Each element, consisting of element name, attributes and element data (only the leaf elements contain data), is represented as a vertex of the tree.

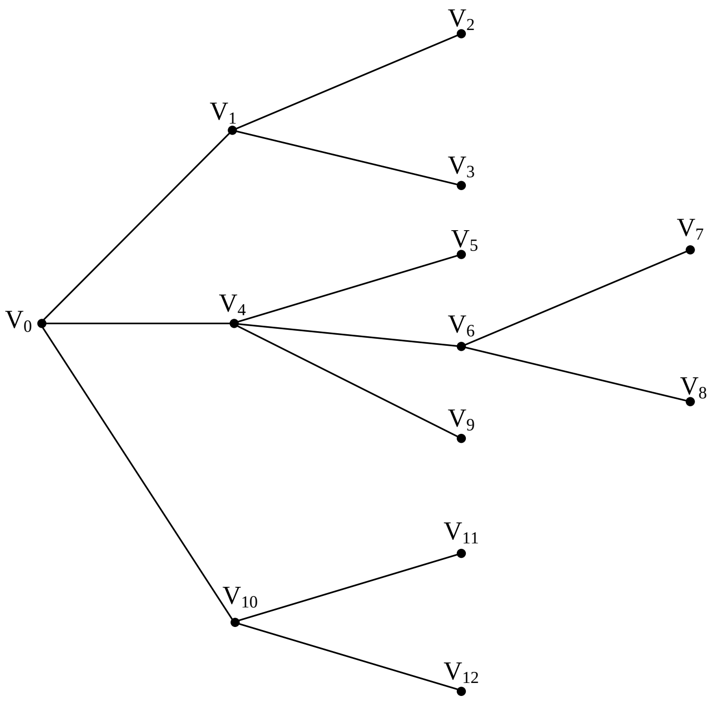


Figure 2-8: XML tree

The XML document is traversed from top to bottom and based on the order of the XML elements from top-to-bottom, the vertices are numbered accordingly. In the example given in Figure 2-8, V_0 is the root element, followed by its first child element V_1. As V_2, the first child of V_1, is already a leaf element, the traverse process continues to the second child of V_1, a sibling of V_2, and names it V_3. Children on the same level are called siblings. After the traverse process has finished with all V_1's children, it continues to V_1's siblings and so on. In this example, there are 13 vertices in total.

Definition 3: The collection of all XML elements of the document is defined as the set below:

$$V = \{(e_i, n_i, A_i, d_i): i \in [0, p) \wedge i \in \mathbb{N}\}$$

Where:

- p is the number of XML elements.
- $A_i = \{(a_k, v_k): k \in [0, n_i) \wedge k \in \mathbb{N}\}$ is the collection of (attribute name, attribute value) pairs in the XML element V_i.

Algorithm 1 traverses the XML document from top to bottom and thus goes through all p XML elements (represented as p vertices of the XML tree). Each iteration of the outer most *while* loop is either *startElement*, *characters*, or *endElement*. A combination of these three iterations forms an iteration that passes over one vertex of the XML tree. Therefore, the loop invariant technique was chosen to apply to the (combined) iteration of (*startElement*, *characters*, or *endElement*). There are p iterations in total for an XML document that has p elements.

2.4.3.1 Initialization

Before the first iteration of the loop in Algorithm 1, no RDF statement has been created yet as expected and therefore the loop invariant holds for the *initialization* property.

2.4.3.2 Maintenance

Assuming that the loop invariant holds before an iteration of $i = j,\ j \in [0, p)$, it is needed to prove that it still holds after the iteration. Having $V_j = (e_j, n_j, A_j, d_j)$ where $A_j = \{(a_k, v_k) : k \in [0, n_j) \wedge k \in \mathbb{N}\}$, there are four cases below:

- $d_j \neq \lambda \wedge n_j > 0$ (leaf element that has attributes):

  This case means that this iteration processes a leaf XML element with n_j attributes. It is expected to have $n_j + 2$ RDF statements created after the iteration. They are $< parent\_uri >< \text{hasResource} >< uri >$ statement, $< uri >< e_j >< d_j >$ statement, and n_j statements created from n_j attributes as the form of $< uri >< a_k >< v_k >$.

  In the *startElement* event, the loop goes to line 10. Since $n_j > 0$, n_j RDF statements of the form of $< uri >< a_k >< v_k >$ are created at line 11 for n_j attributes. In the *characters* event, no RDF statement is created. However, in the *endElement* event, the loop goes to line 24 and 31. At line 24, as this is a leaf element (*leafIdentifier* variable has not been assigned the value of 1, at line 13, yet), the loop passes the *if* condition and goes to line 25 and as $n_j > 0$, the loop goes to line 26 and creates the $< uri >< e_j >< d_j >$ statement. At line 31, as $n_j > 0$, the loop passes the *if* condition and goes to line 32 where the $< parent\_uri >< \text{hasResource} >< uri >$ statement is created. Therefore, $n_j + 2$ expected RDF statements are created after the iteration.

- $d_j \neq \lambda \wedge n_j = 0$ (leaf element that has no attribute):

  This case means that this iteration processes a leaf XML element with no attribute. It is expected to have one RDF statement, $< parent\_uri >< e_j >< d_j >$, created after the iteration.

  In the *startElement* event, the loop goes to line 10. Since $n_j = 0$, no RDF statement is created as the *for* loop condition is not met. In the *characters* event, no RDF statement is created either. However, in the *endElement* event, the loop goes to line 24 and 31. At line

24, as this is a leaf element (*leafIdentifier* variable has not been assigned the value of 1, at line 13, yet), the loop passes the *if* condition and goes to line 25 and as $n_j = 0$, the loop gets to line 28 and creates the $<parent\_uri><e_j><d_j>$ statement. At line 31, since *leafIdentifier = 0* and $n_j = 0$, the loop does not pass the *if* condition and no RDF statement is created. Therefore, the only expected RDF statement, $<parent\_uri><e_j><d_j>$, is created after the iteration.

- $d_j = \lambda \wedge n_j > 0$ (resource element that has attributes):

  This case means that this iteration processes a resource (non-leaf) XML element with n_j attributes. $n_j + 1$ RDF statements are expected to be created after the iteration if the XML element is not the root element; they are $<parent\_uri><\text{hasResource}><uri>$ statement, and n_j statements created from n_j attributes in the form of $<uri><a_k><v_k>$. If the XML element is the root element then only n_j statements of the form of $<uri><a_k><v_k>$ are expected to be created as there is no parent of the root element and thus $parent\_uri$ cannot be constructed.

  In the *startElement* event, the loop goes to line 10. As $n_j > 0$, n_j RDF statements of the form of $<uri><a_k><v_k>$ are created at line 11 for n_j attributes. In the *characters* event, no RDF statement is created. However, in the *endElement* event, the loop first goes to line 23. If this is not the root element, the loop goes to line 24 and line 31. At line 24, as this is a resource element (*leafIdentifier* variable has been assigned the value of 1, at line 13), the loop does not pass the *if* condition and no RDF statement is created. At line 31 the loop passes the *if* condition and goes to line 32 where the $<parent\_uri><\text{hasResource}><uri>$ statement is created. Therefore, $n_j + 1$ expected RDF statements are created after the iteration for this case (not the root element). On the other hand, if this is the root element, the loop does not pass the *if* condition at line 23 and thus only n_j statements in the form of $<uri><a_k><v_k>$ for n_j attributes are created as expected.

- $d_j = \lambda \wedge n_j = 0$ (resource element that has no attribute):

  This case means that this iteration processes a resource (non-leaf) XML element with no attributes. One RDF statement is expected to be created after the iteration if the XML element is not the root element; it is $< parent\_uri >< \text{hasResource} >< uri >$ statement. If the XML element is the root element then no statement is expected to be created as there is no parent for the root element and thus $parent\_uri$ cannot be constructed.

  In the *startElement* event, the loop goes to line 10. Since $n_j = 0$, no RDF statement is created as the *for* loop condition is not met. In the *characters* event, no RDF statement is created either. However, in the *endElement* event, the loop first goes to line 23. If this is not the root element, the loop goes to line 24 and line 31. At line 24, as this is a resource element (*leafIdentifier* variable has been assigned the value of 1, at line 13), the loop does not pass the *if* condition and no RDF statement is created. At line 31, since *leafIdentifier = 1*, the loop passes the *if* condition and creates the $< parent\_uri >< \text{hasResource} >< uri >$ statement which is expected. If this is the root element, the loop does not pass the *if* condition at line 23 and therefore, as expected, no RDF statement is created.

From all the four cases above, it is concluded that if the loop invariant holds before an iteration of the loop then it still holds after the iteration. Thus, the loop invariant holds for the *maintenance* property.

2.4.3.3 Termination

Finally, it is necessary to prove that the loop terminates and examine what happens when the loop terminates [36]. The loop terminates when the outermost *while* loop reaches the end of the document. It is certain that the outermost *while* loop will reach the end of the document at some point as it traverses through all the elements of the XML document from top to bottom and therefore loop termination is proved. At this time, the loop has gone through p (combined) iterations of (*startElement*, *characters*, and *endElement*) and has converted p XML elements, which are the whole XML document, into correct RDF statements. Therefore, it is concluded that the X2R algorithm (Algorithm 1) works correctly.

2.5 Implementation

The X2R tool was developed in Java using the SAX parser, to parse input XML files, and the Jena library to create RDF documents. The SAX parser parses each line of the XML document and provides event handlers for the starting of the document (*startDocument*), the ending of the document (*endDocument*), the starting of an element (*startElement*), the ending of an element (*endElement*), and the *characters* event which, in each call, gives the text string following each XML tag (either an opening tag or a closing tag). In the *startElement* event handler, the XML tag name and attribute list of the element are obtained. In the *characters* event handler, *text* (data) is obtained. While this utilizes the SAX parser instead of developing code from scratch to obtain each XML element, the developed code complies with *Algorithm 1*. The implemented algorithm generates a collection of RDF data (subject, predicate, object). These RDF data are then passed through the Jena library to create the RDF document. The implemented code in Java of *Algorithm 1*, which is the main focus of the X2R tool, is shown in Appendix A.

2.6 Evaluation

2.6.1 Experimental Background

Although an input XML file can belong to any of the six partitions, P1-6 (Figure 2-1), the transformation procedure does not rely on existing ontologies (section 2.4) and therefore input XML files belonging to P1 also belong to P2, input XML files belonging to P3 also belong to P4, and input XML files belonging to P5 also belong to P6. This section prepares XML files that cover all cases of input XML files to test and evaluate both the developed transformation procedure and the X2R tool. These include XML alone, XML with XML Schema, and XML with DTD. In addition, XML files that contain namespaces and empty elements are also covered.

2.6.1.1 Transformation Procedure

An XML file named *books1.xml* that contains attributes, middle elements and leaf elements (three cases for conversion to RDF statements) was chosen for experimentation to cover both partitions P5 and P6 (Figure 2-1). Below is the content of the *books1.xml* file (Figure 2-9).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore country="NZ" city="Auckland">
 <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
 </book>
 <book category="WEB">
    <title lang="en">XQuery Kick Start</title>
    <author>James McGovern</author>
    <author>Per Bothner</author>
    <author>Kurt Cagle</author>
    <year>2003</year>
    <price>49.99</price>
 </book>
 <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
 </book>
</bookstore>
```

Figure 2-9: *books1.xml* document

To cover both partition P3 and P4 (Figure 2-1), another XML file named *shiporder.xml* that adheres to a schema file named *shiporder.xsd* was used in experimentation. The content of *shiporder.xml* is displayed below (Figure 2-10).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<shiporder orderid="889923"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>
```

Figure 2-10: *shiporder.xml* document

It can be seen that, from a transformation procedure perspective, the schema declaration has added two attributes *xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"* and *xsi:noNamespaceSchemaLocation="shiporder.xsd"* to the root element *<shiporder>* and these are the only differences from an XML file belonging to partitions P5 and P6. If the transformation procedure works correctly with the *books1.xml* file, then it should also work correctly with the *shiporder.xml* file as it considers these differences as two extra attributes added to the root element <*shiporder*>.

To cover partitions P1 and P2 (Figure 2-1), an XML file named *notes.xml* that conforms to a DTD file named *notes.dtd* was used in experimentation. Below is the content of the *notes.xml* file (Figure 2-11).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE notes SYSTEM "notes.dtd">
<notes company="AUT" year="2013">
    <note date="16/01" lang="en">
        <to>Tove</to>
        <from>
            <name>Jani</name>
            <age>30</age>
            <address>241 Dominion Rd</address>
        </from>
        <heading>Reminder</heading>
        <body>Don't forget me this weekend!</body>
    </note>
    <note date="17/01" lang="en">
        <to>Steve</to>
        <from>
            <name>Elvis</name>
            <age>30</age>
            <address>168 Herbert Rd</address>
        </from>
        <heading>Reminder</heading>
        <body>Play soccer this weekend!</body>
    </note>
</notes>
```

Figure 2-11: *notes.xml* document

It can be seen that the DTD file declaration *<!DOCTYPE notes SYSTEM "notes.dtd">* is outside the root element *<notes>* and this is the only difference from an XML file belonging to partitions P5 and P6. However, the transformation procedure only cares about what inside the root element and thus, from its perspective, this input file *notes.xml* is also the same as an input file belonging to partitions P5 and P6. Consequently, if the transformation procedure works correctly with the *books1.xml* file then it also works correctly with the *notes.xml* file.

In XML, element names are defined by developers. This results in a conflict when two elements have the same name but mean different things. For example, two elements may be named *<table>* but one means an HTML (HyperText Markup Language) table and the other means a real table. Namespaces are introduced to solve this conflict and can be declared in the elements, where they are used, or in the root XML element [38]. A namespace is defined by the *xmlns* (XML namespace) attribute. Below are two examples of the use of namespaces, *table_ns1.xml,* declares namespaces at the root element and the other, *table_ns2.xml,* declares namespaces at elements where they are used (Figure 2-12 and Figure 2-13).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<root xmlns:h="http://www.w3.org/TR/html4/" xmlns:f="http://www.w3schools.com/furniture">
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
</root>
```

Figure 2-12: *table_ns1.xml* document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<root>
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
<f:table xmlns:f="http://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
</root>
```

Figure 2-13: *table_ns2.xml* document

However, from the transformation procedure perspective, a namespace declaration is just an attribute. Therefore, if the transformation procedure works correctly with the *books1.xml* file then it should also work correctly with *table_ns1.xml* and *table_ns2.xml*.

Cases where the input XML file contains empty elements was also examined in the XML file named *slideshow.xml*. An empty element is an element without *element text*. In this example, four elements listed in bold are empty elements. Below is the content of the *slideshow.xml* file (Figure 2-14).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<slideshow
    title="Sample Slide Show"
    date="Date of publication"
    author="Yours Truly">
```

```
  <!-- TITLE SLIDE -->
  <slide type="all">
    <title>Wake up to WonderWidgets!</title>
  </slide>

  <!-- OVERVIEW -->
  <slide type="all">
      <title>Overview</title>
      <item>Why WonderWidgets are great</item>
      <item />
      <item></item>
      <item id="1" lang="jp" />
      <item id="2" lang="vn"></item>
      <item>Who buys WonderWidgets</item>
  </slide>
</slideshow>
```

Figure 2-14: *slideshow.xml* document

It can be seen that *<item />* and *<item></item>* are empty elements that have no attribute. According to the three conversion rules of the transformation procedure, no RDF statement will be created for these two empty elements. On the other hand, *<item id="1" lang="jp" />* and *<item id="2" lang="vn"></item>* are also empty elements but have attributes and therefore the transformation procedure creates RDF statements for these attributes as usual.

To sum up, *books1.xml*, *shiporder.xml*, *notes.xml*, *table_ns1.xml*, *table_ns2.xml*, and *slideshow.xml*; are being XML files that cover all cases of input XML documents, were examined. While the *books1.xml* file alone was enough to test and evaluate the transformation procedure, the other XML files were used as input for testing and evaluating the X2R tool to see whether it implemented the transformation procedure correctly.

2.6.1.2 X2R tool

In this experiment, the X2R tool was a Java command line tool running on a Windows 7, 64 bit machine (Intel Core i5, 2.67 GHz, 4 GB RAM). The heap size for JVM (Java Virtual Machine) was set to 1024 MB.

Eighteen XML files were created to evaluate the X2R tool (refer to Table 2-3). For file #1 to file #10 which were originated from *books1.xml* containing *<book>* elements, the number of *<book>* elements was increased until the X2R tool threw an *OutOfMemoryError* exception. Files #11 and #12 were accompanied by DTD files while files #13 and #14 had XSD (XML Schema Definition) files. File #11 contained a single note rooted by the *<note>* element while file #12 contained multiple notes rooted by the *<notes>* element, which holds multiple *<note>* elements, and thus has an XML tree one level deeper. File #17 (*slideshow.xml*) has empty elements. File #18 (*special.xml*) contains texts in non-leaf elements for abnormal input testing. Files #1 – 10, file #12, and files #14 – 18 have already been displayed. Figure 2-15 and Figure 2-16 show the contents of files #11 (*note.xml*) and #13 (*addresses.xml*).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
    <to>Tove</to>
    <from>
        <name>Jani</name>
        <age>30</age>
        <address>241 Dominion Rd</address>
    </from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Figure 2-15: *note.xml* document

```
<?xml version="1.0" encoding="UTF-8"?>
<addresses xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="addresses.xsd">

 <address>
  <name>Joe Tester</name>
  <street>Baker street 5</street>
 </address>

 <address>
  <name>Anna Fraser</name>
  <street>Albert Street 100</street>
 </address>

 <address>
  <name>Donna Lynch</name>
  <street>Queen Street 20</street>
 </address>

 <address>
  <name>Elvis Musambi</name>
  <street>Herbert Rd 16</street>
 </address>

</addresses>
```

Figure 2-16: *addresses.xml* document

2.6.2 Results

2.6.2.1 Transformation Procedure

The XML file *books1.xml* was successfully converted into RDF in the experiment. Figure 2-17 shows the transformed RDF graph of the XML file using the transformation procedure in the "walk-through" technique. The ovals denote *resource* nodes while the squares represent *literal* nodes. The *arcs* express the *predicates*. As mentioned in Rule number 2 in the transformation procedure, the three <*book*> elements in the XML file were transferred to correspondingly <*book*>, <*book1*> and <*book2*> nodes as shown in the RDF graph.

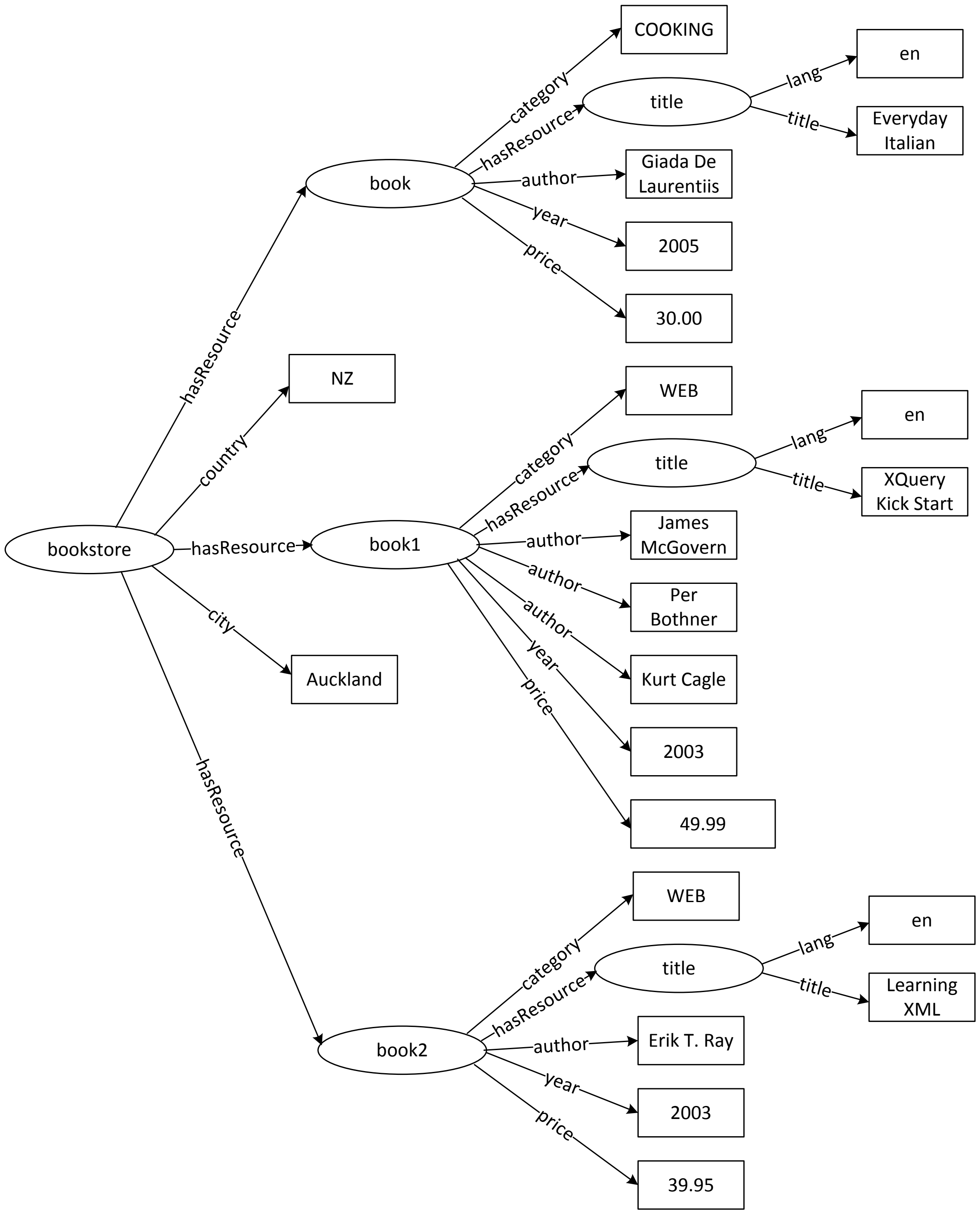


Figure 2-17: Transformed RDF graph

2.6.2.2 X2R tool

Running the X2R tool with the input *books1.xml* file, an RDF document named *books1.rdf* was generated in “RDF/XML” format as the RDF format was set to “RDF/XML”. However, the X2R tool supports five different types of RDF formats, “RDF/XML”, “RDF/XML-ABBREV”, “N-TRIPLE”, “N3”, and “TURTLE”. Appendix B shows the generated RDF files in “RDF/XML-ABBREV”, “N-TRIPLE”, “N3”, and “TURTLE” formats. The content of *books1.rdf* (“RDF/XML” format) is displayed below.

```
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:hms="http://x2r.com/Property/" >
  <rdf:Description rdf:about="http://x2r.com/Resource/bookstore">
   <hms:hasResource rdf:resource="http://x2r.com/Resource/bookstore/book2"/>
   <hms:hasResource rdf:resource="http://x2r.com/Resource/bookstore/book1"/>
   <hms:hasResource rdf:resource="http://x2r.com/Resource/bookstore/book"/>
   <hms:city>Auckland</hms:city>
   <hms:country>NZ</hms:country>
  </rdf:Description>

  <rdf:Description rdf:about="http://x2r.com/Resource/bookstore/book2">
   <hms:price>39.95</hms:price>
   <hms:year>2003</hms:year>
   <hms:author>Erik T. Ray</hms:author>
   <hms:hasResource rdf:resource="http://x2r.com/Resource/bookstore/book2/title"/>
   <hms:category>WEB</hms:category>
  </rdf:Description>

  <rdf:Description rdf:about="http://x2r.com/Resource/bookstore/book1">
   <hms:price>49.99</hms:price>
   <hms:year>2003</hms:year>
   <hms:author>Kurt Cagle</hms:author>
   <hms:author>Per Bothner</hms:author>
```

```
    <hms:author>James McGovern</hms:author>
      <hms:hasResource rdf:resource="http://x2r.com/Resource/bookstore/book1/title"/>
<hms:category>WEB</hms:category>
   </rdf:Description>

   <rdf:Description rdf:about="http://x2r.com/Resource/bookstore/book2/title">
<hms:title>Learning XML</hms:title>
<hms:lang>en</hms:lang>
   </rdf:Description>

   <rdf:Description rdf:about="http://x2r.com/Resource/bookstore/book/title">
<hms:title>Everyday Italian</hms:title>
<hms:lang>en</hms:lang>
   </rdf:Description>

   <rdf:Description rdf:about="http://x2r.com/Resource/bookstore/book">
<hms:price>30.00</hms:price>
<hms:year>2005</hms:year>
<hms:author>Giada De Laurentiis</hms:author>
<hms:hasResource rdf:resource="http://x2r.com/Resource/bookstore/book/title"/>
<hms:category>COOKING</hms:category>
   </rdf:Description>

   <rdf:Description rdf:about="http://x2r.com/Resource/bookstore/book1/title">
<hms:title>XQuery Kick Start</hms:title>
<hms:lang>en</hms:lang>
   </rdf:Description>


</rdf:RDF>
```

Figure 2-18: *books1.rdf* document

Then W3C RDF Validation Service (http://www.w3.org/RDF/Validator/) was used to test the generated RDF document. It resulted positively with the conversion accuracy of 100%. The X2R tool and the XMLtoRDF tool of Petruska and Curda [33] also ran all eighteen input XML files and obtained the results recorded in Table 2-3.

Table 2-3: SUMMARY OF CONVERSION USING XMLTORDF TOOL AND X2R TOOL

No.	File (.xml)	Size (KB)	Expected no. of RDF Statements	Execution time (millisecond)		No. of RDF statements created		Notes
				XMLtoRDF	X2R	XMLtoRDF	X2R	
1	books1	1	28	468	468	28	28	
2	books2	2	54	483	484	54	54	books1 × 2 (*)
3	books3	7	262	515	546	262	262	books1 × 10
4	books4	7	288	530	531	291	288	books3 + books1
5	books5	14	574	577	593	579	574	books4 × 2
6	books6	261	11,442	1,669	1,638	11,484	11,442	books5 × 20
7	books7	2,600	114,402	3,729	3,541	114818	114,402	books6 × 10
8	books8	25,997	1,144,002	23,603	18,127	1148154	1,144,002	books7 × 10
9	books9	51,993	2,288,002	OutOfMemory	251,069	2,296,307	2,288,002	books8 × 2
10	books10	77,989	3,432,002	OutOfMemory	OutofMemory			books9 + books8
11	note	1	7	468	468	7	7	with DTD
12	notes	1	22	493	488	22	22	with DTD
13	addresses	1	14	515	453	14	14	with XSD
14	shiporder	1	18	499	452	18	18	with XSD
15	table_ns1	1	10	546	437	10	10	Declares namespaces at root element.
16	table_ns2	1	10	609	453	10	10	Declare namespaces where they are used.

17	slideshow	1	17	468	499	17	17	Has empty elements
18	special	1	8	468	468	10	8	Contains *texts* not enclosed by *opening tag – closing tag* envelop.

(*): books2's content is created by duplicating books1's content

First, the X2R tool is accurate in terms of the number of RDF statements generated while the XMLtoRDF tool is not. When the input size is small (*books1*, *books2* and *books3*), both tools generate the same number of RDF statements as expected. However, when the input size increases (*books4* onward), the XMLtoRDF tool creates more RDF statements than expected which is not correct. This would suggest there are bugs in the XMLtoRDF tool.

From the eighteen input XML files, the expected number of RDF statements for each input file was calculated as follows. For *books1.xml*, *note.xml*, *notes.xml*, *addresses.xml*, *shiporder.xml*, *table_ns1.xml*, *table_ns2.xml*, *slideshow.xml* and *special.xml* files, the expected number of RDF statements was manually counted by using the transformation procedure. For the nine input XML files (*books2.xml*, *books3.xml…books10.xml*) originated from the *books1.xml* file, the expected number of RDF statements was calculated based on *books1.xml* file.

Syntax checking of the output RDF files generated by the X2R tool was carried out by the W3C RDF Validation Service (http://www.w3.org/RDF/Validator/) on *books1.rdf*, *books2.rdf*, *books3.rdf*, *books4.rdf*, *note.rdf*, *notes.rdf*, *addresses.rdf*, *shiporder.rdf*, *table_ns1.rdf*, *table_ns2.rdf*, *slideshow.rdf* and *special.rdf*. All the RDF documents were validated successfully with an accuracy rate of 100%. The five output files, *books5.rdf*, *books6.rdf*, *books7.rdf*, *books8.rdf and books9.rdf*, were too big to be validated using the W3C RDF Validation Service, which only accepts RDF files of less than 200 KB. However, they should also be valid inasmuch as *books5.xml*, *books6.xml*, *books7.xml*, *books8.xml and books9.xml* all originated from *books1.xml* file.

Second, in terms of running time, the X2R tool can be considered the better. When the input size is small to medium (*books1* to *books7*), the running time of both tools is quite similar. When the input size is large (*books8* onward), the X2R tool outperforms the XMLtoRDF tool. With *books9* (over 2 million RDF statements), the X2R tool continues to work well, while the XMLtoRDF tool runs out of memory.

Third, *note.xml*, *notes.xml*, *addresses.xml*, and *shiporder.xml* were used to test the X2R tool with XML files accompanied by DTD or XML Schema. As expected, it worked on the same principles with XML alone documents and generated correct output. In addition, *table_ns1.xml* and *table_ns2.xml* were used to see how the X2R tool handles namespaces in input XML files and showed the X2R tool works correctly, as expected.

Fourth, with the *slideshow.xml* input XML file, no RDF statements were generated for the completely empty elements *<item />* and *<item></item>* and this is correct. For the other empty elements which have attributes *<item id="1" lang="jp" />* and *<item id="2" lang="vn"></item>*, the X2R tool generated RDF statements for the attributes as normal. The output files *note.rdf*, *notes.rdf*, *addresses.rdf*, *shiporder.rdf*, *table_ns1.rdf*, *table_ns2.rdf*, *slideshow.rdf* and *special.rdf* can be seen in Appendix C.

Finally, with *special.xml* (see section 2.4.2), the XMLtoRDF tool gave the result represented by the RDF graph in Figure 2-19. “bbb” does not belong to the *<tag1_2>* element and therefore the XMLtoRDF tool using *tag1_2* to describe “bbb” is not correct. The X2R tool’s result does not have the two RDF statements constructed from “aaa” and “bbb” as can be seen in Figure 2-20.

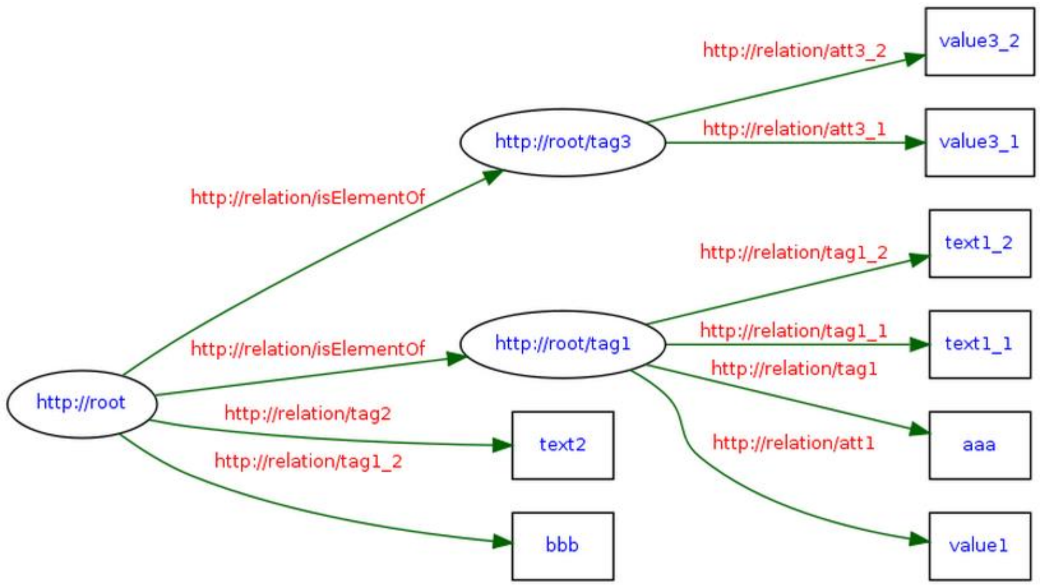


Figure 2-19: RDF graph converted by XMLtoRDF tool for special.xml file

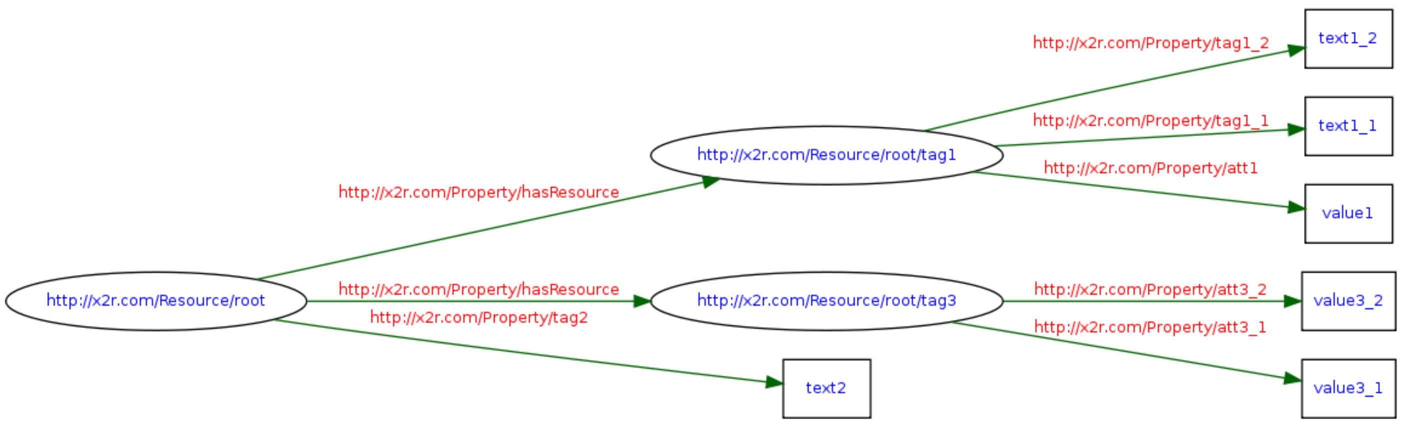

http://x2r.com/Resource/root
http://x2r.com/Property/hasResource
http://x2r.com/Resource/root/tag1
http://x2r.com/Property/tag1_2
text1_2
http://x2r.com/Property/tag1_1
text1_1
http://x2r.com/Property/att1
value1
http://x2r.com/Property/hasResource
http://x2r.com/Resource/root/tag3
http://x2r.com/Property/att3_2
value3_2
http://x2r.com/Property/att3_1
value3_1
http://x2r.com/Property/tag2
text2

Figure 2-20: RDF graph converted by X2R tool for special.xml file

2.6.3 Analysis

This section analyzes the XMLtoRDF and X2R tools, focusing on the algorithms used in the two tools to find the reason why the XMLtoRDF tool is not as accurate and has lower performance than the X2R tool. Table 2-4 shows the analysis.

Table 2-4: ANALYSIS OF XMLTORDF TOOL AND X2R TOOL

No.	Problems with XMLtoRDF	Reason	Solutions in X2R

1	Wrong number of converted RDF statements for *books4*, *books5*, *books6*, *books7*, *and books8*.	SAX parser parses the XML document serially and adds up XML elements into a character array that is 2,048 in length. When the character array is fully filled it is reset and the following characters are filled into the start of the character array. In the *characters*(char[] ch, int start, int length) function, the *text* is obtained by: `String sText = new String(ch, start, length);` Thus, a *text* that would fill over the upper bound position of the character array (start + length >= 2048) is chopped into 2 pieces because of the reset, and the *characters* function is called twice instead of once. For example, the title “Everyday Italian” can be chopped into two pieces “Eve” and “ryday Italian”. This results in broken texts being used to create RDF statements and the number of RDF statements increases.	Add the following code into *characters* function to connect the two broken pieces into one (chopped variable is initialized as false).
			`if ((start + length) >= 2048) {`
			`    chopped = true;`
			`    firstPiece = sText;`
			`    return;`
			`}`
			`if (chopped) {`
			`    sText = firstPiece + sText;`
			`    chopped = false;`
			`}`

2	Wrong number of converted RDF statements and wrongly converted RDF statements for *special.xml*	XMLtoRDF algorithm obtains a *text* within the *characters* function. It decides that if the *text* is not null, the *text* is the content of the XML opening tag immediately before the *text* and carries out the RDF statement creation in the same *characters* function. Therefore, in this case, *<tag1_2>text1_2</tag1_2>bbb</tag1>*, it sees that "bbb" is the content of *tag1_2* tag which is not correct.	In the X2R algorithm, a *text* is obtained within the *characters* function too, but RDF statement creation is carried out in the *endElement* function. In the *endElement* function, it is decided whether the *text* belongs to a leaf XML element or not and the RDF statement contains the *text* is created only in the case of a leaf element.
3	Lower performance (runs slower and runs out memory faster) than the X2R tool	In the XMLtoRDF tool, all created RDF components sets (a set contains a subject string, a predicate string, and an object string) are added into a LinkedList object which is passed to a function where the program iterates the LinkedList object to create each RDF statement and adds them to a Jena model. Constructing the LinkedList object and iterating it is wasteful as this LinkedList object consumes a lot of memory and thus the XMLtoRDF tool runs out of memory faster. Reasons for the low performance of XMLtoRDF tool can also be attributed to the lengthy algorithm as explained in problem #4.	In the X2R tool, an RDF statement is created and added to the Jena model immediately when an RDF components set is obtained. This eliminates creating and iterating the LinkedList object as the XMLtoRDF tool does.

4	The algorithm is lengthy and hard to follow.	The XMLtoRDF algorithm determines which types of RDF statements to create in *characters* function alone. It introduces seven cases for creating RDF statements and makes seven corresponding functions, *createEndElementRDF, createEndElementRDFBegin, createEndElementRDFFirst, createEndElementRDFFromAtribte, createEndElementRDFToTagWithAtribut, createMiddleElementRDF,* and *createMiddleElementRDF*. To determine which case it is, many variables are used to support decision making. They are *wasContent, iterator, startB, processedAttribute,* and *identifikator*. These make the algorithm lengthy and hard to follow.	The X2R algorithm is shorter and easier to follow. It introduces only three cases with three corresponding functions, *createResourceTriple, createLiteralTriple*, and *createLiteralTriplesFromAtts*. These three functions reflect the three conversion rules stated in section 2.4.1. The *createLiteralTriplesFromAtts* function is called in the *startElement* function because in the *startElement* event, the list of attributes of the XML element is obtained and literal RDF statements for these attributes can be created (Conversion rule #1). The *createLiteralTriple* function is called in the *endElement* function as in *endElement* it is decided whether the element is a leaf or not. The *createLiteralTriple* function creates a literal RDF statement for the leaf element. (Conversion rule #3). The *createResourceTriple* function is called in the *endElement* function to create an RDF statement for an element which is not leaf or has attribute(s). (Conversion rule #2).

2.6.4 Discussion

Table 2-5 summarizes the results from applying the performance metric defined in section 2.3.6

Table 2-5: PERFORMANCE METRIC APPLICATION RESULTS FOR X2R TOOL

Execution accuracy	Time	Usability	Cross-platform
100%	OK	OK	Yes

The transformation procedure, which is accompanied by the implemented X2R tool, can be applied to transform real-world data being stored as XML to RDF. This can be confidently stated as the conversion accuracy was recorded as 100% for each of the six input XML partitions, made up of 100% accuracy for all input XML partitions. This outperforms techniques in the existing literature in terms of accuracy.

There could be an argument about the semantic validity of the transformation procedure as it does not care about the meanings of the XML tag names in the input XML file. However, recalling the condition that the input XML file must be valid, its tag names should already be valid and unified. The case of synonyms in tag names should not occur in a valid XML document. The case of homonyms in tag names should already be distinguished by using namespaces in the input XML file and thus also be distinguished in the output RDF document.

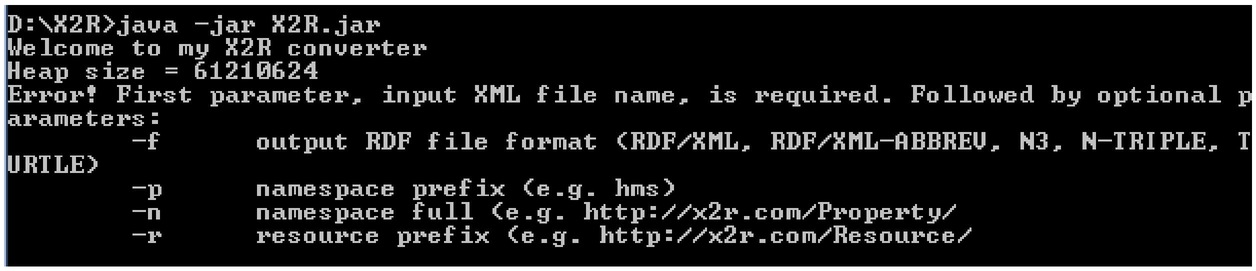
The X2R tool is simply a command line Java tool which takes one input XML document for each run. However, it can easily be upgraded to take multiple XML files (e.g. files stored in a folder). It can also be used as the core to develop a GUI stand-alone application or to develop a Web application to improve usability. As it is written in Java, it can run on any platform making it convenient for any user. In fact, an X2R Web tool version has been developed and is ready to use at http://x2r.aut.ac.nz.

Apart from 100% conversion accuracy, the conversion time of the tool is relatively small (refer to Table 2-3). As the tool uses the SAX parser, which is based on event handling (start/end of document, start/end of element and so on), it does not parse and store the whole XML document into memory (as DOM4J parser does). This helps to avoid out of memory errors which is a big concern with any application, especially a Java application.

However, the current version of the X2R tool still has an out of memory error when dealing with large XML file inputs (e.g. *books10* which contains about 3 million statements). The reason for this is attributed to its dependence on the Jena library to create an RDF document. Jena creates and adds all the RDF statements to one RDF *model* object and finally, it writes the whole *model* object to the output RDF file. The consequence is that the *model* object gets bigger and bigger until at some point the JVM heap space runs out of memory.

To solve this issue, two possible solutions are envisioned. The first one is to report to Apache so they can update their Jena framework. The second solution is to write one's own library based on the Jena code, for creating RDF statements and writing these statements into the output file. It should be about creating a certain number of RDF statements, appending these statements into the output file and discarding them before creating new RDF statements. The idea here is to avoid constructing all the RDF statements and adding them all to one object.

As the X2R tool is a Java application, it satisfies the cross-platform criterion on the performance metric. In terms of usability, the X2R tool gives descriptive prompts for users as exemplified in the following screenshot (Figure 2-21).



```
D:\X2R>java -jar X2R.jar
Welcome to my X2R converter
Heap size = 61210624
Error! First parameter, input XML file name, is required. Followed by optional p
arameters:
        -f      output RDF file format (RDF/XML, RDF/XML-ABBREV, N3, N-TRIPLE, T
URTLE)
        -p      namespace prefix (e.g. hms)
        -n      namespace full (e.g. http://x2r.com/Property/
        -r      resource prefix (e.g. http://x2r.com/Resource/
```

Figure 2-21: Prompts thrown by X2R tool for wrong command line parameters input

2.7 Summary

This chapter provides a transformation procedure to convert any valid XML document to RDF without referencing DTD, XML schemata or existing ontologies. The procedure consists of the three conversion rules based on the structure of XML (combination of attributes, XML tag values and parent-child relationships) and the X2R algorithm, which traverses the XML document to create RDF statements based on the three conversion rules. This procedure gives a conversion accuracy rate of 100%.

Chapter 3 Home Localization System for Misplaced objects

3.1 Introduction

Smart Home (SH) systems assist users with daily activities within the home by applying ubiquitous computing. Implementation of ubiquitous computing infrastructure helps manage objects in the home and enable real-world searches, which differ from the virtual world search of current search engines such as Google [13]. The Global Positioning System (GPS) is considered as the best technology for outdoor localization systems but performs poorly in accuracy and precision in the indoor environment [39]. In addition, satellite signals received in the indoor environment are weak and therefore GPS is unlikely to be able to be used indoor [40, 41]. To overcome these difficulties, various wireless technologies have been experimented with including infrared, bluetooth, *IEEE 802.11* wireless LAN, ultrasonic, video camera, and Radio Frequency IDentification (RFID) [15, 39, 42]. RFID seems to be the most suitable candidate for tracking and localization in SH systems because of its robustness, low price, and flexibility [15]. In addition, RFID can uniquely identify physical artifacts and can be used as a universal entry point to integrate them into the Internet of Things (IOTs) [11], and thus, RFID presents a viable solution in building SH systems. Considered as a main application domain of IOTs, SH systems have attracted much attention recently e.g. [43].

An RFID system has two main components, transponders (RFID tags) and detectors (RFID readers). RFID tags are divided into three categories, active tags, passive tags, and semi-passive tags. The type of RFID system is based on whether the tags are active, passive or semi-passive. For example, if the system uses only active tags then it is called an active RFID system. An active tag has its own power source (battery) to run its internal circuitry and to broadcast signals to readers. A passive tag does not have its own power source and cannot initiate communication with the reader. It is activated by and draws energy from the reader’s signal to send back signals to the reader. A semi-passive tag does not initiate the communication with the reader either, but it has its own battery to run its internal circuitry. Active tags are bigger in size, transmit longer-range signals, and are more expensive than passive tags.

This chapter proposes a solution for localization of easily-lost objects (LOs) within the home and describes a possible implementation of the whole system. The solution uses passive RFID technology for localization and stores information as an ontology. Information stored as an ontology enables the application of reasoning to the search and facilitates its integration with the entire SH system [44]. It also ensures adaptability with the emerging Semantic Web. Prior to this study, many researchers have proposed various RFID solutions for indoor localization. However, these solutions tend to be confined to the prototype or laboratory experimentation stage and much remains to be done to bring them into everyday application due to issues with cost, accuracy, and usability [45]. Unlike these existing solutions, this research presents a solution that is novel and viable to be deployed on a large-scale.

The solution in this study can give the location of the LO with enough detail to show it is on/in (at) a piece of furniture or between some pieces of furniture in the room, for example, "the glasses are at the kitchen table". The fact that the LO is either on/in a piece of furniture or between some pieces of furniture in the room means localization of the LO is classified into two types. Localization of the LO presents on/in a piece of furniture is classified as localization Type 1. On the other hand, localization Type 2 is the localization of the LO between pieces of furniture.

The rest of this chapter is organized as follows. First, section 3.2 analyzes related work to identify challenges with indoor localization and to find out the role RFID can play. Next, section 3.3 provides details of the methodology used in the research for this module, Home Localization System for Misplaced objects (HLSM). Then, section 3.4 describes the solution of this study and section 3.5 explains the implementation and experiments. Further, section 3.6 presents results and discussion of the proposed solution. Finally, Section 3.7 concludes this chapter.

3.2 Literature Review

3.2.1 Challenges with indoor localization

Many studies have emphasized the development of a robust system for indoor localization [45]. However, they all face difficulties in developing a real-world system and thus are still in prototype or experimental stages. Unlike outdoor environments, where GPS has been found effective in localization with an accuracy obtainable of approximately 3 m, indoor environments typically demand more in accuracy of a localization system if the system is to be useful [40, 41]. Combining [40, 41] with another study by Fortin-Simard et al. [15], the following six challenges to an indoor localization system have been derived.

- Accuracy: Indoor environments are limited by the sizes of the rooms and buildings. Indoor objects are relatively small and close to each other. Thus, the accuracy of the system should be high, within 3 feet (about 1 m) [45].

- No need for direct line-of-sight: there are many objects within an indoor environment and therefore these objects can stand in between the detector and the tracking object. A system that needs direct line-of-sight is not suitable.

- Low cost: Due to the number of tracking objects, rooms, and buildings. The cost always needs to be considered to see whether implementing the system is viable.

- Flexibility: The system should be flexible enough to be applied to different tracking items, different rooms and buildings with minimum modification.

- Non-intrusiveness: the localization system should assist home users, not intrude on their everyday lives. For example, a camera system is intrusive and not suitable for indoor tracking and localization for SH systems.

- Usability: The user should find it is easy to use the developed system. Many solutions present the localization result to the user as coordinates for the tracked object and it is hard for the user to interpret where it is in the room. For example, it is not at all easy for the user to quickly imagine the actual location of a tracked object having the coordinates (3.8 m, 2.7 m) in a room. The user would find it much easier if the system tells them that the object is at the kitchen table, for example.

3.2.2 RFID in indoor localization

RFID has been the focus of research for a number of years and has been successfully applied in a number of scientific and technical fields, such as medicine, engineering, the aeronautics industry supply chain, and the retail industry [46]. In recent years, much attention has been paid to using RFID in indoor tracking and localization. Benefits of using RFID include; no need for direct line-of-sight, low cost, flexibility, and non-intrusiveness [15, 42].

Many studies have proposed RFID solutions for indoor localization. Table 4-1 shows some of these studies from the last decade. The table gives a summary of each proposed solution with the technology (active, passive, or semi-passive RFID) and its accuracy. The analysis of these studies is summarized in Table 4-2 that lists the challenges the solutions meet as well as specifying their limitations. It can be seen that no solution has met all six identified challenges mentioned in section 3.2.1. “Usability”, especially, has not been satisfied by any study, though passive RFID technology solutions have already satisfied the “Accuracy”, “No need direct line-of-sight”, “Low cost”, “Flexibility”, and “Non-intrusiveness” requirements. Most of the existing solutions present the localization results as coordinates of the tracking objects which is not user-friendly. Only the solution by Tesoriero et al. [39] developed a virtual map showing the position of the tracking object. However, the user still needs to relate the virtual map with the actual room and identify the physical location of the tracking object from its position on the virtual map. In the next section, this study presents a new solution that addresses the limitations of the existing literature, focusing especially on “Usability”.

Table 3-1: RFID BASED SOLUTIONS

Study	Technology	Solution
Ni et al., 2004 [42]	Active RFID	This is a 2D solution that uses reference tags. The distance between an active RFID tag and the RFID reader is estimated based on the power level (1 – 8) of the tag detected. The position of the tracking tag is decided from its *k* nearest neighbors (*k* nearest reference tags) and the highest accuracy is achieved when *k* = 4. To improve accuracy, multiple readers are used with the optimal number of readers being *n* = 4. The localization error of the system is less than 2 m.
Tesoriero et al., 2009 [39]	Passive RFID	This is a 2D solution that divides the floor surface into a grid of small squared location units. Each physical location unit is tagged by a passive RFID tag and mapped to a position on the virtual map in the system. The tracking object is attached with an RFID reader. Based on which tag the reader is reading, the tracking object’s location is mapped and shown on the virtual map in the system. The localization error of the system is about 0.9 m.
Almaaitah et al., 2010 [47]	Passive RFID	This paper introduces two methods for 3D localization of a passive RFID tag. The first method named Adaptive Power Multilateration (APM) uses 4 RFID readers and localizes the tag based on the minimal interrogation power and multilateration. The second method named Adaptive Power with Antenna Array (APAA) uses a single RFID reader equipped with horizontal and vertical smart antennas and the reader’s adaptive power levels. The APM method gives an accuracy of 0.32 m while the APAA method gives the accuracy of 0.48 m.
Saad and Nakad, 2011 [14]	Passive RFID	This 2D solution also attaches the reader to the tracking object. The passive RFID tags are placed along the object path as reference tags. Calculating the distances between the reader and the reference tags, the position of the object is derived. To have higher accuracy, both RSSI and an angle-dependent loss factor are involved in the proposed algorithm using a Kalman filter. Even though not specified, this can be applied to both 2D and 3D environment settings. The localization error of the system is about 0.1 m.
Yuhong and Ya, 2012 [48]	Passive RFID	This is a 2D solution that calculates the coordinates of the tracked passive RFID tag based on Angle of Arrival (AOA) and Time Difference of Arrival (TDOA) rather than RSSI. The system employs two RFID readers whirling around their fixed axes to scan tags in the room. The localization error of the system is less than 1 m.

Fortin-Simard et al., 2012 [15]	Passive RFID	This is a 2D solution that proposes a new algorithmic approach to localize the tracked passive RFID tag based on elliptical trilateration and fuzzy logic. Though the solution uses only RSSI as the input parameter to the algorithm, it employs multiple filters (iteration based filter, Gaussian mean weighting filter, delta filter and multi-point location filter) to achieve high accuracy. The localization error of the system is about 0.14 m.
Brchan et al., 2012 [49]	Active RFID	This is a 2D solution with an extension to a 3D solution that localizes the tracked active RFID tag base on RSSI with the use of multiple propagation models to improve the accuracy of the system. The system uses reference tags and aims to improve the LANDMARC system. However, the localization error of the 2D model does not show improvement, 58.3% of the time at less than 1 m and 1.7% of the time at more than 2 m. In the 3D model, the error is 9.7% of the time at more than 3 m.
DiGiampaolo and Martinelli, 2012 [16]	Passive RFID	In this solution (2D solution), passive reference tags are placed on the ceiling and the tracking object is attached with the RFID reader. A Quantized Extended Kalman Filter is applied in the algorithm for distance measurements. The localization error of the system is about 0.1 m.
Han et al., 2012 [50]	Active RFID	This is a 3D localization solution based on active reference tags placed as a 3D grid. The distance between 2 adjacent tags is 1 m. Each reader has 2 antennas. The algorithm uses 2 RSSI values reported by the 2 antennas and applies filter rules to estimate the distance between the tag and the reader. The average localization error of the system is about 0.54 m.
Athalye et al., 2013 [51]	Semi-passive RFID	This is both a 2D and 3D solution which uses passive RFID tags as reference tags and a newly developed semi-passive RFID tag component called sensatag for tagging the tracked object. The sensatag has dual functionality, first, detecting and decoding backscatter signals from the passive RFID tags, and second, communicating with the reader just like a standard passive tag. The system’s average error is about 0.30 m. This solution is applied to find the exact placement of items on shelves.

Table 3-2: RFID BASED SOLUTIONS ANALYSIS

Study	Challenges met	Limitations
Ni et al., 2004 [42]	- No need for direct line-of-sight - Flexibility - Non-intrusiveness	- Huge amount of RFID tags need to be placed on the floor. - Expensive due to the use of so many active tags - Effort in maintaining the active RFID tags' batteries - The reference tags make sense to computers but are hard for humans to interpret as physical locations.
Tesoriero et al., 2009 [39]	- Accuracy - No need for direct line-of-sight - Non-intrusiveness	- Huge amount of RFID tags need to be placed on the floor. - Expensive due to each tracking object being attached with an RFID reader. - Hard or impossible to attach the RFID reader to small objects, such as keys, glasses, wallets etc.
Almaaitah et al., 2010 [47]	- Accuracy - No need for direct line-of-sight - Low cost - Flexibility - Non-intrusiveness	- The computed coordinates make sense to computers but are hard for humans to interpret as physical locations.
Saad and Nakad, 2011 [14]	- Accuracy - No need for direct line-of-sight - Low cost - Non-intrusiveness	- Not flexible because the objects need to have fixed routes. - Hard or impossible to attach the RFID reader to small objects, such as keys, glasses, wallets etc.
Yuhong and Ya, 2012 [48]	- Accuracy - No need for direct line-of-sight - Low cost - Flexibility	- The motors required for whirling the RFID readers can generate noise and can be annoying. - The computed coordinates make sense to computers but are hard for humans to interpret as physical locations.

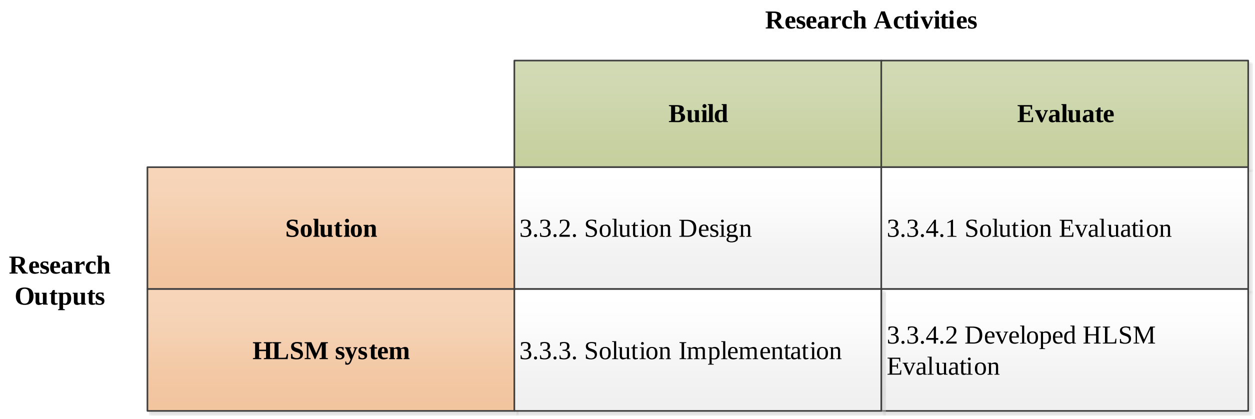
Fortin-Simard et al., 2012 [15]	- Accuracy - No need for direct line-of-sight - Low cost - Flexibility - Non-intrusiveness	- Does not test the solution with various environmental settings, such as different sizes and shapes of tracked objects, relocating furniture in the room etc. - The computed coordinates make sense to computers but are hard for humans to interpret as physical locations.
Brchan et al., 2012 [49]	- No need for direct line-of-sight - Flexibility - Non-intrusiveness	- Expensive due to the use of active tags - Effort to maintain the active RFID tags' batteries - The reference tags make sense to computers but are hard for humans to interpret as physical locations.
DiGiampaolo and Martinelli, 2012 [16]	- Accuracy - No need for direct line-of-sight - Low cost - Non-intrusiveness	- Hard or impossible to attach the RFID reader to small objects, such as keys, glasses, wallets etc. - The accuracy would decrease when the tracked object moves to higher ceiling rooms. - The computed coordinates make sense to computers but are hard for humans to interpret as physical locations.
Han et al., 2012 [50]	- Accuracy - No need for direct line-of-sight - Non-intrusiveness	- Expensive due to the use of a huge amount of active tags - Effort to maintain the active RFID tags' batteries - Not practical and perhaps hazardous to setup a 3D grid of reference tags in the room. - The reference tags make sense to computers but are hard for humans to interpret as physical locations.
Athalye et al., 2013 [51]	- Accuracy - No need for direct line-of-sight - Flexibility - Non-intrusiveness	- Effort to maintain the semi-passive RFID tags' batteries

3.3 Methodology

As stated in Chapter 2, the Design Science methodology was chosen for this research. This section provides detail about the application of the methodology when applied to the Home Localization System for Misplaced objects module.

3.3.1 Research Framework

The framework described in Figure 3-1 was used for this research. It is an adaptation from the framework of March and Smith [18] for information technology research.



		Research Activities	
		Build	**Evaluate**
Research Outputs	**Solution**	3.3.2. Solution Design	3.3.4.1 Solution Evaluation
	HLSM system	3.3.3. Solution Implementation	3.3.4.2 Developed HLSM Evaluation

Figure 3-1: Research framework for HLSM (adapted from [18])

The reason for the adaptation is that given in section 2.3.1 for the Universal XML to RDF conversion research module. Each research output goes through Build and Evaluate activities. The first output is the solution and the second output is the actual implemented HLSM system. Sections 3.3.2 and 3.3.3 explain in detail the Build activities while sections 3.3.4.1 and 3.3.4.2 describe the Evaluate activities for the research outputs.

3.3.2 Solution Design

The solution consists of System Design, RFID Tag placing strategy and Localization Rule, Search Scenarios and LO Ontology Design. The System Design draws a picture of the hardware and software components of the HLSM system. The RFID tag placing strategy describes how the passive RFID tags should be placed within the home so the HLSM system works best. The Localization Rule was decided from experimentation. Search scenarios give examples of how the home user searches for an LO. The LO Ontology Design explains the classes, object properties and individuals in the LO Ontology. The solution was described in plain English and is independent of implementation languages (e.g. Java, C#, PHP).

3.3.3 Solution Implementation

This research used the Java programming language with J2EE technologies to implement the HLSM system. The reasons for this are:

- There is a well-known open source Java framework named Jena for building Semantic Web and Linked Data applications. This framework also provides Jena ontology API for querying ontologies.
- There is an open source library named LLRP Tool Kit for Java which helps Java applications connect to and read data sent from RFID readers.
- The programming language the author is most experienced in is Java.

3.3.4 Evaluation protocol

The evaluation consists of both solution evaluation and developed HLSM system evaluation, as depicted in Figure 3-2.

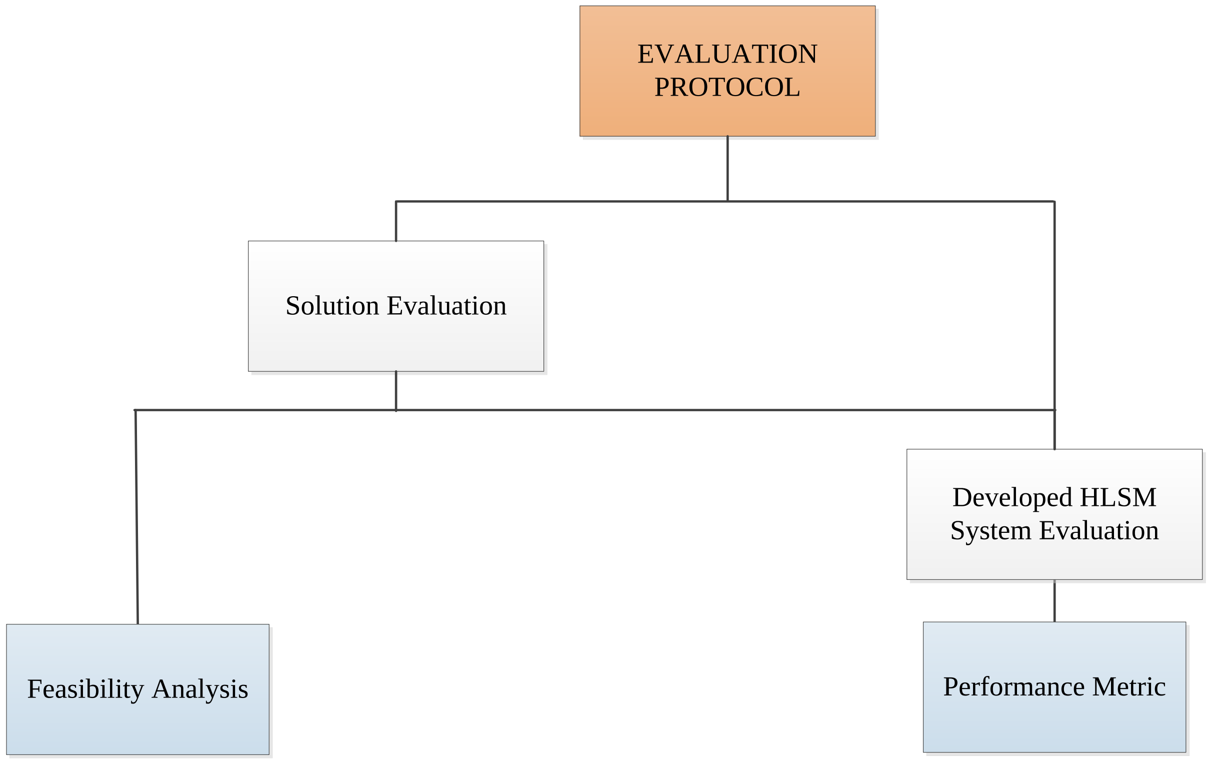


Figure 3-2: Evaluation protocol for Home Localization System for Misplaced objects

3.3.4.1 Solution Evaluation

The solution evaluation contained Feasibility Analysis and the developed HLSM system evaluation. In this study, Feasibility Analysis consisted of Technical Feasibility, Operational Feasibility, and Economic Feasibility [52, 53]. Technical Feasibility focuses on understanding the present hardware and software resources and technologies needed to implement the proposed solution. Operational Feasibility measures how well the proposed system helps solve the current problem and how well it fits to the current environment. Economic Feasibility assesses the benefits of having the system against the cost of developing it to find out whether it is worth developing the system. The developed HLSM system evaluation step was the evaluation of the actual implementation of the proposed solution. Following the spiral model of build and evaluate, results from the developed HLSM system evaluation were used as inputs for decision making in updating the solution to have a better design as well as a better implementation in the next iteration of the development process [19].

3.3.4.2 Developed HLSM Evaluation

The implemented HLSM system evaluation was undertaken by applying a defined performance metric. The performance metric was based on the six challenges of indoor localization identified in section 3.2.1. The six challenges are accuracy, no need for direct line-of-sight, low cost, flexibility, non-intrusiveness, and usability. Accuracy is quantitatively measureable. It can be measured by two accuracy measurements, Localization Type 1's Accuracy and Localization Type 2's Accuracy. Definitions of Localization Type 1 and Type 2 have been defined in section 3.1. These measurements are formulated in Table 3-3.

Table 3-3: ACCURACY MEASUREMENT OF HLSM SYSTEM

	Localization Type 1	**Localization Type 2**
No. of localization experiments	x_1	x_2
No. of correct localization results	r_1	r_2
Accuracy rate	r_1/x_1	r_2/x_2

3.4 Solution

3.4.1 System Design

Figure 3-3 shows an overview of the Home Localization System for Misplaced objects (HLSM). The HLSM system consists of a central web server, fixed RFID readers (D1-4), and mobile RFID readers (e.g. RFID reader smart phones). The fixed readers connect to the central server either by Ethernet or WiFi depending on the reader type, and the mobile readers communicate with the server via WiFi.

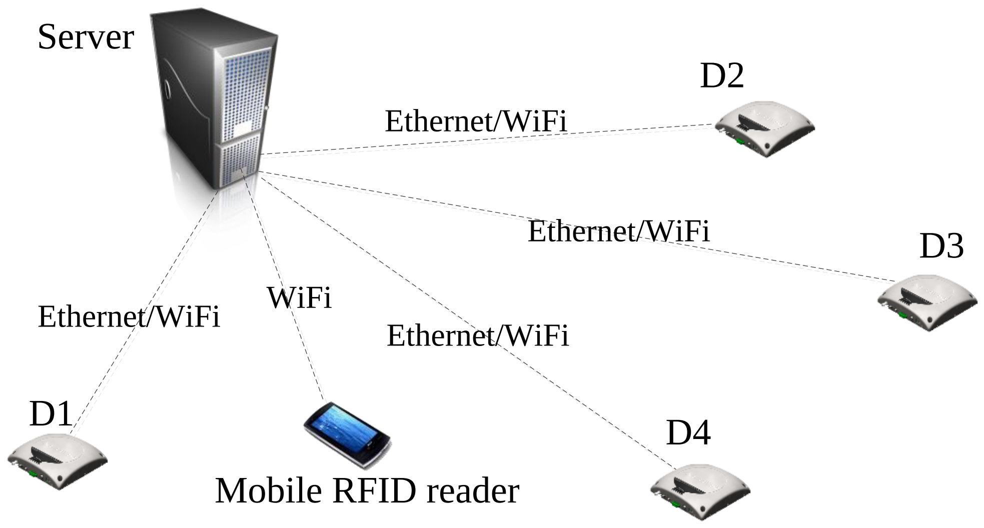


Figure 3-3: HLSM system overview

Figure 3-4 illustrates the software architecture of HLSM. The central server consists of a Web Server, an Application Server, and an Ontology Database. Mobile RFID readers (smart phones) communicate with the Web Server and the fixed RFID readers communicate with the Application Server. The Application Server and the Web Server query the Ontology Database using SPARQL (SPARQL Protocol and RDF Query Language), an RDF query language. RDF (Resource Description Framework) is a language for describing information and resources in Semantic Web environments.

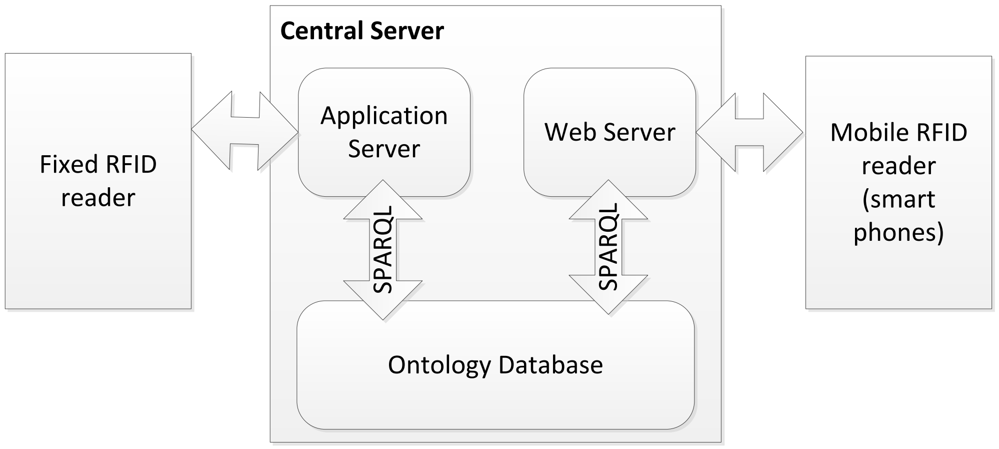


Figure 3-4: HLSM software architecture overview

3.4.2 Tag placing strategy and Localization Rule

Figure 3-5 describes the RFID reader and tag placing strategy in the proposed solution. D1-4 are fixed RFID readers used as high-level navigation landmarks (HLNLs) placed at room entry doors. T1-8 are passive RFID tags used as detailed navigation landmarks (DTNLs), which serve as reference points. Each piece of furniture where the passive RFID tagged LO is possibly located, e.g. kitchen table, is tagged with four DTNLs, one at each corner. The HLNLs help identify in which room the searched for object could possibly be, while the DTNLs help narrow it down to which part of the room the object is located. For the detailed search, the LO is localized from its three nearest identified DTNLs. The system, based on the below *Localization Rule*, then generates the localization result from these three DTNLs.

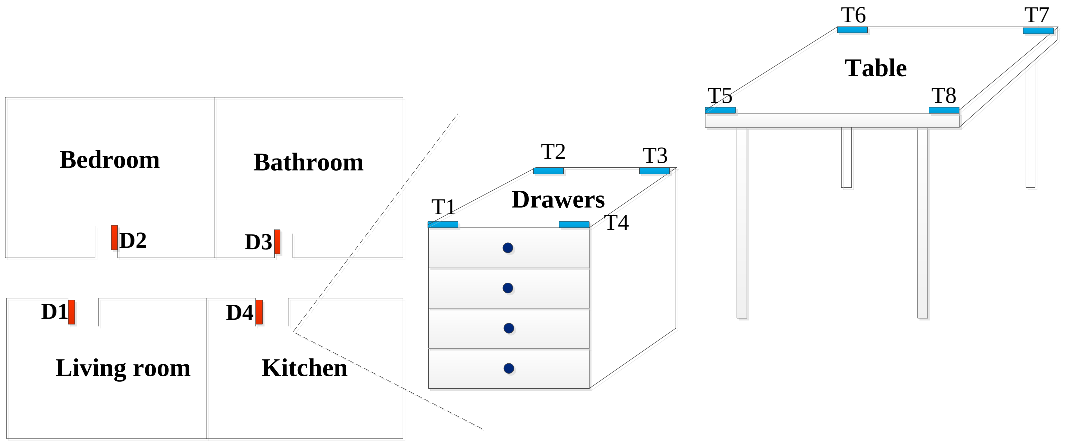


Figure 3-5: RFID tag placements

Localization Rule: If any two of the three nearest DTNLs identified by the HLSM system belong to the same piece of furniture then it is concluded that the LO is “at” (on/in) that piece of furniture. Otherwise, it is said that the LO is between the pieces of furniture that the three nearest DTNLs belong to.

The reason for choosing three as the number of nearest DTNLs (n) and two as the threshold (m) was based on the experimental results of section 3.6. Different sets of (n, m) were experimented on to finalize the optimal set (n = 3, m = 2) for the *Localization Rule*.

3.4.3 Search Scenarios

The HLSM Web Server retrieves data from and saves data to the Ontology Database, which contains the LO Ontology. The LO Ontology stores the last HLNL for the LO. When the LO is moved to another room, the fixed RFID reader at the room entry door detects and sends the RFID tag information of the LO to the Application Server to update the HLNL of the LO in the LO Ontology.

When the home user wants to search for an LO, he/she carries an RFID reader smart phone and accesses the HLSM Web application via the phone's browser. The user can interact with the Web application either via the Web GUI or an interactive voice dialog system (e.g. voiceXML). Once the Web Server receives a request to search for the LO, it accesses the Ontology Database and returns information about the room the LO is currently located based on the last HLNL associated with the LO. The user goes to that room and starts the detailed search. At this stage, an Avalanche search can be applied. An example of an Avalanche search is; if the reading range of the mobile RFID is, for example, 1 m, then the user can choose a point at the room's center as center point *C* and keep rotating around this point with an increasing radius; 1 m, 3 m, 5 m and so forth until the LO is recognized. Choosing the radii of $1(m) + 1(m) \times 2n \quad (n \in \mathbb{N})$, with n starting from 0 and keeps increasing, makes sure that all parts of the room are covered in the search. Once the LO is recognized, DTNLs are read and sent to the Web Server by the phone to identify the three nearest DTNLs of the LO. The Web Server then applies the *Localization Rule* aforementioned to generate the localization result and returns the result to the phone. The user use this generated result to find the LO. Once the LO is found, its actual location is sent back to the server to update the history dataset where data mining technique may be applied.

The above scenarios are illustrated in the following three sequence diagrams (Figure 3-6, Figure 3-7, and Figure 3-8).

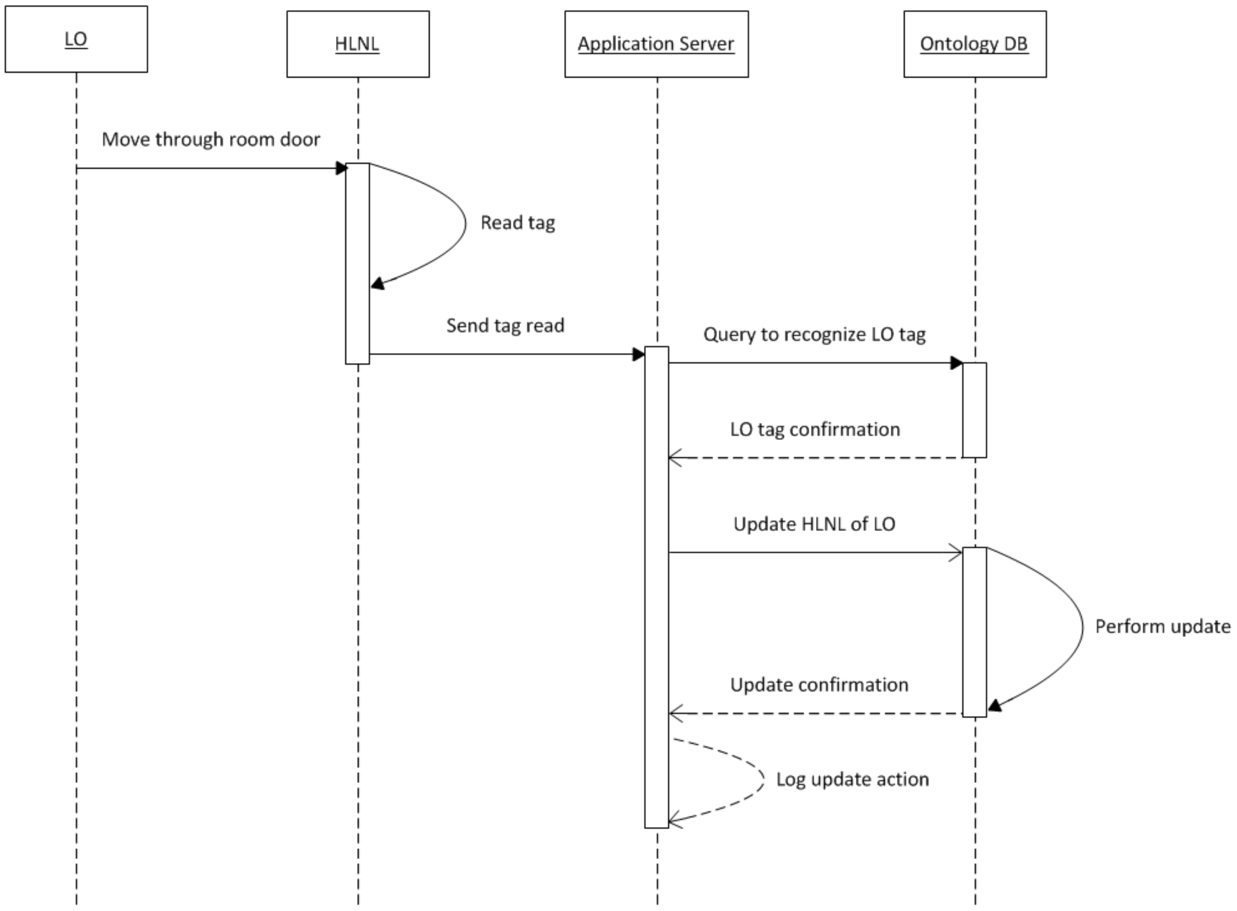


Figure 3-6: LO moves to another room

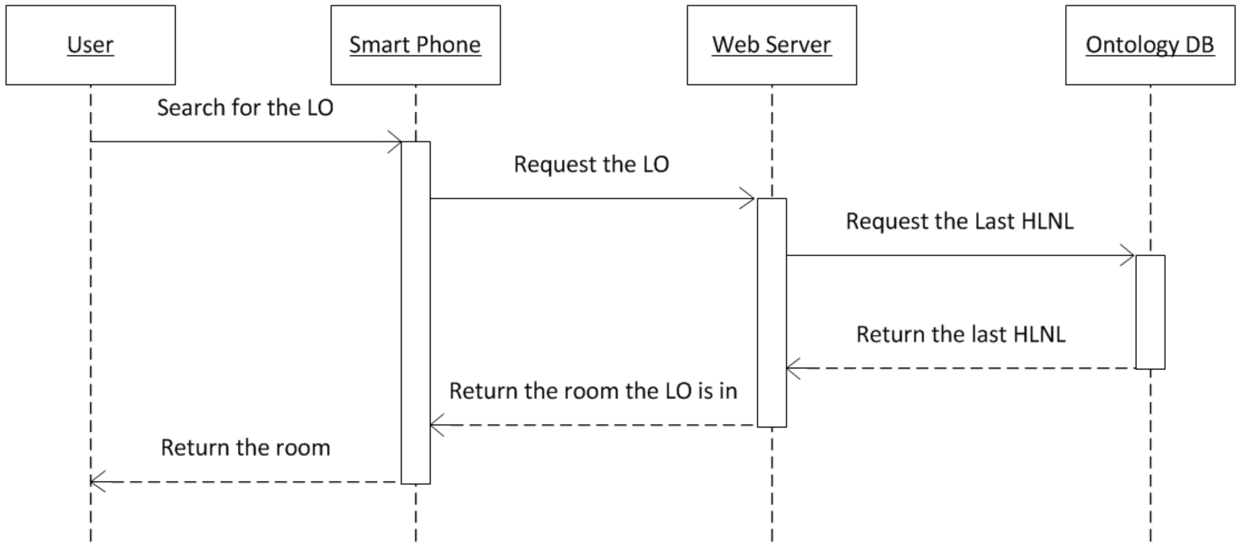


Figure 3-7: High level search

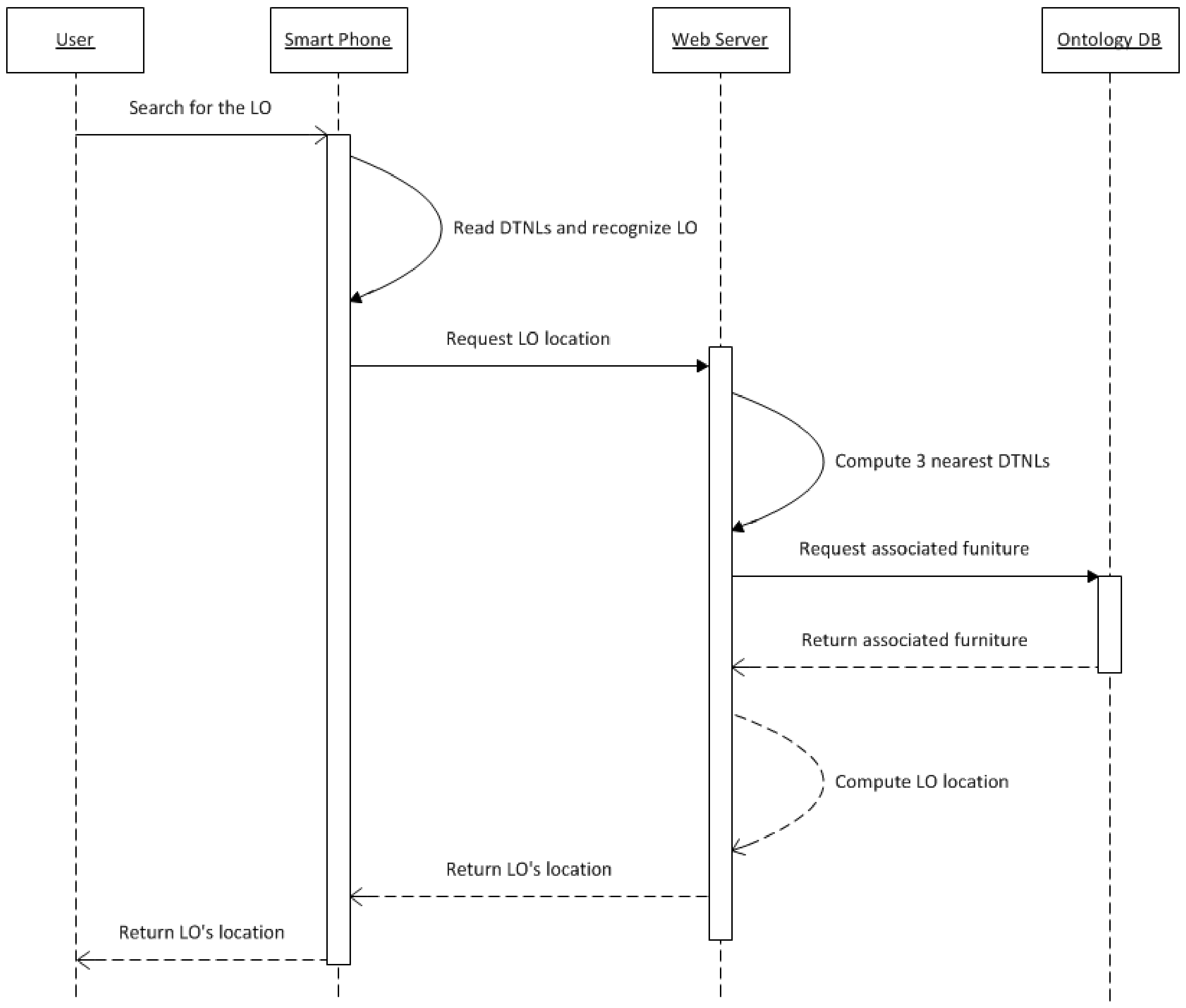


Figure 3-8: Detailed search

3.4.4 LO Ontology Design

Figure 3-9 shows the LO ontology (simplified). Five classes are “LO”, “Landmark”, “Furniture”, “Room”, and “User”. The object properties, “isIn”, “isAt”, “isBetween”, “belongsTo”, “isOwnedBy”, “isNear1”, “isNear2”, “isNear3”, describe the relationships among these classes. “isIn” connects “LO” and “Room”. “isAt” and “isBetween” respectively describe the “LO” being at or between “Furniture”. “belongsTo” indicates that “Landmark” belongs to “Furniture”. “isOwnedBy” connects “LO” and “User”. “isNear1” indicates the nearest “Landmark” of the “LO”. “isNear2” connects the “LO” to the 2nd nearest “Landmark” and so forth. The individuals of the classes are represented as diamonds in the diagram.

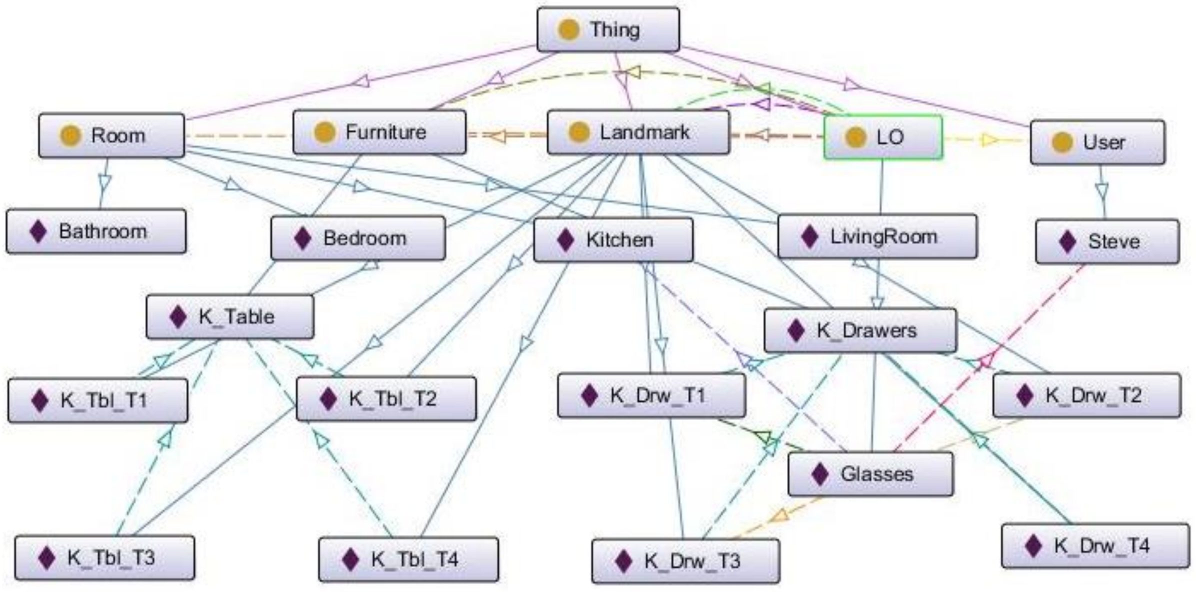


Figure 3-9: LO Ontology

As this is a simplified version, information is stylized and some details are not mentioned. For example, the HLNLs are not presented. Instead, the LO (glasses) is directly connected to the room (Kitchen). The completed LO ontology would have many more classes and individuals. Nevertheless, this simplified version of the LO ontology is enough to illustrate how the LO ontology looks as well as providing enough information to support the ontology query examples given in Appendix G. Below is this simplified version of the LO Ontology written in "RDF/XML" RDF format (Figure 3-10).

```
<?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 lo "http://www.semanticweb.org/steve/ontologies/2013/4/lo#" >
]>


<rdf:RDF xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.w3.org/2002/07/owl"
    xmlns:lo="http://www.semanticweb.org/steve/ontologies/2013/4/lo#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <Ontology rdf:about="http://www.semanticweb.org/steve/ontologies/2013/4/lo"/>

   <!--
   ///////////////////////////////////////////////////////////////////////////////////////
   //
   // Object Properties
   //
   ///////////////////////////////////////////////////////////////////////////////////////
    -->

   <!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#belongsTo -->
   <ObjectProperty rdf:about="&lo;belongsTo">
```

```
    <rdfs:range rdf:resource="&lo;Furniture"/>
    <rdfs:domain rdf:resource="&lo;LO"/>
</ObjectProperty>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#isAt -->
<ObjectProperty rdf:about="&lo;isAt">
    <rdfs:range rdf:resource="&lo;Furniture"/>
    <rdfs:domain rdf:resource="&lo;LO"/>
</ObjectProperty>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#isIn -->
<ObjectProperty rdf:about="&lo;isIn">
    <rdfs:domain rdf:resource="&lo;LO"/>
    <rdfs:range rdf:resource="&lo;Room"/>
</ObjectProperty>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#isNear1 -->
<ObjectProperty rdf:about="&lo;isNear1">
    <rdfs:domain rdf:resource="&lo;LO"/>
    <rdfs:range rdf:resource="&lo;Landmark"/>
</ObjectProperty>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#isNear2 -->
<ObjectProperty rdf:about="&lo;isNear2">
    <rdfs:domain rdf:resource="&lo;LO"/>
    <rdfs:range rdf:resource="&lo;Landmark"/>
    <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
</ObjectProperty>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#isNear3 -->
<ObjectProperty rdf:about="&lo;isNear3">
    <rdfs:domain rdf:resource="&lo;LO"/>
    <rdfs:range rdf:resource="&lo;Landmark"/>
</ObjectProperty>
```

```
<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#ownedBy -->
<ObjectProperty rdf:about="&lo;ownedBy">
   <rdfs:domain rdf:resource="&lo;LO"/>
   <rdfs:range rdf:resource="&lo;User"/>
</ObjectProperty>

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

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#Furniture -->
<Class rdf:about="&lo;Furniture"/>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#LO -->
<Class rdf:about="&lo;LO"/>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#Landmark -->
<Class rdf:about="&lo;Landmark"/>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#Room -->
<Class rdf:about="&lo;Room"/>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#User -->
<Class rdf:about="&lo;User"/>

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

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


<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#Bathroom -->
<NamedIndividual rdf:about="&lo;Bathroom">
   <rdf:type rdf:resource="&lo;Room"/>
</NamedIndividual>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#Bedroom -->
<NamedIndividual rdf:about="&lo;Bedroom">
   <rdf:type rdf:resource="&lo;Room"/>
</NamedIndividual>



<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#Glasses -->
<NamedIndividual rdf:about="&lo;Glasses">
   <rdf:type rdf:resource="&lo;LO"/>
   <lo:isNear1 rdf:resource="&lo;K_Drw_T1"/>
   <lo:isNear2 rdf:resource="&lo;K_Drw_T2"/>
   <lo:isNear3 rdf:resource="&lo;K_Drw_T3"/>
   <lo:isIn rdf:resource="&lo;Kitchen"/>
   <lo:ownedBy rdf:resource="&lo;Steve"/>
</NamedIndividual>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#K_Drawers -->
<NamedIndividual rdf:about="&lo;K_Drawers">
   <rdf:type rdf:resource="&lo;Furniture"/>
</NamedIndividual>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#K_Drw_T1 -->
<NamedIndividual rdf:about="&lo;K_Drw_T1">
   <rdf:type rdf:resource="&lo;Landmark"/>
   <lo:belongsTo rdf:resource="&lo;K_Drawers"/>
```

```
</NamedIndividual>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#K_Drw_T2 -->
<NamedIndividual rdf:about="&lo;K_Drw_T2">
   <rdf:type rdf:resource="&lo;Landmark"/>
   <lo:belongsTo rdf:resource="&lo;K_Drawers"/>
</NamedIndividual>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#K_Drw_T3 -->
<NamedIndividual rdf:about="&lo;K_Drw_T3">
   <rdf:type rdf:resource="&lo;Landmark"/>
   <lo:belongsTo rdf:resource="&lo;K_Drawers"/>
</NamedIndividual>



<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#K_Drw_T4 -->

<NamedIndividual rdf:about="&lo;K_Drw_T4">
   <rdf:type rdf:resource="&lo;Landmark"/>
   <lo:belongsTo rdf:resource="&lo;K_Drawers"/>
</NamedIndividual>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#K_Table -->
<NamedIndividual rdf:about="&lo;K_Table">
   <rdf:type rdf:resource="&lo;Furniture"/>
</NamedIndividual>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#K_Tbl_T1 -->
<NamedIndividual rdf:about="&lo;K_Tbl_T1">
   <rdf:type rdf:resource="&lo;Landmark"/>
   <lo:belongsTo rdf:resource="&lo;K_Table"/>
</NamedIndividual>

<!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#K_Tbl_T2 -->
```

```
    <NamedIndividual rdf:about="&lo;K_Tbl_T2">
        <rdf:type rdf:resource="&lo;Landmark"/>
        <lo:belongsTo rdf:resource="&lo;K_Table"/>
    </NamedIndividual>

    <!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#K_Tbl_T3 -->
    <NamedIndividual rdf:about="&lo;K_Tbl_T3">
        <rdf:type rdf:resource="&lo;Landmark"/>
        <lo:belongsTo rdf:resource="&lo;K_Table"/>
    </NamedIndividual>

    <!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#K_Tbl_T4 -->
    <NamedIndividual rdf:about="&lo;K_Tbl_T4">
        <rdf:type rdf:resource="&lo;Landmark"/>
        <lo:belongsTo rdf:resource="&lo;K_Table"/>
    </NamedIndividual>

    <!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#Kitchen -->
    <NamedIndividual rdf:about="&lo;Kitchen">
        <rdf:type rdf:resource="&lo;Room"/>
    </NamedIndividual>

    <!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#LivingRoom -->
    <NamedIndividual rdf:about="&lo;LivingRoom">
        <rdf:type rdf:resource="&lo;Room"/>
    </NamedIndividual>

    <!-- http://www.semanticweb.org/steve/ontologies/2013/4/lo#Steve -->
    <NamedIndividual rdf:about="&lo;Steve">
        <rdf:type rdf:resource="&lo;User"/>
    </NamedIndividual>
</rdf:RDF>
```

Figure 3-10: Simplified version of the LO Ontology written in "RDF/XML" RDF format

3.5 Implementation and Experiments

3.5.1 Implementation

The System Design part in section 3.4.1 for the Web Server and the Application Server is technology independent. They can be implemented using any suitable technology, such as Java, .NET or PHP. In this research, Java technology is used to develop the HLSM system. The Web Server is developed using JSP technology and the Jena ontology API to query the LO ontology. The Application Server is written in Java and also uses the Jena ontology API. Jena Fuseki is used to host the LO ontology.

To experiment and test the HLSM system, focusing on the detailed search, which is the main part of the HLSM system, the author has developed a Java application prototype that uses LLRP Tool Kit for Java (ltkjava) to read and process data received from the mobile reader. The mobile reader keeps sending passive RFID tag information it reads to the Java application prototype. The Java application prototype checks all these tags to find the LO. Once the LO is recognized, the Java application prototype identifies the three nearest DTNLs to the LO based on signal strength (RSSI). Finally, the Java application prototype, which implements the *Localization Rule*, determines whether the LO is at (on/in) a specific piece of furniture or between some pieces of furniture. The main part of the Java application prototype code is presented in Appendix D.

3.5.2 Experiments

The experimental setup is shown in Figure 3-11. 24 UHF RFID tags were placed as DTNLs on 6 pieces of furniture (f) in the experiment room and 1 UHF RFID tag was placed as the LO. The sizes of the tables were 45cm×45cm, 110cm×75cm, and 160cm×80cm. The distances between any two pieces of furniture were from 170 cm to 250 cm to guarantee that when the LO was placed on any piece of furniture, its four actual nearest DTNLs were the four tags on that piece of furniture.

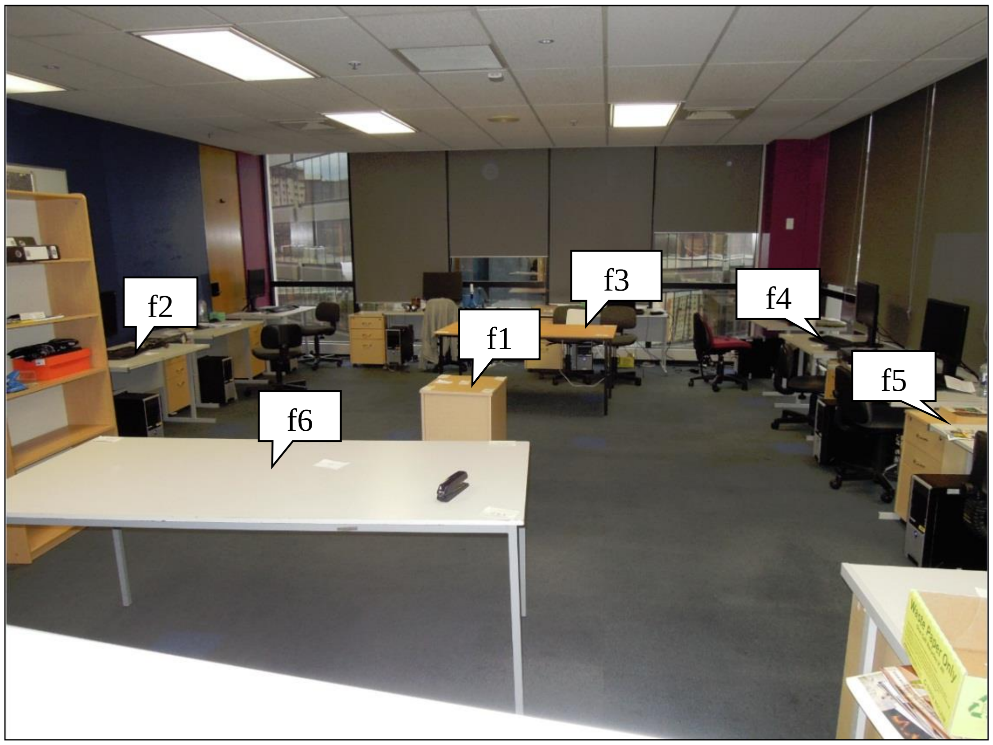


Figure 3-11: Experimental setup

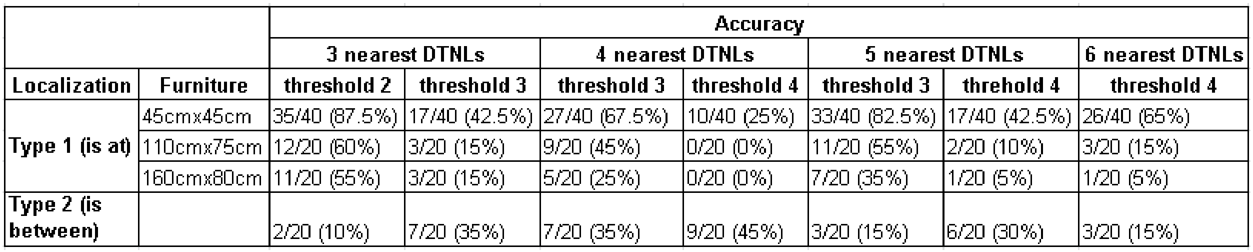
Experiments were conducted on different combinations of the number of nearest DTNLs (*n*) and the threshold (*m*) to find out the optimal set of (*n*, *m*) to form the *Localization Rule* in section 3.4.2. *m* should be greater than *n*/2 and less than or equal to 4, which is the number of DTNLs placed on a piece of furniture. In total, 7 sets of (*n*, *m*) were experimented on. They were (3, 2), (3, 3), (4, 3), (4, 4), (5, 3), (5, 4), and (6, 4). Results generated from the Java application prototype for each combination of (*n*, *m*) were recorded and compared to determine the optimal combination (Table 3-4).

In total, 100 test cases were conducted to formulate the *Localization Rule* (section 3.4.2) as well as to obtain the localization accuracy of the HLSM system. A UHF Gen2 RFID tag reader was used as a mobile reader. The RFID antenna was pointed and moved towards the LO until the Java application prototype gave out the results. For localization Type 1 (LO is at furniture), the LO was placed on three differently sized furniture types, five different positions on each piece of furniture and the antenna was approached from different directions for each position. 80 test cases in total were conducted for this localization type. In these 80 test cases, 40 test cases were for 45cm×45cm furniture, 20 test cases were for 110cm×75cm furniture, and 20 test cases were for 160cm×80cm furniture. For localization Type 2, LO is between furniture, the LO was placed in between 2, 3, and 4 pieces of furniture and 20 test cases in total were conducted.

3.6 Results and Discussion

Table 3-4 shows the experimental results generated from the raw testing result data in Appendix E. These results can be confirmed using the manual method described in Appendix F. The successful localization rate reaches 87.5% when the LO is at the furniture (Type 1). The 3 nearest neighbor technique with a threshold of 2, (n = 3, m = 2), works the best for all three furniture sizes and hence is chosen for the proposed solution. The smaller the furniture, the better the accuracy. This suggests placements of more than four reference tags on larger sized furniture would increase localization accuracy. Though accuracy for localization Type 2 is low (10%), the localization result still narrows down the location of the LO. For example, the actual location was between f3 and f4, and the program gave a result between f3, f4 and f5, which helped in the search of the LO (Table E-4 shows the results of localization Type 2 in detail). In addition, it should be pointed out that localization Type 1 is more important than localization Type 2 and therefore should be given priority. Usually, objects are put in or on furniture ("is at" or type 1) rather than on the floor ("is between" or type 2).

Table 3-4: EXPERIMENTATION RESULTS FOR HLSM SYSTEM PROTOTYPE



		Accuracy						
		3 nearest DTNLs		4 nearest DTNLs		5 nearest DTNLs		6 nearest DTNLs
Localization	**Furniture**	**threshold 2**	**threshold 3**	**threshold 3**	**threshold 4**	**threshold 3**	**threhold 4**	**threshold 4**
Type 1 (is at)	45cmx45cm	35/40 (87.5%)	17/40 (42.5%)	27/40 (67.5%)	10/40 (25%)	33/40 (82.5%)	17/40 (42.5%)	26/40 (65%)
	110cmx75cm	12/20 (60%)	3/20 (15%)	9/20 (45%)	0/20 (0%)	11/20 (55%)	2/20 (10%)	3/20 (15%)
	160cmx80cm	11/20 (55%)	3/20 (15%)	5/20 (25%)	0/20 (0%)	7/20 (35%)	1/20 (5%)	1/20 (5%)
Type 2 (is between)		2/20 (10%)	7/20 (35%)	7/20 (35%)	9/20 (45%)	3/20 (15%)	6/20 (30%)	3/20 (15%)

The graph in Figure 3-12 depicts the localization accuracy results of localization Type 1 for all three furniture sizes.

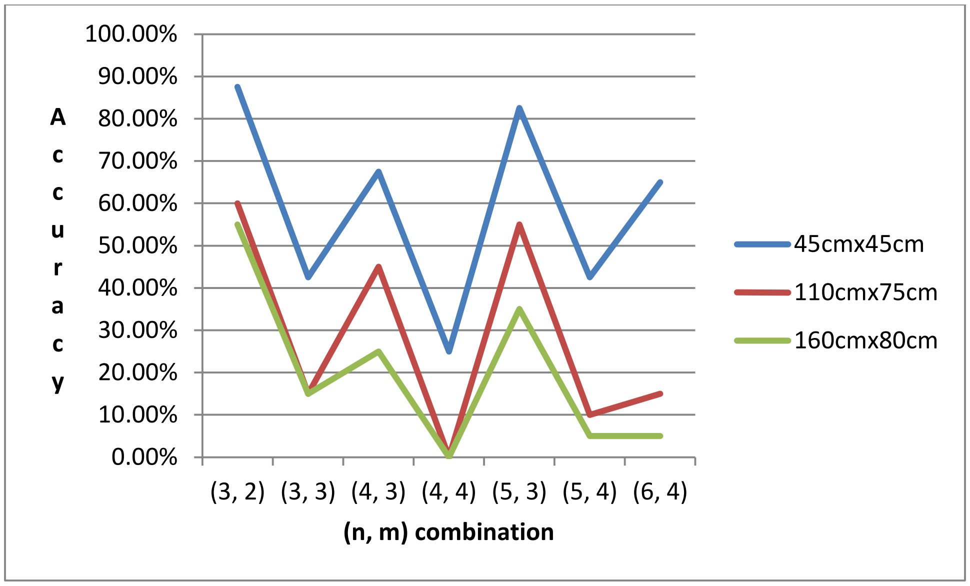


Figure 3-12: HLSM system's localization Type 1 accuracy results

In addition to less popularity of localization Type 2, the actual location of this type is sometimes itself at issue. The consequence is that the incorrect localization result might in fact be correct. Figure 3-13 illustrates this situation.

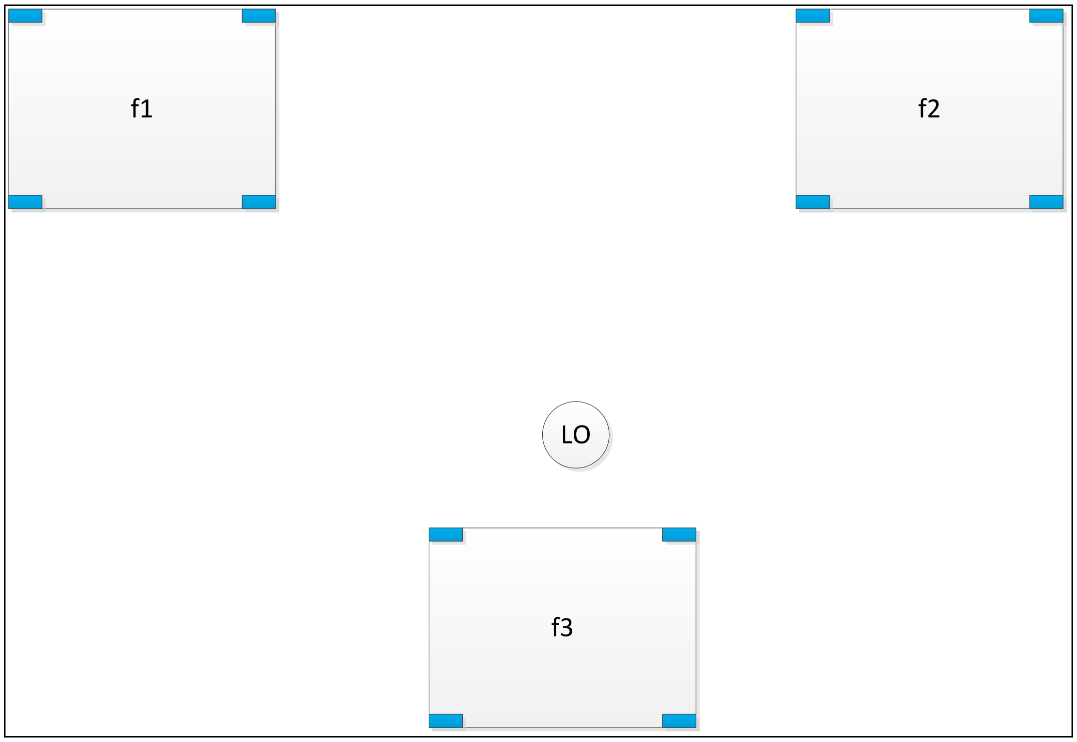


Figure 3-13: HLSM system's localization Type 2: disputed actual location

It can be seen that, in this case, the actual location is stated to be “is between f1, f2, and f3”. Therefore, the resulting location of “is at f3” is considered incorrect. However, this result is technically correct as the LO’s four nearest DTNLs all belong to f3. From this example, the accuracy value of localization Type 2 does not really reflect how well the HLSM system performs.

Table 3-5 summarizes the results of applying the performance metric defined in section 3.3.4.

Table 3-5: PERFORMANCE METRIC APPLICATION RESULTS FOR HLSM SYSTEM PROTOTYPE

Accuracy		**No need direct line-of-sight**	**Low cost**	**Flexibility**	**Non-intrusiveness**	**Usability**
Type 1	Type 2					
87.5%	10%	OK	OK	OK	OK	OK

In terms of accuracy, as explained above, the accuracy value of localization Type 1 should be used to measure the performance of the HLSM system rather than the localization Type 2 accuracy rate. In addition, though, there were three furniture sizes in the experiment and the larger sizes gave lower accuracy rates, the 87.5% accuracy rate coming from the smallest furniture size can be used as the accuracy rate of the HLSM system. The reason being that the larger sized furniture can be tagged with more than 4 RFID tags as can be seen from the experimentation results.

The other criteria constituting the performance metric also provides positive results. The fact that the HLSM system uses RFID technology means it does not need direct line-of-sight and it is non-intrusive [54]. As the system uses passive RFID tags which do not require batteries or maintenance and therefore have an indefinite operational life [54], the cost for building and operating the system should be low. Passive RFID tags are small in size and can be easily attached to different types and sizes of objects [54]. In addition, in the HLSM system, larger sized furniture can be tagged with more RFID tags than smaller sized furniture to achieve the same level of localization accuracy. These make the HLSM system flexible and being easily adapted to various home environment settings. Finally, the system gives out the location in the form, the tracked object is at a specific piece of furniture or between pieces of furniture, e.g. “The glasses are at the kitchen table”, which provides great usability for home users.

As HLSM is a web-based system, it is highly accessible to mobile RFID readers that can be smart phones or tablets. In addition, web-based system helps minimize fragmentation issue when developing software for mobile platforms. In HLSM, most of the functionalities are done on the Web Server, the mobile readers (smart phones/tablets) are mainly to read the RFID tags, pass the information to the Web Server and receive the result as HTML pages. As a result of this, the developed application is lightweight and focuses on reading and passing RFID tags’ details.

Three major limitations have been found in this study. First, at the current stage, the nearest neighbor is only calculated based on signal strength. This can be improved if RF phase information is applied [48, 55]. Second, cases where the LO is inside drawers, which would decrease the signal strength, have not been tested. Third, an actual RFID smart phone was not used in the experiment, a powerful UHF Gen2 RFID tag reader connected to a laptop was used instead. This reader has a long reading range that is unlikely to be achieved with current RFID smart phones. However, with development of these technologies, this limitation should be solved.

3.7 Summary

This chapter has introduced an RFID technique to localize misplaced objects in home environments as well as designing and recommending the implementation of an entire HLSM system. The system generates the localization results in the form of user-friendly statements, such as “the glasses are at the kitchen table”. The localization rule relies on the three nearest detailed navigation landmarks decided by RSSI. Experimental results show that this technique is promising.

Chapter 4 Conclusion

This chapter consists of three sections. First, section 4.1 provides the summary of this research emphasizing what has been achieved. Second, section 4.2 points out the limitations in the research. Finally, possible future work for this research area is presented in section 4.3.

4.1 Research Achievements

This research has proposed a transformation procedure for converting valid XML documents into RDF documents. The transformation procedure relies on the structure of XML (a combination of attributes, XML tag values and parent-child relationships) rather than relying on DTD, XML schemata or existing ontologies because a valid XML document already organizes data in a comprehensible way. The transformation procedure consists of three conversion rules and an algorithm expressed in pseudo code. In this study, the transformation procedure has been implemented in a Java tool called X2R tool that can be used to convert XML document to RDF documents on a large scale. The X2R tool was tested against eighteen input XML files covering all input cases. This universal XML to RDF conversion solution will help hasten the process of transforming current resources stored as XML into RDF documents, ready to be used on the Semantic Web.

This research has also provided a solution for indoor localization that uses passive RFID technology and can operate in Semantic Web environments. The core of the solution is the localization rule that is based on the three nearest detailed navigation landmarks, which are currently decided using RSSI. The developed system prototype was developed and the localization technique was experimented with one hundred test cases. The solution also provides a hardware list, a system software design, a tag placing strategy, search scenarios, and an LO ontology design for building the entire HLSM system.

The HLSM system servers can be deployed on the home existing computers/servers. The fixed RFID readers can be mounted at the room doors. As HLSM is a web-based system where most of the functionalities are done on the web server, the home users can use their smart phones/tablets to read RFID tags, pass the information to the web server and get the localization results from the web server as HTML pages. As a result of this, the developed application is lightweight and focuses on reading and passing RFID tags' details.

The main contributions of this research are summarized as below:

1. A transformation procedure consisting conversion rules and a pseudo code written algorithm to convert XML documents to RDF has been developed. The X2R tool, which is written in Java and implements the transformation procedure, has also been built to perform the conversion on a large scale. The solution in this research has a conversion accuracy rate of 100% that is better than the existing solutions.

2. A novel indoor localization system named HLSM (Home Localization System for Misplaced objects) has been designed and prototype developed. The system generates a meaningful description of the location of the misplaced object instead of its coordinates, such as "the key is on the kitchen table". None of the existing literature has the same approach. In addition, this smart home automation system uses ontology and adheres to the Semantic Web protocols, and therefore is forward compatible as on the move towards Semantic Web.

4.2 Limitations

A few limitations have been found in this research. First, an out of memory error occurs on the X2R tool when working with large sized input XML files due to its dependence on the Jena library to create RDF documents. Jena adds all created RDF statements to one RDF model object and so this model object gets larger and larger until at some point the JVM heap runs out of memory. Second, the HLSM system calculates the nearest neighbors based on only RSSI which may not be enough to maximize accuracy. Third, cases where the LO is inside drawers or cupboards etc. were not tested. Signal strength can be decreased when the LO is put inside a piece of furniture. Fourth, a laptop attached with a powerful UHF Gen2 RFID tag reader was used in the experiments rather than a real RFID smart phone; which may have made searching easier than with an RFID smart phone as it normally has a shorter reading range.

4.3 Future Work

Two possible future work projects have been identified from the Universal XML to RDF conversion research. The first possible work would be enhancing the X2R tool to avoid the out of memory issue and to accept multiple input XML files. The first enhancement to solve the memory issue would probably take quite a lot of effort as the Jena library may need to be examined and rewritten. The second enhancement to allow multiple input XML files would not take a lot of effort inasmuch as it could be to just add a loop in the Java code to iterate through all input XML files. The second possible future work would be to research how to convert XML to OWL. This would be more difficult and require more effort than converting XML to RDF as OWL is more expressive and more powerful than RDF [8].

Future work following the HLSM research would be to enhance the current HLSM system which so far has just high level design and the localization rule implemented. The first enhancement would be to improve the three nearest neighbors finding algorithm by applying RF (Radio Frequency) phase information to increase accuracy. The second enhancement would be to implement the whole HLSM system.

References

[1] D. Brickley and R. V. Guha, "RDF vocabulary description language 1.0: RDF schema," W3C., IBM., Feb. 10, 2004.

[2] G. Antoniou and F. Van Harmelen, *A semantic Web primer*. Cambridge, Mass: MIT Press, 2004.

[3] D. Fensel, F. M. Facca, E. Simperl, and I. Toma, "Semantic Web," *Semantic Web Services,* pp. 87-104, 2011.

[4] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, "Learning to map between ontologies on the semantic web," in Proc. *11th International Conference on World Wide Web*, 2002, pp. 662-673.

[5] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, "What are ontologies, and why do we need them?," *IEEE Intelligent Systems and Their Applications*, vol. 14, pp. 20-26, 1999.

[6] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American,* vol. 284, pp. 28-37, 2001.

[7] D. Parry, "Fuzzification of a standard ontology to encourage reuse," in Proc. *IEEE International Conference on Information Reuse and Integration*, 2004, pp. 582-587.

[8] G. Antoniou and F. Van Harmelen, "Web ontology language: OWL," in *Handbook on ontologies,* S. Staab and R. Studer, Eds., ed: Springer Berlin Heidelberg, 2004, pp. 67-92.

[9] E. Wenger, "Artificial intelligence and tutoring systems," *International Journal of Artificial Intelligence in Education,* vol. 14, pp. 39-65, 2004.

[10] T. T. T. Pham, L. Young-Koo, L. Sungyoung, and J. Byeong-Soo, "Transforming Valid XML Documents into RDF via RDF Schema," in Proc. *3rd International Conference on Next Generation Web Services Practices,* 2007, pp. 35-40.

[11] J. Ziegler, M. Graube, and L. Urbas, "RFID as universal entry point to linked data clouds," in Proc. *IEEE International Conference on RFID-Technologies and Applications*, 2012, pp. 281-286.

[12] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, *et al.*, "Building the Internet of Things using RFID: the RFID ecosystem experience," *IEEE Internet Computing,* vol. 13, pp. 48-55, 2009.

[13] B. Guo, S. Satake, and M. Imai, "Home-Explorer: Ontology-based physical artifact search and hidden object detection system," *Mobile Information Systems,* vol. 4, pp. 81-103, 2008.

[14] S. S. Saad and Z. S. Nakad, "A standalone RFID indoor positioning system using passive tags," *IEEE Transactions on Industrial Electronics,* vol. 58, pp. 1961-1970, 2011.

[15] D. Fortin-Simard, K. Bouchard, S. Gaboury, B. Bouchard, and A. Bouzouane, "Accurate passive RFID localization system for smart homes," in Proc. *IEEE 3rd International Conference on Networked Embedded Systems for Every Application*, 2012, pp. 1-8.

[16] E. DiGiampaolo and F. Martinelli, "A passive UHF-RFID system for the localization of an indoor autonomous vehicle," *IEEE Transactions on Industrial Electronics,* vol. 59, pp. 3961-3970, 2012.

[17] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly,* vol. 28, pp. 75-105, 2004.

[18] S. T. March and G. F. Smith, "Design and natural science research on information technology," *Decision support systems,* vol. 15, pp. 251-266, 1995.

[19] B. W. Boehm, "A spiral model of software development and enhancement," *Computer,* vol. 21, pp. 61-72, 1988.

[20] J. Collis and R. Hussey, *Business research: A practical guide for undergraduate & postgraduate students*. Basingstoke, UK: Palgrave Macmillan, 2009.

[21] Y. Q. Yang, M. H. Dai, and H. Chen, "An Automatic Semantic Extraction Algorithm for XML Document," in Proc. *International Conference on Machine Vision and Human-Machine Interface,* 2010, pp. 41-44.

[22] I. R. Cruz, X. Huiyong, and H. Feihong, "An ontology-based framework for XML semantic integration," in Proc. *International Conference on Database Engineering and Applications Symposium,* 2004, pp. 217-226.

[23] D. Van Deursen, C. Poppe, G. Martens, E. Mannens, and R. Walle, "XML to RDF Conversion: A Generic Approach," in Proc. *International Conference on Automated solutions for Cross Media Content and Multi-channel Distribution,* 2008, pp. 138-144.

[24] N. Yihua, W. Haibo, H. Nasha, L. Yan, and W. Zhengxiao, "A heterogeneous system integration framework for business collaboration," in Proc. *International Conference on Intelligent Computing and Intelligent Systems,* 2009, pp. 217-221.

[25] H. Bohring and S. Auer, "Mapping XML to OWL ontologies," *Leipziger Informatik-Tage,* vol. 72, pp. 147-156, 2005.

[26] S. Battle, "Gloze: XML to RDF and back again," in *Jena User Conference*, 2006.

[27] M. Klein, "Interpreting XML documents via an RDF schema ontology," in Proc. *13th International Workshop on Database and Expert Systems Applications,* 2002, pp. 889-893.

[28] M. Ferdinand, C. Zirpins, and D. Trastour, "Lifting XML Schema to OWL," *Web Engineering,* pp. 776-777, 2004.

[29] N. Bikakis, N. Gioldasis, C. Tsinaraki, and S. Christodoulakis, "Querying XML Data with SPARQL," in Proc. *20th International Conference on Database and Expert Systems Applications*, 2009, pp. 372-381.

[30] S. Bischof, S. Decker, T. Krennwallner, N. Lopes, and A. Polleres, "Mapping between RDF and XML with XSPARQL," *Journal on Data Semantics,* vol. 1, pp. 147-185, 2012.

[31] P. M. Fischer, D. Florescu, M. Kaufmann, and D. Kossmann, "Translating SPARQL and SQL to XQuery," *XML Prague*, pp. 81-98, 2011.

[32] S. Groppe, J. Groppe, V. Linnemann, D. Kukulenz, N. Hoeller, and C. Reinke, "Embedding SPARQL into XQuery/XSLT," in Proc. *ACM Symposium on Applied Computing*, 2008, pp. 2271-2278.

[33] R. Petruska and P. Curda. (2012, 06 Mar 2013). *XMLtoRDF 1.2*. Available: http://mac.softpedia.com/get/Utilities/XMLtoRDF.shtml

[34] T. S. Chow, "Testing Software Design Modeled by Finite-State Machines," *IEEE Transactions on Software Engineering*, vol. SE-4, pp. 178-187, 1978.

[35] N. Shadbolt, W. Hall, and T. Berners-Lee, "The Semantic Web Revisited," *IEEE Intelligent Systems*, vol. 21, pp. 96-101, 2006.

[36] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. Cambridge, Mass: MIT Press, 2009.

[37] P. Buneman, S. Davidson, W. Fan, C. Hara, and W.-C. Tan, "Reasoning about keys for XML," *Information Systems*, vol. 28, pp. 1037-1063, 2003.

[38] T. Bray, D. Hollander, A. Layman, R. Tobin, and H. S. Thompson, "Namespaces in XML 1.0 (Third Edition)," Textuality., Contivo, Inc., Microsoft., University of Edinburgh and Markup Technology Ltd., University of Edinburgh., W3C., Dec. 8, 2009.

[39] R. Tesoriero, J. A. Gallud, M. D. Lozano, and V. M. R. Penichet, "Tracking autonomous entities using RFID technology," *IEEE Transactions on Consumer Electronics*, vol. 55, pp. 650-655, 2009.

[40] A. Ibrahim and D. Ibrahim, "Real-time GPS based outdoor WiFi localization system with map display," *Advances in Engineering Software*, vol. 41, pp. 1080-1086, 2010.

[41] R. Mautz, "The challenges of indoor environments and specification on some alternative positioning systems," in Proc. *6th Workshop on Positioning, Navigation and Communication*, 2009, pp. 29-36.

[42] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "LANDMARC: indoor location sensing using active RFID," *Wireless Networks*, vol. 10, pp. 701-710, 2004.

[43] Y. Song, B. Han, X. Zhang, and D. Yang, "Modeling and simulation of smart home scenarios based on Internet of Things," in Proc. *3rd IEEE International Conference on Network Infrastructure and Digital Content*, 2012, pp. 596-600.

[44] H. Dae-Man and L. Jae-Hyun, "Design and implementation of smart home energy management systems based on ZigBee," *IEEE Transactions on Consumer Electronics*, vol. 56, pp. 1417-1425, 2010.

[45] N. Fallah, I. Apostolopoulos, K. Bekris, and E. Folmer, "Indoor Human Navigation Systems: A Survey," *Interacting with Computers*, vol. 25, pp. 21-33, 2013.

[46] K. Domdouzis, B. Kumar, and C. Anumba, "Radio-Frequency Identification (RFID) applications: A brief introduction," *Advanced Engineering Informatics*, vol. 21, pp. 350-355, 2007.

[47] A. Almaaitah, K. Ali, H. S. Hassanein, and M. Ibnkahla, "3D passive tag localization schemes for indoor RFID applications," in Proc. *IEEE International Conference on Communications*, 2010, pp. 1-5.

[48] L. Yuhong and Q. Ya, "An indoor localization of UHF RFID using a hybrid approach," in Proc. *2nd International Conference on Consumer Electronics, Communications and Networks*, 2012, pp. 1653-1656.

[49] J. L. Brchan, L. Zhao, J. Wu, R. E. Williams, and L. C. Pérez, "A real-time RFID localization experiment using propagation models," in Proc. *IEEE International Conference on RFID*, 2012, pp. 141-148.

[50] J. Han, Y. Zhao, Y. S. Cheng, T. L. Wong, and C. H. Wong, "Improving Accuracy for 3D RFID Localization," *International Journal of Distributed Sensor Networks*, vol. 2012, 2012.

[51] A. Athalye, V. Savic, M. Bolic, and P. M. Djuric, "Novel Semi-Passive RFID System for Indoor Localization," *IEEE Sensors Journal*, vol. 13, pp. 528 - 537, 2013.

[52] J. A. O'Brien and G. M. Marakas, *Management information systems*. New York: McGraw-Hill/Irwin, 2011.

[53] L. D. Bentley, J. L. Whitten, and G. Randolph, *Systems analysis and design for the global enterprise*. Boston: McGraw-Hill Irwin, 2007.

[54] R. Want, "An introduction to RFID technology," *IEEE Pervasive Computing*, vol. 5, pp. 25-33, 2006.

[55] P. V. Nikitin, R. Martinez, S. Ramamurthy, H. Leland, G. Spiess, and K. Rao, "Phase based spatial identification of UHF RFID tags," in Proc. *IEEE International Conference on RFID*, 2010, pp. 102-109.

[56] V. Vaneková, J. Bella, P. Gurský, and T. Horváth, "Fuzzy RDF in the Semantic Web: Deduction and induction," in Proc. *Workshop on Data Analysis*, 2005, pp. 16-29.

Appendices

Appendix A X2R code

This section provides the main part of the X2R code, the implemented Java code of *Algorithm 1* mentioned in section 2.4.2. The Java class which implements the algorithm extends org.xml.sax.helpers.DefaultHandler class of the SAX parser. The focus methods are *startElement*, *characters* and *endElement*.

```
public class RDFMaker extends DefaultHandler {

        private LinkedList<Elem> xmlElemlist = new LinkedList<Elem>();

        private Element temp;       // to represent details of an XML element

        private String nsPrefix;    // namespace prefix, e.g. "hms"
        private String nsFull;      // namespace full, e.g. "http://x2r.com/Property/"
        private String rsPrefix;    // resource prefix, e.g. "http://x2r.com/Resource/"

        private Model model = null;         // Jena RDF model to holds RDF triples for writing to an RDF file

        private boolean chopped = false;    // To solve an error of SAX parser
        private String firstPiece;          // To solve an error of SAX parser

        int nscounter = 1;          // To set namespace index when there are more than 1, e.g.  hms, hms1

        /**
         * Constructor
         */
        public RDFMaker(String nsPrefix, String nsFull, String rsPrefix) {
                this.nsPrefix = nsPrefix;
                this.nsFull = nsFull;
                this.rsPrefix = rsPrefix;
                model = ModelFactory.createDefaultModel();
                model.setNsPrefix(nsPrefix, nsFull);
        }

        @Override
        public void startDocument() {
        }

        @Override
        public void endDocument() {
        }
```

```
@Override
public void startElement(String uri, String name, String qName, Attributes atts) {

        temp = new Element();
        temp.setName(qName);
        temp.setAtts(atts);
        temp.setUri(generateUri());

        // Whether the element has attribute(s)
        boolean hasAtts = atts.getLength() > 0 ? true : false;

        // Set status "1" to the parent tag. This status will be used later
        // to indicate this is not a leaf element
        int size = xmlElemlist.size();
        if (size > 0) {
                xmlElemlist.get(0).getArr()[0] = "1";
        }

        // Store the current element's resources
        String[] arr = { "0", ""};
        xmlElemlist.addFirst(new Elem(temp.getName(), temp.getUri(), hasAtts, arr));
        createLiteralTriplesFromAtts();
}

@Override
public void endElement(String uri, String name, String qName) {

        if (xmlElemlist.size() > 1) { // Not the root element

                // Create <S><P><data> triple for leaf element
                if (xmlElemlist.get(0).getArr()[0].equals("0")
                                && xmlElemlist.get(0).getArr()[1].length() > 0) {
                        createLiteralTriple(xmlElemlist.get(0).getArr()[1]);
                }

                // Create <S><hasResource><O> triple for element which is not leaf or
                // element which has got attributes
                if (xmlElemlist.get(0).getArr()[0].equals("1") || temp.getAtts().getLength() > 0) {
                        createResourceTriple();
                }
        }

        xmlElemlist.removeFirst();
}

@Override
public void characters(char ch[], int start, int length) {
        String eTxt = new String(ch, start, length);
        eTxt = eTxt.trim();
```

```
		/*
		 * When start reaches 2048, it is reset back to 0. Thus, the text lies
		 * over 2048 is chopped into 2 pieces. This code is to solve this error
		 * of SAX parser by connecting these two pieces back to 1 piece as it
		 * should be.
		 */
		if ((start + length) >= 2048) {
			chopped = true;
			firstPiece = eTxt;
			return;
		}
		if (chopped) { // chopped == true
			eTxt = firstPiece + eTxt;
			chopped = false;
		}

		// Store text
		if (eTxt.length() > 0) {
			xmlElemlist.get(0).getArr()[1] = eTxt;
		}
	}

	/**
	 * Create <S><hasResource><O> triple
	 *
	 */
	private void createResourceTriple() {
		addTriple(xmlElemlist.get(1).getURI(), "hasResource", xmlElemlist.get(0).getURI(), true);
	}

	/**
	 * Create <S><P><data> triple
	 *
	 */
	private void createLiteralTriple(String text) {

		if (xmlElemlist.get(0).isHasAtts()) { // The element has got attributes
			addTriple(xmlElemlist.get(0).getURI(), xmlElemlist.get(0).getName(), text, false);
		} else { // The element has got no attribute
			addTriple(xmlElemlist.get(1).getURI(), xmlElemlist.get(0).getName(), text, false);
		}

	}

	/**
	 * Create triples from XML attributes
	 *
	 */
	private void createLiteralTriplesFromAtts() {
		int len = temp.getAtts().getLength();
		for (int i = 0; i < len; i++) {
			addTriple(temp.getUri(), temp.getAtts().getQName(i), temp.getAtts().getValue(i), false);
		}
	}

	/**
	 * Function to create an RDF triple and add it to the model from
	 * the subject, predicate and object identified
	 *
	 */
	private void addTriple(String subject, String predicate, String object, boolean objectIsResource) {

		Resource resource = model.createResource(subject);
		Property property = ResourceFactory.createProperty(nsFull, predicate);

		// Add the name space to the list it is not there
		if (model.getNsPrefixMap().containsValue(property.getNameSpace()) == false) {
			model.setNsPrefix(nsPrefix + "" + nscounter, property.getNameSpace());
			nscounter++;
		}

		if (objectIsResource) {
			resource.addProperty(property, model.createResource(object));
		} else {
			resource.addProperty(property, object);
		}
		tripleCounter++;
	}

	/**
	 * Generate URI for a resource, if the parent already contains n resource
	 * with the same name then increase the counter so that the source is unique
	 * when presenting on the RDF graph
	 *
	 */
	private String generateUri() {

		String uri = null;
		if (xmlElemlist.size() == 0) { // root element
			uri = rsPrefix + temp.getName();
		} else { // Not root element, create or update childList

			List<Child> childList = xmlElemlist.get(0).getChildList();
			Iterator<Child> iterator = childList.iterator();
			boolean match = false;

			while (iterator.hasNext()) {

				Child child = iterator.next();
```

```
                                        if (child.getName() == temp.getName()) { // Already in the child list -> update

                                                  child.setCounter(child.getCounter() + 1); // Increase counter
                                                  uri = xmlElemlist.get(0).getURI() + "/" + temp.getName()
                                                                      + child.getCounter();
                                                  match = true;
                                        }
                              }

                              if (!match) { // Child list is empty or the element is not already in the child list

                                        xmlElemlist.get(0).getChildList().add(new Child(temp.getName()));
                                        uri = xmlElemlist.get(0).getURI() + "/" + temp.getName();

                              }
                    }

                    return uri;
          }

          public Model getModel() {
                    return model;
          }
}
```

Appendix B *books1.rdf* file in other formats

This section provides the RDF file generated for *books1.xml* using the X2R tool, in RDF formats other than “RDF/XML” one; “RDF/XML-ABBREV”, “N-TRIPLE”, “N3”, and “TURTLE” formats. TURTLE is a subset of N3 and therefore in simple data structure, it would be syntactically identical to N3 [56]. In this *books1.xml* example, the generated *books1.n3* in N3 format and *books1.ttl* in Turtle format have identical contents. Below are the contents generated in these four RDF formats.

“RDF/XML-ABBREV” format

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:hms="http://x2r.com/Property/">
 <rdf:Description rdf:about="http://x2r.com/Resource/bookstore">
  <hms:hasResource>
   <rdf:Description rdf:about="http://x2r.com/Resource/bookstore/book2">
    <hms:price>39.95</hms:price>
    <hms:year>2003</hms:year>
    <hms:author>Erik T. Ray</hms:author>
    <hms:hasResource>
     <rdf:Description rdf:about="http://x2r.com/Resource/bookstore/book2/title">
      <hms:title>Learning XML</hms:title>
      <hms:lang>en</hms:lang>
     </rdf:Description>
    </hms:hasResource>
    <hms:category>WEB</hms:category>
   </rdf:Description>
  </hms:hasResource>
  <hms:hasResource>
   <rdf:Description rdf:about="http://x2r.com/Resource/bookstore/book1">
    <hms:price>49.99</hms:price>
    <hms:year>2003</hms:year>
```

```
        <hms:author>Kurt Cagle</hms:author>
        <hms:author>Per Bothner</hms:author>
        <hms:author>James McGovern</hms:author>
        <hms:hasResource>
          <rdf:Description rdf:about="http://x2r.com/Resource/bookstore/book1/title">
            <hms:title>XQuery Kick Start</hms:title>
            <hms:lang>en</hms:lang>
          </rdf:Description>
        </hms:hasResource>
        <hms:category>WEB</hms:category>
      </rdf:Description>
    </hms:hasResource>
    <hms:hasResource>
      <rdf:Description rdf:about="http://x2r.com/Resource/bookstore/book">
        <hms:price>30.00</hms:price>
        <hms:year>2005</hms:year>
        <hms:author>Giada De Laurentiis</hms:author>
        <hms:hasResource>
          <rdf:Description rdf:about="http://x2r.com/Resource/bookstore/book/title">
            <hms:title>Everyday Italian</hms:title>
            <hms:lang>en</hms:lang>
          </rdf:Description>
        </hms:hasResource>
        <hms:category>COOKING</hms:category>
      </rdf:Description>
    </hms:hasResource>
    <hms:city>Auckland</hms:city>
    <hms:country>NZ</hms:country>
  </rdf:Description>
</rdf:RDF>
```

“N-TRIPLE” format

```
<http://x2r.com/Resource/bookstore/book/title> <http://x2r.com/Property/title> "Everyday Italian" .
<http://x2r.com/Resource/bookstore/book/title> <http://x2r.com/Property/lang> "en" .
<http://x2r.com/Resource/bookstore/book1/title> <http://x2r.com/Property/title> "XQuery Kick Start" .
<http://x2r.com/Resource/bookstore/book1/title> <http://x2r.com/Property/lang> "en" .
<http://x2r.com/Resource/bookstore> <http://x2r.com/Property/hasResource>
<http://x2r.com/Resource/bookstore/book2> .
<http://x2r.com/Resource/bookstore> <http://x2r.com/Property/hasResource>
<http://x2r.com/Resource/bookstore/book1> .
<http://x2r.com/Resource/bookstore> <http://x2r.com/Property/hasResource>
<http://x2r.com/Resource/bookstore/book> .
<http://x2r.com/Resource/bookstore> <http://x2r.com/Property/city> "Auckland" .
<http://x2r.com/Resource/bookstore> <http://x2r.com/Property/country> "NZ" .
<http://x2r.com/Resource/bookstore/book2/title> <http://x2r.com/Property/title> "Learning XML" .
<http://x2r.com/Resource/bookstore/book2/title> <http://x2r.com/Property/lang> "en" .
<http://x2r.com/Resource/bookstore/book1> <http://x2r.com/Property/price> "49.99" .
<http://x2r.com/Resource/bookstore/book1> <http://x2r.com/Property/year> "2003" .
<http://x2r.com/Resource/bookstore/book1> <http://x2r.com/Property/author> "Kurt Cagle" .
<http://x2r.com/Resource/bookstore/book1> <http://x2r.com/Property/author> "Per Bothner" .
<http://x2r.com/Resource/bookstore/book1> <http://x2r.com/Property/author> "James McGovern" .
<http://x2r.com/Resource/bookstore/book1> <http://x2r.com/Property/hasResource>
<http://x2r.com/Resource/bookstore/book1/title> .
<http://x2r.com/Resource/bookstore/book1> <http://x2r.com/Property/category> "WEB" .
<http://x2r.com/Resource/bookstore/book2> <http://x2r.com/Property/price> "39.95" .
<http://x2r.com/Resource/bookstore/book2> <http://x2r.com/Property/year> "2003" .
<http://x2r.com/Resource/bookstore/book2> <http://x2r.com/Property/author> "Erik T. Ray" .
<http://x2r.com/Resource/bookstore/book2> <http://x2r.com/Property/hasResource>
<http://x2r.com/Resource/bookstore/book2/title> .
<http://x2r.com/Resource/bookstore/book2> <http://x2r.com/Property/category> "WEB" .
<http://x2r.com/Resource/bookstore/book> <http://x2r.com/Property/price> "30.00" .
<http://x2r.com/Resource/bookstore/book> <http://x2r.com/Property/year> "2005" .
<http://x2r.com/Resource/bookstore/book> <http://x2r.com/Property/author> "Giada De Laurentiis" .
```

```
<http://x2r.com/Resource/bookstore/book> <http://x2r.com/Property/hasResource>
<http://x2r.com/Resource/bookstore/book/title> .
<http://x2r.com/Resource/bookstore/book> <http://x2r.com/Property/category> "COOKING" .
```

“N3” format

```
@prefix hms:     <http://x2r.com/Property/> .

<http://x2r.com/Resource/bookstore>
    hms:city "Auckland" ;
    hms:country "NZ" ;
    hms:hasResource <http://x2r.com/Resource/bookstore/book2> , <http://x2r.com/Resource/bookstore/book1> ,
<http://x2r.com/Resource/bookstore/book> .

<http://x2r.com/Resource/bookstore/book2>
    hms:author "Erik T. Ray" ;
    hms:category "WEB" ;
    hms:hasResource <http://x2r.com/Resource/bookstore/book2/title> ;
    hms:price "39.95" ;
    hms:year "2003" .

<http://x2r.com/Resource/bookstore/book1>
    hms:author "Per Bothner" , "James McGovern" , "Kurt Cagle" ;
    hms:category "WEB" ;
    hms:hasResource <http://x2r.com/Resource/bookstore/book1/title> ;
    hms:price "49.99" ;
    hms:year "2003" .

<http://x2r.com/Resource/bookstore/book2/title>
    hms:lang "en" ;
    hms:title "Learning XML" .

<http://x2r.com/Resource/bookstore/book/title>
```

```
    hms:lang "en" ;
    hms:title "Everyday Italian" .

<http://x2r.com/Resource/bookstore/book>
    hms:author "Giada De Laurentiis" ;
    hms:category "COOKING" ;
    hms:hasResource <http://x2r.com/Resource/bookstore/book/title> ;
    hms:price "30.00" ;
    hms:year "2005" .

<http://x2r.com/Resource/bookstore/book1/title>
    hms:lang "en" ;
    hms:title "XQuery Kick Start" .
```

“TURTLE” format

```
@prefix hms:     <http://x2r.com/Property/> .

<http://x2r.com/Resource/bookstore>
    hms:city "Auckland" ;
    hms:country "NZ" ;
    hms:hasResource <http://x2r.com/Resource/bookstore/book2> , <http://x2r.com/Resource/bookstore/book1> ,
<http://x2r.com/Resource/bookstore/book> .

<http://x2r.com/Resource/bookstore/book2>
    hms:author "Erik T. Ray" ;
    hms:category "WEB" ;
    hms:hasResource <http://x2r.com/Resource/bookstore/book2/title> ;
    hms:price "39.95" ;
    hms:year "2003" .

<http://x2r.com/Resource/bookstore/book1>
    hms:author "Per Bothner" , "James McGovern" , "Kurt Cagle" ;
    hms:category "WEB" ;
    hms:hasResource <http://x2r.com/Resource/bookstore/book1/title> ;
    hms:price "49.99" ;
    hms:year "2003" .

<http://x2r.com/Resource/bookstore/book2/title>
    hms:lang "en" ;
    hms:title "Learning XML" .

<http://x2r.com/Resource/bookstore/book/title>
    hms:lang "en" ;
    hms:title "Everyday Italian" .

<http://x2r.com/Resource/bookstore/book>
    hms:author "Giada De Laurentiis" ;
    hms:category "COOKING" ;
    hms:hasResource <http://x2r.com/Resource/bookstore/book/title> ;
    hms:price "30.00" ;
    hms:year "2005" .

<http://x2r.com/Resource/bookstore/book1/title>
    hms:lang "en" ;
    hms:title "XQuery Kick Start" .
```

Appendix C Output RDF files

This section provides the RDF files generated in "RDF/XML" format for the input XML files other than those originated from the *books1.xml* file. These RDF files are *note.rdf, notes.rdf, addresses.rdf, shiporder.rdf, table_ns1.rdf, table_ns2.rdf, slideshow.rdf,* and *special.rdf* .

<u>note.rdf</u>

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:hms="http://x2r.com/Property/" >
  <rdf:Description rdf:about="http://x2r.com/Resource/note/from">
   <hms:address>241 Dominion Rd</hms:address>
   <hms:age>30</hms:age>
   <hms:name>Jani</hms:name>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/note">
   <hms:body>Don't forget me this weekend!</hms:body>
   <hms:heading>Reminder</hms:heading>
   <hms:hasResource rdf:resource="http://x2r.com/Resource/note/from"/>
   <hms:to>Tove</hms:to>
  </rdf:Description>
</rdf:RDF>
```

<u>notes.rdf</u>

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:hms="http://x2r.com/Property/" >
  <rdf:Description rdf:about="http://x2r.com/Resource/notes/note1">
   <hms:body>Play soccer this weekend!</hms:body>
   <hms:heading>Reminder</hms:heading>
```

```
    <hms:hasResource rdf:resource="http://x2r.com/Resource/notes/note1/from"/>
    <hms:to>Steve</hms:to>
    <hms:lang>en</hms:lang>
    <hms:date>17/01</hms:date>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/notes/note/from">
    <hms:address>241 Dominion Rd</hms:address>
    <hms:age>30</hms:age>
    <hms:name>Jani</hms:name>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/notes/note">
    <hms:body>Don't forget me this weekend!</hms:body>
    <hms:heading>Reminder</hms:heading>
    <hms:hasResource rdf:resource="http://x2r.com/Resource/notes/note/from"/>
    <hms:to>Tove</hms:to>
    <hms:lang>en</hms:lang>
    <hms:date>16/01</hms:date>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/notes">
    <hms:hasResource rdf:resource="http://x2r.com/Resource/notes/note1"/>
    <hms:hasResource rdf:resource="http://x2r.com/Resource/notes/note"/>
    <hms:year>2013</hms:year>
    <hms:company>AUT</hms:company>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/notes/note1/from">
    <hms:address>168 Herbert Rd</hms:address>
    <hms:age>30</hms:age>
    <hms:name>Elvis</hms:name>
  </rdf:Description>
</rdf:RDF>
```

addresses.rdf

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:hms="http://x2r.com/Property/"
   xmlns:hms1="http://x2r.com/Property/xmlns:"
   xmlns:hms2="http://x2r.com/Property/xsi:" >
  <rdf:Description rdf:about="http://x2r.com/Resource/addresses">
   <hms:hasResource rdf:resource="http://x2r.com/Resource/addresses/address3"/>
   <hms:hasResource rdf:resource="http://x2r.com/Resource/addresses/address2"/>
   <hms:hasResource rdf:resource="http://x2r.com/Resource/addresses/address1"/>
   <hms:hasResource rdf:resource="http://x2r.com/Resource/addresses/address"/>
   <hms2:noNamespaceSchemaLocation>addresses.xsd</hms2:noNamespaceSchemaLocation>
   <hms1:xsi>http://www.w3.org/2001/XMLSchema-instance</hms1:xsi>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/addresses/address1">
   <hms:street>Albert Street 100</hms:street>
   <hms:name>Anna Fraser</hms:name>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/addresses/address2">
   <hms:street>Queen Street 20</hms:street>
   <hms:name>Donna Lynch</hms:name>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/addresses/address">
   <hms:street>Baker street 5</hms:street>
   <hms:name>Joe Tester</hms:name>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/addresses/address3">
   <hms:street>Herbert Rd 16</hms:street>
   <hms:name>Elvis Musambi</hms:name>
  </rdf:Description>
</rdf:RDF>
```

shiporder.rdf

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:hms="http://x2r.com/Property/"
   xmlns:hms1="http://x2r.com/Property/xmlns:"
   xmlns:hms2="http://x2r.com/Property/xsi:" >
  <rdf:Description rdf:about="http://x2r.com/Resource/shiporder">
   <hms:hasResource rdf:resource="http://x2r.com/Resource/shiporder/item1"/>
   <hms:hasResource rdf:resource="http://x2r.com/Resource/shiporder/item"/>
   <hms:hasResource rdf:resource="http://x2r.com/Resource/shiporder/shipto"/>
   <hms:orderperson>John Smith</hms:orderperson>
   <hms2:noNamespaceSchemaLocation>shiporder.xsd</hms2:noNamespaceSchemaLocation>
   <hms1:xsi>http://www.w3.org/2001/XMLSchema-instance</hms1:xsi>
   <hms:orderid>889923</hms:orderid>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/shiporder/item1">
   <hms:price>9.90</hms:price>
   <hms:quantity>1</hms:quantity>
   <hms:title>Hide your heart</hms:title>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/shiporder/item">
   <hms:price>10.90</hms:price>
   <hms:quantity>1</hms:quantity>
   <hms:note>Special Edition</hms:note>
   <hms:title>Empire Burlesque</hms:title>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/shiporder/shipto">
   <hms:country>Norway</hms:country>
   <hms:city>4000 Stavanger</hms:city>
   <hms:address>Langgt 23</hms:address>
   <hms:name>Ola Nordmann</hms:name>
  </rdf:Description>
</rdf:RDF>
```

table_ns1.rdf

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:hms="http://x2r.com/Property/"
   xmlns:hms2="http://x2r.com/Property/h:"
   xmlns:hms1="http://x2r.com/Property/xmlns:"
   xmlns:hms3="http://x2r.com/Property/f:" >
  <rdf:Description rdf:about="http://x2r.com/Resource/root/h:table/h:tr">
   <hms2:td>Bananas</hms2:td>
   <hms2:td>Apples</hms2:td>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/root">
   <hms:hasResource rdf:resource="http://x2r.com/Resource/root/f:table"/>
   <hms:hasResource rdf:resource="http://x2r.com/Resource/root/h:table"/>
   <hms1:f>http://www.w3schools.com/furniture</hms1:f>
   <hms1:h>http://www.w3.org/TR/html4/</hms1:h>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/root/h:table">
   <hms:hasResource rdf:resource="http://x2r.com/Resource/root/h:table/h:tr"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/root/f:table">
   <hms3:length>120</hms3:length>
   <hms3:width>80</hms3:width>
   <hms3:name>African Coffee Table</hms3:name>
  </rdf:Description>
</rdf:RDF>
```

table_ns2.rdf

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:hms="http://x2r.com/Property/"
```

```
  xmlns:hms2="http://x2r.com/Property/h:"
  xmlns:hms1="http://x2r.com/Property/xmlns:"
  xmlns:hms3="http://x2r.com/Property/f:" >
 <rdf:Description rdf:about="http://x2r.com/Resource/root/h:table/h:tr">
  <hms2:td>Bananas</hms2:td>
  <hms2:td>Apples</hms2:td>
 </rdf:Description>
 <rdf:Description rdf:about="http://x2r.com/Resource/root">
  <hms:hasResource rdf:resource="http://x2r.com/Resource/root/f:table"/>
  <hms:hasResource rdf:resource="http://x2r.com/Resource/root/h:table"/>
 </rdf:Description>
 <rdf:Description rdf:about="http://x2r.com/Resource/root/h:table">
  <hms:hasResource rdf:resource="http://x2r.com/Resource/root/h:table/h:tr"/>
  <hms1:h>http://www.w3.org/TR/html4/</hms1:h>
 </rdf:Description>
 <rdf:Description rdf:about="http://x2r.com/Resource/root/f:table">
  <hms3:length>120</hms3:length>
  <hms3:width>80</hms3:width>
  <hms3:name>African Coffee Table</hms3:name>
  <hms1:f>http://www.w3schools.com/furniture</hms1:f>
 </rdf:Description>
</rdf:RDF>
```

slideshow.rdf

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:hms="http://x2r.com/Property/" >
 <rdf:Description rdf:about="http://x2r.com/Resource/slideshow/slide1/item4">
  <hms:lang>vn</hms:lang>
  <hms:id>2</hms:id>
 </rdf:Description>
 <rdf:Description rdf:about="http://x2r.com/Resource/slideshow">
```

```
    <hms:hasResource rdf:resource="http://x2r.com/Resource/slideshow/slide1"/>
    <hms:hasResource rdf:resource="http://x2r.com/Resource/slideshow/slide"/>
    <hms:author>Yours Truly</hms:author>
    <hms:date>Date of publication</hms:date>
    <hms:title>Sample Slide Show</hms:title>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/slideshow/slide1/item3">
    <hms:lang>jp</hms:lang>
    <hms:id>1</hms:id>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/slideshow/slide">
    <hms:title>Wake up to WonderWidgets!</hms:title>
    <hms:type>all</hms:type>
  </rdf:Description>
  <rdf:Description rdf:about="http://x2r.com/Resource/slideshow/slide1">
    <hms:item>Who buys WonderWidgets</hms:item>
    <hms:hasResource rdf:resource="http://x2r.com/Resource/slideshow/slide1/item4"/>
    <hms:hasResource rdf:resource="http://x2r.com/Resource/slideshow/slide1/item3"/>
    <hms:item>Why WonderWidgets are great</hms:item>
    <hms:title>Overview</hms:title>
    <hms:type>all</hms:type>
  </rdf:Description>
</rdf:RDF>
```

special.rdf

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:hms="http://x2r.com/Property/" >
  <rdf:Description rdf:about="http://x2r.com/Resource/root/tag1">
    <hms:tag1_2>text1_2</hms:tag1_2>
    <hms:tag1_1>text1_1</hms:tag1_1>
    <hms:att1>value1</hms:att1>
```

```
 </rdf:Description>
 <rdf:Description rdf:about="http://x2r.com/Resource/root">
   <hms:hasResource rdf:resource="http://x2r.com/Resource/root/tag3"/>
   <hms:tag2>text2</hms:tag2>
   <hms:hasResource rdf:resource="http://x2r.com/Resource/root/tag1"/>
 </rdf:Description>
 <rdf:Description rdf:about="http://x2r.com/Resource/root/tag3">
   <hms:att3_2>value3_2</hms:att3_2>
   <hms:att3_1>value3_1</hms:att3_1>
 </rdf:Description>
</rdf:RDF>
```

Appendix D HLSM code

This section provides the main part of the HLSM system prototype code, the implemented Java code of the *Localization Rule* mentioned in section 3.4.2, which contains *messageReceived* function and *Algorithm* class. The *messageReceived* function gets called each time the RFID reader sends the details of the read tags to the Java application. In this function, all the reported tags are looped through to recognize the LO. Once the LO is recognized, the RSSIs of the reported tags are gathered. The program then starts the *Algorithm* thread to apply the *Localization Rule* and to generate the localization result.

messageReceived function code:

```
  /*
  * This function gets called asynchronously
  *  when a tag report is available.
 */
  public void messageReceived(LLRPMessage message)
  {
     if (message.getTypeNum() == RO_ACCESS_REPORT.TYPENUM)
     {
        // The message received is an Access Report.
        RO_ACCESS_REPORT report = (RO_ACCESS_REPORT) message;
        // Get a list of the tags read.
        List <TagReportData> tags = report.getTagReportDataList();

        // Loop through the list and get the EPC of each tag.
        for (TagReportData tag : tags)
        {
           String id = Util.getTagID(tag.getEPCParameter().toString());

           // Check the received RFID tag to see wether it is the tracking lost object.we are looking for
           if (id.equals(LOConstants.LO_ID)) {
                   System.out.println("Found LO: " + id);
                   lo_rssi = tag.getPeakRSSI().getPeakRSSI().toInteger();
                   LOFound = true;
           }

           // When LO found,  the user is expected to stop mobilizing and  the program
           // gather the RSSI of the neighbor tags for further processing
           if (LOFound) {
                   counter++;
                   if (!id.equals(LOConstants.LO_ID) && !id_hash.containsKey(id)) {
                            Integer rssi = tag.getPeakRSSI().getPeakRSSI().toInteger();
                            id_hash.put(id, rssi);
```

```
                        }
                }

                // When the program gathers a certain amount of tags (maximum is 20 as there can be duplication),
                // the program start the Algorithm thread which implements the localization rule and generate the
                // localization result
                if (counter >= 20 && !processing) {
                        processing = true;
                        Runnable algorithm = new Algorithm(id_hash, lo_rssi);
                        Thread thread = new Thread(algorithm);
                        thread.start();
                }
            }
        }
    }
```

Algorithm class code:

```
public class Algorithm implements Runnable {

        Hashtable<String, Integer> id_hash;
        int lo_rssi;
        List threeNearest = new ArrayList<String>();

        /*
         * Constructor:
         * + Set the RSSI of the LO
         * + Set the hash table contain (tag ID, RSSI) of the recognized reference tags
         */
        public Algorithm(Hashtable<String, Integer> id_hash, int lo_rssi) {
                this.id_hash = id_hash;
                this.lo_rssi = lo_rssi;
        }


        /*
         * + Get the absolute value of the RSSI difference between the LO and each reference tag
         * + Create an array list contains (tag ID, RSSI difference) of reference tags
         * + Sort this array list in ascending order of RSSI difference
         * + Get the first 3 elements to have the 3 nearest reference tags (DTNLs)
         * + Call function to decide the location of the LO
         */
        @Override
        public void run() {

                // Get the absolute value of the RSSI difference between the LO and each reference tag
                Iterator<Map.Entry<String, Integer>> it = id_hash.entrySet().iterator();
                while (it.hasNext()) {
                        Map.Entry<String, Integer> entry = it.next();
                        entry.setValue(Math.abs(entry.getValue() - lo_rssi));
                }

                // Create an array list contains (tag ID, RSSI difference) of reference tags
                ArrayList dtnlList = new ArrayList(id_hash.entrySet());

                // Sort this array list in ascending order
                Collections.sort(dtnlList, new DTNL_Comparator());

                // Get the first 3 elements to have the 3 nearest reference tags
                System.out.println("LO: " + LOConstants.LO_ID + ": " + lo_rssi);
                Iterator itr = dtnlList.iterator();
                int count = 1;
                while (itr.hasNext()) {

                        Map.Entry e = (Map.Entry)itr.next();
                        System.out.println(count + ":           " + e.getKey().toString() + ": " + e.getValue());

                        if (Util.isContained(e.getKey().toString(), LOConstants.allReferenceTags)) {
                                if (count <= 3) {
                                        threeNearest.add(e.getKey());
                                }
                                count++;
                        }
                }

                // Call function to decide the location of the LO
                decideLocation();
        }



        /*
         * Decide the location of the LO:
         * + Create a hash table contains (furniture, number of reference tags belong to)
         * + Create and sort the furniture list in descending order number of reference tags
         * + Decide the location based on the localization rule
         */
        private void decideLocation() {
                System.out.println("////////////////////////////////////////");
                System.out.println("SIZE = " + threeNearest.size());

                // Create a hash table contains (furniture, number of reference tags it has)
                Hashtable<String, Integer> f_hash = new Hashtable<String, Integer>();
                Iterator it = threeNearest.iterator();
                while (it.hasNext()) {
                        String tagID = (String) it.next();
                        System.out.println(tagID);

                        String f = getFurniture(tagID);
```

```
                if (!f_hash.containsKey(f)) {
                        f_hash.put(f, 1);
                } else {
                        int value = f_hash.get(f);
                        f_hash.put(f, value + 1);
                }
        }
        System.out.println("------------------------------");

        // Create and sort the furniture list in descending order of number of reference tags
        ArrayList fList = new ArrayList(f_hash.entrySet());
        Collections.sort(fList, new F_Comparator());

        // Print the furniture list out
        String betweenStr = "";
        Iterator itr = fList.iterator();
        int count = 1;
        while (itr.hasNext()) {
                Map.Entry e = (Map.Entry)itr.next();
                System.out.println(count + ":          " + e.getKey().toString() + ": " + e.getValue());
                betweenStr = betweenStr + ", " + e.getKey().toString();
                count++;
        }
        System.out.println("==================================");

        // Decide the location based on the localization rule:
        // + If the first furniture in the list has >= 2 tags then "isAt"
        // + Otherwise: "isBetween"
        Map.Entry first = (Entry) fList.get(0);
        int value = (int) first.getValue();
        if (value >= 2) {
                System.out.println("LOCALIZATION RESULT: isAt " + first.getKey().toString());
        } else {
                System.out.println("LOCALIZATION RESULT: isBetween" + betweenStr);
        }

        System.out.println("//////////////////////////////////////////");
}


/*
 * Get the furniture the reference tag belongs to
 */
private String getFurniture(String tagID) {
        String f = "";
        if (Util.isContained(tagID, LOConstants.f1)) { // tag belongs to f1
                f = "f1";
        } else if (Util.isContained(tagID, LOConstants.f2)) {
                f = "f2";
        } else if (Util.isContained(tagID, LOConstants.f3)) {
```

```
                        f = "f3";
                } else if (Util.isContained(tagID, LOConstants.f4)) {
                        f = "f4";
                } else if (Util.isContained(tagID, LOConstants.f5)) {
                        f = "f5";
                } else if (Util.isContained(tagID, LOConstants.f6)) {
                        f = "f6";
                }

                return f;
        }
}
```

Appendix E Raw experimental result data of HLSM system

Table E-1 displays the localization results for localization Type 1 with 45cm×45cm sized furniture. “Resulting Location” is the location generated by the Java application prototype (HLSM system prototype). The results were generated for each of the seven sets of (*n*, *m*) combinations. In a “Resulting Location” cell, if the value is only one piece of furniture, e.g. “f1”, it is an “is at” result. If the value contains more than one pieces of furniture, e.g. “f1, f2” then it is an “is between” result. In addition, “Resulting Location” cells have one of three colors: white, yellow, or red. White means localization is correct. Yellow means the localization is incorrect but still includes the actual location in the resulting location. For example, with (*n* = 3, *m* = 3) in test case #1, although the resulting location is incorrect, it still has the actual location f1included in the resulting location of “is between f1 and f6”. Red means the resulting location is “totally incorrect” and does not include the actual location at all. For example, with (*n* = 5, *m* = 3) in test case #3, the actual location is f1 but the resulting location was f6. These interpretation rules are also applied to Table E-2, Table E-3, and Table E-4.

Table E-1: RAW EXPERIMENTATION RESULTS FOR LOCALIZATION TYPE 1 WITH 45CM×45CM SIZED FURNITURE

#	Actual Location	Resulting Location (*n* = 3, *m* = 2)	Resulting Location (*n* = 3, *m* = 3)	Resulting Location (*n* = 4, *m* =3)	Resulting Location (*n* = 4, *m* = 4)	Resulting Location (*n* = 5, *m* = 3)	Resulting Location (*n* =5, *m* = 4)	Resulting Location (*n* =6, *m* =4)
1	f1	f1	f1, f6	f1, f6	f1, f6	f1, f5, f6	f1, f5, f6	f1, f5, f6
2	f1	f1	f1	f1	f1	f1	f1	f1
3	f1	f6	f1, f6	f6, f1, f5	f6, f1, f5	f6	f6, f1, f5	f6, f1, f5
4	f1	f1	f1, f6	f1	f1, f6	f1	f1, f6, f3	f1, f6, f3
5	f1	f6	f6, f5	f6	f6, f5	f6	f6, f1, f5	f6, f1, f5
6	f1	f1	f1, f6	f1	f1, f6	f1	f1, f6, f3	f1
7	f1	f1	f1, f3	f1, f3, f6	f1, f3, f6	f1	f1, f3, f6	f1
8	f1	f1	f1, f6	f1	f1, f6	f1	f1, f2, f6	f1
9	f1	f1, f2, f6	f1, f2, f6	f1, f2, f6	f1, f2, f6	f1, f2, f6, f5	f1, f2, f6, f5	f1, f2, f6, f5
10	f1	f1	f1, f5	f1	f1, f5	f1	f1, f5, f3	f1
11	f1	f1	f1	f1	f1, f3	f1	f1, f3, f5	f1, f3, f5, f6
12	f1	f1	f1, f6	f1, f6, f5	f1, f6, f5	f1	f1, f6, f5	f1, f6, f5, f2
13	f1	f1	f1, f6	f1, f6, f3	f1, f6, f3	f1, f6, f3	f1, f6, f3	f1, f6, f3
14	f1	f1	f1	f1	f1	f1	f1	f1
15	f1	f1	f1	f1	f1	f1	f1	f1
16	f1	f1	f1	f1	f1, f6	f1	f1, f3, f6	f1, f3, f6
17	f1	f1	f1	f1	f1	f1	f1	f1
18	f1	f1	f1, f6	f1	f1, f6	f1	f1	f1
19	f1	f1	f1	f1	f1	f1	f1	f1
20	f1	f1	f1, f6	f1	f1, f6	f1	f1, f6	f1, f6

21	f1	f1	f1, f3	f1	f1, f3	f1	f1	f1
22	f1	f1	f1, f3	f1, f3	f1, f3	f1	f1, f3	f1
23	f1	f1	f1	f1	f1, f3	f1	f1, f2, f3	f1
24	f1	f1	f1	f1	f1	f1	f1	f1
25	f1	f1	f1	f1	f1, f3	f1	f1	f1
26	f1	f1	f1, f3	f1	f1, f3	f1	f1	f1
27	f1	f1, f2, f3	f1, f2, f3	f1, f2, f3	f1, f2, f3	f1	f1, f2, f3	f1, f2, f3
28	f1	f1	f1	f1	f1	f1	f1	f1
29	f1	f1	f1, f3	f1	f1, f3	f1	f1	f1
30	f1	f1	f1, f2	f1, f2, f3	f1, f2, f3	f1	f1, f2, f3	f1
31	f1	f1	f1	f1	f1	f1	f1	f1
32	f1	f1	f1	f1	f1	f1	f1	f1
33	f1	f1, f5, f3	f1, f5, f3	f1, f6, f5, f3	f1, f6, f5, f3	f1, f6, f5, f3	f1, f6, f5, f3	f1, f6, f5, f3, f4
34	f1	f1	f1	f1	f1, f5	f1	f1	f1
35	f1	f1	f1, f2	f1, f5, f2	f1, f5, f2	f1, f5, f3, f2	f1, f5, f3, f2	f1, f5, f3, f2
36	f1	f1	f1	f1	f1, f3	f1	f1, f3	f1
37	f1	f1	f1	f1	f1, f3	f1	f1	f1
38	f1	f1	f1	f1	f1	f1	f1	f1
39	f1	f1	f1, f2	f1, f3, f2	f1, f3, f2	f1	f1, f2, f3	f1
40	f1	f1	f1, f2	f1	f1, f2	f1	f1, f2, f3	f1, f2, f3
	Total	**35/40 (87.5%)**	**17/40 (42.5%)**	**27/40 (67.5%)**	**10/40 (25%)**	**33/40 (82.5%)**	**17/40 (42.5%)**	**26/40 (65%)**

Table E-2 displays the localization results for localization Type 1 with 110cm×75cm sized furniture. It should be mentioned that in test cases #11 to #20, although f3 was 160cm×80cm, the four reference tags at the four corners were moved closer to one another to create a size of 110cm×75cm.

Table E-2: RAW EXPERIMENTATION RESULTS FOR LOCALIZATION TYPE 1 WITH 110CM×75CM SIZED FURNITURE

#	**Actual Location**	**Resulting Location (*n* = 3, *m* = 2)**	**Resulting Location (*n* = 3, *m* = 3)**	**Resulting Location (*n* = 4, *m* = 3)**	**Resulting Location (*n* = 4, *m* = 4)**	**Resulting Location (*n* = 5, *m* = 3)**	**Resulting Location (*n* = 5, *m* = 4)**	**Resulting Location (*n* = 6, *m* = 4)**
1	f2	f2	f2, f3	f2, f3	f2, f3	f1, f2, f3	f1, f2, f3	f1, f2, f3, f5
2	f2	f3	f2, f3	f2, f3	f2, f3	f2, f3, f4	f2, f3, f4	f2, f3, f4
3	f2	f2	f2	f2	f1, f3	f2	f1, f2, f4	f1, f2, f3, f4
4	f2	f2	f2	f2	f1, f2	f2	f1, f2	f1, f2
5	f2	f2	f2, f3	f2	f2, f3	f2	f1, f2, f3	f2
6	f2	f2	f2, f3	f2, f3	f2, f3	f3	f2, f3	f1, f2, f3
7	f2	f2	f1, f2	f2	f1, f2	f2	f1, f2, f5	f1, f2, f3, f5
8	f2	f1, f2, f5	f1, f2, f5	f1, f2, f4, f5	f1, f2, f4, f5	f1, f2, f3, f4, f5	f1, f2, f3, f4, f5	f1, f2, f3, f4, f5
9	f2	f1, f2, f3	f1, f2, f3	f1, f2, f3	f1, f2, f3	f1, f2, f3	f1, f2, f3	f1, f2, f3
10	f2	f2	f2, f3	f2	f2, f3	f2	f2	f2
11	f3	f3	f3, f1	f3	f3, f1	f3	f3, f1, f6	f3, f1, f6, f2
12	f3	f1, f3, f2	f1, f3, f2	f1, f3, f2	f1, f3, f2	f1, f3, f2, f6	f1, f3, f2, f6	f1, f3, f2, f6, f5
13	f3	f3	f3	f3	f3, f1	f3	f3	f3
14	f3	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5
15	f3	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5
16	f3	f3	f3, f1	f1, f3, f5	f1, f3, f5	f3	f1, f3, f5	f1, f3, f5, f4
17	f3	f3, f1	f3, f1	f1, f3, f5	f1, f3, f5	f3, f1, f5, f6	f3, f1, f5, f6	f3, f1, f5, f6
18	f3	f3	f3, f1	f3	f3, f1	f3	f1, f3	f1, f3
19	f3	f3	f3, f1	f3	f3, f1	f3	f1, f3	f1, f3, f5
20	f3	f3, f1	f3, f1	f1, f3	f1, f3	f3	f1, f3	f1, f3
	Total	**12/20 (60%)**	**3/20 (15%)**	**9/20 (45%)**	**0/20 (0%)**	**11/20 (55%)**	**2/20 (10%)**	**3/20 (15%)**

Table E-3 shows the localization results for localization Type 1 with 160cm×80cm sized furniture.

Table E-3: RAW EXPERIMENTATION RESULTS FOR LOCALIZATION TYPE 1 WITH 160CM×80CM SIZED FURNITURE

#	Actual Location	Resulting Location (*n* = 3, *m* = 2)	Resulting Location (*n* = 3, *m* = 3)	Resulting Location (*n* = 4, *m* =3)	Resulting Location (*n* = 4, *m* = 4)	Resulting Location (*n* = 5, *m* = 3)	Resulting Location (*n* =5, *m* = 4)	Resulting Location (*n* =6, *m* =4)
1	f3	f2	f2, f3	f2, f3	f2, f3	f1, f2, f3	f1, f2, f3	f1, f2, f3
2	f3	f1, f3, f2	f1, f3, f2	f1, f3, f2	f1, f3, f2	f1, f3, f2	f1, f3, f2	f1, f3, f2
3	f3	f1, f3	f1, f3	f1, f3	f1, f3	f1, f3, f2	f1, f3, f2	f1, f2, f3, f4
4	f3	f3	f3	f3	f1, f3	f3	f3	f3
5	f3	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5
6	f3	f3	f1, f3	f3	f1, f3	f3	f1, f2, f3	f1, f2, f3
7	f3	f1, f3, f2	f1, f3, f2	f1, f3, f2	f1, f3, f2	f1	f1, f3, f2	f1, f3, f2
8	f3	f3	f1, f3	f1, f3	f1, f3	f1, f2, f3	f1, f2, f3	f1, f2, f3, f5
9	f3	f4	f1, f4	f1, f4	f1, f4	f1, f3, f4	f1, f3, f4	f1, f2, f3, f4
10	f3	f3, f4	f3, f4	f4	f3, f4	f4	f1, f3, f4	f1, f3, f4, f5
11	f3	f3	f3	f3	f3, f1	f3	f3, f1	f3, f1, f4
12	f3	f1, f5, f3	f1, f5, f3	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5	f1, f3, f5
13	f3	f3	f3, f1	f3, f1, f4	f3, f1, f4	f3	f1, f3, f4	f1, f3, f4
14	f3	f3	f3, f1	f1, f3	f1, f3	f1, f3, f2	f1, f3, f2	f1, f3, f2
15	f3	f1, f2	f1, f2	f1, f2	f1, f2	f1, f2	f1, f2	f1, f3, f2
16	f3	f3	f3, f5	f3, f5, f2	f3, f5, f2	f3, f2, f5	f3, f2, f5	f3, f2, f5
17	f3	f3	f3, f1	f1, f3, f2	f1, f3, f2	f1, f3, f2	f1, f3, f2	f1, f3, f2, f5
18	f3	f3	f3	f3	f3, f2	f3	f3, f2, f5	f3, f2, f5
19	f3	f3	f3, f6	f3	f3, f6	f3	f3, f2, f6	f3, f2, f6
20	f3	f3	f1, f2	f1, f3	f1, f3	f3	f3, f1	f3, f1, f2
Total		11/20 (55%)	3/20 (15%)	5/20 (25%)	0/20 (0%)	7/20 (35%)	1/20 (5%)	1/20 (5%)

Table E-4 shows the localization results for localization Type 2. It can be seen that all the incorrect results (yellow colored cells) contain the actual locations in part. No result is “totally incorrect”. “Totally incorrect” means that the resulting location does not overlap with the actual location at all. For example, with (*n* = 3, *m* = 2) in test case #1, the resulting location of “is between (f1, f3, f4)” overlaps the actual location of “is between (f3, f4)”. The overlapped part here is (f3, f4) and therefore the localization result in this case is incorrect but not totally incorrect.

Table E-4: RAW EXPERIMENTATION RESULTS FOR LOCALIZATION TYPE 2

#	Actual Location	Resulting Location (*n* = 3, *m* = 2)	Resulting Location (*n* = 3, *m* = 3)	Resulting Location (*n* = 4, *m* =3)	Resulting Location (*n* = 4, *m* = 4)	Resulting Location (*n* = 5, *m* = 3)	Resulting Location (*n* =5, *m* = 4)	Resulting Location (*n* =6, *m* =4)
1	f3, f4	f1, f3, f4	f1, f3, f4	f1, f3, f4, f5	f1, f3, f4, f5	f1, f3, f4, f5	f1, f3, f4, f5	f1, f3, f4, f5
2	f3, f4	f1, f4	f1, f4	f1, f3, f4	f1, f3, f4	f1, f3, f4	f1, f3, f4	f1, f3, f4
3	f3, f1	f1	f1, f3	f1	f1, f3	f1	f1, f3, f5	f1, f3, f4, f5
4	f3, f1	f1	f1	f1	f1	f1	f1	f1
5	f3, f2	f3	f2, f3	f2, f3	f2, f3	f2	f2, f3	f2
6	f3, f2	f3	f2, f3	f2, f3	f2, f3	f2	f2, f3	f2
7	f3, f1, f4	f1	f1, f4	f1, f3, f4	f1, f3, f4	f1	f2, f3, f4	f1
8	f3, f1, f4	f4	f1, f4	f4, f1, f5	f4, f1, f5	f4	f4, f1, f5	f4, f1, f5, f3
9	f1, f5, f6	f1	f1	f1	f1	f1	f1	f1
10	f1, f5, f6	f2, f3, f6	f2, f3, f6	f2, f3, f6	f2, f3, f6	f2, f3, f5, f6	f2, f3, f5, f6	f2, f3, f5, f6
11	f3, f4	f3, f4	f3, f4	f3, f6, f4	f3, f6, f4	f4, f3, f6	f4, f3, f6	f4, f3, f6
12	f3, f4	f3	f3, f4	f3, f4	f3, f4	f4, f3, f6	f4, f3, f6	f4, f3, f6
13	f3, f1	f3	f3, f1	f3, f1, f2	f3, f1, f2	f3	f3, f1, f2	f3, f1, f2, f4
14	f3, f1	f3	f3	f3	f3, f1	f3	f3, f1, f2	f3, f1, f2
15	f3, f2	f3, f2	f3, f2	f3, f2	f3, f2	f3, f2, f5	f3, f2, f5	f3, f2, f5
16	f3, f2	f3, f2, f5	f3, f2, f5	f3, f5, f2	f3, f5, f2	f3, f2, f5	f3, f2, f5	f3, f2, f5
17	f3, f1, f4	f3	f3, f1	f3, f1, f4	f3, f1, f4	f4, f3, f1	f4, f3, f1	f4, f3, f1
18	f3, f1, f4	f3	f3, f1	f3, f1, f4	f3, f1, f4	f4, f3, f1	f4, f3, f1	f4, f3, f1
19	f3, f4, f5, f6	f3	f3, f4	f4, f3	f4, f3	f4	f4, f3	f4, f3
20	f3, f4, f5, f6	f3	f3, f4	f3, f6, f4	f3, f6, f4	f3, f6, f5, f4	f3, f6, f5, f4	f3, f6, f5, f4
	Total	**2/20 (10%)**	**7/20 (35%)**	**7/20 (35%)**	**9/20 (45%)**	**3/20 (15%)**	**6/20 (30%)**	**3/20 (15%)**

Appendix F How to calculate localization results manually from the nearest DTNLs list printed out by the HLSM system prototype?

Table F-1 lists the 24 RFID tags used as DTNLs placed on the six pieces of furniture in the experiment mentioned in section 3.5.2. For example, the first RFID tag in the list which has the tag number of **300833b2ddd906c000000005** was labeled **5** and placed on table **f1**.

Table F-1: 24 UHF RFID TAGS USED AS DTNLS ON 6 PIECES OF FURNITURE

Furniture Name	DTNL (RFID tag number)	DTNL label
f1	300833b2ddd906c000000005	5
	300833b2ddd906c000000006	6
	300833b2ddd906c000000007	7
	300833b2ddd906c000000008	8
f2	303643303030303030303130	30
	303643303030303030303131	31
	303643303030303030303132	32
	303643303030303030303133	33
f3	303643303030303030303134	34
	303643303030303030303135	35
	303643303030303030303136	36
	303643303030303030303137	37
f4	303643303030303030304130	40
	303643303030303030304131	41
	303643303030303030304132	42
	303643303030303030304133	43
f5	303643303030303030304134	44
	303643303030303030304135	45
	303643303030303030304136	46
	303643303030303030304137	47
f6	303643303030303030304141	141
	303643303030303030304142	142
	303643303030303030304143	143
	303643303030303030304144	144

The HLSM system prototype (the Java application prototype) was programmed to print out the list of the nearest DTNLs for each of the 100 test cases (80 test cases for localization Type 1 and

20 test cases for localization Type 2). This list can be used for manually checking the localization results generated by the Java application prototype. For example, below is the list of the 9 nearest DTNLs in a test case.

```
1:  300833b2ddd906c000000008: 1
2:  300833b2ddd906c000000007: 2
3:  300833b2ddd906c000000006: 2
4:  300833b2ddd906c000000005: 2
5:  303643303030303030304142: 3
6:  303643303030303030304141: 7
7:  303643303030303030304136: 12
8:  303643303030303030303136: 13
9:  303643303030303030304143: 13
```

There are three parameters in a row. The first parameter is the serial number to indicate the nearest DTNL, the second nearest DTNL and so forth. The second parameter is the RFID tag number of the DTNL. The third parameter is the difference in the DTNL's and the LO's received RSSI. The list is sorted in ascending order of the third parameter values.

The location of the LO from the list above can be manually calculated as follows. With the Localization Rule (*n* = 3, *m* = 2), for example, take the first 3 rows. It can be seen that all the three tags, 300833b2ddd906c000000008, 300833b2ddd906c000000007, and 300833b2ddd906c000000006 belong to f1 (Table F-1) and therefore the LO is concluded to be at f1.

Appendix G LO ontology querying

The LO Ontology in section 3.4.4 was deployed on a Jena Fuseki server version 1.0.0. Below are some sample SPARQL queries for querying the LO Ontology and query results in JSON format.

Query 1: Find which room the Glasses are in.

```
PREFIX lo: <http://www.semanticweb.org/steve/ontologies/2013/4/lo#>
SELECT * {lo:Glasses lo:isIn ?z}
```

Query 1's result: the answer is Kitchen (the bold text in the JSON result script below).

```
{
  "head": {
    "vars": [ "z" ]
  } ,
  "results": {
    "bindings": [
      {
        "z": { "type": "uri" , "value": "http://www.semanticweb.org/steve/ontologies/2013/4/lo#Kitchen" }
      }
    ]
  }
}
```

Query 2: Find the Glasses' nearest DTNL.

```
PREFIX lo: <http://www.semanticweb.org/steve/ontologies/2013/4/lo#>
SELECT * {lo:Glasses lo:isNear1 ?z}
```

Query 2's result: the answer is K_Drw_T1 (the bold text in the JSON result script below).

```
{
  "head": {
    "vars": [ "z" ]
  } ,
  "results": {
    "bindings": [
      {
        "z": { "type": "uri" , "value": "http://www.semanticweb.org/steve/ontologies/2013/4/lo#K_Drw_T1" }
      }
    ]
  }
}
```

Query 3: Update the nearest DTNL of the Glasses from K_Drw_T1 to K_Tbl_T4. This update consists of 2 steps. The first step is deleting the existing RDF statement *<Glasses><isNear1><K_Drw_T1>*. The second step is inserting the new RDF statement *<Glasses><isNear1><K_Tbl_T4>*.

```
PREFIX lo: <http://www.semanticweb.org/steve/ontologies/2013/4/lo#>
DELETE DATA {lo:Glasses lo:isNear1 lo:K_Drw_T1};
INSERT DATA {lo:Glasses lo:isNear1 lo:K_Tbl_T4};
```