Semantically Driven
Web Content Management Systems Adaptation

Lai Wei

A dissertation submitted to the University of Dublin,

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

2010

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed : _____

Lai Wei

13 Sep, 2010

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

Signed : _____

Lai Wei

13 Sep, 2010

Acknowledgements

There are a large number of people who have been supporting me and guiding my study in many ways. It is their kind help that makes this dissertation and the work described herein possible.

Firstly, I thank my supervisor, Dr. Owen Conlan, whose expertise and knowledge, along with his guidance has made this work finished so smoothly. I thank Kevin Koidl, who has provided many hints and helpful suggestions along the research, design and implementation process.

As importantly I thank Fu Chen, whose love, understanding and support gave me the courage to start the course and finish the dissertation. I also thank my parents, for giving me the opportunity to come over to Ireland in the first place, and for their consistent support and belief in me and my education and work.

I would like to express my gratitude to my fellow classmates in the 2009 - 2010 MSc NDS class. They have all been very supportive by giving me inspiring ideas and volunteering to participate in the evaluation.

Finally, I would like to thank those who are behind the various tools, resources and technologies that I have studied - some Web Content Management Systems (WCMS), especially MediaWiki, several adaptive systems, especially Adaptive Personalized e-Learning Service (APeLS), the semantic web technologies and projects such as Linking Open Data project. I may not know most of these people, but it is on their wonderful contributions that my dissertation and work is based.

Thank you all.

Summary

As the background of this dissertation, a number of Web Content Management Systems (WCMS) are used to manage content on the Web. The amount of Web content has become so large that finding a specific piece of information is difficult and time-consuming. Several adaptation services have been developed to solve this problem by adapting Web content to better fit user needs. However, the use of the these services is limited to closed environments in which the intelligence and knowledge source are manually maintained and adaptable content is restricted.

This dissertation asks the question to what extent can a third party adaptation service built on public knowledge be used to adapt content in WCMS.

In order to answer the question, some existing adaptation services were studied to determine their system architecture, adaptation methods and techniques being used; the extensibility of some popular WCMS were analysed to examine the possibilities for their content to be adapted; semantic web technologies, especially the linked data initiative and its resources were researched for their suitability to act as the knowledge source of the desired adaptation service.

With all the technologies and resources studied, the novel design of an adaptation service for WCMS is proposed. The design features the separation of the concerns of content, knowledge and intelligence, and follows Service-oriented Architecture (SOA) principles. The design is implemented in the Semantically Enhanced Open Wiki (SEOW) system, which adapts content on MediaWiki, a state of the art WCMS that forms the basis of Wikipedia, based on simple rules using knowledge from the publicly available DBpedia dataset. The quality of the adaptation is evaluated by analysing

the system performance and user feedback from the user tests which involves eight users each answering six questions.

The evaluation shows that third part adaptation service using public knowledge can successfully adapt content on open WCMS. Moreover, the experience that the adaptation service provides to the users are valuable. It is also noticed that the modular design of the system has great potential because it ensures that each component of the system can be independently extended.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACM Association for Computing Machinery. 12

AHA! Adaptive Hypermedia for All. 16, 17

AHS Adaptive Hypermedia System. 14

APeLS Adaptive Personalized e-Learning Service. 2, 18, 19, 31, 33

API Application Programming Interface. 24–26

BBC British Broadcasting Corporation. 2, 12

CSS Cascading Style Sheets. 37, 46, 49–51

DOM Document Object Model. 51

FOAF Friend of a Friend. 2, 9

GPL GNU General Public License. 23, 25, 26

GUMS General User Modeling System. 28

HTML HyperText Markup Language. 1, 11, 25, 38, 46, 50

HTTP Hypertext Transfer Protocol. 11

JSON JavaScript Object Notation. 36, 45–48, 54, 58, 59

LIP IMS Learner Information package. 29

LMS Learning Management System. 19

MVC Model-View-Controller. 25

OLR Open Learning Repository. 20, 21

OWL OWL Web Ontology Language. 2, 8–10, 13, 26

OWL 2 OWL 2 Web Ontology Language. 10

PAPI IEEE Personal and Private Information. 29

PHP PHP: Hypertext Preprocessor. 13, 45–48

PLA Personal Learning Assistant. 21

RDF Resource Description Framework. 2, 8–14, 20, 21, 24–26, 29, 35

RDFa Resource Description Framework - in - attributes. 24, 26

RDFS RDF Schema. 2, 8, 10, 13, 20, 29

REST Representational State Transfer. 36, 45, 52, 55

SEOW Semantically Enhanced Open Wiki. 6, 43–47, 52, 55, 57–59, 61, 70, 73, 74, 76, 89–91

SIOC Semantically-Interlinked Online Communities. 9

SKOS Simple Knowledge Organization System. 9

SMW Semantic MediaWiki. 26

SOA Service-oriented Architecture. 34, 36, 74

SPARQL SPARQL Protocol and RDF Query Language. 9, 12–14, 24, 35, 39, 41, 52, 54, 57, 76

SQL Structured Query Language. 23, 35

SUS System Usability Scale. 63, 68, 69, 86

SweoIG Semantic Web Education and Outreach Interest Group. 12

URI Uniform Resource Identifier. 2, 9, 11–13, 33, 48, 52, 55–58

W3C World Wide Web Consortium. 8–10, 12

WCMS Web Content Management Systems. 1, 3–5, 7, 14, 22–26, 30–33, 35, 44, 60, 66, 74–76, 86, 88

Web World Wide Web. 1, 7–9, 11, 12, 22, 28, 33, 38, 39, 90

XHTML Extensible Hypertext Markup Language. 24

XML Extensible Markup Language. 10, 17, 22, 45, 55, 76

Chapter 1

Introduction

1.1 Motivation

The World Wide Web (Web) has been flooded with enormous amount of content since it was invented, majority of which are in the form of markup languages such as HyperText Markup Language (HTML), thus are for human reading. In order to manage this huge amount of content, a number of Web Content Management Systems (WCMS) are used, among which MediaWiki[1], Drupal[2] and Joomla![3] are the most popular and widely used ones [50]. While all of the three WCMS are highly extensible and customizable, they tend to have slightly different use: Drupal and Joomla! are used more often for personal and corporate website with blogs, and MediaWiki specializes in providing a wiki platform for multiple users to collaboratively access and edit large amount of information. In most WCMS use cases, system users are given a lot of text to read, making finding a specific piece of information an extremely

[1] http://www.mediawiki.org/
[2] http://drupal.org/
[3] http://www.joomla.org/

complicated task. In order to solve the problem, it is desired to allow computers to understand the semantics of page content, which can then assist users' reading by adapting the page markup.

Resource Description Framework (RDF) is an initiative to represent concept in computer readable formats using triples consisting of an subject in Uniform Resource Identifier (URI) form, a predicate using some common vocabularies, and an object which can be another resource or a value of a primitive data type [40]. Since it was proposed over ten years ago, RDF has been adopted well and enhanced by vocabularies such as RDF Schema (RDFS) [15], OWL Web Ontology Language (OWL) [30], and Friend of a Friend (FOAF) [16] published by various organizations. Linked data is an initiative to use such vocabularies to represent the semantic relationships between resources described in RDF format [7]. Many influential companies and organizations, such as Wikipedia[4], British Broadcasting Corporation (BBC)[5], and the United States Census Office[6], are participating in the linked data project by publishing their data in linked data format [49]. Linked data is a source perfectly suitable for finding semantic relationships between concepts.

A number of adaptation systems, such as AHA! [28], Adaptive Personalized e-Learning Service (APeLS) [25], KnowledgeTree [18], and ELENA [32], were developed to adapt web content. Such systems are capable of altering the web content based on decisions made by applying adaptive rules on knowledge of the web content, possibly with the help of user models which describe some aspects of a user such as acquired knowledge, browsing history, and preferences. By applying different sets of adaptive

[4]http://www.wikipedia.org/
[5]http://www.bbc.co.uk/
[6]http://www.census.gov/

rules and distinct adaptive techniques, the developed adaptation systems have been used in relatively small scope over the years, many of which were proved effective in assisting users to some extent. Besides adaptive systems, a slightly different approach of adaptation is adaptable systems, in which users have control over when to apply adaptation at what items. One similarity of most existing adaptation systems is that most of them are developed for closed educational environment, where the possible content of the system is limited and the semantics are predefined.

A main reason why existing adaptation systems are difficult to be extended to the wider open web is the fact that the knowledge on which the adaptation decisions are based is manually maintained by the service owner. Because a knowledge base for the open web may require enormous amount of knowledge, keeping such a knowledge base seems to be an impossible mission for any individual or single organization. However, the linked data project has great potential to be the knowledge base for the open web: it is in a widely adopted format, freely available to the public, and actively updated and maintained by many organizations and individuals throughout the world. An adaptation service that uses a public knowledge base to adapt open web content on WCMS is a potentially powerful combination, but has not been tried out.

1.2 Research Question

This dissertation asks to what extent can a third party adaptation service using knowledge available in the public domain be used to adapt the content of open WCMS.

1.3 Objectives

The objectives of the study are:

- In order to answer the research question, I will first perform state of the art analysis on closed-environment adaptive services/systems, with focus on how adaptation happens and the adaptive methods and techniques are used;
- The potential of semantic web technologies will be examined, specifically the linked data initiative which may help finding the semantic relationships between things;
- I will examine WCMS for their potential and feasibility to be enhanced by third party extensions or programs;
- A set of services will be designed and implemented, along with complementary plug-ins to allow open WCMS to be adapted using open knowledge;
- The quality of adaptation on the designed and implemented system will then be evaluated by analysing system performances and studying user feedback.

1.4 Contribution

This study introduces a new approach to adaptation on open WCMS, which has three significant advantages:

- The knowledge comes from publicly available sources that do not require specific maintenance by the adaptation service owner;

- The intelligence, typically in the form of rules, is generalized and can be used across various knowledge sources without modification;
- The WCMS only requires a simple plug-in to invoke the adaptation; no change is required on the WCMS content.

On successful evaluation, this study would help understanding the feasibilities and limitations of deploying such adaptation service with the off-the-shelf resources and technologies.

1.5 Thesis Overview

The remaining chapters will be arranged as follows:

- Chapter 2 discusses the state of the art of adaptive systems, public semantic resources, and features of various WCMS. It specifically looks at the various adaptive methods and techniques being used in the existing adaptive systems, the features provided by linked data, and the extensibility of MediaWiki and Drupal, along with their support for semantics. The purpose of looking at these topics is to determine the challenges of introducing adaptivity to open WCMS;
- Chapter 3 presents the design of the open WCMS adaptable system. It will show the separation of the concerns of content, knowledge and intelligence. It proposes an approach that allows adaptation to be used to leverage semantics in order to adapt third party content repositories;
- Chapter 4 presents the implementation of the approach, which is manifested in

the Semantically Enhanced Open Wiki (SEOW) system that consists of a pair of services and complementary plug-ins;

- Chapter 5 introduces a set of user tests performed by eight volunteers. Both the usability of the adaptation service and the completeness and accuracy of the adaptation results are evaluated based on the system performance during the user tests and users' feedback;

- Chapter 6 concludes the dissertation by highlighting the significance of the proposed architecture over previous studies, and summarizing the findings from the evaluation of the prototype system. Some possible future works are also discussed in this chapter.

Chapter 2

State-of-the-Art

2.1 Overview

The desired prototype system of the project will facilitate a WCMS with the ability to adapt its content according to the semantic relationships between the objects mentioned on the page. In order to implement such a system, it is worthwhile to study the following five areas:

1. Representation of objects and their relationships in semantics format, and their availability on the public Web;
2. Adaptation methods and techniques and how they are used in existing adaptive systems;
3. The extensibility and support of semantics on the various WCMS;
4. Similarities and differences between adaptive systems and adaptable systems;
5. The user modeling process.

In this chapter, the state of the art of the five topics are discussed and alternative technologies and systems are compared when applicable.

2.2 Semantic Web Technologies

The semantic web is designed as a framework on which different applications and organizations can share and reuse data [9]. It was first introduced in 2001, aiming at expanding the current Web of documents to the web of data. One of the two main focuses of semantic web is a common data format for integrating data from various sources. The other focus is how the data and the object it represents in the real world are related. The semantic web has gained extraordinary attention and has been developing very fast in recent years.

In this section, a number of semantic web technologies will be discussed for their suitability of being used in this project. I will start from RDF - the fundamental data format used across all semantic web technologies, then cover RDFS and OWL - some of the common vocabularies, and finally reach the linked data, which interlinks semantic data from various sources and helps defining the semantic relationships between the objects.

2.2.1 RDF

As a common data format of semantic web, RDF became a World Wide Web Consortium (W3C) specification in 1999 [40]. A newer version was published in 2004 [4]. It was designed as a data model for information exchange on the Web.

RDF data models are in the form of subject-predicate-object expressions, which

are called triples. Traditionally when describing a relation like this, it is common to use URI to identify the two ends in the relation. However, RDF extends the use of URI by using it to name the predicate itself, which is called a vocabulary. The most fundamental vocabulary in the semantic web is OWL [30] which is built upon RDF. Other popular vocabularies that based on RDF include FOAF [16], Semantically-Interlinked Online Communities (SIOC) [14] and Simple Knowledge Organization System (SKOS) [42]. Although not required, it is also common in RDF for a URI to contain an actual link to the resource on the Web. As a result, structured and semi-structured data from different sources can be easily mixed and integrated using RDF.

RDF metadata can be saved in special databases called triple stores. These triple stores, such as Sesame [23], can support file systems as well as relational databases as the underlying store, and support one or more RDF query languages. Compared to other types of RDF metadata stores, triple stores are optimized for the storage and retrieval of RDF triples.

2.2.2 SPARQL

In order to search RDF and other ontology languages based on it, a number of RDF query languages were introduced, among which SPARQL Protocol and RDF Query Language (SPARQL) [48] is most widely used and became a W3C Recommendation in 2008. SPARQL can be used to write queries on any data either stored in RDF or may be treated as RDF from various sources. Its syntax is capable to express triple patterns, conjunctions and disjunctions.

2.2.3 RDFS

The predicate component in the a normal RDF triple is not specifically defined. This reduces the portability of the RDF data being created because a locally defined vocabulary may not be understood at a remote system. In order to overcome this problem, some organizations released various vocabularies and made them available for public use. All the vocabularies are defined using RDFS - an extensible knowledge representation language that describes how to use RDF to describe RDF vocabularies [15]. RDFS is a W3C Recommendation and the latest version was 1.0 published in 2004 [15].

2.2.4 OWL

OWL is an ontology language for the semantic web with formally defined meaning [52]. It first became a W3C Recommendation in 2004 and a newer version OWL 2 Web Ontology Language (OWL 2) was published in 2009 as an extension of the original one. OWL aims at allowing applications rather than human beings to access the content of the information through facilitating more machine interpretability than Extensible Markup Language (XML), RDF, and RDFS [30]. In OWL, ontologies are formalized vocabularies of terms defined using their relations with other terms that are covering a domain and shared by users [52]. In implementation, OWL and OWL 2 ontologies can be stored and transmitted in various formats whereas RDF/XML is most commonly used.

2.2.5 Linked Data

The current Web is a web of hypertext where massive number of documents are linked together using relationship anchors written in HTML. A hypertext file is typically an HTML file that contains data in marked up format. It might be easier for human beings to understand the data by marking them up using HTML, but it makes it hard to understand for machines. In order to solve this problem and to build a more machine friendly Web, Tim Berners-Lee proposed the idea of linked data [7]. Compared to current Web of hypertext, linked data can be seen as a web of data where data is represented using URIs and the links are described in semantic vocabularies represented in RDF rather than hyperlinks written in HTML.

There are four principles of constructing linked data:

1. Use URIs to name things;
2. Use Hypertext Transfer Protocol (HTTP) URIs so that the name can be looked up;
3. Provide useful information in standard format such as RDF when a name is looked up;
4. Link to external URIs. [7]

By following these four rules, any resource in the world, both information resource such as images and non-information resource such as places, can be expressed, identified, and linked to the rest of the world using linked data [11].

Although the idea of linked data is relatively new, there have been a number of projects attempting to implement the idea and build the web of data. The most

notable project among them has been the linking open data project [49] from W3C Semantic Web Education and Outreach Interest Group (SweoIG). Linking open data project aims at creating the web of data by identifying existing data sets, converting them to RDF, linking them by following linked data principles, and publishing the linked data on the Web [12]. Since its launch in 2007, the project has extended its data sets to include over 13.1 billion RDF triples that are interlinked by about 142 million RDF links by November 2009 [49]. Major datasets of the project include academic institutes such as Association for Computing Machinery (ACM), government official data such as US Census Data, media industrial products such as BBC Programmes, as well as on-line encyclopedia such as Wikipedia's DBpedia [49]. Figure 2.1 shows some of the datasets published in the linking open data project.

In an interview [8], Tim Berners-Lee stated that a more famous part of the linked data initiative is the DBpedia project[1] - a project aims at extracting structured information from the on-line encyclopedia Wikipedia [13]. It has a massive knowledge base describing more than 2.9 million entities using 479 million RDF triples [10]. A dereference-able URI is given to each entity, along with a description in RDF format that contains human-readable definitions and its relations with other resources. Increasing numbers of external data publishers have been linking to DBpedia in the last few years, making it a central hub in the web of data [13]. Despite the enormous space required on the hard disk, the DBpedia datasets can be downloaded under certain licences [35].

In order to consume linked data, many linked data publishers, including DBpedia, Musicbrainz, and U.S. Census [26], have been publishing SPARQL endpoints that

[1]http://dbpedia.org/

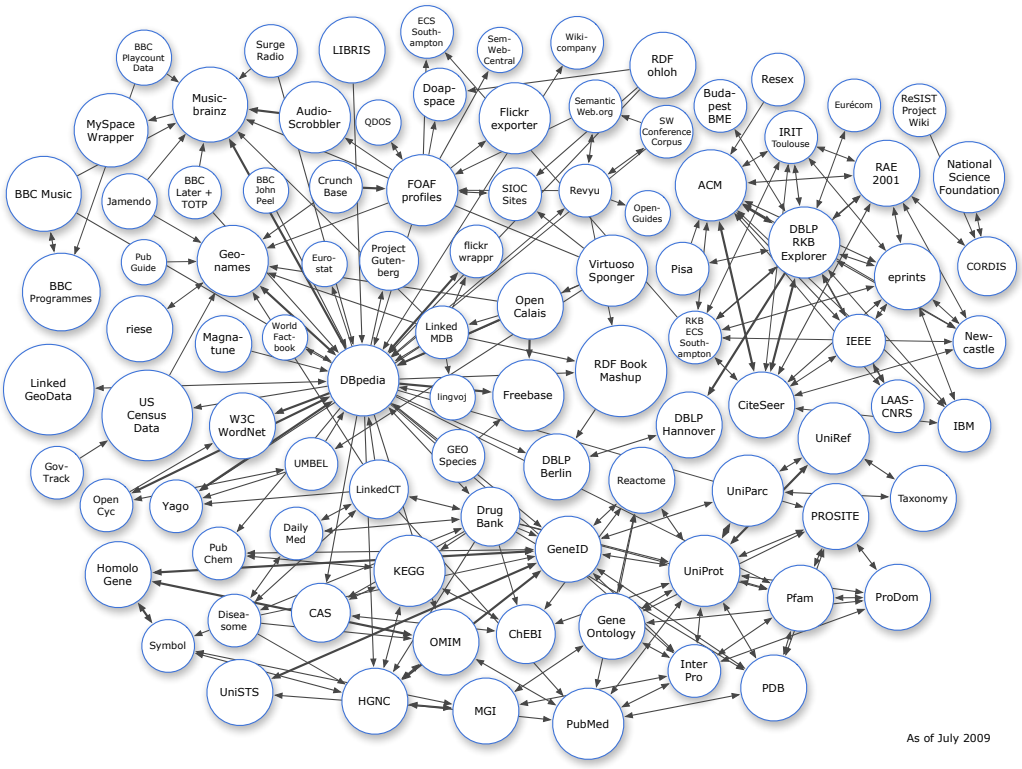

Figure 2.1: Some of the datasets published in the linking open data project as of July 2009 [27]

are capable of accepting SPARQL queries and returning query results in various formats. A number of libraries for different programming languages are also available to assist consuming linked data in application development. For example, Jena for Java, PySPARQL for Python, and RAP for PHP: Hypertext Preprocessor (PHP).

2.2.6 Summary

In this section, the basics of semantic web technologies were introduced. RDF is the fundamental data model that represents resources and their relationships. It is also the building blocks of ontological languages such as OWL and RDFS. Linked data initiative aims at linking abstract resources represented by URIs using RDF links.

SPARQL, the query language of RDF, may then be used to perform queries on the linked data.

The next section will overview adaptive methods and techniques, and illustrate how they are used in several existing adaptive services.

2.3 Adaptive Services

Over the years, a number of adaptation services have been developed for usage primarily in e-learning scenarios [25][18][32][28]. In order to design and implement the desired adaptation service to work with WCMS, it is worthwhile to study how the existing adaptation e-learning systems perform adaptation.

In this section, common adaptation methods and techniques are discussed first, followed by how they are applied in existing adaptation e-learning systems.

2.3.1 Adaptation Methods and Techniques

Typically, there are two parts involved in the adaptation process in an adaptation system, namely user modeling and adaption. User modeling consists of collection and processing of user data in order to build a user model. Adaption, on the other hand, is about using the established user model to adapt the target platform. These two parts usually exist as user modeling services and adaptive services in adaptation systems.

There have been a number of research efforts on Adaptive Hypermedia System (AHS) in recent years, from which many adaptation methods and techniques were developed. AHS refers to hyper-document systems that can be adapted using user

models in some forms. Various adaptation methods and technologies can be used to customize user's view of the subject. Depending on the target of adaptation, these techniques are usually classified as adaptive presentation and adaptive navigation [17].

Adaptive Presentation

In adaptive presentation, the target of adaptation is the subject content's appearance to the users. The motivation behind adaptive presentation is that a same item may be understood differently by users depending on their knowledge and background of the subject, thus providing a most appropriate presentation for a user may help the user understanding the content better in shorter time. For example, when reading about a term, a learner may expect a short definition and a brief explanation, while an expert in the area may think the definition redundant. Content may be tailored by using additional details, prerequisite explanations, or explanations based on comparison with another piece of knowledge that the user is already familiar with. The most obvious technique for adaptive presentation is conditional text, in which all possible information about a term is divided into several chunks and only the chunks meeting current user's knowledge level are displayed to the user. However, conditional text suffers that information needs to be manually divided and programmed to map user's knowledge level. A more advanced and more common technique is fragment variants in which several variants of explanations for each concept are stored and the system decides which variant to present according to user model.

Adaptive Navigation

Adaptive navigation is about providing users with an adapted path in the system by customizing form and structure of links between pages. Most often used link adaptation technologies include adaptive ordering, hiding, and adaptive annotation. Adaptive ordering refers to ordering links on the page in a relativity order based on the user model. It can be useful but suffers from the fact that it makes orders of links unstable, which is undesired by non-professionals [36]. Hiding is a technique in which irrelevant links are hidden from users. Adaptive annotation, which is most commonly used, has similar ideas with hiding, but provides more flexibilities by using more types of annotations than simply visibility. It is notable that user models, especially goal of the user, normally have a significant influence on adaptive navigation results.

Adaptation services normally use a combination of one or more adaptive presentation and adaptive navigation techniques in the adaptation process.

The rest of the section illustrates the use of such adaptive techniques in four existing e-learning adaptation systems.

2.3.2 AHA!

Adaptive Hypermedia for All (AHA!) is an open source adaptive hypermedia platform primarily used for educational purposes [29]. The motivation of the project is to provide adaptive learning resources for learners with different background. Although the goal is similar to other adaptive e-learning systems, AHA! has its unique architecture and features.

At the centre of AHA! are a web server and the AHA! engine. System managers,

course authors, as well as students interact with the system through the web server. In addition, it has a combined domain/adaptation model and a user model. The domain/adaptation model contains the conceptual structure of the application along with the adaptation rules created by course authors; the user model represents a user's knowledge in the form of the number of times each conceptual model has been visited. Learning resources in AHA! can be either hosted on the web server in case of static files, or linked from the cloud if they are dynamic resources such as web servers. When a resource request is received by the web server, it is passed to the AHA! engine, which updates user model and creates adapted web pages using the adaptation rules in domain/adaptation model [28]. Adaptation types supported by AHA! are adaptive link hiding and annotation, adaptive link destinations, and conditional inclusion of fragments, all of which were described in section 2.3.1.

The domain/adaptation model and the user model in AHA! are stored in databases or XML files, depending on system manager's decision. Models and concepts need to be predefined by course authors. Adaptation rules such as adaptive conditions and actions also need to be declared.

2.3.3 APeLS

Traditionally, there are two flavours in research and development of e-learning systems: standard-based courseware reuse systems such as ARIADNE[2] and adaptive web-based educational systems such as ELM-ART [20] [21]. Standard-based courseware reuse systems normally contain a database of existing courseware and allow course creators to search the database for relevant ones and use them as a part of

[2]http://www.ariadne-eu.org/

the course. There are two main problems with this approach. First, every student needs to use same selected courseware which cannot be tailored according to student's background and existing knowledge; second, courseware needs to be copied to the database, which makes newer courseware such as web services unfeasible to the system. Adaptive web-based educational systems, on the other hand, use individual student's model and educational material with pedagogical metadata to dynamically select learning materials that are most relevant. One main problem of these systems is that adaptivity is normally embedded in the learning material so that course creators have to accept learning goals set by original material creators if they would like to use the material. Another drawback of the approach is the inability to easily include external content.

The APeLS [21] is an attempt to combine the benefits of both approaches. The aim of APeLS is to allow teachers to use their own teaching goals to structure courses and reuse existing teaching materials in various formats including both static files and web services. Compared to traditional courseware reuse systems and adaptive web-based educational systems, APeLS has two significant features. The first is to separate the course management systems from the content, which allows same content service to be used in multiple contexts through different course management system portals. The other is to separate content specification from the actual content, which allows teachers to specify desired content type and let the system selects suitable real content at run time. This latter feature enables APeLS to perform adaptation, which involves selecting from a list of narratives predefined by domain experts, each of which describes a possible approach or route to finish the course [24].

Adaptive service in APeLS is driven by learning content model, narrative model,

and learner model. Learning content model is a description of actual learning content in metadata format. Narrative model contains concepts of learning outcomes along with learning strategy. Narrative model and content model are mapped via metadata vocabulary. Learner model represents the knowledge of a learner based on the learner's interaction with the system. An adaptive engine processes the three models at run time to dynamically generate learning content for a learner. Extra models such as device model which allows adaptivity based on user's device can also be added if required.

2.3.4 KnowledgeTree

KnowledgeTree is an architecture for adaptive e-learning based on distributed reusable learning activities [19]. Similar to APeLS, the aim of the architecture is to fulfill the gap between the large amount of resources in the existing Learning Management System (LMS) and the potentially powerful adaptive systems to provide a new generation of e-learning systems featuring personalized content.

The architecture consists of four core components: activity servers, value-adding services, learning portals, and student model servers [18]. A learning portal is a platform for course providers to structure and manage the course, and for learners to access the learning tools and resources. Like an educational repository, activity servers manage various reusable content and services that act as learning resources in the system. It is capable to manage both traditional static files and dynamic content such as web services. Value-adding service could add value to the resources in the activity servers using techniques such as annotation, visualization, and sequencing.

The value-adding services can be queried by the learning portal in the same way as activity servers. Value-adding services are independent from activity servers, thus can be used on different learning resources. The optional student model servers represent a student's goal and current states in the system. These models are independent from learning content and can be applied in different courses.

The KnowledgeTree architecture is flexible in a way that several instances of each component may exist at the same time and instances are exchangeable. It is also an open architecture in which new components such as value-adding services and external activity servers/services can be added to the system as needed.

In KnowledgeTree architecture, adaptivity could take place in two components. An activity server/service can be intentionally designed to be adaptable. More often, adaptivity is achieved by various value-adding services that normally annotate activity servers' content. An example of such value-adding services is QuizPACK[3], which creates dynamic parameterized quizzes for programming-related courses using adaptive presentation technique [46].

2.3.5 ELENA

Peter Dolog et al. proposed an approach of personalization in distributed e-learning environments [32] and implemented it in the ELENA project[4]. This approach takes consideration of Open Learning Repository (OLR) - a metadata-based e-learning repository modeled using RDF/RDFS as modeling language [31]. On one hand, OLR can provide a much wider range of learning materials which may significantly enrich

[3]http://www2.sis.pitt.edu/$\sim$ *taler/QuizPACK.html*
[4]http://www.elena-project.org/

an e-learning system; on the other hand, the learning resources may be created for special purposes, the learning repositories may also appear or disappear in unexpected manner. OLR increases difficulty of adaptation, however, adaptation on open learning resources has much more utility value.

In the ELENA project, the e-learning system is built upon a number of interactive services. At the system level, Personal Learning Assistant (PLA) Services is at the centre of the architecture. It integrates and uses other services to find learning materials for learners, who interact with the system using interaction components. The user interaction components may be provided in several formats to support learners using different devices. The personalization services are composed of three core services - query rewriting service, recommendation service, and link generation service. Query rewriting service reforms user queries using additional information such as user model. Recommendation service adds annotation to learning resources based on user preferences described in user model. Link generation service creates hyperlinks between learning resources according to the semantic relations of them as well as user model whereas applicable. There are also some supporting services, each of which focuses on a specific less critical functionality of the system. A notable one is repository services which provide access to resource repositories connected to the network. Most of these services can be extended to include new functionalities. New services can also be added to extend the system.

Information that needs to be described in an e-learning system includes descriptions of learning resources, user profiles, as well as specific learning goals. In order to describe such information, ELENA represents the properties in RDF classes, combined with optional additional domain ontologies. Ontologies are built based on such

descriptions and managed by ontology service of the system. A mapping service is in charge of managing the relations between ontologies.

2.3.6 Summary

This section introduced the various adaptation presentation and navigation techniques, followed by an overview of four adaptive e-learning systems. These systems have distinct architectures and use different mechanisms to drive their adaptation, however, they all use one or more methods and techniques described in 2.3.1 to achieve adaptive personalisation.

So far, the Web content and user models are represented in computer readable format, and the possible adaptive techniques are known. The next step would be performing adaptive personalisation on some content in a WCMS. The next section will discuss the alternative WCMS.

2.4 WCMS

Up to a few years ago, the Web was authored and managed by computer professionals who had knowledge in markup languages and web application management. This has been changed over the recent years by the existence of a number of WCMS. A WCMS is a web application that is used for the online management and control of web content via a web interface [41].

WCMS are typically facilitated with content creation, control and editing, along with Web maintenance functions. WCMS usually use database or XML files to store the created content, which can be displayed in the required format when requested

by an end user of the WCMS. Many advanced WCMS also provide programming interface for developers to produce additional plug-ins to extend their functionalities.

In this section, a selection of popular WCMS based on WCMS market research [50], including Drupal, Joomla! and MediaWiki, will be discussed and compared for their suitability to be used in this project. The comparison focuses on their extensibility and support for semantic web technologies.

2.4.1 Drupal

Drupal is an open source WCMS, distributed free of charge under GNU General Public License (GPL) [51]. It allows users to publish, manage, and organize content on a website. Dries Buytaert developed early versions of Drupal, and it is currently maintained and developed by developers from the Drupal community. The standard version of Drupal, Drupal core, provides features such as user and access control management, content management and feed generation, page layout and appearance control, as well as logging and other administrative functions. More functions can be added to the platform in the form of modules, most of which are developed by the community. Drupal is one of the three most popular WCMS [50] and is used by many organizations and individuals including the White House[5], and Sony Music[6].

Drupal is developed in PHP, same as all of its numerous external modules. MySQL and PostgreSQL databases are natively supported, with the possibility of any other Structured Query Language (SQL) database support by implementing fourteen back-end supporting functions. Internally, Drupal has a layered structure in order to

[5]http://www.whitehouse.gov/
[6]http://www.sonymusic.com/

separate content with displaying elements, settings and configurations. The layers of Drupal include data, modules, blocks and menus, user permissions, and templates. In order to personalise appearance of the content on a page, changes are normally made at data, modules and template layers. An Application Programming Interface (API) is provided by Drupal, which can be used to interact with the three layers to develop the personalisation plug-in. Drupal 6 has over 3,600 external modules to download to extend its functionality.

Since version 6, Drupal officially started RDF support through its RDF API from RDF module [5]. As described in the module introduction blog entry [6], the API provides a storage abstraction layer for storing RDF triples so that other Drupal modules can connect to the API and use it as a data store for their own higher level usage. A SPARQL module that works with the RDF module is also available on Drupal, which enables query on the stored RDF triples. It is also revealed that the new Drupal 7, which is currently under development, will natively support semantic generation in its core by creating metadata in Resource Description Framework - in - attributes (RDFa) format - a specification for attributes to express structured data such as RDF in markup language such as Extensible Hypertext Markup Language (XHTML) [3] - whenever new content is created in Drupal [22]. This will potentially convert massive amount of information managed by Drupal into RDF triples.

2.4.2 Joomla!

Joomla! is an open source WCMS that allows users to build websites and online applications [44]. It is developed in PHP and uses MySQL database as its data store.

Joomla! features easy installation and comes with an extension directory with a large number of extensions, most of which are free under GPL [44]. Like Drupal, Joomla! is also among the top three most popular WCMS [50].

The architecture of Joomla! follows Model-View-Controller (MVC) architectural pattern, with a controller component sitting in the middle, getting data from database through model component, and getting layout settings from view component. The controller component is also responsible for communicating with web server to respond to users' requests.

Functionality of Joomla! can be extended using over 4,600 extensions falling into several categories from its official extension directory. An API is also publicly available for Joomla! framework, on which Joomla! WCMS is built [45]. Content personalisation may be achieved by developing plug-in that uses the API. However, the core components of Joomla! can only process data saved in the database and is not capable of dealing with semantic web technologies such as RDF. There is also a lack of relative plug-ins for the platform. Therefore semantics is neither natively nor externally supported in Joomla!.

2.4.3 MediaWiki

MediaWiki is an open source server-based wiki engine that features advanced content management and customizable information representation [1]. Besides basic content creation, management and organisation functions, MediaWiki also provides version control on all the stored content. For content representation, MediaWiki uses wikitext format, a lightweight alternative to HTML, to markup display elements on the page.

MediaWiki was first developed for Wikipedia[7] and other Wikimedia projects, and was later released for others to use. It is unique compared with other WCMS because it tends to be used for knowledge management much more often than others. MediaWiki is freely available, licensed under GPL. It is relatively new in the WCMS market but a fast growth on its user base has been witnessed since its launch [50].

MediaWiki is developed in PHP and uses MySQL as its main data store. A developer API is provided for extension development purposes, which uses callback functions to include external codes in the code base. The MediaWiki extension matrix offers about 1,500 extensions to its users.

There have been a few projects aiming at adding semantic support to the MediaWiki platform, among which Semantic MediaWiki (SMW)[8] is an outstanding one. SMW allows content authors to annotate the wiki's content with machine-readable information in the form of OWL ontology language descriptions [39], which makes publishing semantic web content easier on MediaWiki platform [2]. The annotated wiki with machine-readable semantics can then serve as a data store for other semantic web services to use. SMW has found its use cases in many MediaWiki instances.

2.4.4 Summary

For the WCMS examined, API with display changing functionality are available on all three systems. However, only Drupal and MediaWiki are capable of dealing with semantics in RDF or similar formats. The native RDFa support on Drupal 7 is desired, but the system is under development at the moment and is not available

[7] http://wikipedia.org/
[8] http://semantic-mediawiki.org/

yet. Compared to Drupal, MediaWiki is more often used in knowledge management systems, which would make an ideal prototype system for this project.

With all the essential components of the prototype system examined, the next section will discuss the two approaches that an adaptation system can take and their differences.

2.5 Adaptive and Adaptable System

In general, there are two adaptation approaches that a system can use to adapt its content - adaptive system and adaptable system [43].

In the first approach, assumptions of users' needs are made by the system, using which the system is adapted. Such systems are called adaptive systems, and they implement implicit adaptation. User modeling services are normally used to improve the quality of the assumptions.

In the other approach, users are allowed to change certain parameters of the system, which perform adaptation by changing system's behaviour accordingly. Such systems are called adaptable systems [43] and the adaptation functions provided by these systems are explicit.

Which approach to use in an adaptation system really depends on the nature of the system and users' needs. In general, however, adaptable systems tend to be more flexible and give users more control over their use.

2.6 User Modeling

User modeling can be a complex process and may apply a number of techniques. Although user modeling is not the core of the this dissertation, it is still worthwhile to learn the requirements of the user modeling and what a user model looks like.

In order to distinguish users of the application and know their individual status and need, many systems need a way to recognize and record characteristics associated with each user. This is the process of user modeling.

The potential value that user models can bring to computer applications was recognized very early. Pioneering user modeling systems were initially proposed and developed in the late 70s and early 80s [47], most of which did not clearly separate user modeling components from the rest of the application, thus seriously reduced re-usability. The General User Modeling System (GUMS) [34] published in 1986 was the first user modeling system that saved assumptions about users in lively updated stereotype hierarchies. The first user modeling system that clearly separates user modeling component was witnessed in 1990 [37], and most systems after that followed the design. With the popularity of the Web and networked applications, a number of centralized user modeling services were developed around 2000. Centralized user modeling systems have many advantages over the previous systems especially in terms of consistency among applications and re-usability. Such systems are still actively being used nowadays.

Most common requirements of user modeling systems are:

- *Generality.* Multiple applications should be able to make use of a single user modeling system.

- *Expressiveness.* The system should be capable to describe more than one type of assumption for a user.
- *Inferential capabilities.* It should be able to perform different types of reasoning.
- *Comparisons of different user's model.* Different user models should be comparable in order to predict user patterns.
- *Import external user model.* External user information such as other user models should be usable in the system. [38]

In recent years, there were a few attempts to use semantic web technologies to assist user modeling. In the ELENA project [32], for example, developers implemented a semantic web driven user modeling service [33], which uses a combination of RDF, RDFS, along with two specific learner profile standards, IEEE Personal and Private Information (PAPI) and IMS Learner Information package (LIP), to represent a user model. Figure 2.2 shows an example user model for a student in ELENA.

2.7 Summary

In this chapter, I started by proposing the possible essential technologies required in the desired adaptation process:

- A common knowledge representation format for both objects and their relationships;
- Publicly available knowledge sources;
- Various adaptation methods and techniques;

```
<rdf:something>
 <papi:performance>
     ...
  <rdf:Description
   rdf:ID="Advanced Security Technologies for Networking II">
   <papi:issued_from_identifier
    rdf:resource="http://www.elena.../Test AST345"/>
   <papi:learning_competency
    rdf:resource="http://www.kbs.uni-hannover.de/Uli/ACM CCS.rdf\#
    C.2.0.2"/>
   <papi:learning_experience_identifier
    rdf:resource="http://www.elena.../AST34.pdf"/>
   <papi:granularity>
    topic
   </papi:granularity>
   <papi:performance_coding>
    number
   </papi:performance_coding>
   <papi:performance_metric>
    0-1
   </papi:performance_metric>
   <papi:performance_value>
    0.9
   </papi:performance_value>
  </rdf:Description>
 </papi:performance>
<rdf:Description>
```

Figure 2.2: An example user model of a student in ELENA project [33]

- WCMS as the interaction platform and content repository;
- Alternative adaptation approaches;
- User modeling technologies that may help making more appropriate adaptation decisions.

The state of the art of each of these areas was then studied one by one. According to my study of the existing adaptation systems, there has not been a significant

adaptation system designed specifically for open content in WCMS. However, the following findings from the various areas are concerned for constructing one:

- The multi-model design of APeLS system gives the possibility of modularity separation of concerns of content, knowledge and intelligence;
- The plugability of the WCMS provides the feasibility to create plug-ins to manipulate WCMS content;
- Linked data has appropriate instance information to represent resources and their relationships.

By combining these technologies and resources, an adaptation system for open content is not only feasible, but it can also be rather powerful.

Chapter 3

Design

In chapter 2, I have discussed the alternative technologies that might be used in the adaptation service, as well as the possible approaches that such a service can take. In this chapter, I will start by describing the desired adaptation service and stating the requirements of its design. This will be followed by a proposed architecture that fulfills the requirements. After all of the core components of the adaptation service and their communications are analysed individually, I will discuss the advantages of the design and summarize the chapter.

3.1 Design Requirements

Directly derived from the research question and as a piece of innovation being introduced in this dissertation, the desired adaptation service is an adaptable system that is able to adapt user created content of any topic in a WCMS using publicly available knowledge. Content annotation and adaptation often try to guide users to the appropriate information on the page. One way to achieve that is highlighting the

terms that might be interested to the users. The adaptation system uses MediaWiki as the chosen WCMS and linked data as the source of knowledge. MediaWiki is chosen because of its popularity to be used as the basis of information centric on-line systems. Linked data, on the other hand, is selected as it is the most appropriate and suitable knowledge source.

From the study of the various related technologies in chapter 2, the key requirements of the adaptation service are:

- Derived from existing adaptation services such as APeLS and KnowledgeTree, the concerns of content, knowledge, as well as intelligence that drives the decision making need to be separated, which effectively means the three core components of the adaptation service would be content component, knowledge component, and intelligence component.

- The content component should have the following functionalities:

  - Use a WCMS to manage the creation and editing of the Web content;

  - Adaptable items need to be identified in the Web content;

  - Adaptable items need to be highlight-able in the Web content.

- The knowledge component should have the following functionalities:

  - Find the resource type of the describing resource;

  - Leverage the knowledge of linked data, which enables getting resources[1] of

[1] A resource is a object or "thing". For example, "Dublin Airport" is a resource, which has the URI "http://dbpedia.org/resource/Dublin_Airport", and can be displayed as "Dublin Airport or "DUB".

relative topic based on type[2].

- The intelligence component should have the following functionalities:
  - Store, manage and process adaptation rules in the predefined form;
  - Make appropriate adaptation decision based on the rules.

The major non-functional requirement - the separation of concerns of content, knowledge and intelligence - is significant to the system because it guarantees the openness and extensibility and allows more advanced sub-systems to replace the existing ones when new technologies mature.

3.2 System Architecture

An adaptation service is designed to fulfill the requirements. The proposed service has a Service-oriented Architecture (SOA) style architecture, which provides a set of loosely-integrated services including content repository, knowledge repository, and intelligence component. Figure 3.1 illustrates the architecture along with the positions of each component in the system and the major interactions between them.

SOA is selected as the architecture because it best fits the need of the separation of the concerns of content, knowledge and intelligence. In SOA, the core components are completely loosely coupled. Services are not required to be at a particular system or network, thus achieving dynamic connectivity. This can hugely benefit the desired adaptation service because of the separation nature of the services and the possibility for each service to grow indefinitely in the future to include enhanced functionalities.

[2]A resource is said to be of a type if it is regarded as a member of the class or category. For example, "Dublin Airport"is in the type "airport".

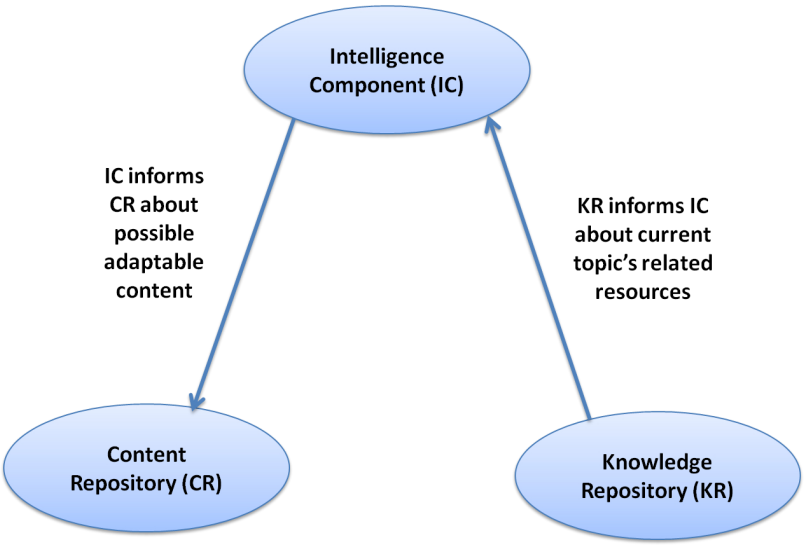

Figure 3.1: Architecture of the desired adaptation service

In the adaptation service, content repository is a service that contains an instance of MediaWiki WCMS and a plug-in that can manipulate the content following adaptation decisions. The MediaWiki instance provides the mechanism to store the content in a SQL database and to manage the saved content. The plug-in uses the tag extension to introduce a specific tag which is hooked up with a parser function that pre-parse the page content in wikitext format before it is processed by the built-in wikitext parser in MediaWiki. The content repository service has a simple interface to communicate with the intelligence component in order to inform the latter about page title and receive adaptation decisions from it.

Knowledge repository is a service that acts as a gateway to access knowledge in the linked data. DBpedia and linked data datasets offer information about resources and their relationships represented using RDF and the various semantic vocabularies respectively. Most of these datasets are capable of processing SPARQL queries and returning results. The knowledge repository service is able to query DBpedia dataset

for instances of relative topic based on types. It also provides an interface for the intelligence component to utilize the service.

Intelligence component is another service that is in charge of managing adaptation rules and making adaptation decisions based on the rules. A simple adaptation rule contains a list of object types that are relevant to the page topic. The intelligence component service uses the object types to create a list of relevant resources to the page topic, and uses it as the adaptation decision. The service has an interface to communicate to the content repository and another to talk to the knowledge repository.

Following the SOA principles, the two services that take requests from other components - knowledge repository and intelligence component - provide their interface as Representational State Transfer (REST) web service for its simplicity. The messages being passed between the services are encoded in JavaScript Object Notation (JSON) format.

In the following sections, each of the components will be discussed individually with in-depth details.

3.3 Content Repository

Content repository has the following functions and responsibilities in the system:

- Provide storage of all the content to be displayed to the system users;
- Provide an interface for content authors to manage content;
- Provide an interface for other system users to browse the pages and navigate

between them;

- Include a communication mechanism so that it can inform other components of the system about current page topic and receive response that contains adaptation decisions;

- Have the ability to manipulate page display without affecting the saved content based on the adaptation decisions received.

Figure 3.2 shows the internal architecture of the content repository. As can be seen from the figure, the content repository service exists in the form of a MediaWiki instance with a third party plug-in, along with necessary Cascading Style Sheets (CSS) styles and JavaScript functions.

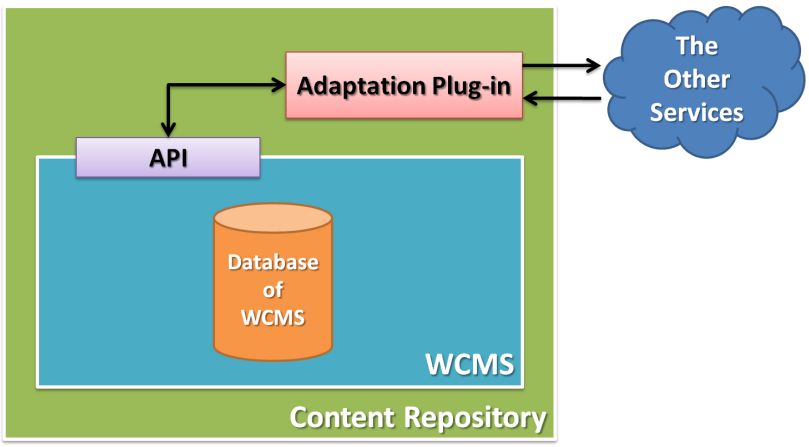

Figure 3.2: Internal architecture of content repository service

MediaWiki provides a sophisticated solution for content storage, management and hosting. The plug-in has a parser function that is hooked up with each page that the content author or the service owner wants to be adaptable using a special tag. The function is called every time these pages get loaded or reloaded, and it requests

the other services for appropriate adaptation decision, which contains a list of object types along with the related items of each type. Upon receiving the response, the function amends parsing results by making two types of modification:

1. A highlighting control element with all the available highlighting options is added to the top of the page;

2. The recognized potential highlighting items are edited to be in certain displaying style.

The highlight-able items can be turned on and off using the highlighting control on the page without making additional requests to the other services or external resources. This approach forces the adaptation results to use the knowledge at the exact time when the knowledge repository queries the DBpedia dataset, but considering the fact that DBpedia dataset is only updated every a few months, the effect of this can be ignored.

The manipulated wikitext is then sent back to the built-in parser of MediaWiki, which converts the wikitext to normal HTML tags that get displayed on the Web page. With the help of customized JavaScript functions, users are able to switch on and off the highlighting items using the added control element.

It is notable that throughout the process, neither the MediaWiki instance nor the plug-in is capable of understanding the semantics of the content. All actions that are performed by both parts are about editing wikitext and HTML, which are markup languages.

3.4 Knowledge Repository

As discussed in section 3.1, linked data is selected to be the source of knowledge of the adaptation service. Thus the knowledge repository needs to act as a gateway to linked data. The knowledge repository has the following functions and responsibilities in the system:

- Provide query service on linked data and support queries in the form of "find all displayable resources of type X that is related to resource Y", in which a displayable resource is one with a name that can indicate itself on the page. When adapting a Web page, only displayable resources matter because only these resources can be identified and potentially adapted;
- Provide an interface so that the other services are able to use the query service and receive results with minimum effort.

Figure 3.3 shows the design of the knowledge repository service. Upon receiving a request from the other services, the knowledge repository validates the querying conditions before constructing a SPARQL query for the relevant resources. The query is sent to a chosen SPARQL endpoint with the help of an appropriate library. When the results get back, the knowledge repository forms it in an predefined format and transfers it back to the requesting component.

Notably, the knowledge repository does not have any previous knowledge of the querying items until the request is received. Linked data, as the knowledge source, is solely managed by various organizations and individuals across the world rather than the adaptation service owner. The only intelligence that the knowledge repository has

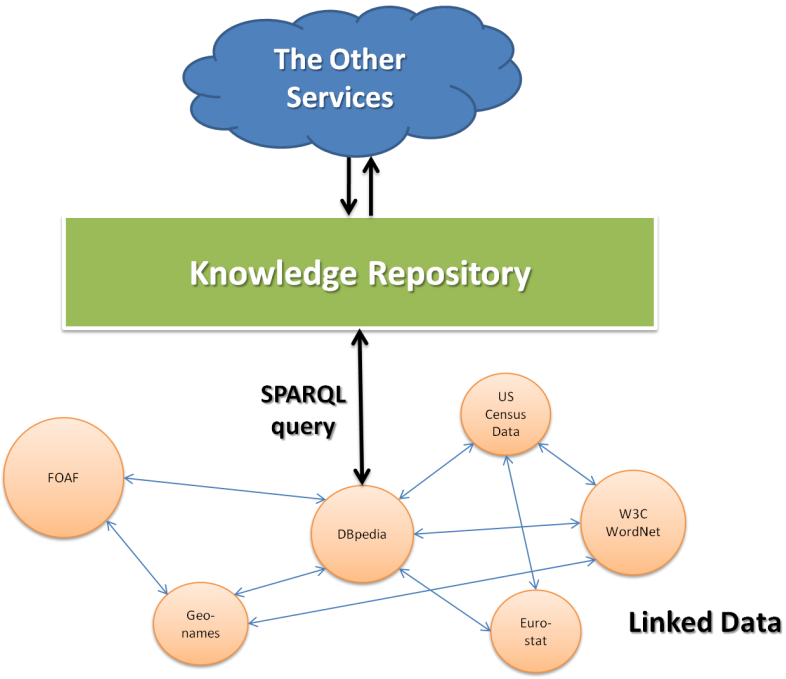

Figure 3.3: Design of knowledge repository service

is the ability to decide whether a resource is displayable, thus no adaptation decision is made by the component.

3.5 Intelligence Component

As its name suggests, the intelligence component is the only component in the system that has intelligence to make adaptation decisions. Its functions and responsibilities are:

- Receive adaptation requests in the form of page titles from the content repository;
- Store and have the ability to process the adaptation rules, which contain a list of item types to be highlighted;

- Request the knowledge repository service for related items of the topic to construct adaptation results;
- Return adaptation results to content repository in a format that is resolvable by the content repository.

Because content repository is unaware of the meanings of the page content, all it can inform the intelligence component is the title of the page in the form of plain text. When the intelligence component receives the title, it first finds out the resource that the title represents and the type of the resource by making a SPARQL request to an endpoint. Based on the resource type, appropriate rules are loaded, each containing a list of object types that relevant to the queried object. The intelligence component then sends a separate request to the knowledge repository for each relevant object type. The results are then combined to form the adaptation decision, which is sent back to content repository in a predefined format. Figure 3.4 shows the design of the intelligence component.

The significance of the intelligence component is that it has neither previous knowledge of the content in content repository, nor the knowledge in knowledge repository. All it has is a set of simple rules and the ability to select which rules to use according to the request received by the service. It is likely that the simple rules would be replaced by more complicated reasoning services such as user models in the future.

In the last few sections, the architecture of the system was introduced and the core components of the system were discussed individually. The next section will cover the advantages of the design and its potentials.

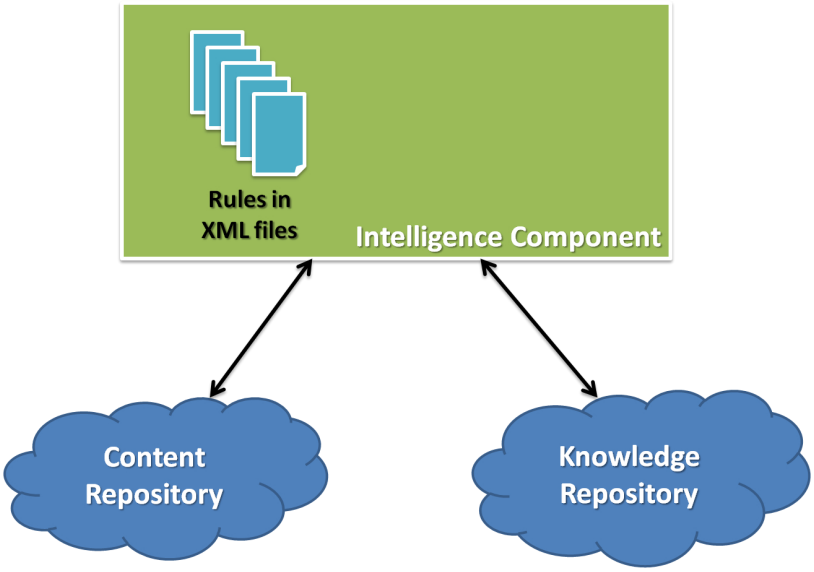

Figure 3.4: Design of intelligence component service

3.6 Design Benefits

The most significant advantage that the proposed adaptation service features over existing adaptation systems is the separation of concerns of content, knowledge and intelligence, and locating them in three independent and communicating services. Therefore, content repository consists of only displaying elements that computers do not necessarily need to understand; knowledge repository contains no more than resources and their relationships that are managed and maintained by various organisations and individuals rather than the adaptation service owners; the intelligence component has neither content nor knowledge, but it can make adaptation decisions based on adaptation rules and status of the other services.

Previous adaptation systems did not manage to achieve adaptation in open environment primarily because of the complexity of maintaining a knowledge base of the unpredictable open content. This architecture opens the gate of adaptation in open

environment because it makes use of publicly available resources as its knowledge base. The architecture has huge potential because:

- Using the adaptation service on different content repository requires no modification on the other services;
- Newly added content in content repository can only bring more adaptation possibilities to the system;
- As time goes by, when the linked data get extended through adding new knowledge sources and updating knowledge from existing sources, knowledge repository will grow and more adaptation results and better adaptation qualities can be achieved without changes in the other services;
- Having adaptation rules amended by domain experts, adaptation results can be tailored by user's specific needs.

3.7 Summary

In this chapter, I introduced the architecture of the desired adaptation service, focusing on the separation of the concerns of content, knowledge and intelligence. I then discussed the core components with their functions and responsibilities individually. This is followed by the benefits and potential of the design. The next chapter will take the design and implement it in the SEOW system.

Chapter 4

Implementation

In chapter 3, I introduced the core components of the system and the communications between them. In order to demonstrate the proposed benefits of the design and to evaluate how well it performs, in this chapter, I will introduce an implementation of the design in the SEOW system.

SEOW system, as its name suggests, is a set of tools and services that cooperate with each other to provide adaptation services to MediaWiki WCMS in open environment with the help of semantics. It consists of three sub-systems - a MediaWiki instance with the semanticTag plug-in, a linked data gateway web service, and an adaptation decision making web service, each of which reflexes a design component in the system architecture. The MediaWiki plug-in is responsible for informing the rest of the system about the topic of the current web page. Upon receiving the page topic, the adaptation web service finds a list of object types that are relevant to the topic, and requests the linked data gateway web service to find all objects falling in the types from the cloud and puts them into the adaptation decision. After the decision

is sent back to the MediaWiki plug-in, the page content is altered for any occurrence of the relevant objects, and if found in the page, they are marked highlight-able using a significant background color.

In the rest of the chapter, the off-the-shelf technologies and resources being used in the SEOW system will be discussed first for the reasons why they are chosen from the alternatives. This will be followed by the implementing details of each of the three services that implement the three core design components respectively.

4.1 Technologies and Resources Selections

In order to ensure the separation of the concerns, the core components of SEOW are implemented as three individual communicating sub-systems.

- The content repository has a MediaWiki instance as a base. Some third party plug-ins are used, and a plug-in is specifically developed for SEOW using PHP, which is the only supported language on MediaWiki platform.

- The knowledge repository is implemented as a REST web service because its simplicity in use. Java is selected as the programming language because of its supports of various web application servers.

- The intelligence component is also implemented as a REST web service and uses Java for the same reason with the knowledge repository. The simple rules used in SEOW, which will be introduced in section 4.4, have XML syntax.

JSON is used as the standard message format because it is supported in both Java and PHP either natively or via third party library. All messages passing between the

sub-systems are encoded in JSON.

4.2 Content Repository

The content repository of SEOW contains a MediaWiki instance along with the necessary plug-ins.

The MediaWiki instance uses MediaWiki version 15.4. The hosting machine is a Dell XPS M1530 laptop, and has Ubuntu 10.04 operating system based on 2.6.32-24-generic Linux kernel. PHP version 5.3.2-1ubuntu4.2 is installed to run PHP script, and MySQL version 5.1.41-3ubuntu12.6 is used as the database for MediaWiki.

The MediaWiki has the following plug-ins installed:

- CSS extension[1] is used to allow the MediaWiki instance to load user-created CSS file that contains the displaying styles of the highlighting items. The additional CSS file will be discussed in section 4.2.2.

- Simple Forms extension[2] is used to introduce new tags in wikitext that can be converted to form and its various control types in HTML.

- JavaScript extension[3] is used to import external JavaScript file in the output HTML document. The added JavaScript functions will be discussed in section 4.2.3.

- SemanticTag extension is the plug-in that performs the adaptation on the content and communicates with the rest of the adaptation system. While all of

[1] http://www.mediawiki.org/wiki/Extension:CSS
[2] http://www.mediawiki.org/wiki/Extension:Simple_Forms
[3] http://www.organicdesign.co.nz/Extension:JavaScript

the three previous plug-ins are found from the MediaWiki extension catalogue, the SemanticTag plug-in is developed for the SEOW system specifically. The SemanticTag plug-in will be discussed in more details in section 4.2.1.

4.2.1 SemanticTag Plug-in

The SemanticTag plug-in is developed in PHP following the MediaWiki extension development guideline. It is both the applier of the adaptation and the communication system of the content repository.

The plug-in registers a special tag in the wikitext - the <semanticTag> tag. The tag takes one parameter, which is the title of the current MediaWiki page, and wraps all the wikitext of the page between its starting and ending tags. Figure 4.1 shows how a MediaWiki page looks like after using the <semanticTag> tag.

```
<semanticTag title="PAGE_TITLE">
NORMAL_PAGE_CONTENT_IN_WIKITEXT_FORMAT
...
</semanticTag>
```

Figure 4.1: Sample wikitext of a MediaWiki page using <semanticTag > tag

If a <semanticTag> tag is reached when rendering the wikitext, a PHP function that is hooked with the tag in the plug-in is called. The function makes a request to the adaptation web service, which will be introduced in section 4.4, and receives response in JSON format. A sample intelligence component service response is shown in figure 4.2.

As can be seen from figure 4.2, the response has a layered organization in which the first layer is a list of all available object types. For each type, the second layer contains

```
{
  "Aircraft": {},
  "Airline": {},
  "Airport": {
    "0": {
      "subject": "http:\/\/dbpedia.org\/resource\/British_Airways",
      "predicate": "http:\/\/dbpedia.org\/ontology\/hubAirport",
      "object":
          "http:\/\/dbpedia.org\/resource\/London_Gatwick_Airport",
      "displayNames": {
        "0": "EGKK",
        "1": "LGW",
        "2": "Gatwick Airport",
        "3": "London Gatwick Airport"
      },
      "objectProperties": {}
    },
    "1": {
      "subject": "http:\/\/dbpedia.org\/resource\/British_Airways",
      "predicate": "http:\/\/dbpedia.org\/ontology\/hubAirport",
      "object":
          "http:\/\/dbpedia.org\/resource\/London_Heathrow_Airport",
      "displayNames": {
        "0": "EGLL",
        "1": "LHR",
        "2": "London Heathrow Airport"
      },
      "objectProperties": {}
    }
  }
}
```


Figure 4.2: Sample intelligence component web service response

a list of related objects falling in the type. The third layer is a representation of the object with its URI and possible displaying names. This layered structure simplifies the processing the adaptation results in the plug-in.

After the response is received by the function, it is decoded from JSON and converted to a PHP array. Each object type is given a highlighting background color,

depending on their orders in the object type list. The available colors are defined in the additional CSS file, which will be introduced in section 4.2.2. The function then inserts a table to the beginning of the wikitext, which contains all available object types, each attached with a switch in the form of two radio buttons. The radio button control is supported by using Simple Forms MediaWiki extension. The radio buttons are not within any form, thus their events are not connected to any form actions. However, the switching on and off events are handled by two JavaScript function respectively, which will be introduced in section 4.2.3. Figure 4.3 shows the switches' appearance on the MediaWiki page after being rendered.

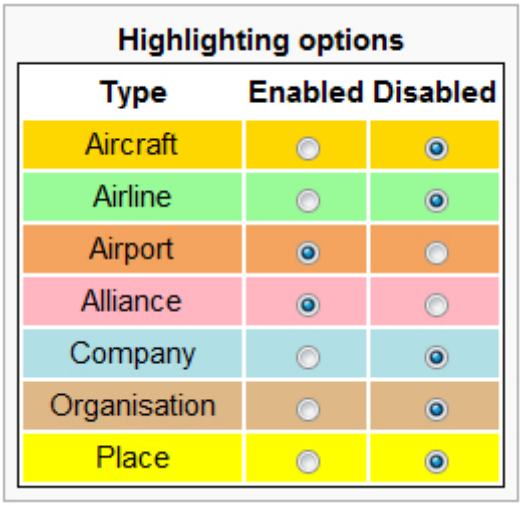

Figure 4.3: Screenshot of the highlighting options controlling switches

Moreover, the function looks into the original wikitext of the page for any appearance of the possible displaying names of the related objects. If one is found, the text is marked as a part of a with an id. All displaying names of the objects in the same type share a same id.

After finishing all the adaptation work, the altered wikitext is returned to the default MediaWiki parser and converted to the final HTML document that end users can see from their browsers. Figure 4.4 shows the result of highlighting on a MediaWiki page.

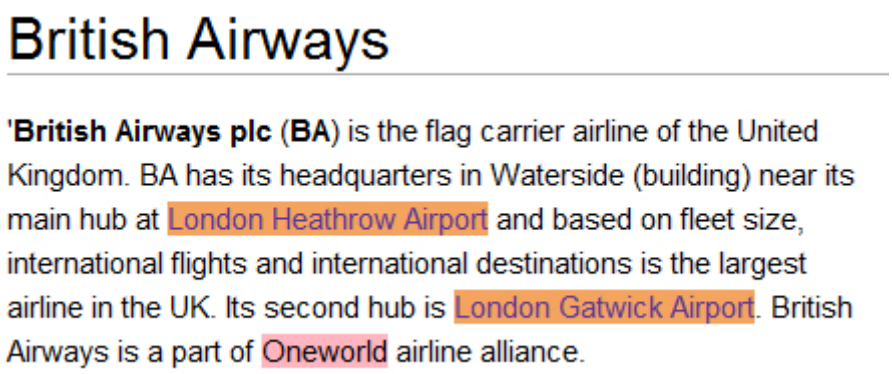

Figure 4.4: Screenshot of the highlighted items on a MediaWiki page

4.2.2 Additional CSS Styles

Two types of displaying styles are used for the highlighting items. The styles only change the background colors of the enclosed text, the font colors are determined by the default MediaWiki render according to whether the text is a hypertext link to other pages. The two types of styles are:

1. "nohighlighting" class defines transparent background, which is used to show the highlight-able items when the highlighting is disabled by the user;

2. A list of 20 classes are defined, each with a different background color recognized by name. The colors are carefully selected from the CSS colors supported by

all major browsers. They are arranged in a way that similar colors have small possibilities of being shown together on the page. The names of the classes are in the form of "highlighting" followed by a sequence number starting from 0, i.e. "highlighting0" to "highlighting19".

These classes are defined in a CSS file called highlighting.css and are automatically loaded using the CSS MediaWiki extension.

4.2.3 Additional JavaScript Functions

In order to give users more control over the adaptation process, some JavaScript functions are used for dynamically altering page content and saving necessary data.

Function js_enable() and js_disable() are the major functions that hook up with the onClick() event of the "active" and "inactive" radio buttons in the highlighting options control respectively. They enable or disable the highlighting of a object types on the page according to user action. Both functions take two parameters - a sequence number of the object type in the available types, which is used to name the highlighting items in their id, and the object type name, which is used to store user's highlighting preferences in the cookie.

A number of supporting JavaScript functions are also used:

1. js_cookieEnableTag() and js_cookieDisableTag() functions are where the saving Cookie action are actually executed;

2. getElementsByID() function is an extension to Document.getElementById() function that gets multiple elements with same id in the Document Object Model (DOM);

3. in_array() function checks if an value is in a JavaScript array and is used when saving cookie.

All the JavaScript functions are saved in a file called tags.js and is included by using the JavaScript MediaWiki extension.

4.3 Knowledge Repository

The knowledge repository of SEOW acts as a gateway that other components can use to query linked data. It is implemented as a REST web service. The web service is coded in Java and hosted on a GlassFish application server version 3.0.1[4] on the same machine that hosts the MediaWiki instance. It is a lightweight web service without database layer, and can be deployed on any machine that can connect to the chosen SPARQL endpoint.

The web service takes two parameters in the form of URI, one representing the resource of the page topic, for example "http://dbpedia.org/resource/British_Airways" for the page topic "British Airways"; the other represents the related object type, for example "http:// dbpedia.org/ontology/Airline" for the object type "Airline". The web service then constructs a SPARQL query that looks for all display-able objects semantically related to the topic resource and in the received type. Figure 4.5 shows the SPARQL query constructed to search for "Airline" objects related to "British Airways".

Because DBpedia URIs are used extensively in SEOW to represent resources and object types, it is natural to use DBpedia's SPARQL endpoint for querying linked

[4]https://glassfish.dev.java.net/downloads/3.0.1-final.html

```
PREFIX dbpediao:<http://dbpedia.org/ontology/>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?predicate ?object WHERE
{
  {
    {
      <http://dbpedia.org/resource/British_Airways> ?predicate ?object
          FILTER(!regex (?object, 'Category:')) .
      {
        {
          ?object rdfs:label ?x
              FILTER langMatches(lang(?x),'en')
        } UNION {
          ?object foaf:name ?y
              FILTER langMatches(lang(?y),'en')
        }
      } .
      ?object rdf:type <http://dbpedia.org/ontology/Airline> .
    }
  } UNION {
    {
      ?object ?predicate <http://dbpedia.org/resource/British_Airways>
          FILTER(!regex (?object, 'Category:')) .
      {
        {
          ?object rdfs:label ?x
              FILTER langMatches(lang(?x), 'en')
        } UNION {
          ?object foaf:name ?y
              FILTER langMatches(lang(?y), 'en')
        }
      } .
      ?object rdf:type <http://dbpedia.org/ontology/Airline> .
    }
  }
}
```

Figure 4.5: Sample knowledge respository web service SPARQL query

data.

The web service uses Jena[5] SPARQL query engine to send the constructed SPARQL query to the DBpedia SPARQL endpoint at "http://dbpedia.org/sparql", and saves the response in a ResourceRelationships object. A ResourceRelationships object contains a list of semantic relationships and the displaying names of the related resource in each relationship retrieved by a separate SPARQL query. The ResourceRelationships object is then converted to a JSON string and is used as the response of the web service to be sent back to the requesting component. Figure 4.6 shows the response sent back for the "British Airways" example.


```
{
  "0": {
    "subject": "http:\/\/dbpedia.org\/resource\/British_Airways",
    "predicate": "http:\/\/dbpedia.org\/property\/parent",
    "object": "http:\/\/dbpedia.org\/resource\/Go_Fly",
    "displayNames": {
      "0": "Go Fly"
    },
    "objectProperties": {}
  },
  "1": {
   "subject": "http:\/\/dbpedia.org\/resource\/British_Airways",
    "predicate": "http:\/\/dbpedia.org\/property\/parent",
    "object": "http:\/\/dbpedia.org\/resource\/CityFlyer_Express",
    "displayNames": {
      "0": "CityFlyer Express"
    },
    "objectProperties": {}
  }
}
```


Figure 4.6: Sample knowledge respository web service response

[5]http://jena.sourceforge.net/

4.4 Intelligence Component

The intelligence component of SEOW has the ability to make adaptation decisions on the request received from the SemanticTag MediaWiki plug-in, using simple adaptation rules and the knowledge obtained by requesting the knowledge repository web service. It is implemented as another REST web service, which is coded in Java and hosted on the same GlassFish application web server with the knowledge repository web service for development and testing use. It can, however, be deployed on any other application server and be hosted on a separate server or host with the knowledge repository web service.

In this section, I will start by introducing the adaptation rules used by the web service, then discuss how the web service applies the rules to make decisions.

4.4.1 Adaptation Rules

The SEOW system uses very simple adaptation rules to drive adaptation. A rule file contains a list of DBpedia URIs of object types that are relevant to a certain type of objects or topics, and is stored in XML format. Therefore, there is a separate rule file for objects of different types. In order to create the list, a domain expert who knows exactly what the most relevant object types are needs to find the corresponding URIs of the types from DBpedia. The list can also be intentionally altered according to the special needs of different systems. Figure 4.7 shows the rule for civil aviation industry.

As can be seen from the figure, the <rule> tag is the root tag that declares this file contains adaptation rule. Currently, <highlight> is the only child of <rule> because

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rule>
  <highlight>
    <resourceType>
      <uri>http://dbpedia.org/ontology/Airport</uri>
      <name>Airport</name>
    </resourceType>
    <resourceType>
      <uri>http://dbpedia.org/ontology/Company</uri>
      <name>Company</name>
    </resourceType>
    <resourceType>
      <uri>http://dbpedia.org/class/yago/AirlineAlliances</uri>
      <name>Alliance</name>
    </resourceType>
    <resourceType>
      <uri>http://dbpedia.org/ontology/Airline</uri>
      <name>Airline</name>
    </resourceType>
  </highlight>
</rule>
```

Figure 4.7: Sample adaptation rule

it is the only supported type of adaptation, but other types can be added to the same file if the rest of the system is capable of processing them. Within the <highlight> tag are the list of highlighting object types, each represented by a <resourceType> tag and consists of two parts - a <uri> tag that contains the URI of the object type, and a <name> tag that has the display name.

All the rule files are saved in a specific folder, the path to which is saved in an environment variable in the intelligence component web service.

4.4.2 Adaptation Process

The intelligence component web service in SEOW takes a single parameter - the page title. After a request is received, the following steps are taken to understand the request, make the adaptation decisions, and return the results:

1. Resolve page topic;
2. Load adaptation rules;
3. Construct adaptation results;
4. Return adaptation results to the content repository.

In this section, the four steps will be introduced one by one.

Resolve Page Topic

When a request is received, the web service first determines the URI of the page topic by sending a simple SPARQL query to the DBpedia SPARQL query endpoint. Figure 4.8 shows the query sent for "British Airways".

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?resource WHERE
{
  ?resource rdfs:label "British Airways"@en .
      FILTER (!regex (?resource, 'Category:'))
}
```

Figure 4.8: Sample SPARQL query for object URI

Sending the SPARQL query to the DBpedia endpoint using the Jena SPARQL query engine returns the URI of the page topic, for example "http://dbpedia.org/

resource/British_Airways", and this is the output of the first step.

Load Adaptation Rules

Proposed in the design but not implemented in SEOW, the web service should also get the object type of the base object, and refer to it to determine which set of rules to use in the adaptation. At present, however, only one set of rules is created in the SEOW system for civil aviation industry and all current MediaWiki pages are closely related to the topic. As a result, the selected rules are set to the civil aviation rules by default.

The output of this step is a list of object types represented in the form of URI.

Construct Adaptation Results

Using the topic resource URI resolved in the first step and the list of object type URIs fetched from the rule file in the second step, the web service can request knowledge repository web service for other related objects. A separate request is sent for each object type.

The web service uses ResourceRelationship object to manage all the semantic relationships with other resources that a resource has. A ResourceRelationship object may contain multiple ResourceRelationships objects, each of which represents a type of semantically related objects and is restored from the JSON decoded results of a knowledge repository web service response.

The output of this step is a ResourceRelationship object representing all the semantic relationships that the topic object has.

Return Adaptation Results

In this step, all the ResourceRelationships objects in the ResourceRelationship object are converted in JSON and stored in a string. The string is used as the web service response and sent back to the SemanticTag MediaWiki plug-in. A sample response can be found in figure 4.2.

4.5 Summary

In this chapter, I introduced the SEOW system which implements the design proposed in chapter 3 using off-the-shelf technologies and resources. The SEOW system meets most of the requirements of the desired system, especially achieves the separation of the concerns of content, knowledge and intelligence by decoupling these sub-systems and using only open formats for the communications between them.

In the next chapter, the performance of the design will be evaluated by letting some users to use the SEOW system and gathering usage data and feedback.

Chapter 5

Evaluation

5.1 Evaluation Goals

This dissertation asks to what extent can a third party adaptation service using public knowledge be used to adapt content on open WCMS. This can be broken down into the design considerations around the separation of the concerns of content, knowledge and intelligence. With the separation, there is concern that users may have a broken or disjoint experience while using the system. Therefore, assessing the usability and utility of the system is essentially important in order to engage the users' experience and potential to use the system. Moreover, I looked at the completeness and accuracy of the adaptation performed, which consists to what extent does each of component accomplish their designated missions. This evaluation describes both the user based usability trial and the completeness, accuracy and performance of the adaptation.

The usability evaluation has the following goals:

- Verify that the adaptation service is easy to learn and use;

- Confirm the adaptation process does not distract normal MediaWiki usage;
- Examine how the adaptation results affect user activities on MediaWiki.

The completeness and accuracy evaluation focuses on the following aspects:

- Evaluate the completeness and accuracy of the adaptation results;
- Evaluate other aspects in the performance of the adaptation service such as response speed and message payload.

The setup and process of the user evaluation are introduced in the next section. This is followed by the usability evaluation and the completeness and accuracy evaluation in section 5.3 and 5.4 respectively. Section 5.5 concludes the evaluation results.

5.2 Evaluation Process

For the evaluation of SEOW, a total of 52 distinct pages about civil aviation topics, including airlines, airports, aeroplanes, and their manufacturers, are created in the MediaWiki. The content of the pages are copied from their corresponding Wikipedia pages with slight modifications to hide irrelevant content and links. The correctness of most content is verified from their original sources.

Six tasks are prepared about the created content, the solutions to which can be found using no more than the provided pages. The tasks are designed so that they can be resolved by browsing a single page or navigating through a number of interlinked pages. The tasks are divided into three groups, and the questions in the same group require similar amount of effort to be solved. Table 5.1 shows the designed tasks.

ID	Group	Task
A	I	List all airlines that ever operated Concorde.
B	I	Find the airline that received the delivery of the first Boeing 717 aircraft.
C	II	Find all hub airports of British Airways, and for each airport, find other airlines that also use it as a hub.
D	II	List the founding members of the airline alliance that Lufthansa is in.
E	III	Among the airlines that belong to the parent company of City-Jet, find the one with the biggest fleet size.
F	III	Among all the airlines that use Dublin Airport as hub, find the one with the smallest fleet size.

Table 5.1: Tasks in SEOW user tests

A group of eight volunteers, five male and three female, with mixed Wikipedia experience and knowledge level about civil aviation are recruited via invitation emails to participate the user tests. The participants are first informed about the nature of the tasks and instructed on how to use the highlighting controls. They are then asked to finish the six tasks in a particular order. For the first three tasks, which include one task from each task group, users are given a clean MediaWiki instance without the adaptation service; for the last three tasks, which also contain one task from each group, the highlighting options are provided and the users are encouraged to use them to assist finishing the tasks.

The order of the tasks are decided as follows:

1. Pick one task from each of the three groups, and make them the first three tasks in the same order as their group numbers;

2. Make the remaining tasks as the last three in the same order as their group numbers.

For example, the first user gets the set of tasks in the order of ACEBDF, where

the adaptation options are only available for task B, D, and F. The second user may then get them in the order of ACFBDE, where task E and F, which are in the same group and similar complexity levels, are swapped.

There are exactly eight distinct orders of tasks, therefore, effectively every participant finished the same six tasks in different order, and each task was attempted exact four times with and without the assistance of the adaptation system respectively.

Each user is asked to fill in a questionnaire while they perform the tasks. They are asked to provide information about their familiarity of MediaWiki and civil aviation before answering any question. Users are also required to give answers to the tasks, which are used as a measure of the adaptation performance. After finishing each of the last three tasks, questions are asked to what degree the adaptation service helps in answering the question. At the end of the questionnaire, there are also some System Usability Scale (SUS) questions in order to evaluate the usability of the adaptation service. The questionnaire can be found in Appendix A.

While the users are approaching the tasks, a screen video capture program - Camtasia Studio version 7.0.1[1] - is used to record users' interactions with the MediaWiki. Information such as time and steps taken to answer each question are gathered from analysing the recording.

5.3 Usability Evaluation

In this section, the evaluation data from two sources - the system usage data and the questionnaire feedback - are stated and analysed separately.

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

System usage data includes nature of the participants' actions on the system and the characteristics of the solutions the participants give. It is the objective measure of the adaptation service performance. Meanwhile, the answers to the questionnaire questions are the users' subjective view of how well the adaptation service performs. Combining the views from the two points can generate a more general assessment of the value added by the adaptation service.

5.3.1 System Usage Data

The changes of the following measures that the adaptation service brings to the system are taken into consideration and analysed:

- Accuracy of the answers;
- Time taken to give answers.

Answers Accuracy

Figure 5.1 shows each participant's accuracy rate in answering the questions with and without the highlighting options. Among the eight participants, two managed to answer all task questions correctly. The highlighting options' effects on the accuracy of the other six participants vary: four users gave more correct questions with the assistance of the adaptation service, one saw no difference, while the other one got more correct answers without the highlighting options. Therefore, the majority of the participants performed at least as good with the highlighting options in terms of accuracy.

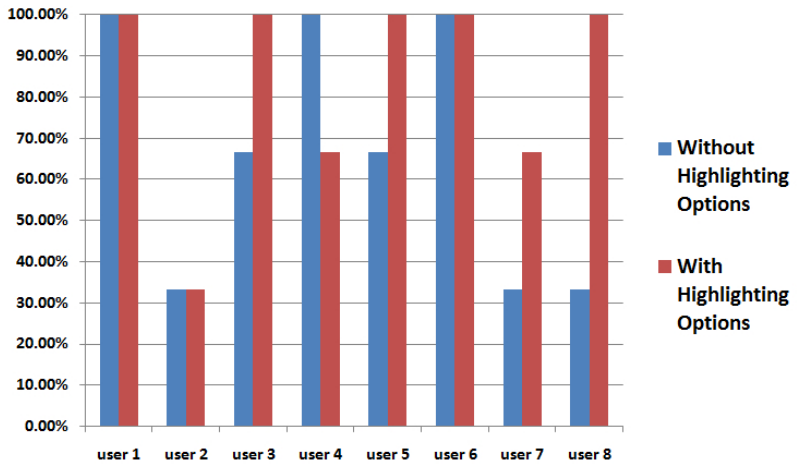

Figure 5.1: Comparison of answer accuracy rate with and without adaptation service by user

Figure 5.2 shows the comparison of answer accuracy rate from all users with and without the adaptation service. The accuracy rate increased dramatically by 16.66 percent, which effectively means the adaptation service managed to half the amount of incorrect answers on same set of tasks.

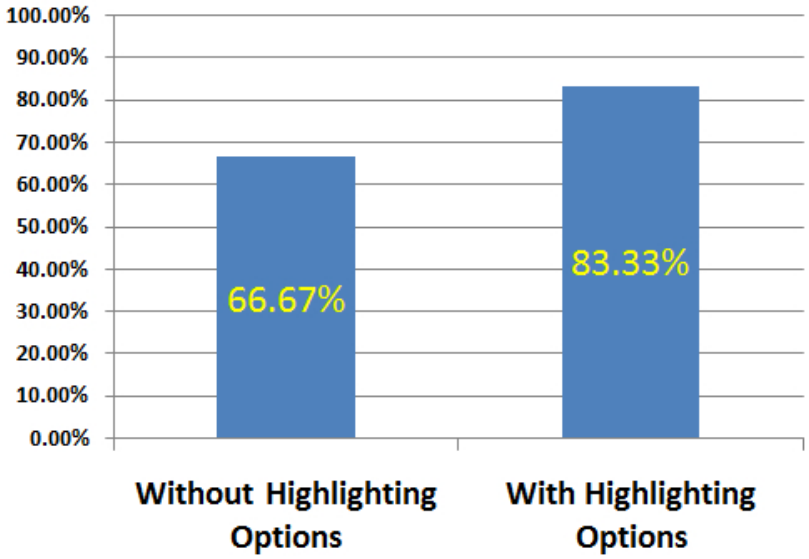

Figure 5.2: Comparison of answer accuracy rate from all users with and without the adaptation service

The adaptation service was seen extremely efficient in answering the question asked in one particular task. For task F, which is "Among all the airlines that use Dublin Airport, find the airline that has the smallest fleet size", none of the four participants was able to give the right answer without using the highlighting options. On the contrast, when assisted by the adaptation service, three out of four users answered the question correctly.

Time Used To Answer Questions

Figure 5.3 indicates the differences in the average time each user took to answer each question with and without the adaptation options. From the figure, the eight participants reacted differently to the addition of the highlighting options in the WCMS in terms of time used to answer the questions. The adaptation service shortened the average time of answering questions for three participants, two of which only used half the time as they were not given the highlighting options. Unexpectedly, the other five participants took longer to finish tasks using the highlighting options. A possible reason for this is the participants may need time to get used to the highlighting controls and many times, they were puzzled by why their expecting highlighting items were not marked by the adaptation service.

Figure 5.4 presents the differences in the average time the users took to finish each task before and after the adaptation service is available. From the figure, the average time taken to finish most tasks are not affected by the introduction of the adaptation service in a particular pattern. The only significant exception is task B, which has the question "Find the airline that received the delivery of the first Boeing 717 aircraft". The adaptation service is extremely efficient in this case because finding the answer

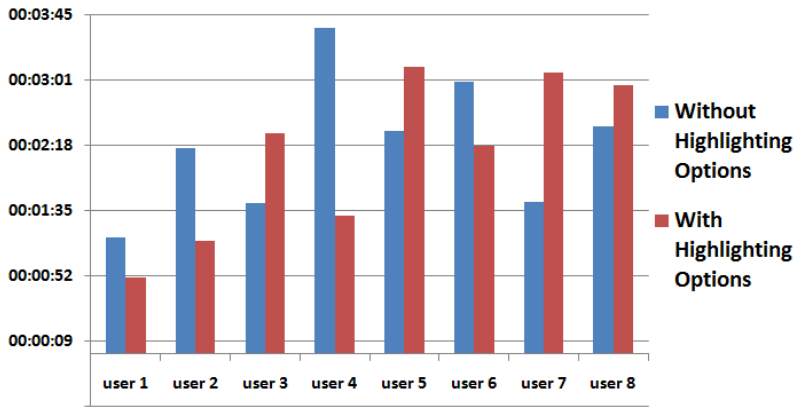

Figure 5.3: Comparison of average time used to answer each question with and without the adpatation options by user

without highlighting options requires users to read a number of long paragraphs, but with the adaptation service, the exact answer to the question is among the only few highlight-able words on the page.

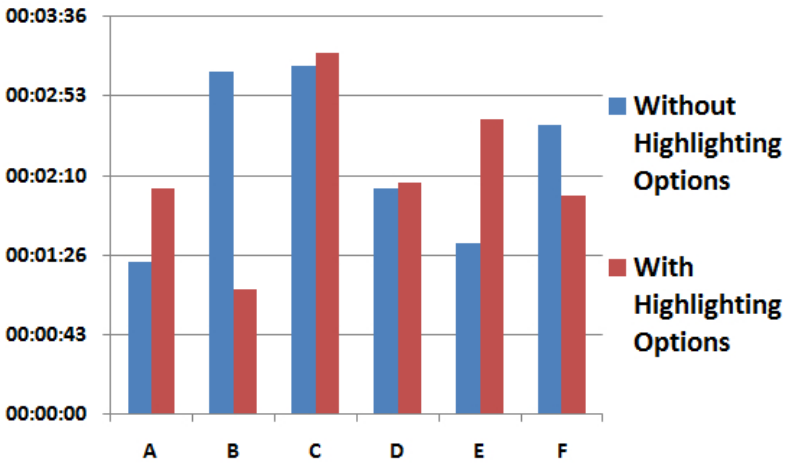

Figure 5.4: Comparison of average time used to answer each question with and without the adaptation service

Summary

In summary, the adaptation service generally had positive effects in assisting the participants to finish the tasks. Although it did not reduce the time taken to answer some questions, the participants were able to find solutions to the tasks more easily with the assistance of the adaptation service.

5.3.2 Questionnaire Answers

In this subsection, the answers that the participants gave in the questionnaire are analysed. Answers to the following two sets of questions are considered:

- Questions after finishing each task;
- SUS questions.

Questions After Each Task

The purpose of asking task specific questions after users finish each task is to determine in what circumstances and to what extent is the adaptive service helpful from users' points of view.

Figure 5.5 and 5.6 shows the participants' ratings on how helpful the highlighting options were in answering the questions in the tasks and their distribution respectively. The results show that the highlighting options effectively helped in solving 75 percent of questions, compared to in only 8 percent of cases users considered it unhelpful.

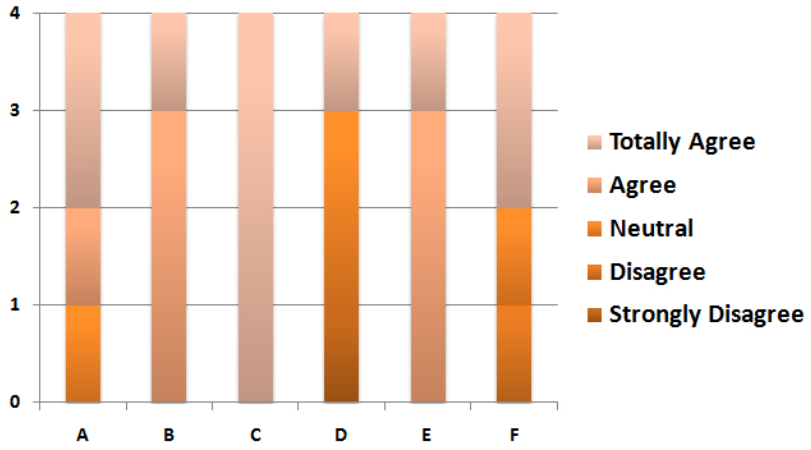

Figure 5.5: Participants' ratings on the helpfulness of the adaptation service in finishing the tasks

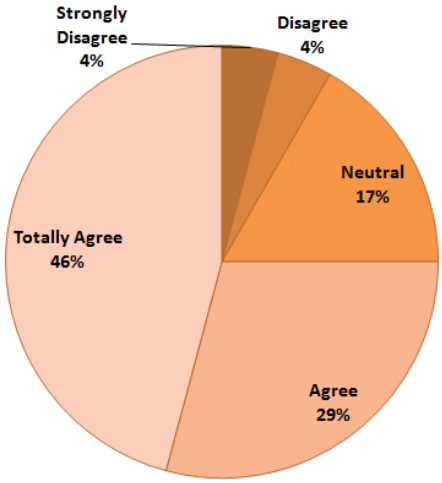

Figure 5.6: The distribution of the participants' ratings on the helpfulness of the adaptation service in finishing the tasks

SUS Questions

A typical SUS test was carried out at the end of the user tests after the participants finished all the tasks, in which the adaptation service received very positive feedback. According to the answers, all the participating users agree that the adaptation functions in the user tests are easy to learn, easy to use, and highly usable. All par-

ticipants also find the highlighting features intuitive to use while not intruding their normal MediaWiki experience.

There is an open question on the questionnaire about what can be improved in SEOW. The participants' answers to this question give some interesting insights of what the users' ideal adaptation service looks like. Four out of the eight users mentioned that they would like to have the adaptation applied faster, and after being explained how the the content is adapted, most of them understand that time is a trade-off of adaptable content. Although all participants agreed that the current background colors used for the seven highlight-able object types can easily be told apart, two users showed their concerns about the difficulty of distinguishing more background colors if the amount of highlight-able object types grows. Instead of finding more distinguishable colors, the problem might be solved by using a different types of highlighting, as suggested by a participant.

5.4 Completeness, Accuracy and Performance Evaluation

The completeness and accuracy of the adaptation is evaluated in two ways. The participants were asked for their opinions on the adaptation results for every task that the adaptation options were available. The results gave a subjective view of the completeness and accuracy. Moreover, the log files of the various services in the system were also analysed for performance measures.

5.4.1 User Feedback

Figure 5.7 presents the distribution of the participants' ratings on the completeness and accuracy of the highlighted items. This is an important measurement of the quality of the adaptation results. According to the statistics, the highlighting results are considered accurate and complete in over half of the cases. Among the negative feedback, most users mentioned the incompleteness of the adaptation results. The reasons for this are discussed in subsection 5.4.2.

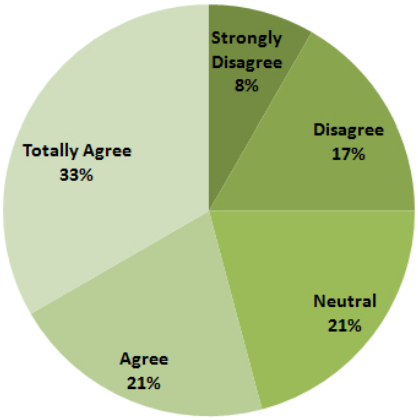

Figure 5.7: The distribution of the participants' ratings on the completeness and accuracy of the highlighted items

5.4.2 System Measurements

An analysis of the adaptation decisions indicates that most adaptation decisions made by the intelligence component has been accurate, thus users can completely trust the adaptation results. However, the completeness is not as good because many items that should be highlighted are not included in the adaptation results.

For example, when the page "Dublin Airport" is loaded with the adaptation

service, a number of airlines, including “Aer Lingus”, “Ryanair”, “CityJet”, and “Aer Lingus Regional” are marked as highlight-able because they all have “http://dbpedia.org/ontology/hubAirport” relationship with the airport, which indicates they use Dublin Airport as hub airport. However, another airline - “Aer Arann” - is not marked as highlight-able because such information is not included in the DBpedia dataset in the linking open data project.

This is a current shortcoming of the linked data resources rather than a flaw of the design. Despite the enormous size of datasets in the linking open data project, the incompleteness of information is a nature of linked data. This problem may become less significant as the continuous growth of the datasets and the introduction of new datasets, but will not vanish.

Other system measurements are also evaluated. An analysis on the recorded video reveals that the average loading time for an adaptable MediaWiki page during all the user tests is just under six seconds, compared to less than two seconds on a clean MediaWiki installation. An analysis on the server log shows that an average response from the intelligence component web service has the size of eight kilobytes while an average response from the knowledge repository web service is about one kilobyte.

The response time is notably long but acceptable in most cases according to the users. The response size can hardly be reduced because it is already the adaptation results in its smallest possible format. A concern about the response size is that if the knowledge sources such as DBpedia dataset grow indefinitely, the response of the knowledge repository can potentially explode as well, so does the response of the intelligence component web service. This can potentially harm the system performance somehow, but to what extent the performance will suffer is difficult to

predict.

5.5 Summary

Based on the analysis of both the participants' evaluation results and the system's performance, the following findings are concluded:

- To the extent that linked data is complete and accurate, the adaptation service successfully gives appropriate results;
- To the extent that adaptation rules can be enhanced and modified with more intelligent rules, the adaptation service successfully gives appropriate results;
- To the extent that the answers to the questions that users are asked to answer are included in the content, the adaptation service works;
- Even with limited content, very simple rules, and incomplete linked data knowledge, the adaptation service in SEOW is still undoubtedly usable;
- The user experience on the adaptation service is neither invasive nor intrusive, and intuitive;
- There is great extra value being added to the MediaWiki by the adaptation service.

The next chapter will conclude this dissertation. An overview of the potential of the proposed framework, as well as the possible future works will be included.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This dissertation sought to answer the question to what extent can a third party adaptation service using knowledge available in the public domain be used to adapt the content of open WCMS. Specifically, it looked at the extensibility of various WCMS, the feasibility of semantic web technologies as knowledge source, as well as mainstream adaptation approaches, methods and techniques.

The dissertation design employed an approach from an existing adaptive system which features the separation of the concerns of content, knowledge and intelligence, and locates them as the three core components respectively. The design was implemented using SOA approach in SEOW adaptable system, which consists of a set of services and a front-end using MediaWiki. After being fitted with some content from Wikipedia and a set of simple rules, the system was evaluated from two aspects. On one hand, the usability evaluation showed that the adaptation service was user-

friendly and highly usable; on the other hand, the accuracy evaluation showed that although the current adaptation results are mostly correct, it is still incomplete due to the issues of linked data.

The most significant advantage of the designed approach is that each of the three core components is independently extensible without affecting the other services. Content authors can add new rules indefinitely; knowledge experts can improve adaptation rules indefinitely; and the data in linked data can grow indefinitely. Therefore, the architecture has great potential.

With regard to the research question, the system shows that a third party adaptation service using public knowledge can successfully adapt content of open WCMS. Moreover, the experience that the adaptation service provides to the users are valuable. It is also noticed that the modular design of the system, which ensures that each component of the system is independently extensible, has great potential.

6.2 Future Work

Some interesting potential future work of the dissertation are discussed in this section.

Apply Adaptation on Alternative WCMS

The only requirement that a WCMS needs to meet to be used as the platform of the content repository component in the system design is that the appearance of its content can be modified by an extension or plug-in. Besides MediaWiki, most current mainstream WCMS meet the requirement thus are potential platforms of the content repository. Developing the corresponding extension or plug-in for these

WCMS so that they can use the knowledge repository and the intelligence component to perform adaptation makes an interesting future work.

Use Alternative Knowledge Source

DBpedia - the knowledge source being used in this project - is only one of the hundreds of datasets in the Linking Open Data project. There are many other datasets specialising on certain areas that also provide SPARQL endpoints. Using the alternative datasets as knowledge source may generate different adaptation results, and is also to be explored.

More Intelligent Rules

The rules that SEOW can process are in very simple XML format and can only process one type of semantic relationships - object type. It would be desired for the system to be able to process more sophisticated single rules and combinations of multiple rules.

Introducing User Modeling Service

Through users' interaction with the system, a lot of information about the users, such as their interested areas and browsing styles, can be obtained. Allowing a user modeling service to collect such data gives the possibility to re-use such information to drive adaptation. Therefore, instead of providing only adaptable type of adaptation, the system will use a combination of adaptive and adaptable approaches by making adaptation decisions based on the user model beforehand. The feasibility of the user modeling service in the existing system is to be studied. The additional value of the

adaptation driven by the user model is also to be evaluated.

References

[1] How does mediawiki work?, Oct. 2009. Retrieved on 24 Mar, 2010 from http://www.mediawiki.org/wiki/How_does_MediaWiki_work.

[2] Help:introduction to semantic mediawiki, Jan. 2010. Retrieved on 24 Mar, 2010 from http://semantic-mediawiki.org/wiki/Help:Introduction_to_Semantic_MediaWiki.

[3] B. Adida, M. Birbeck, S. McCarron, and S. Pemberton. Rdfa in xhtml: Syntax and processing, Oct. 2008. Retrieved on 24 Mar, 2010 from http://www.w3.org/TR/rdfa-syntax/.

[4] D. Beckett. Rdf/xml syntax specification (revised), Feb. 2004. Retrieved on 04 Apr, 2010 from http://www.w3.org/TR/rdf-syntax-grammar/.

[5] A. Bendiken. Resource description framework (rdf), Oct. 2007. Retrieved on 24 Mar, 2010 from http://drupal.org/project/rdf.

[6] A. Bendiken. Rdf api for drupal 6.x, Feb. 2008. Retrieved on 24 Mar, 2010 from http://groups.drupal.org/node/8930.

[7] T. Berners-Lee. Linked data - design issues, 2006. Retrieved on 19 Mar, 2010 from http://www.w3.org/DesignIssues/LinkedData.html.

[8] T. Berners-Lee. Sir tim berners-lee talks with talis about the semantic web, Feb. 2008. Retrieved on 27 Mar, 2010 from http://talis-podcasts.s3.amazonaws.com/twt20080207_TimBL.html.

[9] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284:28–37, May 2001.

[10] C. Bizer. About dbpedia, Mar. 2010. Retrieved on 27 Mar, 2010 from http://wiki.dbpedia.org/About.

[11] C. Bizer, R. Cyganiak, and T. Heath. How to publish linked data on the web, July 2007. Retrieved on 19 Mar, 2010 from http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/.

[12] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web & Infomation Systems*, 5(3):1–22, 2009.

[13] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia : a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, Sept. 2009.

[14] U. Bojars and J. G. Breslin. Sioc core ontology specification, Jan. 2010. Retrieved on 19 Mar, 2010 from http://rdfs.org/sioc/spec/.

[15] D. Brickley and R. Guha. Rdf vocabulary description language 1.0: Rdf schema, Feb. 2004. Retrived on 04 Apr, 2010 from `http://www.w3.org/TR/rdf-schema/`.

[16] D. Brickley and L. Miller. Foaf vocabulary specification 0.97, Jan. 2010. Retrieved on 19 Mar, 2010 from `http://xmlns.com/foaf/spec/`.

[17] P. Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, 6(2–3):87–129, 1996.

[18] P. Brusilovsky. Knowledgetree: A distributed architecture for adaptive e-learning. In *WWW2004: Proceedings of the 13th international World Wide Web conference*, pages 104–113, 2004.

[19] P. Brusilovsky and H. Nijhavan. A framework for adaptive e-learning based on distributed re-usable learning activities. In *E-Learn 2002: Proceedings of 7th World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education*, pages 154–161. AACE, Oct. 2002.

[20] P. Brusilovsky, E. Schwarz, and G. Weber. Elm-art: An intelligent tutoring system on world wide web. In *Third International Conference on Intelligent Tutoring Systems*, volume 1086 of *Lecture Notes in Computer Science*, pages 261–269. Pringer Verlag, 1996.

[21] P. Brusilovsky, V. P. Wade, and O. Conlan. *From Learning Objects to Adaptive Content Services for E-Learning*, chapter 14, pages 243–261. IGI Global, 2007.

[22] D. Buytaert. Drupal, the semantic web and search, Oct. 2008. Retrieved on 24 Mar, 2010 from `http://buytaert.net/drupal-the-semantic-web-and-search`.

[23] A. B.V. *User Guide for Sesame 2.3*, 2010. Retrieved on 18 Mar, 2010 from `http://www.openrdf.org/doc/sesame2/2.3.1/users/userguide.html`.

[24] O. Conlan, V. Wade, C. Bruen, and M. Gargan. Multi-model, metadata driven approach to adaptive hypermedia services for personalized elearning. In *Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 100–111. Springer-Verlag, 2002.

[25] O. Conlan and V. P. Wade. *Evaluation of APeLS - An Adaptive eLearning Service Based on the Multi-model, Metadata-Driven Approach*, volume 3137 of *Lecture Notes in Computer Science*, pages 291–295. Springer Berlin / Heidelberg, 2004.

[26] W. W. W. Consortium. Currently alive sparql endpoints. Retrieved on 27 Aug, 2010 from `http://esw.w3.org/SparqlEndpoints`.

[27] R. Cyganiak. About the linking open data dataset cloud, July 2007. Retrieved on 22 Mar, 2010 from `http://richard.cyganiak.de/2007/10/lod/`.

[28] P. De Bra, A. Aerts, B. Berden, B. de Lange, B. Rousseau, T. Santic, D. Smits, and N. Stash. Aha! the adaptive hypermedia architecture. In *HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 81–84, New York, NY, USA, 2003. ACM.

[29] P. De Bra, D. Smits, and N. Stash. *Creating and Delivering Adaptive Courses with AHA!*, volume 4227 of *Lecture Notes in Computer Science*, pages 21–33. Springer Berlin / Heidelberg, 2006.

[30] M. Dean and G. Schreiber. Owl web ontology language reference, Feb. 2004. Retrieved on 04 Apr, 2010 from `http://www.w3.org/TR/owl-ref/`.

[31] H. Dhraief, W. Nejdl, B. Wolf, and M. Wolpers. Open learning repositories and metadata modeling. In *International Semantic Web Working Symposium (SWWS)*, pages 495–514, 2001.

[32] P. Dolog, N. Henze, W. Nejdl, and M. Sintek. Personalization in distributed e-learning environments. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 170–179, New York, NY, USA, 2004. ACM.

[33] P. Dolog and W. Nejdl. Challenges and benefits of the semantic web for user modelling. In *AH2003: Proceedings of Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems*, Twelfth International World Wide Web Conference, 2003.

[34] T. Finin and D. Drager. Gums: a general user modeling system. In *HLT '86: Proceedings of the workshop on Strategic computing natural language*, pages 224–230, Morristown, NJ, USA, 1986. Association for Computational Linguistics.

[35] A. Jentzsch. Dbpedia 3.4 downloads, Feb. 2010. Retrieved on 27 Mar, 2010 from `http://wiki.dbpedia.org/Downloads34`.

[36] V. Kaptelinin. Item recognition in menu selection: the effect of practice. In *NTERACT '93 and CHI '93: Conference Companion on Human Factors in Computing Systems*, pages 183–184. ACM, 1993.

[37] A. Kobsa. Modeling the user's conceptual knowledge in bgp-ms, a user modeling shell system. *Computational Intelligence*, 6:193–208, 1990.

[38] A. Kobsa. Generic user modeling systems. *User Model. User-Adapt. Interact.*, 11(1-2):49–63, 2001.

[39] M. Krötzsch, D. Vrandecic, and M. Völkel. Semantic mediawiki. In I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 935–942. Springer, 2006.

[40] O. Lassila and R. R. Swick. Resource description framework (rdf) model and syntax specification, Feb. 1999. Retrieved on 04 Apr, 2010 from `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/`.

[41] L. Luinenburg, S. Jansen, J. Souer, I. van de Weerd, and S. Brinkkemper. Designing web content management systems using the method association approach. In *MDWE 2008: Proceedings of the 4th International Workshop on Model-Driven Web Engineering*, pages 106–120, Sept. 2008.

[42] A. Miles and S. Bechhofer. Skos simple knowledge organization system reference, Aug. 2009. Retrieved on 19 Mar, 2010 from `http://www.w3.org/TR/2009/REC-skos-reference-20090818/`.

[43] A. Mullery, M. Besson, M. Campolargo, R. Gobbi, and R. Reed, editors. *Personalized Hypermedia Information Provision through Adaptive and Adaptable System Features: User Modelling, Privacy and Security Issues*, volume issue 1328 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, May 1997.

[44] Open Source Matters, Inc. What is joomla? Retrieved on 22 Mar, 2010 from `http://www.joomla.org/about-joomla.html`.

[45] Open Source Matters, Inc. Joomla! 1.5 api reference, Nov. 2009. Retrieved on 22 Mar, 2010 from `http://api.joomla.org/`.

[46] S. Pathak and P. Brusilovsky. Assessing student programming knowledge with web-based dynamic parameterized quizzes. In *ED-MEDIA2002: Proceeding of World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 1548–1553, 2002.

[47] C. R. Perrault, J. F. Allen, and P. R. Cohen. Speech acts as a basis for understanding dialogue coherence. In *Proceedings of the 1978 workshop on Theoretical issues in natural language processing*, pages 125–132, Morristown, NJ, USA, 1978. Association for Computational Linguistics.

[48] E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf, Jan. 2008. Retrieved on 18 Mar, 2010 from `http://www.w3.org/TR/rdf-sparql-query/`.

[49] Semantic Web Education and Outreach Interest Group. Linkingopendata, Mar. 2010. Retrieved on 22 Mar, 2010 from `http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`.

[50] R. Shreves. Open source cms market share, 2008. Retrieved on 16 Mar, 2010 from http://
www.waterandstone.com/downloads/2008OpenSourceCMSMarketSurvey.pdf.

[51] the Drupal community. About drupal, July 2009. Retrieved on 16 Mar, 2010 from http://drupal.org/about.

[52] W3C OWL Working Group. Owl 2 web ontology language document overview, Oct. 2009. Retrieved on 04 Apr, 2010 from
http://www.w3.org/TR/owl2-overview/.

Appendix A

SEOW Evaluation Questionnaire

In the real evaluation, participants answer the questions and fill in the questionnaire in electronic format on a form provided by Google Docs. This appendix lists all questions in the questionnaire.

An introduction is given to the participants before they approach any task, followed by five questions about their previous experience in WCMS and civil aviation industry. For each task, the participants are asked to give a answer; for the last three tasks, additional questions about the contribution of the adaptation service are also asked. After the participants finish all the tasks, they will be asked to answer a number of SUS questions.

Introduction

Thank you for attending this evaluation user test.

In this user test, you will be using a Wiki which is similar to Wikipedia to perform six tasks, each of which asks a question. When answering the last three questions, you will be able to use SEOW (stands for Semantically Enhanced Open Wiki) - a tool that gives you a list of type switches. When a switch is on, all objects of that type which the computer considers relevant to the page's main topic will be highlighted. For example, on a page talking about "UK", switching on "Place" will highlight "London", but "Tokyo" on the same page will not be highlighted because the computer thinks it is irrelevant. At the end of each task, you will be asked to give the answer to the question and perhaps to give feedback on how SEOW performed and helped you answering the question.

You are free to base your answer on your own knowledge, but please do not use Wikipedia.org or Google during the experiment. The Wiki you are provided has sufficient information to answer all the questions.

You have rights to refuse to answer any question in the questionnaire. You are also free to terminate the test at any time without penalty.

Section 1: Background Questions

Before starting the first task, I would like to ask some background information.

- How often do you use Wikipedia and other MediaWiki instances to seek information?

  Five grades rating from “Seldom ” to “Everyday ”.

- How often do you use other WCMS such as Drupal, Joomla! and WordPress etc to seek information?

  Five grades rating from “Seldom” to “Everyday ”.

- How often do you contribute content to WCMS such as Wikipedia, Drupal and Joomla?

  Five grades rating from “Seldom” to “Everyday ”.

- How often do you actively contribute in the development of WCMS?

  Five grades rating from “Seldom” to “Everyday ”.

- How much do you know about civil aviation stuff? Such as airlines, airports, aircraft etc.

  Five grades rating from “Hardly anything” to “Almost everything ”.

Section 2: Questions After Tasks

For the last three tasks, the questions are as follows.

- What is your answer to the task question?

  Open question.

- SEOW switches helped me finishing this task.

  Five grades rating from "Disagree" to "Agree ".

- Loading the Wiki page took too long for this task.

  Five grades rating from "Disagree" to "Agree ".

- The available type switches were relevant to this task.

  Five grades rating from "Disagree" to "Agree ".

- All items highlighted by SEOW were accurate and complete.

  Five grades rating from "Disagree" to "Agree ".

- Some potentially more useful type switches were not provided by SEOW for this task.

  Five grades rating from "Disagree" to "Agree ".

- Do you have any comments on this task?

  Open question.

Section 3: Final Questions

The following questions are asked after the participants finished all tasks.

- Learning how to use SEOW switches was complex.

  Five grades rating from “Disagree” to “Agree ”.

- SEOW switches were easy to use.

  Five grades rating from “Disagree” to “Agree ”.

- Highlighted items were easily found on the page.

  Five grades rating from “Disagree” to “Agree ”.

- When multiple items were highlighted, they were easily distinguishable.

  Five grades rating from “Disagree” to “Agree ”.

- I wish I could have more control over the available type switches.

  Five grades rating from “Disagree” to “Agree ”.

- SEOW switches helped me finding information on the Wiki.

  Five grades rating from “Disagree” to “Agree ”.

- I found the SEOW highlighting switches to be intrusive.

  Five grades rating from “Disagree” to “Agree ”.

- I found the SEOW highlighting switches intuitive to use.

  Five grades rating from “Disagree” to “Agree ”.

- In general, I’m concerned about personalisation on the Web due to privacy issues.

Five grades rating from “Disagree” to “Agree ”.

- What can be improved in SEOW?

  Open question.