

---

The Australian National University  
COMP 6703 eScience Project

Semantic Web for Museums  
Final Report

Version Date: June 21<sup>st</sup> 2006

---

Author	Content	Submitted to	Date
Junran Lei U4143677	COMP6703 eScience Project	Client/Technical Supervisor: Tom Worthington Academic Supervisor : Peter Strazdins	21/06/2006

## Table of Content

<b>TABLE OF CONTENT .....</b>	<b>3</b>
<b>ACKNOWLEDGE.....</b>	<b>6</b>
<b>ABSTRACT.....</b>	<b>7</b>
<b>GLOSSARY OF TERMS .....</b>	<b>8</b>
<b>1. INTRODUCTION.....</b>	<b>10</b>
1.1. MOTIVATION .....	10
1.2. PROJECT BACKGROUND AND OBJECTIVES.....	12
<b>2. REQUIREMENTS.....</b>	<b>13</b>
2.1. CHAPTER OVERVIEW .....	13
2.2. BUSINESS REQUIREMENTS.....	13
2.3. CLIENT REQUIREMENTS .....	13
2.4. USER REQUIREMENTS.....	14
2.5. POSSIBLE RESEARCH TOPICS.....	14
2.6. REQUIREMENTS PRIORITIES.....	14
<b>3. PROJECT PLAN .....</b>	<b>15</b>
3.1. CHAPTER OVERVIEW .....	15
3.2. DELIVERABLES .....	15
3.3. MILESTONES.....	15
3.4. LIMITS AND EXCLUSIONS .....	16
3.5. CONSTRAINTS AND ASSUMPTIONS.....	16
3.6. RESOURCE PLAN.....	16
3.6.1. <i>Technical Resources</i> .....	16
3.6.2. <i>Human Resource</i> .....	17
3.7. PROJECT SCHEDULE.....	17
3.7.1. <i>Work Breakdown Structure</i> .....	18
3.7.2. <i>Timetable</i> .....	18
3.8. RISK ANALYSIS .....	18
3.8.1. <i>Risk Assessment Form</i> .....	18
3.8.2. <i>Risk Response Matrix</i> .....	18
<b>4. DESIGN ANALYSIS AND MODELLING .....</b>	<b>20</b>
4.1. CHAPTER OVERVIEW .....	20
4.2. DATA MODEL.....	20
4.3. METADATA STANDARD .....	21
4.4. STORAGE METHODS .....	21
4.5. QUERY METHODS .....	21
4.6. MUSEUM BASED EXHIBITION AND VIRTUAL EXHIBITION PLANNING.....	22
4.7. SUB-DOMAIN ONTOLOGY .....	23

4.8.	MODEL EXAMPLES AND EXPLANATION .....	24
4.9.	ELEMENTS RELATIONSHIPS .....	26
4.10.	RDF VOCABULARY DESIGN .....	28
<b>5.</b>	<b>IMPLEMENTATION .....</b>	<b>30</b>
5.1.	CHAPTER OVERVIEW .....	30
5.2.	ARCHITECTURAL DESIGN .....	30
5.3.	STORAGE APPLICATION COMPONENT .....	31
5.3.1.	<i>Component Structure</i> .....	31
5.3.2.	<i>Single Entity Storage</i> .....	32
5.3.3.	<i>Batch Storage</i> .....	33
5.3.4.	<i>RSS</i> .....	34
5.3.5.	<i>Input Batch for Testing</i> .....	34
5.3.6.	<i>Remove Model for Testing</i> .....	34
5.4.	QUERY APPLICATION COMPONENT .....	34
5.4.1.	<i>Component Structure</i> .....	34
5.4.2.	<i>Keyword Search</i> .....	34
5.4.3.	<i>Relation Search</i> .....	35
5.4.4.	<i>Detail Search</i> .....	36
5.5.	INTERFACE MODEL .....	36
5.5.1.	<i>Component Structure</i> .....	36
5.5.2.	<i>Introduction</i> .....	37
5.5.3.	<i>TextTable</i> .....	38
5.5.4.	<i>Physical Object Storage</i> .....	40
5.5.5.	<i>Activity Storage</i> .....	40
5.5.6.	<i>Representation Storage</i> .....	41
5.5.7.	<i>Location Storage</i> .....	41
5.5.8.	<i>Batch Storage</i> .....	41
5.5.9.	<i>RSS Storage</i> .....	42
5.5.10.	<i>CenterRela</i> .....	42
5.5.11.	<i>Keywords Search</i> .....	44
5.5.12.	<i>Related Resource Search</i> .....	45
5.5.13.	<i>Detail</i> .....	45
5.5.14.	<i>Relationships Diagram</i> .....	46
5.5.15.	<i>Show RDF</i> .....	46
5.5.16.	<i>Print</i> .....	47
5.5.17.	<i>UserGuide</i> .....	47
5.5.18.	<i>Help</i> .....	47
<b>6.</b>	<b>TESTING.....</b>	<b>49</b>
6.1.	CHAPTER OVERVIEW .....	49
6.2.	TEST DESCRIPTION .....	49
6.3.	TEST ENVIRONMENT.....	49
6.4.	FUNCTIONAL TEST.....	49
6.5.	PERFORMANCE TEST .....	50

6.5.1.	<i>Storage Components</i>	51
6.5.2.	<i>Search Components</i>	53
6.6.	TEST ASSESSMENT AND RECOMMENDED IMPROVEMENTS	55
<b>7.</b>	<b>CONFIGURATION AND USER GUIDE</b>	<b>57</b>
7.1.	INSTALLATION AND CONFIGURATION	57
7.1.1.	<i>Tested Environment</i>	57
7.1.2.	<i>Server Installation</i>	57
7.1.3.	<i>Library Installation</i>	57
7.1.4.	<i>Test Installation</i>	58
7.2.	USER GUIDE	58
7.2.1.	<i>Structure</i>	58
7.2.2.	<i>Introduction</i>	58
7.2.3.	<i>Storage</i>	58
7.2.4.	<i>Search</i>	60
7.2.5.	<i>User Guide</i>	61
7.2.6.	<i>Help</i>	61
7.3.	HELP	62
<b>8.</b>	<b>CONCLUSIONS AND DISCUSSION</b>	<b>63</b>
<b>9.</b>	<b>FUTURE WORK</b>	<b>64</b>
<b>APPENDIX A.</b>	<b>MAPPING TABLE</b>	<b>65</b>
A.1.	ENTITY	65
A.2.	PHYSICAL OBJECT	65
A.3.	REPRESENTATION	67
A.4.	ACTIVITY	69
A.5.	LOCATION	72
A.6.	CONDITION ASSESSMENT	72
A.7.	COLLECTION	74
<b>APPENDIX.B.</b>	<b>SWM RDF SCHEMA</b>	<b>75</b>
<b>APPENDIX.C.</b>	<b>RSS EXAMPLE</b>	<b>79</b>
<b>APPENDIX.D.</b>	<b>TIMETABLE</b>	<b>80</b>
<b>REFERENCE</b>		<b>82</b>

## Acknowledge

I would like to thank my client and supervisor, Mr Tom Worthington, for guiding me to Semantic Web this interesting area and giving me so much support and inspiration.

I would like to thank my supervisor, Dr. Peter Strazdins, for teaching me professional skills and giving me so much feedback and encouragement.

I would also like to thank project advisor, Mr Chris Blackall, for his useful advice.

And thanks to all the support and help from my family and friends.

## Abstract

The current management methods and storage features of museums collections constrain the integration, interoperability and discovery of cultural heritage knowledge. The promising semantic web technology is proposed to solve the constraints and provide a better way for cultural heritage preservation and management. Several semantic web elements and technologies have been researched and chosen for the design and implementation of a semantic web system for museums. The system shows the benefits of relationships expression, data interoperability, integration and application independent, which could not be provided by traditional museums systems using text document or database record methods. The research and design results also show that the semantic web technology still have large potential benefits to offer.

**Keyword:** Semantic Web, Metadata, RDF, Ontology, CIDOC CRM, Dublin Core, Vocabulary, Museum, Collection, Cultural heritage

## Glossary of Terms

AMICO	Art Museum Image Consortium
CIDOC CRM	The International Committee for Documentation of the International Council of Museums Conceptual Reference Model
CIMI	Consortium for the Computer Interchange of Museum
ICOM-CIDOC	The International Committee for Documentation of the International Council of Museums
IFLA FRBR	International Federation of Library Associations and Institutions Functional Requirements for Bibliographic Records
Metadata	Metadata is structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use or manage an information resource. Metadata is often called data about data or information about information. (Hodge 2001:P3)
RDF	The Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web. (W3C RDF Primer)
RDF Schema	RDF's vocabulary description language, RDF Schema, is a semantic extension of RDF. It provides mechanisms for describing groups of related resources and the relationships between these resources. (W3C RDF Schema)
Ontology	In the technical view of ontological engineering, an ontology is the vocabulary for expressing the entities and relationships of a conceptual model for a general or particular domain, along with semantic constraints on that model that limit what that model means. Both the vocabulary and the semantic constraints are necessary in order to correlate that information model with the real-world domain it represents. (Daconta, Obrst and Smith 2003: Chap8)
OWL	The OWL Web Ontology Language is intended to provide a language that can be used to describe the classes and relations between them that are inherent in Web documents and applications. (W3C OWL Guide)
URI	Uniform Resource Identifier



W3C                      Word Wide Web Consortium

XML                      XML is not a language; it is actually a set of syntax rules for creating semantically rich markup languages in a particular domain. In other words, you apply XML to create new languages. (C. Daconta, et al 2003:Chap3)

## 1. Introduction

### 1.1. Motivation

With the quick development and wide application of web technology, many museums have built their websites to publish digital format of collections online. However, in this chapter we would discuss web technology has not been taken the most advantage to exploit and manage the cultural heritage preserved in museums. The storage features and the current management methods of the collections both constrain the knowledge exploitation and interoperability. And the emerging semantic web technology (W3C Semantic Web) would be proposed to solve the constraints and provide a better way for cultural heritage preservation and management.

The first constraint originates from the natural features of cultural collections. The contents of collections in museums have rich associations, which are equally important as the contents themselves. For example, the creator of painting A had other paintings with the same style, and this style actually originates from another artist, who drew painting B with the same topic... These types of relationships between the items or entities are even more valuable for domain experts or general public to explore the subjects. The innumerable associations could not be revealed to users under current storage methods.

To majority of museums, collections are preserved as isolated objects in individual museums. There are not interoperability and association management for the collections between museums or in the same museum. Even if the museums have archive systems to manage their collections, the systems might use different vocabulary to describe and index the collections contents, which is another constraint for interoperability. For example, one museum could describe author as author-name, while other might regard it as creator.

Semantic web, as an emerging and promising technology, could promote the information integration, knowledge exploitation and interoperability in museums.

As said by Tim Berners-Lee, the inventor of the Web and director of the World Wide Web Consortium (W3C), "The Semantic Web is the next generation of the World Wide Web". It provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries through giving information well defined and machine understandable meaning. (W3C Semantic Web)

Metadata (Hodge 2001:P3) and ontology (Daconta, Obrst and Smith 2003: Chap8) are two basic elements of current semantic web technology. In the metadata level, all data are described with a set of predefined vocabulary and syntax, then meaning of

information could be perceived by human and machine according the common reference schema. Information integration could be enhanced in the metadata level. However, the enhancement is not enough for association expression and interoperability. From the psychology analysis, there are mental models in our mind to hold the relation and rule of information. We could simulate the models acting the role of mental models to express the association and rule. Ontology is one of these models. There are many ways of writing down ontology, such as RDFS (W3C RDF Schema) and OWL (W3C OWL Guide).

W3C recommends the Resource Description Framework (RDF) (W3C RDF Primer) data to describe metadata. The resource described in RDF could be identified by URI (Uniform Resource Identifier). The statement about resource is combined of three elements, or triple, including subject, predicate and object. “Subject, in grammar, is the noun or noun phrase that is the doer of the action; predicate is the part of a sentence that modifies the subject and includes the verb phrase; object is a noun that is acted upon by the verb.” (Daconta, Obrst and Smith 2003: Chap5) A simple example of RDF Triple is listed below:

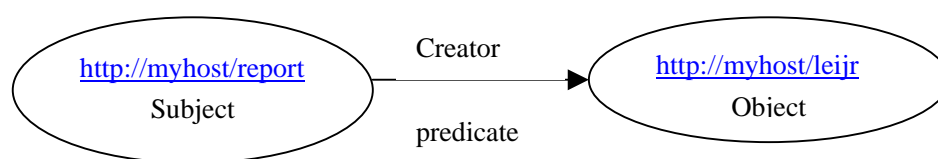


Figure 1.1 RDF Triple example (Drawn after W3C RDF Premier)

Using the RDF model above, the relationships and properties of resource are explicitly expressed. RDFS and OWL have the similar structure as RDF, which make the rich relationships and semantic network of semantic web application.

We have introduced the concept and basis elements of semantic web technology, metadata, ontology and RDF, but how could they benefit the application of museum? First of all the RDF Triple model is very suitable for the association management of museum collection. It defines a common reference frame for human and machine. Its feature of machine understandable makes the automatic inference and reasoning possible. The semantic application could infer the concerned information based on the associations embedded in metadata and the rules or relationships in ontology. Using this storage method, the collections are not longer isolated with each other. The associations could be revealed to user with great facility.

Secondly, ontology not only describes the rule and relationship in the domain, it also defines the domain vocabulary. Then information could be shared across different institutes using domain vocabulary, and even across different domain if mapping

between ontologies is constructed and applied. As discussed above, different museums might have various indexes for their collections. We could develop the mediator application to transfer the indexes or require the collections description in all developing applications comply with the same vocabulary to the domain ontology. Interoperability would be enhanced by these methods based on metadata and ontology.

## 1.2. Project Background and Objectives

The project originates from a workshop presented by Mr Tom Worthington for museums of the Pacific islands region. One of the recommendations from the workshop suggested building an on-line repository across the Pacific museums. The Semantic Web for Museum project aims to research the promising semantic technology for creating the knowledge management network or regional digital archive among museums to help the Museum Community preserve their distinct cultural heritage knowledge for the future. The current objective of the project is to develop an effective semantic web archive system for museums.

## 2. Requirements

### 2.1. Chapter Overview

There are no particular users that we can get requirements from. The requirements listed below are obtained from the client – Mr Tom Worthington and two reports: Report on a Workshop on the Use of Technology for Museums of the Pacific Islands Region 2005 (Worthington 2005)

Digital Heritage for South Pacific Museums Project Findings and Report (Yew 2005)

### 2.2. Business Requirements

Business Requirements are high level extraction of organization, project and users requirements. It should be used as guide to every activity throughout the project.

1. Promote collaboration and knowledge sharing between museums.
2. Create a resource and knowledge management network or a regional digital archive of materials among museums.
3. Help the Museum Community preserve their distinct cultural heritage knowledge for the future.

### 2.3. Client Requirements

Client requirements are the most important direction to the general objectives and deliverables of the project, and also the compulsory contents of the project.

1. How to present semantic data (meaningful to computer) to users (meaningful to user). Design a usable semantic web interface specifically for museums.
2. Design or select metadata suitable to museum situation.
3. Design or select RDF, Ontology, repository, and other components suitable to museum situation.
4. What tools and standards should be used? Select suitable tools and standards for the project context.
5. Produce an exercisable and effective online semantic archive system for museums.

For the time limitation of the project, it is more realistic to use existing metadata, RDF, Ontology standards and probably add customised contents to them. As for user interface, we could analyze the existing web interface of museums around the world and other semantic webs, and also apply some theories from user interface development area to our design. According to the analysis in 3.6.1 Technical Resource, we would develop application model using Java to communicate interface model, metadata model and repository; and implement user interface with JSP on the Tomcat platform. For further technical detail such as candidate languages and tools, design

and implementation issues, please refer to 3.6.1 Technical Resource.

## 2.4. User Requirements

There are just a few user requirements that we can get from the reports. However, they can still give useful direction and reference to system development.

The user requirements include providing an online digital archive system used by scholarly, non-expert users and general public and with the concern of limited technical staff, equipment and telecommunication links for some museums situation.

## 2.5. Possible Research Topics

1. Compare and choose repositories from Dspace, Fedora or other Repositories.
2. How to communicate and transfer data between decentralised organisations. (Webservice? Harvesting? )
3. How to store RDF Model, Document or DB?
4. How to interoperate with existing systems?
5. Can we incorporate existing technologies and components together to produce a usable system?

We would choose one to two topics from the above or other topics that come up during the development process. We would not fix the research topics as we do to the implementation topics, because the design and implementing process is actually a research process as well since there are so many uncertainties and possibilities in the semantic web technology area. In fact, designing and implementing is getting our research results and turning it to implementation. Using this strategy could add more flexibility and imagination to the project and also make it more adapted to the time limitation.

## 2.6. Requirements Priorities

The requirements above are listed with priorities order. All requirements are originated from business requirements. Client requirements are the most important and compulsory components. In order to meet the client requirements at the maximum level, user requirements and research topics would be reduced or cancelled according to time limitation. With the client requirements, producing an exercisable and effective online semantic archive system for museums, the implementation of which must be guaranteed, has the highest priority

### 3. Project Plan

#### 3.1. Chapter Overview

This chapter describes the initial project plan and also the alternative plan during the development process. For each section, actual plan are listed after initial plan for comparison, and followed by explanation for the change if applicable.

#### 3.2. Deliverables

Project Plan

Software Requirements Analysis

Software Design Description

Test Plan

Code

Installation and Configuration Guide

User Guide

Final Report

#### 3.3. Milestones

Initial Plan:

Milestone	Date
Initial Presentation & Report	Thursday 09 March
Design Completed	Sunday 02 April
Mid-project results due	Thursday 20 April
Implementation due	Friday 26 May
Final Presentation	Monday 08 June
Final Report due	Wednesday 21 June

Actual Plan:

Milestone	Date
Initial Presentation & Report	Thursday 09 March
Design Completed	Thursday 13 April
Mid-project results due	Thursday 27 April
Implementation due	Thursday 01 June
Final Presentation	Thursday 15 June
Final Report due	Wednesday 21 June

--	--

The change is due to the extension of design period and official timetable alteration.

### 3.4. Limits and Exclusions

Implementation would be limited to repositories centralised situation.

### 3.5. Constraints and Assumptions

Assuming there are no existing systems to be interoperated with in museum.

### 3.6. Resource Plan

#### 3.6.1. Technical Resources

##### 3.6.1.1. Open source software

#### 1. RDF/Ontology Development Tool: Jena, Protégé

**The Resource Description Framework (RDF)** is a language for representing information about resources in the World Wide Web. (W3C, RDF Primer) The expression of associations between entities is the major benefit of RDF.

**The OWL Web Ontology Language** is intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called an ontology. (W3C, OWL Web Ontology Language Overview)

**Jena** (<http://jena.sourceforge.net/>) is a Java framework for writing Semantic Web applications. It has the features of RDF API, RDF/XML Parser, Reasoning and Ontology subsystem. (Hewlett Packard Laboratories, Bristol, Jena2 Overview)

**Protégé** (<http://protege.stanford.edu/>) is a visual development tool for RDF, RDFS and OWL Web Ontology.

The languages and tools listed above are either developing prospectively or widely used in the semantic web technology area.

#### 2. Application and Interface Development Tool: Tomcat, Java



Java language has many excellent features such as unrelated platform, mobility; furthermore, it is compatible with other determining tools. We would develop an application model using Java to communicate interface model, metadata model and repository. As to the user interface development, JSP is compatible with Java very well and most familiar to me, the system programme. So we would use Tomcat and Java to build the interface and application development environment.

### **3. Repository: Dspace, Fedora**

**Dspace** (<http://dspace.org/>) and **Fedora** (<http://www.fedora.info/>) are the leading repositories, from which we should choose the most suitable and compatible one for our system design and other development environment.

### **4. Database: PostgreSQL, MySql**

According to the repository configuration files, PostgreSQL and Mysql are the open source databases to be used as backend database of Dspace and Fedora.

### **5. RSS Development Tool**

**ROME** is a set of Atom/RSS Java utilities that make it easy to work in Java with most syndication formats. (Sun Microsystems 2006)

**JDOM** is the dependencies are required to compile and run ROME.

### **6. Others: TBD**

We should keep our mind open for seeking other possible development languages or tools and comparing them with the ones on hand to build the better compatible developing environment and then the high quality semantic web system.

#### **3.6.1.2. Hardware**

One laptop, which set up with mobile AMD Athlon™ XP-M0+ would be used as implementation and testing machine. Other configurations are:

CPU: x86 Family 1600 Mhz

Memory: 256 MB

Operation System: Window 2000 Server (5.0.2195 Service Pack 4)

#### **3.6.2. Human Resource**

Over 350 hours workload would be cost in this project.

### **3.7. Project Schedule**

### 3.7.1. Work Breakdown Structure

1.0	Project Plan & Report
2.0	Requirement
2.1	Requirement Analysis
3.0	Knowledge Learning and Research
3.1	Development Tools
3.2	RDF
3.3	Ontology
3.4	Repository
3.5	Usable Interface
	(Learning & research throughout all project, but more at the beginning)
4.0	Building Development Environment
5.0	Design Design is actually combined with research and learning. It might include Domain Chart, Object Information Model, State Transition Diagram, Class Communication Model.
6.0	Implementation
7.0	Test
8.0	Project Closeout

### 3.7.2. Timetable

We use Microsoft Project to produce and manage the project schedule. See Appendix D for initial timetable and actual timetable.

## 3.8. Risk Analysis

### 3.8.1. Risk Assessment Form

Risk Event	Likelihood (0-6)	Impact (0-6)
The time required to analyse, design, implement and test underestimated.	2	5
Not enough time to implement all the functions designed.	4	3

### 3.8.2. Risk Response Matrix

Risk Event	Contingency plan
The time required to analyse, design, implement and test underestimated.	Reconstruct WBS and reschedule

Not enough time to implement all the functions designed.	Set priorities to the functions and implement them with the priorities order. Leaves the remaining functions to the future plan or future works.
--	--

## 4. Design Analysis and Modelling

### 4.1. Chapter Overview

This chapter would analyze the elements and technologies in semantic web and discuss how to apply them to museum knowledge management systems. It also includes the sections about the possible functions and other design considerations, such as storage, query methods and application model.

### 4.2. Data Model

There are many data models in the cultural heritage conservation and interchange area, such as CIDOC CRM (CIDOC 2005a), AMICO Data Model (Trant 2002), IFLA FRBR (IFLA 2002) and ABC Model from CIMI (CIMI 2002). These models have different emphases on the description of heritage objects and their attributes. For example, AMICO describes museum objects and especially the art-based collections and associated digital media; FRBR focuses on expression of objects properties and taxonomic bibliographic description, while CIDOC CRM is event centred or context focused to the museum objects and the relevant activities, people and time. (Gill 2003)

Between all these models, CIDOC CRM might be the most comprehensive and widely accepted one, which due to the reason of building upon many museums standards and processes. And mappings (crosswalks) have been established with major data schemas and metadata standards such as Dublin Core, ABC, and Spectrum. XML and RDFS implementations have been developed using the CIDOC CRM and data migrations have been made using test data provided by different institutions. (Cover Pages 2002) Furthermore, as mentioned in the introduction section of CIDOC CRM Homepage, CIDOC CRM is currently being elaborated by the International Standards Organisation as Draft International Standard ISO/DIS 21127. It is also a domain ontology providing definitions and a formal structure for concepts and relationships used in cultural heritage documentation.

Except the advantages above, due to its event centred and context focused specialities, CIDOC CRM is suitable for the context search and activity planning functions of the SMW system. However, CIDOC CRM includes 81 classes and 132 properties, it would not be realistic to apply and implement the whole model in the SWM system. We should concentrate on certain parts of the model, which would be most suitable to our defined functions of context search and activity planning.

### 4.3. Metadata Standard

There are some metadata standards have been mapped to CIDOC CRM, such as Dublin Core (DCMI 1999), AMICO Data Dictionary and SPECTRUM (MDA 2005) data elements. And CIMI has built a XML Schema based on SPECTRUM and CIDOC CRM. But we found CIMI XML Schema and SPECTRUM are complicated to use directly because SPECTRUM just includes data elements while not well defined namespace as Dublin Core; CIMI XML Schema need two processes of mapping from CIMI XML Schema to SPECTRUM and SPECTRUM to CIDOC CRM. The format of Dublin Core, including fifteen elements, seems simpler than the former two standards. But probably results from its simplicity that it becomes the foundation of many other standards and the most important metadata standard. From this point, using Dublin Core could make the share and communication with other systems and domains easier. We could add the customized vocabulary from the analysis of other standards compatible with CIDOC CRM, if the gap of data and model representation emerges during the practical process.

### 4.4. Storage Methods

We would choose Fedora (Flexible Extensible Digital Object Repository Architecture) as the system repository to store museum data with RDF format. The distinction between Fedora and other repository is that it could act as the foundation layer for a variety of multi-tiered systems, service-oriented architectures, and end-user applications, while not only storing and manipulating complex objects through a fixed user interface as other repositories like Dspace, arXiv, ePrints and Greenstone. (Lagoze et al 2005) And “the architecture includes a generic RDF-based relationship model that represents relationships among objects and their components.” (Lagoze et al 2005) Furthermore, the features of storing and searching data with RDF format make it more compatible with semantic technology based on RDF.

In the current implementation, Jena database package are used to store RDF data in MySQL, instead of Fedora. Using Jena package has shorten the development process for the prototype system, however, RDF model is stored in a few table in database, which would limit the system performance when RDF model turns bigger in the future. Fedora or other repository should be used for further development.

### 4.5. Query Methods

The Query methods mentioned here have two type of meanings, one is how to search

the data of RDF Format from the repository, another is how to present the collection data to users or how users search the information they want from the system.

We could use RDF query languages to search RDF data from Fedora repository. The RDF query languages supported by Fedora share a similar syntax to SQL and response the similar results as the result sets returned by a SQL query.

For the search services provided to users, we could use Finnish Museum Semantic Web for reference, whose data model is also based on CIDOC CRM. There are three types of semantics-based facilities for FMW. They are View-based filtering, Topic-based navigation and Ontological search. View-based filtering is the search function based on ontology used in annotating collection data and the search profile expressed by users. For example, by selecting Object Type=carpet and Material=silk, silk carpets are found. Topic-based navigation lists the links to the topics of interests according to the collection domain ontology and actual collection data. (Hyvönen et al) We would discuss Topic-based navigation later on in the section of Sub-domain Ontology. When user is browsing one topic of the search results from the two search methods above, system also list the related topics based on the Ontological search, such as the topics based on the relevant people, events or objects. We could adopt the methods of View-based filtering and Ontological search, which are taking use of the context-focused speciality of CIDOC CRM.

#### 4.6. Museum Based Exhibition and Virtual Exhibition Planning

Another service that SWM can provide is exhibition planning. The organisers of the exhibition need to know the relevant objects or artefacts and their condition, location... If the artefact is a temporal entity, which might be borrowed from other institute, then the returning date should be taken into consideration for the exhibition period. The search methods discussed above could fulfil these tasks. Virtual exhibitions offer another channel for general public to get information about the cultural heritage resource. Non-expert users or general public sometimes have no ideas about the search profiles for the interesting topics or even have not concept about what type of topics would interest them. Exhibition collections based on the knowledge of domain experts could help users to explore certain topics or the related domain. The difference between museum-based exhibition and virtual exhibition would be the locations and conditions of the artefacts do not need to be considered into the exhibition planning, because the collections are presented as text, image or other digital formats to users. And the virtual exhibition would be held permanently online as links provided to users.

## 4.7. Sub-domain Ontology

As depicted in Figure 2.3, ontology levels are consisted of upper ontology presenting the common generic information of all domains, middle ontology representing knowledge that spans several domains and may not be as general as the knowledge of the upper level, and the lower levels representing ontology at the individual domain or sub-domain level, which is knowledge about more or less specific subject areas. (Daconta et al 2003) The CIDOC CRM ontology belongs to the individual domain, the lower layer of the graph. It describes the data management and operation process in the heritage collection and conservation. As to the ontology in the sub-subjects of the collection such as archaeology, history, art and music...we could not get from the CIDOC CRM layer of ontology. But we also need the relationships or rules behind the sub-domains to further present, manage and utilize the knowledge. For example, if we have the ontology of the time span and dynasty, then that certain entity belongs to which dynasty could be inferred from its created time. Topic-based navigation would benefit from this type of ontology, because topic classifications rely on taxonomy and rule from the classified domain.

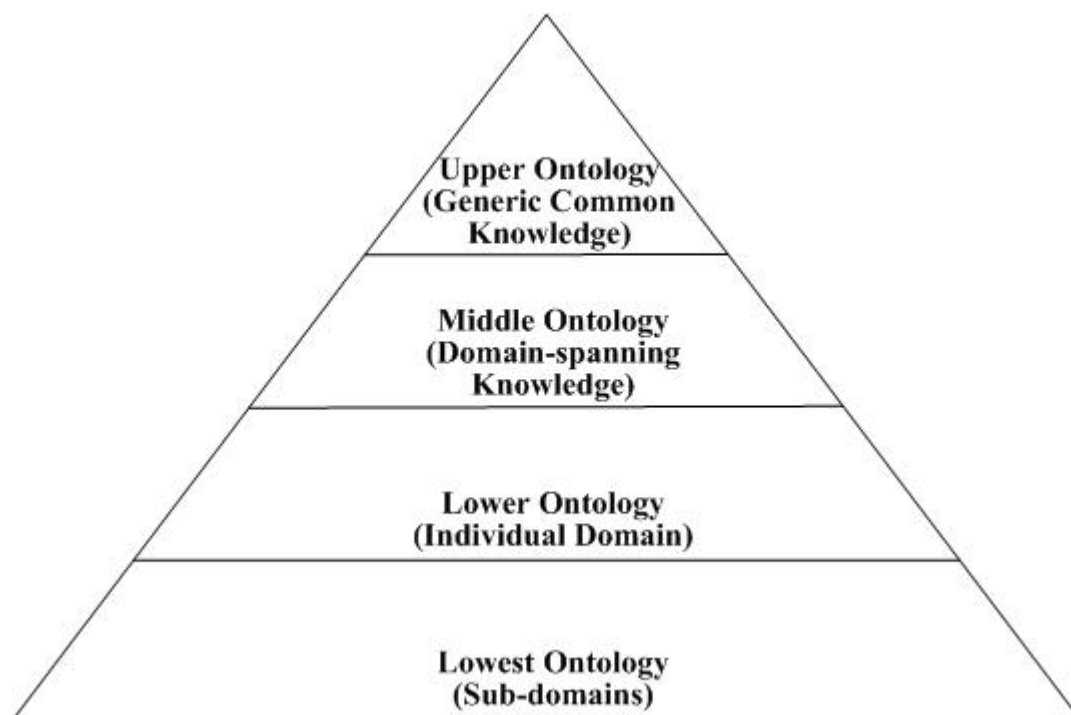


Figure 4.1. Ontology Levels (Drawn after C. Daconta, J. Obrst and T. Smith 2003)

As discussed in The Semantic Web (Daconta et al 2003): “domain experts have to address specific knowledge about their domains. They can be guided by ontologists for semantic modelling issues, ... and this knowledge must be provided to ontologists to represent their domains accurately.” So we could provide an interface convenient for domain experts to input their knowledge and produce the ontology automatically. Then the semantic web system could utilize it with the functions like topic map or

intelligent management. If the sub-domain ontology can be applied to the system, then we would prove that the designing methods used in the systems could cooperate with other ontology or other systems. This could be a future work for the project due to the scope and time limitation.

#### 4.8. Model Examples and Explanation

CIDOC CRM includes 81 classes and 132 properties, it would not be practical to apply and implement the whole model in the SWM system. So we would use a partial model in our system. The model graphs together with example data are list below:

As seen from the graphs, RDF Triple model concept is used in the CRM Model. Class E1 to E81 stand for the subjects or objects in Triple; the relationship link, such as `has_type`, `took_place_at`, could be regarded as predicate.

The models in graphs are concerned about the object properties, event centred and acquisition model separately.

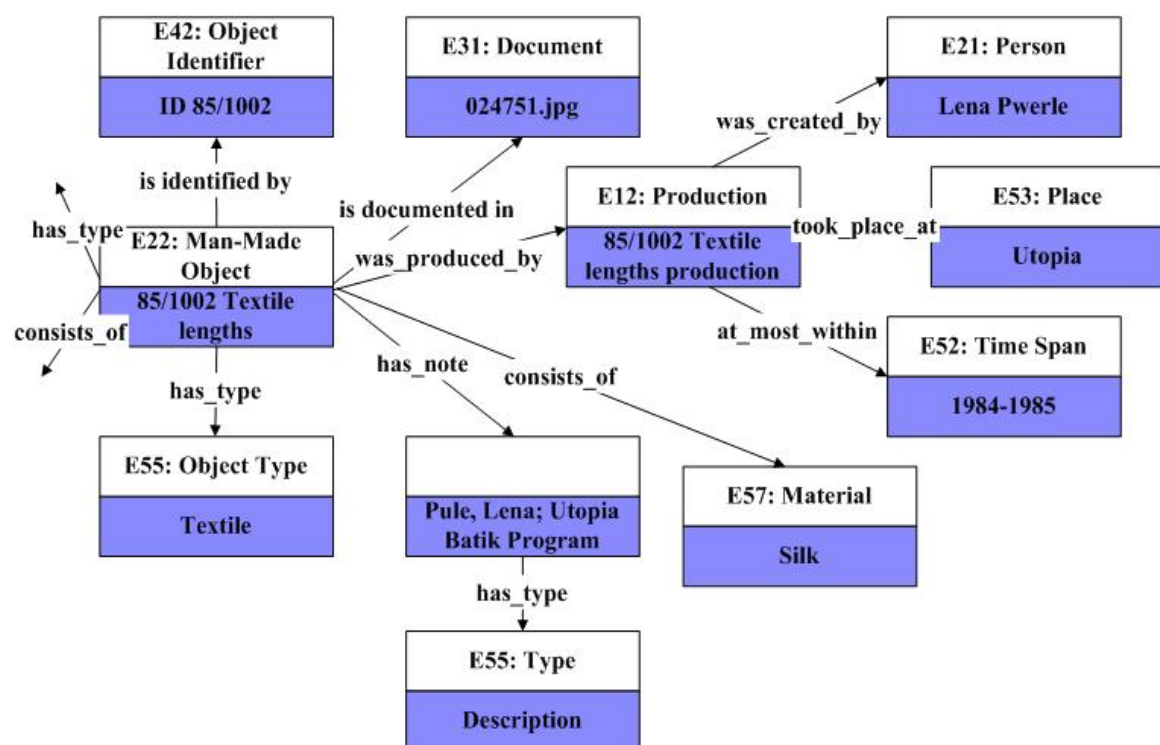


Figure 4.2. Object Properties Model (Drawn after CIDOC 2005c)

The first example is an artefact named Textile Lengths. As we could see, it has the properties of type, note, identification and production... And all these properties are not limited to a single value. For example, it could have one more type as “aboriginal animal motif” or other material property, such as thread.



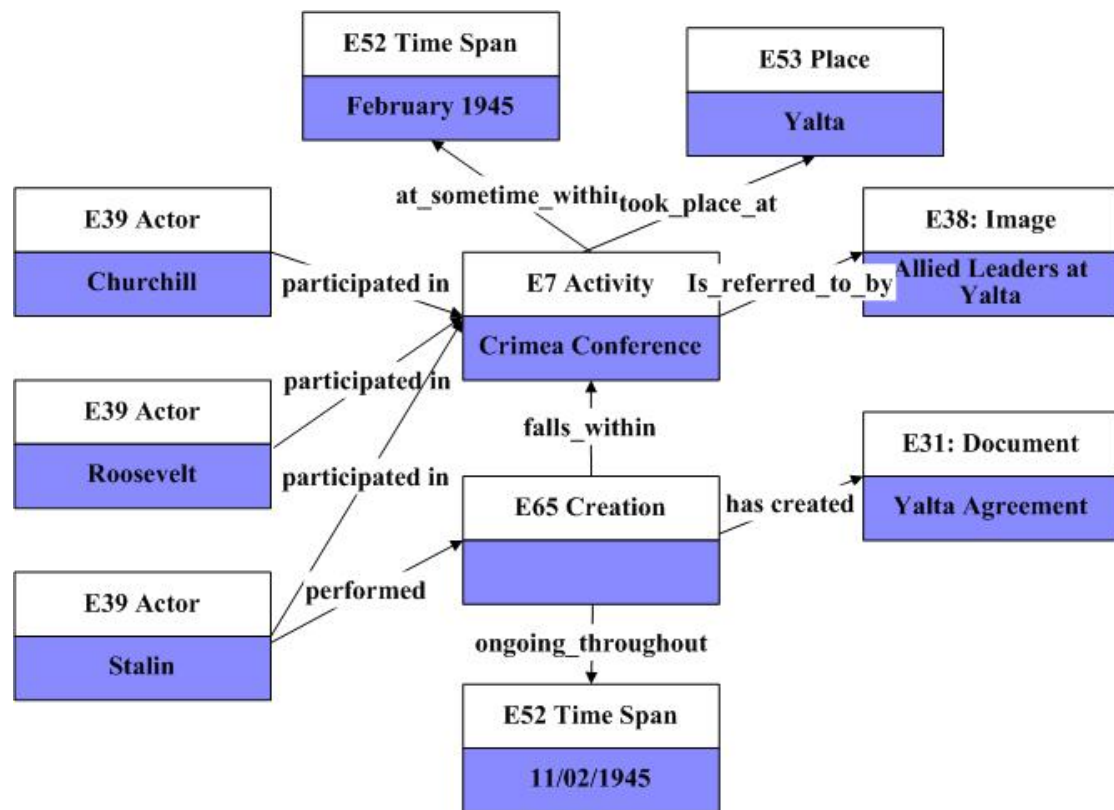


Figure 4.3.Event Centred Model (Drawn after CIDOC 2005c)

The Second example is about a conference after World War II. The graph presents the participant, event time, place and the created agreement during the conference. The event centred models are related to the classes of Actor, Activity, Creation, Time Span and Place.

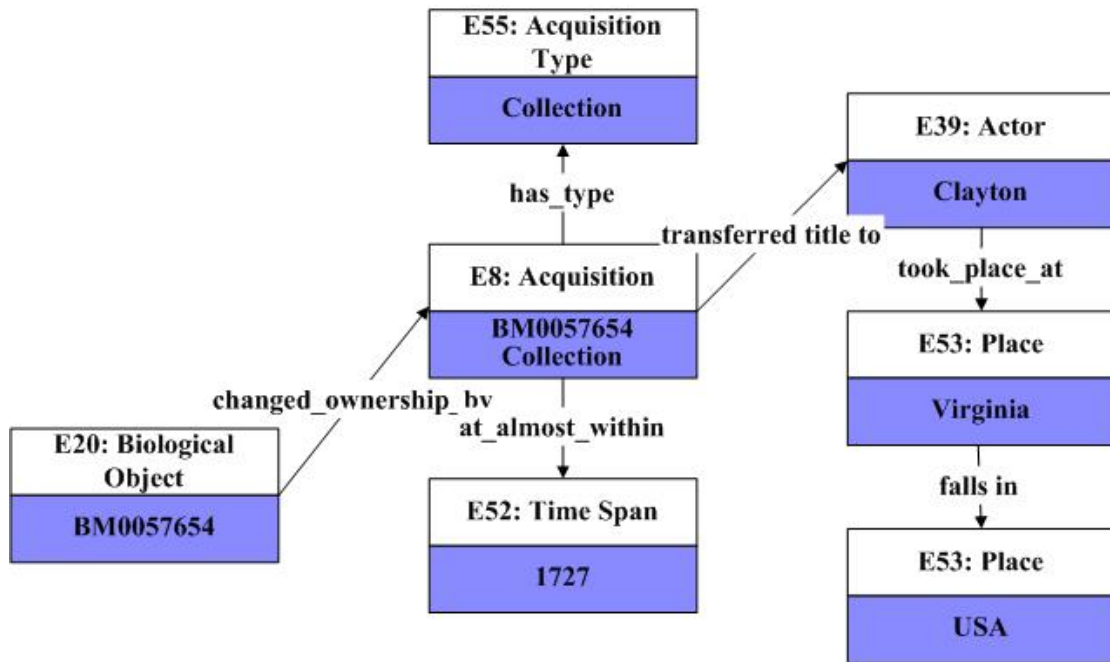


Figure 4.4. Acquisition Model (Drawn after CIDOC 2005c)

The Third example is more specific to the museum domain. It presents the object acquisition activity and process in museum.

All these models above describe the simple extraction of CRM Model application for SWM System.

## 4.9. Elements Relationships

Several semantic web elements and technologies have been discussed so far. For applying them to the semantic web system design, how they related to each other need to be clarified for further design and implementation.

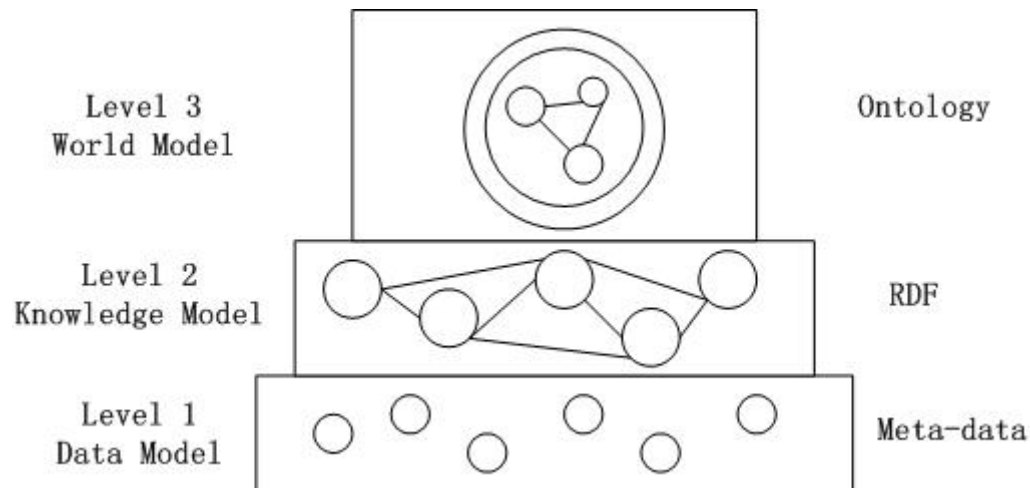


Figure.4.5. Semantic Levels (Redrawn after C. Daconta, et al 2003)

Figure.4.4 shows the information stack composed of semantic levels. Level 1 is meta-data describing singular concepts and objects. Level 2 RDF enable us to model the relationships between level 1 objects. Level 3 ontology describes semantics of the world in our mind.

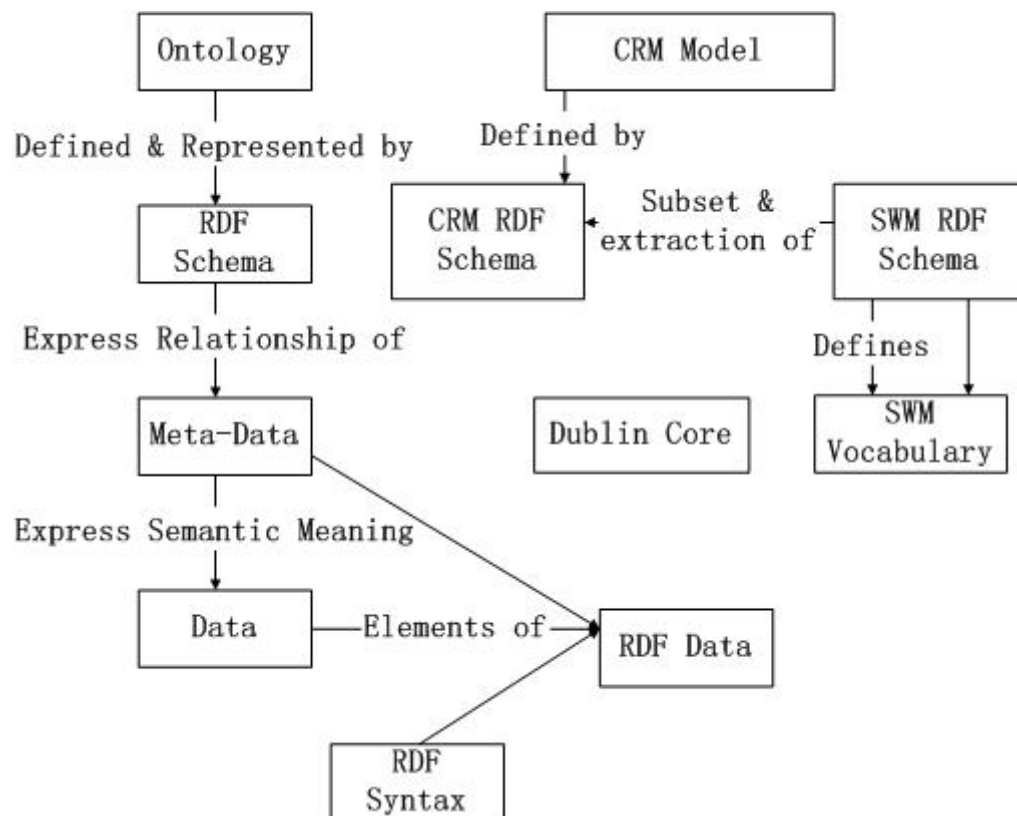


Figure.4.6.Elements Relationships

Figure.4.5 shows the elements relationships in our system design. The left side of the figure describes the similar meaning of Figure.4.4.Semantic Level, except showing the RDFS to define and represent ontology. RDF Schema is a lightweight ontology

language to represent ontology. Elements on the Right side of the figure correspond to the elements on the left side. CRM Ontology need to be extracted to subset ontology and vocabulary, since it includes 81 classes and 132 properties, such a large structure, and its vocabulary definition is too detailed to be used as metadata with Dublin Core. Dublin Core and SWM Vocabulary are used as metadata. Dublin Core is widely accepted by many domains, so SWM Vocabulary Mixed use with Dublin Core can increase the interoperability with other domains and applications. All data would be stored and transferred as RDF data, which composed of Data, Metadata and RDF syntax. The Example RDF data and mixed use of metadata vocabulary could be:

```
<swm:activity rdf:about = "&basens; activity / Textile Lengths 85 – 1002
Production">
  <DC:type>production</DC:type>
  <DC:identifier>Textile Lengths 85-1002 Production </DC:identifier>
  <swm:beginDate>1984</swm:beginDate>
  <swm:endDate>1985</swm:endDate>
  <swm:locateAt rdf:resource = "&basens;location/Ngkwarlerlaneme camp"/>
</swm:activity>
```

#### 4.10.RDF Vocabulary Design

CRM Model and DC metadata standard have completely different vocabularies. For example, CRM uses the following RDF Schema (W3C 2004) to describe the identifier of object: (CIDOC 2005b)

```
<rdf:Property rdf:ID="P47F.is_identified_by">
  <rdfs:domain rdf:resource="#E19.Physical_Object"/>
  <rdfs:range rdf:resource="#E42.Object_Identifier"/>
  <rdfs:subPropertyOf rdf:resource="#P1F.is_identified_by"/>
</rdf:Property>
<rdf:Property rdf:ID="P47B.identifies">
  <rdfs:domain rdf:resource="#E42.Object_Identifier"/>
  <rdfs:range rdf:resource="#E19.Physical_Object"/>
  <rdfs:subPropertyOf rdf:resource="#P1B.identifies"/>
</rdf:Property>
```

Resources E19.Physical\_Object, E42.Object\_Identifier and properties P47B.identifies, P47F.is\_identified\_by are defined in CRM to express the relationship between physical object and its identifier. According to the mapping between Dublin Core metadata element set and CRM (Doerr 2000), DC.identifier, one of the fifteen elements in Dublin Core metadata standard, could correspond to E42.Object\_Identifier.

Similarly we could find the corresponding Dublin Core elements to other resources and properties in CRM for our application. However, as seen from the example above, CRM and Dublin Core are just like two extremes, one uses very detailed definition for every RDF resource and property, while the other only has a number of elements. Sometimes the meaning in CRM could not be completely expressed using Dublin Core elements because of the gap between these two and the need of compliance with RDF format. So we need to design the RDF vocabulary to fulfil the gap and use the designed vocabulary and Dublin Core together to express the meaning of CRM model for the application.

Several steps are followed in the design process.

1. List the application components and find the corresponding CRM definitions.
2. Dublin Core elements and designed vocabulary are mapped to the CRM definitions.
3. Write the RDF examples according to the Dublin Core elements, designed vocabulary and applications.
4. RDF vocabulary could be formalized with RDF Schema.

These steps are listed in **Appendix A. Mapping Table** and **Appendix B. SWM RDF Schema** with the application components. The components we could extract from the application are Entity, Physical Object, Representation, Activity, Location and Condition Assessment.

## 5. Implementation

### 5.1. Chapter Overview

The chapter describes the system architecture and implementation details, such as system components, units purpose, design consideration, program library and example usage. It provides the useful supplements to program comments for later maintenance or further development.

### 5.2. Architectural Design

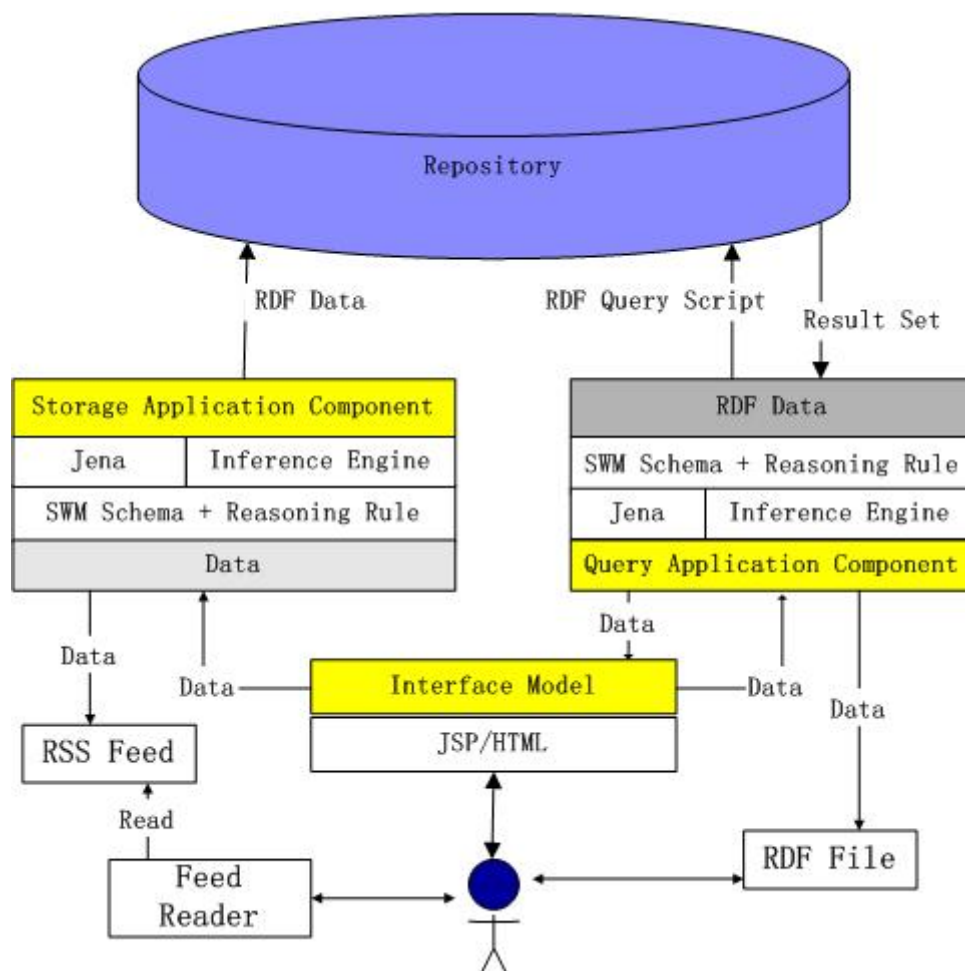


Figure 5.1. System Architecture

The architecture shows Storage Application and Query Application component act as intermediate to communicate with repository and interface model. Using the pipes-and-filters architecture styles has the benefits of Efficiency by parallel

processing and Reuse and Recombination of filter elements. (Buschmann et al 1996)

The Architecture Components, as seen from the architecture graph, include Storage Application, Query Application and Interface Model Components.

**Storage Application:** Transfer the data obtained from Interface Model to RDF Data based on SWM schema and reasoning rule. Inference engine is one function of Jena framework. It can validate the data against the RDF schema and rules.

**Query Application:** Use Jena library to execute the RDF query scripts in repository and return the results set. Produce the RDF Data with results set and validate it against the RDF schema and rules.

**Interface Model Component:** Provide the content input, context search and activity planning interface with JSP and HTML technologies; transfer the data from content input, context search components to storage and query components; transfer the data obtained from Query Application Component to JSP/HTML pages.

### 5.3. Storage Application Component

#### 5.3.1. Component Structure

Storage application component is composed of five units, which including Single Entity Storage, Batch Storage, RSS Storage, Input Batch for Testing and Remove Model for Testing. Input Batch and Remove Model Units provide simple functions for entering data to or removing data from the system when interface model is not ready or under development.

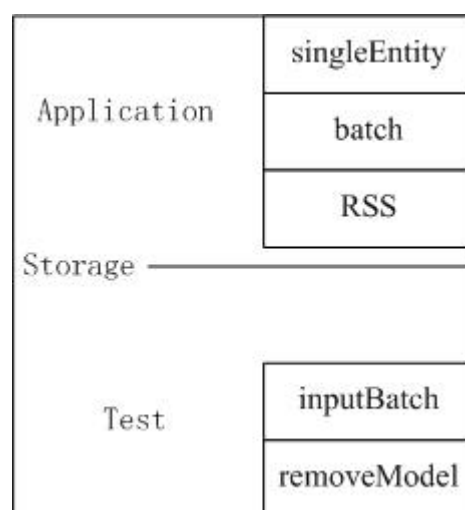


Figure 5.2.Storage Application Component Structure

### 5.3.2. Single Entity Storage

#### Unit Purpose

This unit is used to transfer single physical object, activity, representation, location data from user input data to RDF data, which complies with SWM vocabulary schema, RDF and Dublin Core format, for repository storage and further search requirements.

#### Design Decision

Instead of making class to each component saving function, passing value, such as “*physicalObject*”, “*activity*”, “*representation*” and “*location*”, to “*rootType*” parameter in “*singleEntity*” construction function, would provide concise structure and reduce the size of code.

Before saving the data as RDF in repository, unit should check the validation of RDF data against the SWM schema and DC, RDF format. This could be done by inference model validate function from Jena library.

#### Program Library

The unit is composed of *singleEntity* class of *swm.storage.\** package and several jsp files for entity save purpose, such as *objectsave.jsp*, *activitysave.jsp*, *representsave.jsp* and *locationsave.jsp*.

*singleEntity* Construction Parameters:

*aoDBConn* - Database connection, *asModelName* – name of Model used to store RDFdata , *asSchemaUri* – schema file address, *asRuleUri* – rule file address, String *asRootUri* – resource URL, *asRootType* – resource type

#### Function:

*checkModelValid ( )*: Check the input data complies with schema, rule, RDF and Dublin Core format before save as RDF data in repository.

*addPredRes ( )*: Add predicate and object to the root resource, in which predicate acts as property to root subject (resource) and object acts as resource attribute to property.

**Parameters:** *predNs* - Name space of predicate; *predLocal* – Local name of predicate; *resUri* – Object resource URI.

#### Example usage:

```
so.addPredRes(nsSwm, "locateAt", nsBase+"location/d8-c2-r7-f6-wn-benaki");
```

#### Result of added property:

```
<swm:locateAt rdf:resource="&basens;location/d8-c2-r7-f6-wn-benaki"/>
```

*addPredLiteral ( )*: Add predicate and object to the root resource, in which predicate acts as property to root subject (resource) and object acts as literal value to property.

**Parameters:** *predNs* - Name space of predicate; *predLocal* – Local name of predicate; *literal* – Object literal value.



**Example usage:**

```
so.addPredLiteral(nsDC, "type", "ecclesiastical embroidery");
```

**Result of added property:**

```
<DC:type>ecclesiastical embroidery</DC:type>
```

*addPredNode ( )*: Add predicate and object to the root resource, in which predicate acts as property to root subject (resource) and object acts as RDFNode to property.

**Parameters:** *predNs* - Name space of predicate; *predLocal* - Local name of predicate; *type* - type of object as RDFNode; *value* - value of object as RDFNode.

**Example usage:**

```
so.addPredNode(nsDCTerms, "requires", "shows", nsBase + "object/Epitaphios  
GE34604");
```

**Result of added property:**

```
<DCTerms:requires rdf:parseType="Resource">  
  <rdf:value>&basens;/object/Epitaphios GE34604</rdf:value>  
  <DC:type>shows</DC:type>  
</DCTerms:requires>
```

*Save ( )*: save all the added data as RDF data to repository.

*Close ( )*: every time the storage process finished, close function should be called to close the collection with database and relieved the memory.

### 5.3.3. Batch Storage

**Unit Purpose**

The purpose of batch unit is to save the batch or several entity items with RDF file or text at one time in repository.

**Design Decision**

Before they are stored in repository, RDF file and text would be validated against SWM schema, RDF and DC format. If the validation fails, invalid message about RDF file/text will be thrown to users through interface model.

RDF text will be handled with the same method as to RDF file, since RDF text would be transferred to temporary RDF file firstly and then the file address could be pass to batch function.

**Program Library**

batch Construction Parameters:

*checkFileValid ( )*: Check the RDF file complies with schema, rule, RDF and Dublin Core format before save as RDF data in repository.

**Parameters:** *asFile* - address of RDF file

**Return:** Boolean

Save ( ): Save the valid RDF file as RDF data in repository.

Close ( ): Same as singleEntity.

#### 5.3.4. RSS

##### **Unit Purpose**

New item would be added to RSS file when resource added to the system. User can view RSS files through RSS interface page or FeedReader.

##### **Design Decision**

RSS1.0 format are used for RSS file, since RSS1.0 are more compatible with RDF than other version. Rome.jar and Jdom.jar are used for RSS file reading and RSS items adding. The codes for RSS item adding locate at entities saving page.

#### 5.3.5. Input Batch for Testing

##### **Unit Purpose**

This component is designed for convenient test. It can input RDF file with a simple way, which means not checking validation or through interface model.

##### **Program Library**

*InputBatch* class

#### 5.3.6. Remove Model for Testing

##### **Unit Purpose**

This component is also designed for convenient test. It can delete the test model from database.

##### **Program Library**

*removeModel* class

### 5.4. Query Application Component

#### 5.4.1. Component Structure

Query application component is composed of Keyword Search, Relation Search and Detail Search Unit.

#### 5.4.2. Keyword Search

##### **Unit Purpose**

Get keywords search results from database and return the results to interface model as

List.

### **Design Decision**

It supports multiple keywords and or/and condition search. Keywords were assumed separated with one or more space. Resource identifier and resource description, resource relationships with keyword would be return as List respectively.

### **Program Library**

*SearchKeyword* class of *swm.search.\** package

*checkModelValid ( )*: Check the searched model complies with schema, rule, RDF and Dublin Core format before search operated on this model.

*doSearch ( )*: search the resource according to the keywords and condition.

**Parameters:** *asKeyword* – the search keyword which may contains one or more keywords separated by one or more space; *abAnd* – true:search according to and condition, false:search according to or condition.

## 5.4.3. Relation Search

### **Unit Purpose**

Get relationships search results from database and return the results to interface model as List.

### **Design Decision**

It supports one or two levels relationships search for certain resource. As for two levels relationships, the item of the second level may be the same resource as the base resource. The same node would be deleted from the results list before returned to the interface model.

### **Program Library**

*SearchRlation* class of *swm.search.\** package

*checkModelValid ( )*: Check the searched model complies with schema, rule, RDF and Dublin Core format before search operated on this model.

*doSearch()*: search one level relationship according to the resource.

**1 Parameters:** *asResUri* – resource URL

**2 Parameters:** *asRes* – resource identifier, *asType* – resource type.

*do2LSearch()*: search two level relationships according to the resource.

**2 Parameters:** *asResUri* – resource URL, *aoList1L* – one level relationships list used to remove the same node with 1 level list from 2 level list.

**3 Parameters:** *asRes* – resource identifier, *asType* – resource type, *aoList1L* – one level relationships list

#### 5.4.4. Detail Search

##### **Unit Purpose**

Get properties of certain resource and return them to interface model as List.

##### **Program Library**

*SearchDetail* class of *swm.search.\** package

*checkModelValid ( )*: Check the searched model complies with schema, rule, RDF and Dublin Core format before search operated on this model.

*doSearch()*: search properties of the resource.

**1 Parameters:** *asResUri* – resource URL

**2 Parameters:** *asRes* – resource identifier, *asType* – resource type.

### 5.5. Interface Model

#### 5.5.1. Component Structure

Interface Model is composed of Introduction, Storage, Query, User Guide and Help components. Storage component includes storing Physical Object, Activity, Representation, Location, RSS and Batch function, while *textTable* provides input and delete text functions to other components in Storage. Query component includes Search Keyword, Related Resource, Detail, Relationship Diagram, Show RDF and Print function. *CenterRela HTC* provides drawing functions to Relationships Diagram component.

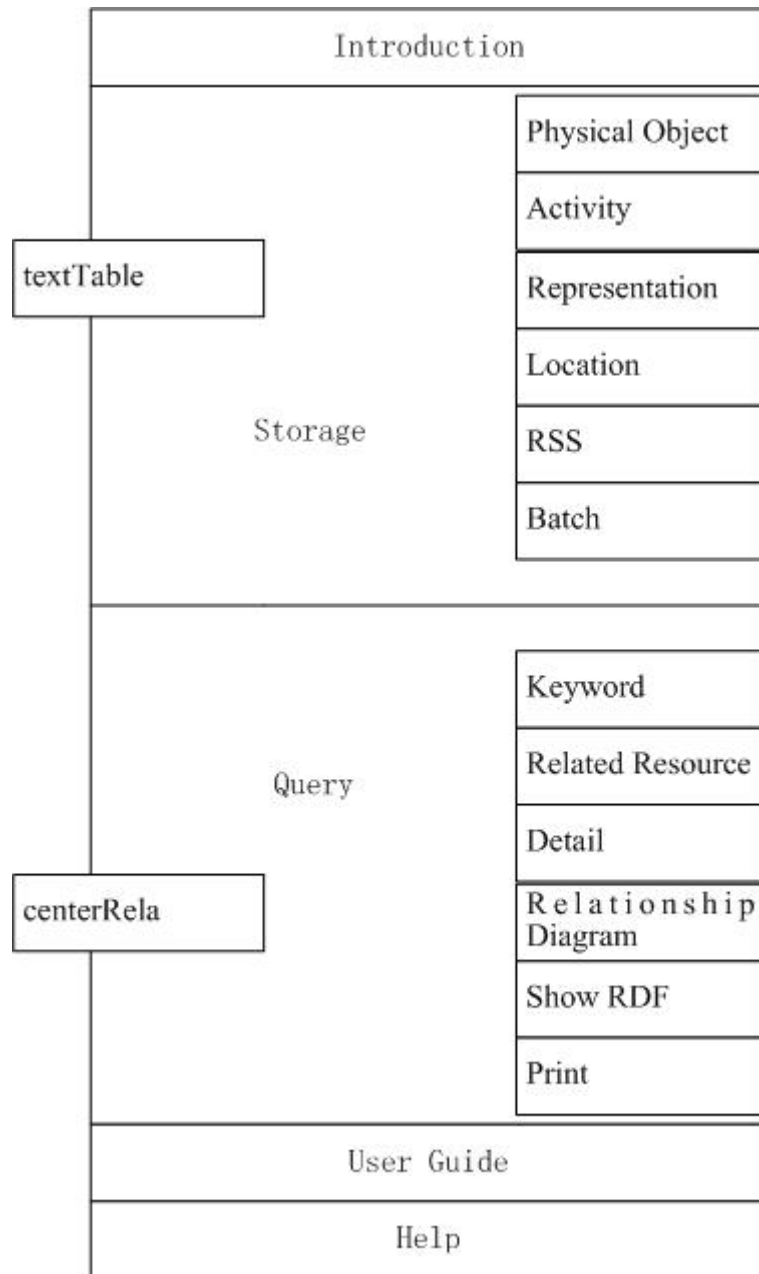


Figure.5.3. Interface Component Structure

### 5.5.2. Introduction

#### Unit Purpose

It provides an overview of the Semantic Web project. The contents include the semantic web technology, the motivations, background and objectives of the Semantic Web Project.

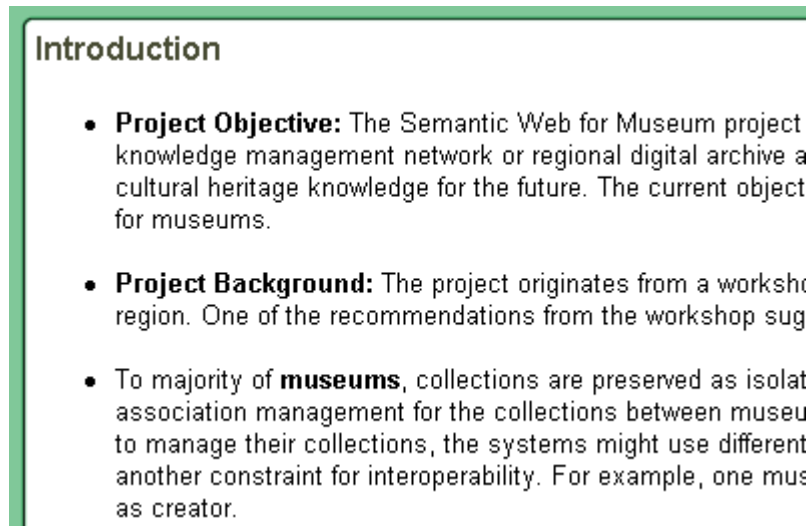


Figure.5.4.Partial Snapshot of Introduction

### 5.5.3. TextTable

#### Unit Purpose

The component is used to provide multiple purposes users input functions, such as single value input, multiple value input, value adding, value deleting, multiple choices and the corresponding value adding.

#### Design Decision

Javascript Language is used to control the display and layout of the HTML elements in *textTable*, in which the major element controls are table row selecting, adding and deleting.

#### Program Library

textTable.js

#### Function:

***appendTable* ( )**: append the *textTable* component in certain container of page.

#### Parameters:

***tableID*** – The ID of the created *textTable* component. It is used for later reference.

***ContainerID*** – The ID of element that contains the *textTable* component. Example element: <SPAN> <DIV>

***arraySize*** –table body column and row number and size. Example- [2,4,30%, 30%, 40%]: 4 rows and 2 columns of 40%, and 60% respectively.

***arrayHead*** – table head content with the same array size with body column size, Example- new Array('<b>Contributor</b>','Example: Georgios, son of Kyriazes');

***arrayData*** – table body content. Each cell is composed of an array with length of 2. array[0] is 1 (input) or 0 (text); array[1] is content of cell. Example of a 2 columns and 3 rows table: [[0,title1] , [1,input1] , [0,title2] , [1,input2] , [0,title3] , [1,input3]]

***asRows*** – Height of Cell. It uses the default value 1 if no data is passed to this parameter.

**Example usage:**

Table with head:

```
var idenASize = new Array(2,1,"30%","70%");
var idenAData = new Array(new Array(0,'<b>Identifier</b>'),new Array(1,""));
var idenAHead = new Array('&nbsp;','Example: Transfer-1923 of Epitaphios
GE34604');
appendTable('p_identifier','p_identifier_span',idenASize,idenAHead,idenAData);
```

	Example: EpitaphiosGE34604.png
<b>Identifier</b>	

Figure5.1.textTable component with head title

Table without head and with input choice:

```
var ctypeASize = new Array(2,1,"30%","70%");
var ctypeHTML = '<select width=100% id="p_ctype"><option... </select>';
var ctypeAData = new Array(new Array(0,'<b>Controlled Type</b>'),new
Array(0,ctypeHTML));
appendTable('p_ctypeTA','p_ctype_span',ctypeASize,null,ctypeAData);
```

<b>Controlled Type</b>	Image 
------------------------	---

Figure5.5.textTable component with input choice and without head title

**addRow ( ):** Add one more row at the end of the *textTable* component.

**Parameters:**

*tableID*, *arraySize*, *arrayData*, *asRows*: Same as the parameters in *appendTable*.

*bHead* – If the *textTable* component has head title or not.

**delRow ( ):**Delete the selected row. The cell will be selected when click event is activated in this cell. The row, which this selected cell belongs to, will be the selected row. The present change for selection is controlled by “*selTd*” and “*unselTd*” class from style sheet.

**Example usage of addRow() and delRow():**

```
var typeASize = new Array(2,1,"30%","70%");
var typeAData = new Array(new Array(0,'<b>Type</b>'),new Array(1,""));
<button onclick = "addRow('p_type',typeASize, typeAData, true);" title="Add
Type"><b>+</b></button>
<button onclick="delRow('p_type');" title="Delete Type"><b>-</b></button>
```



	Example: Lament Image	
<b>Type</b>		

Figure5.6.textTable component with add and delete row function

### 5.5.4. Physical Object Storage

#### Unit Purpose

It provides the input interface for attributes of physical object.

#### Design Decision

The interface is composed of several “*textTable*” component. The data gotten from the *textTable* component will be checked for validation before posted to *objectsave.jsp* page for further transfer and storage, in which identifier and controlled type are compulsory. Example usages are provided in the head of *textTable* as convenient reference.

#### Diagrams:

	Example: Textile Lengths 85-1002
Identifier	<input type="text"/>

Controlled Type	Man-Made Object ▼
-----------------	-------------------

	Example: embroidery	+
Type	<input type="text"/>	-

	Example: Textile Lengths	+
Title	<input type="text"/>	-

Figure.5.7.Partial graphics of physical object input interface

### 5.5.5. Activity Storage

#### Unit Purpose

It provides the input interface for attributes of activity.

#### Design Decision

Contributor, related objects and object location have multiple choices and values input. Component *textTable* has more complicated usages here, which means added and deleted rows size will be two rows, and one of them is multiple choices content. Other contents are the same as Physical Object.

#### Diagrams:



Related Object	Example: Epitaphios GE34604	
Relation	transferred	+
Related Object		-

Object Location	Example: d0-c4-r8-f6-Metropolitan	
Type	object former location	+
Object Location		-

Save Activity

Figure.5.8.Partial graphics of activity input interface

### 5.5.6. Representation Storage

#### Unit Purpose

It provides the input interface for attributes of representation.

#### Design Decision

Same as physical object and activity

#### Diagrams

Similar to physical object and activity

### 5.5.7. Location Storage

#### Unit Purpose

It provides the input interface for attributes of Location.

#### Design Decision

Same as physical object and activity

#### Diagrams

Similar to physical object and activity

### 5.5.8. Batch Storage

#### Unit Purpose

It provides the input interface for batch input. Batch input function increases the interoperability with other semantic systems and provide a convenient transfer way to existing data.

#### Design Decision

RDF file or text information can be input by user and then passed to batchsave.jsp page for further operation and storage. The special characters in RDF Text, such as &, #, \n, cannot be accepted by page link parameter, because they conflict with the

reserved character. So these characters will be transferred to the corresponding URL coding, such as '%26', '%23', '%0A' before posted to batchsave.jsp page.

### Diagrams

**Save Date from RDF file:**

**Save Date from RDF Text:**

```

<?xml version="1.0"?>
<!--
RDF Example for the Semantic Web for Museums Element Set
This Version:
  RDF Example 1 the Semantic Web for Museums Element Set 2006/05/17
-->
<!DOCTYPE rdf:RDF [
  <!ENTITY swmns "http://escience.anu.edu.au/project/06S1/semanticWeb/SWM/eleme
  <!ENTITY basens "http://escience.anu.edu.au/project/06S1/semanticWeb/">
]>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:DC="http://purl.org/dc/elements/1.1/"

```

Figure.5.9.Partial graphics of batch input interface

### 5.5.9. RSS Storage

#### Unit Purpose

It provides the interface to show the RSS file.

#### Design Decision

See also Storage.RSS Unit and Appendix.C. RSS Example

### 5.5.10. CenterRela

#### Unit Purpose

CenterRela provide the functions to draw the relationships of a focused entity and its related entity. The component can be extended to draw different layout and expression graphics, such as pie chart for portion-value expression.

#### Design Decision

HTML Component (HTC) is used as the framework of CenterRela. HTC provides the object-oriented programming facility, which help to capsule all the drawing operations in component. Javascript and Vector Markup Language (W3C VML) are used for elements controlling and graphics rendering. Vector graphics are preferred choices for web graphics rendering, since they have better rendering performance and much smaller size than bitmap graphics. (Morrison et al, 2000 Chp34.2) So it is more suitable to use Vector rendering methods to draw the large number of museums entity relationships.

The component can be extended to three version, flow branch, encirclement and pie chart style. Currently only flow branch style is used in the system. However, the figures of three styles are all listed below to show the extendable of this component.

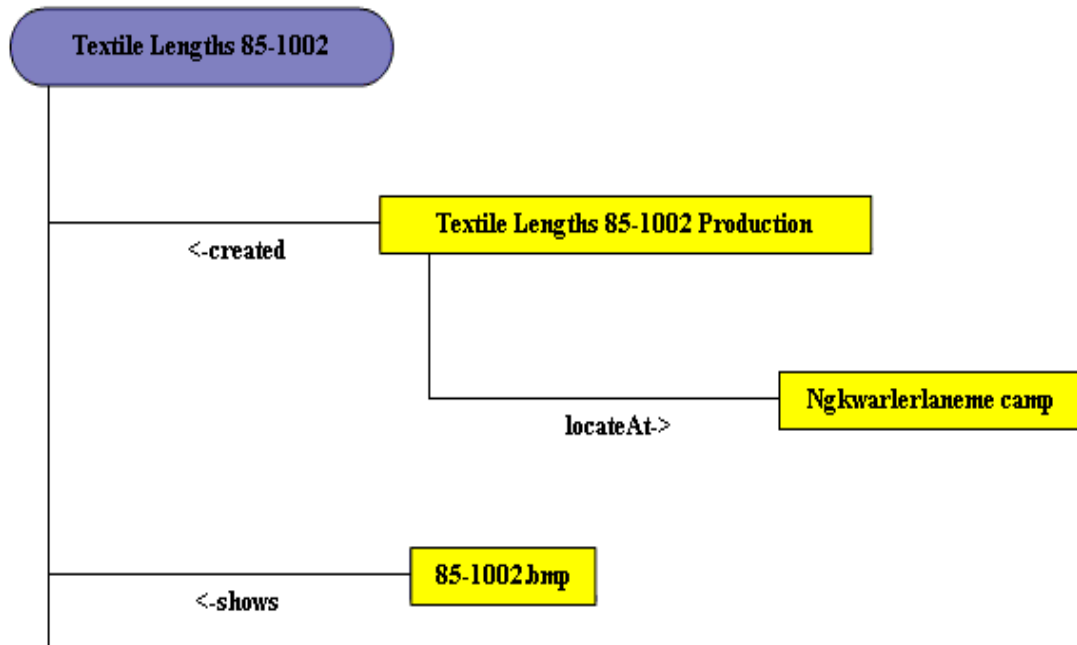


Figure.5.10.Flow Branch Style for CenterRela Component

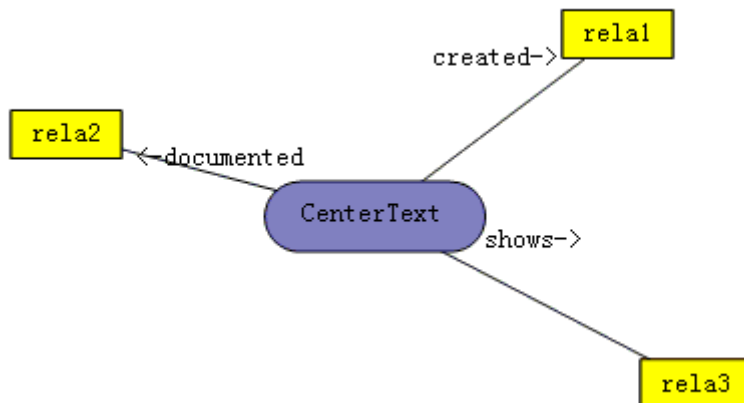


Figure.5.11.Encirclement Style for CenterRela Component

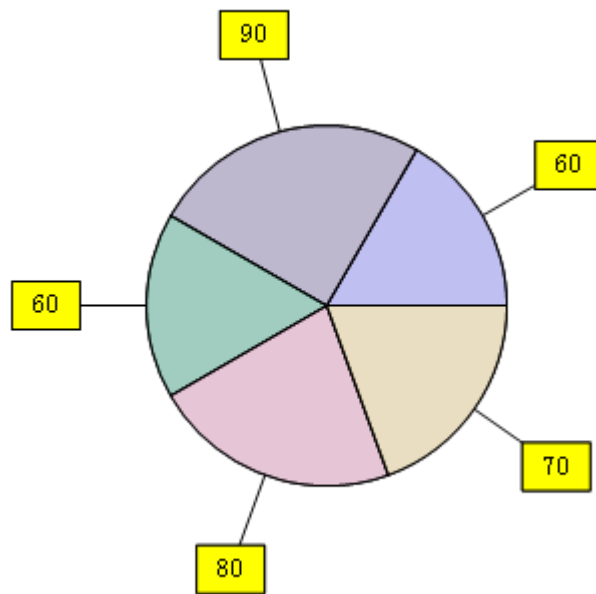


Figure.5.12.Pie chart Style for CenterRela Component

**Program Library:**

CenterRelas.htc, Sector.htc (for extension to other style)

### 5.5.11. Keywords Search

**Unit Purpose**

It provides the interface to search keywords and display results.

**Design Decision**

For multiple keyword searching, each keyword can be separated with one or more space. The search also bases on condition or/and. The keywords and condition would be submitted to *searchKeywordRsp.jsp* from *searchKeyword.jsp* page. *searchKeyword* class and methods would be used in *searchKeywordRsp.jsp* page. Returning results would be classified according to the resource types. The results show resource identifier, resource description and the relationships with keyword, which provides the relevancy to user for target locating or search refining.

**Diagrams**

Keyword:  OR ☒ AND ☐

---

Physical Objects

- [Epitaphios GE34604](#)  
EpitaphiosGE34604 is instance of Iconographic object:  
Epitaphios GE34604 *medium*→ **silk cloth**
- [Textile Lengths 85-1002](#)  
Textile length, batik, silk, Lena Pwerle/Utopia Batik Program, Ngkwarlerlaneme camp, Utopia, Northern Territory, Australia, 1984-1985  
Textile Lengths 85-1002 *medium*→ **silk**

Activities

Figure.5.13.Search keyword interface

### 5.5.12. Related Resource Search

#### Unit Purpose

It provides the reference of resource context to users when they viewing the resource pages.

#### Design Decision

Float blocks are used to show the related resource items. Each item includes relationships, relationship direction arrow and related resource identifier. The item is also the link to the page of this related resource.

#### Diagrams

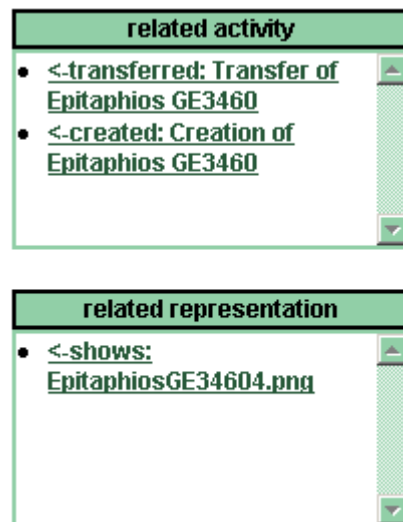


Figure.5.14.Related Resource Window

### 5.5.13. Detail

#### Unit Purpose

It shows the properties of resource. For representation object, it could show image.

### Design Decision

Class *searchDetail* and its methods would be used in the page.

### Diagrams

**object: Epitaphios GE34604**

- **extent:** folio,length 1:5m,width 1:2m
- **subject:** Lament
- **title:** TA 959a
- **medium:** silk cloth
- **medium:** silver thread
- **description:** EpitaphiosGE34604 is instance of
- **identifier:** Epitaphios GE34604

Figure.5.15.Resource Properties

## 5.5.14. Relationships Diagram

### Unit Purpose

It shows the one or two level relationships diagram.

### Design Decision

*CenterRela.htc* would be used to draw the levels relationship diagram. The diagram would not be initialized until users press “1 Level” or “2 Levels” button. And “Clear” button is provided to remove the drawn diagram from screen.

### Diagrams



Figure.5.16.Drawing Button

**See also:** Figure.5.4.Flow Branch Style for CenterRela Component

## 5.5.15. Show RDF

### Unit Purpose

It shows the RDF data embedded in the resource page. Then the RDF data could be move to other system, which increases the data exchangeable and interoperability.

### Design Decision

Firstly the RDF data would be put into RDF file under search/RDF directory, and then presented to users using HTML window.

### 5.5.16. Print

**Unit Purpose**

It provides the printing function to resource properties and relationship diagram.

**Design Decision**

A *print.css* is used instead of the default style sheet *style.css*. Related resource window would be hidden when printing. Relationship diagram will be printed if it is showed. Users can clear the diagram if they don't want to print it.

### 5.5.17. UserGuide

**Unit Purpose**

It provides the operations and functions of the system to users.

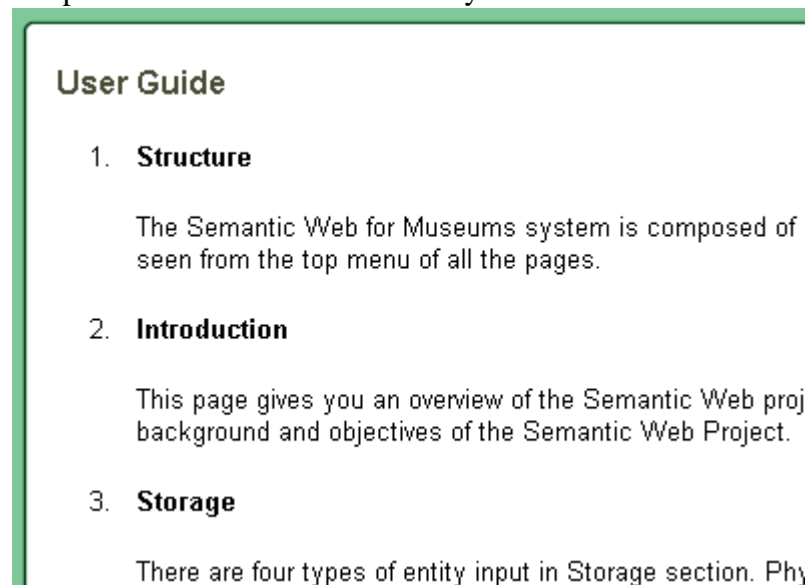


Figure.5.17.Partial Snapshot of User Guide

### 5.5.18. Help

**Unit Purpose**

It Provides the tips or FAQ to users.

## Help

### The thing you may not know about the system:

1. When you input the RDF data from resource page to Batch Storage, rer displayed on web browser has additional "-"sign before each resource, i
2. The yellow rectangle in the relationship diagram can be moved around in diagram branch or to just have fun.
3. You can get a larger image when you click the image on the representa
4. If you are interested in the theory and design methods behind the syste the top menu of the right hand side or send email to [leijr@hotmail.com](mailto:leijr@hotmail.com).

Figure.5.18.Partial Snapshot of Help



## 6. Testing

### 6.1. Chapter Overview

The chapter describes the test methods and analyzes the test results. Recommended improvements are given according to the test results analysis and evaluation.

### 6.2. Test Description

Automated Test Tool TestComplete would be used to test functions and performance of the system. TestComplete can record actions when user interacts with system, then these recorded actions can be played back when tester need to perform the functional and performance test of the system. Results and performance statistics diagrams would be automatically created by TestComplete.

### 6.3. Test Environment

One laptop, which set up with mobile AMD Athlon™ XP-M0+ was used as testing machine. Other configurations are:

CPU: x86 Family 1600 Mhz

Memory: 256 MB

Operation System: Window 2000 Server (5.0.2195 Service Pack 4)

Test Tool: TestComplete 4 Demo

### 6.4. Functional Test

Functional Tests includes Storage Function Test and Search Function Test. Each of these tests are composed of several tasks. Each task tests one function separately, such as physical object, activity, location storage, keyword search, description and relationships search. A serial of actions are recorded in one task. For example, submitting identifier and description to saving page. Example of tasks lists is showed in Figure.6.1.

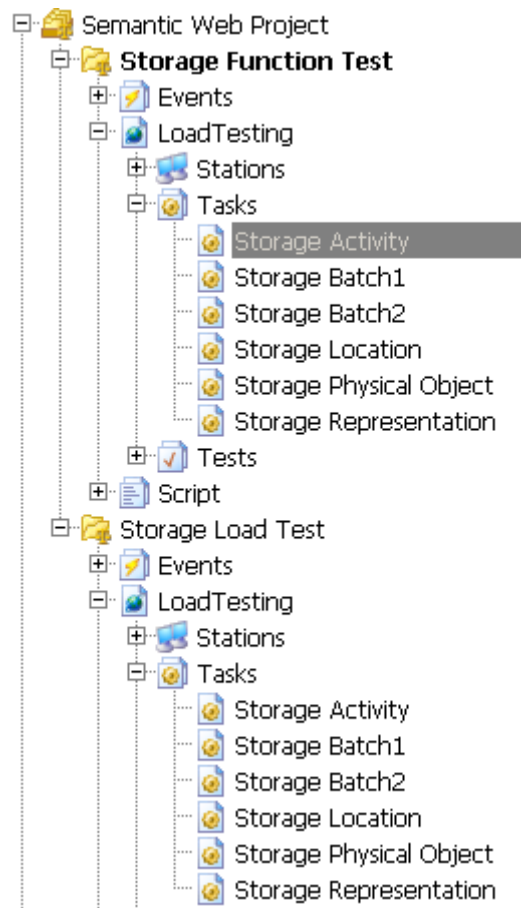


Figure.6.1.TaskList in TestComplete

All the records could be played back by running the sub-item of “Tests” under each functional test. And before running the test, task, start delay and browser need to be configured. Figure.6.2 Test Configuration shows one of the examples.

Task	Workstation	Browser	Start Delay
Search 1 Keyword	Master	Internet Explorer 6.0	5 ms

Figure.6.2 Test Configuration

The record could be used for regression test of further development. However, manual testing is still necessary, since the TestComplete just provides simple functions for Functional Test and it cannot evaluate the correctness of the test results.

## 6.5. Performance Test

Performance Test uses the same tasks with functional test and has similar operations except that it needs setting user number. The number of simultaneous users can be set to maximum of 5 users, since we used the demo version of TestComplete. The general

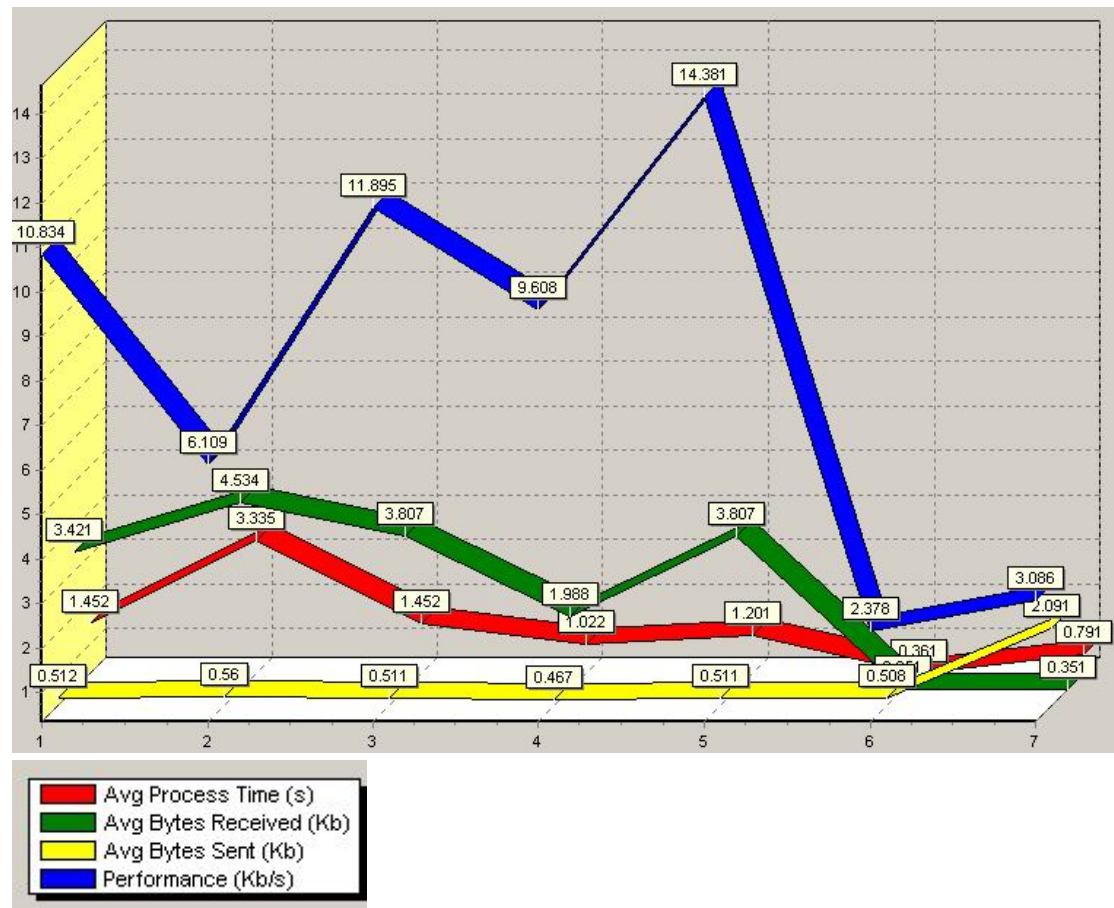
purposes of performance test include testing workload of certain parts, demonstrating system meets performance criteria, addressing the possible causes of the performance bottleneck. However, we used the same machine to act as server and client, it can not tell how much of the hardware workload, such as CPU, memory usage, caused by server and client separately, since most of the test tools just calculate the overall workload change in testing machine. And for the same reason, the performance measure is not accurate under current testing environment. But we could still find the possible causes of bottleneck by comparing the performance of different software components, because impacts from other elements are the same to all software components and software always contribute most to bottleneck rather than hardware. Performance of storage and search components would be compared and analyzed separately in the following sections.

### 6.5.1. Storage Components

Performance Results of storage components tested by one user are showed as Figure.6.3. One User Testing Results for Storage

As we can see from the figure, activity storage has the lowest performance than other components, such as physical object, representation and location, which using the same storage methods. The major difference between activity storage and other storage components might be activity storage has more parameters with multiple values, for example, contributor role, contributor name, object relationship, object name, location type and location name. These values are combined with tag before passed as a parameter to saving page, and then separated by the saving page according to the tag position. This data storage method might become the bottleneck of the system when users input many items with multiple values.

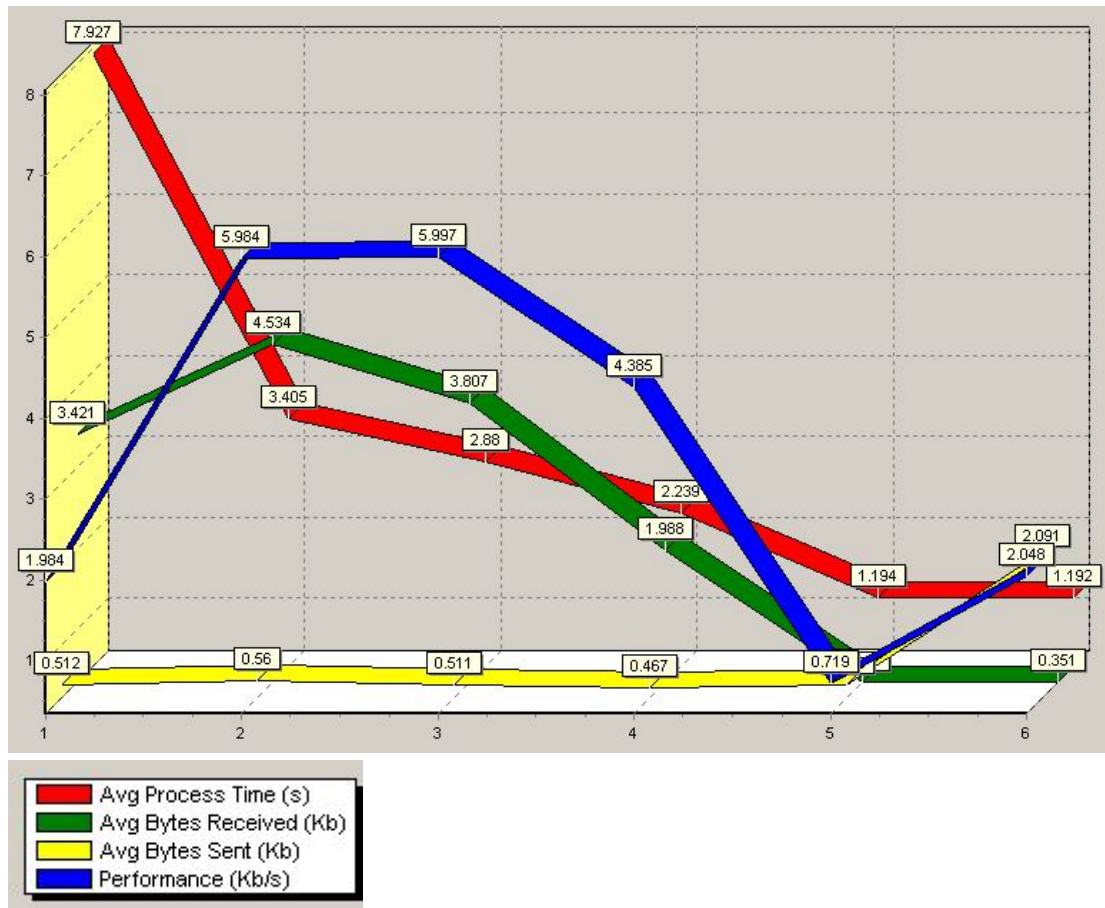
Batch storage has much lower performance than single entity storage, since it need to process files in hard disk. Batch text has lower performance than batch file saving, because some special characters in RDF Text, such as &, #, \n, conflict with the reserved character. So these characters will be transferred to the corresponding URL coding, such as '%26', '%23', '%0A' before posted to batchsave.jsp page. Additional workload comes from this transfer process and also the process of printing the RDF Text to file in hard disk.



1-Physical Object 2-Activity 3-Representation 4-Location 6- Batch Text 7- Batch File  
Figure.6.3.One User Testing Results for Storage

Performance Results of five users simultaneous testing are showed as Figure.6.3.five User Testing Results for Storage

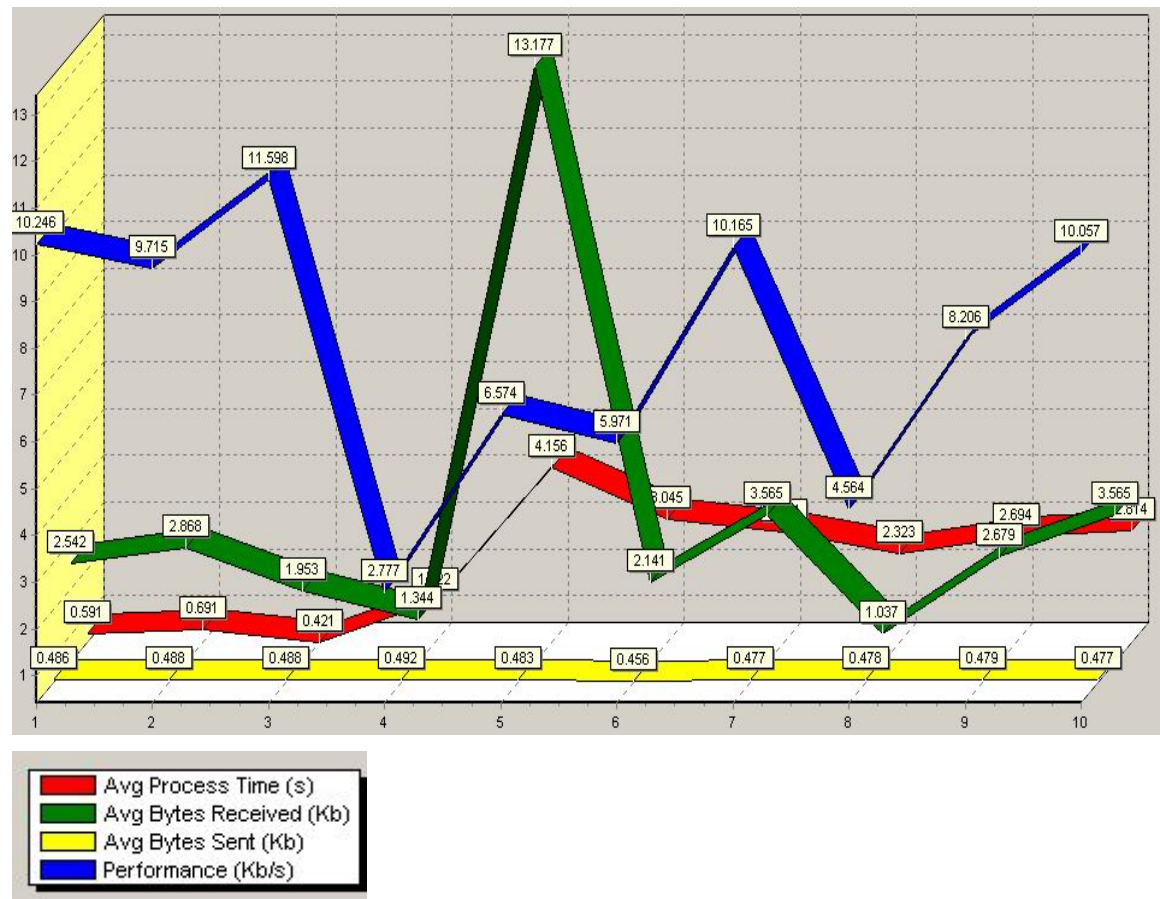
Physical object storage performs worst in multiple users simultaneous testing among single entity storage functions.



1-Physical Object 2-Activity 3-Representation 4-Location 5- Batch Text 6- Batch File  
Figure.6.4.Five User Simultaneous Testing Results for Storage

### 6.5.2. Search Components

Performance Results of search components tested by one user are showed as Figure.6.5.One User Testing Results for Search



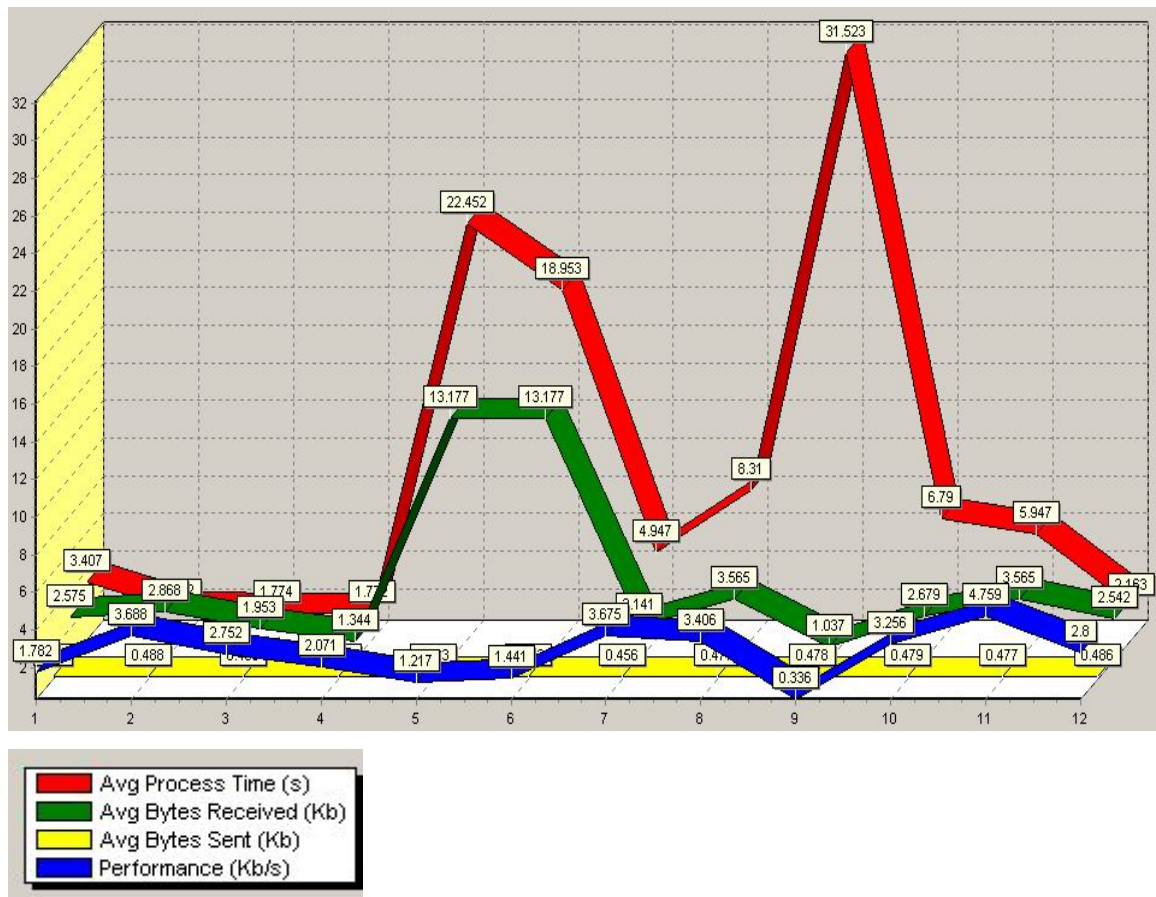
1-1 Keyword, 2-2 Keyword (or), 3-2 Keyword (and), 4-3 Keyword (and), 5-1 Keyword (long results), 6-Resource Page (long descriptions), 7- Resource Page (short descriptions), 8-Resource Page (Relationship Diagram), 9- Resource Page (linked from relationship diagram), 10-Resource Page (lined from related window)

Figure.6.5.One User Simultaneous Testing Results for Search

Results 1-5 are obtained from keyword search testing. The performance falls dramatically when keyword number reaching 3. The size of returned results also affects performance. However, the major bottleneck would be the process of keyword search in *searchKeyword* class. Currently the search function in the class is too busy and need to be reconstructed.

Results 6-10 are obtained from resource page testing. The two lower performance tests are 6 (resource with long descriptions) and 8 (resource with relationship diagram). In test 8, the results in *TestComplete* show relationship diagram loads extra *CenterRela* instance each time it draw one more 2 level relationships. Because each *CenterRela* instance just can draw 1 level directly, the second level is drawn by another instance of *CenterRela* component. If *CenterRela* can be reconstructed to support 2 or more level directly, it will enhance the performance greatly.





1-1 Keyword, 2-2 Keyword (or), 3-2 Keyword (and), 4-3 Keyword (and), 5-1 Keyword (long results), 6-Resource Page (long descriptions), 7- Resource Page (short descriptions), 8-Resource Page (short relationships), 9- Resource Page (relationship diagram), 10- Resource Page (linked from relationship diagram), 11-Resource Page (lined from related window)

Figure.6.6.Five User Simultaneous Testing Results for Search

When number of testing users reaching 5, keyword and resource search with long results perform badly, so does the relationship diagram. The reasons for relationship diagram have been discussed in one user testing. The reasons for bad performance in long results testing might due to the data structure of returned List from application components. Currently nesting Lists are used for returning results to interface model, which leaves the separation burden to interface and cause large workload. The structure of the data that communicating between application component and interface model, need to be reconstructed. XML data, which is the original design, would be a solution.

## 6.6. Test Assessment and Recommended Improvements

Although the performance test results are not very accurate under current testing

environment, we can still estimate majority of the process time is under five seconds, which meets the user psychological need for webpage speed of eight seconds, plus or minus two. That means we could assume the system can provide satisfying responding speed. However, based on the analysis of performance results, there are many bottlenecks and improvement space in software components. Although some of the reasons of bottleneck have not been determined, such as physical object storage in multiple users testing, several improvement methods could be provided according the analysis.

1. Reconstruct data that communicating between application component and interface model with XML.
2. Redesign *CenterRela* component for supporting 2 or more level directly.
3. Reconstruct the format of multiple value parameters between storage input and saving page or use other saving method.
4. Redesign the search function in *searchKeyword* class.



## 7. Configuration and User Guide

### 7.1. Installation and Configuration

#### 7.1.1. Tested Environment

Currently the system has been tested under the environment below.

Hardware: CPU 1600Mhz, 256 MB RAM

Operation System: Window 2000 Server

Web Browser: Internet Explorer 6.0

#### 7.1.2. Server Installation

**Required Software:** Tomcat5.0, MySQL

**Configuration:**

1. Install Tomcat5.0.
2. Open web.xml file in \conf directory; add  
    <welcome-file>index.jsp</welcome-file> in <welcome-file-list>.
3. Copy swm.xml file to \conf\Catalina\localhost directory.
4. Copy swm folder to \webapps directory.
5. Install MySQL.
6. Add Database-jenatest, User-test in MySQL

Database Server URL, User Name, Password, Model Name, schema and rules address can be changed in swm/Util/ PageImport.jsp.

Default Value and Example:

```
DB_URL = "jdbc:mysql://localhost/jenatest"; // URL of database server
DB_USER = "test";                          // database user id
DB_PASSWD = "";                            // database user id
modelName = "batchmodel";                  // Model Name in Database
schemaUri = "./webapps/swm/RDF/swm.rdfs"; // Schema address
ruleUri = "./webapps/swm/RDF/batch.rules"; // rules address
```

#### 7.1.3. Library Installation

**Required Software and Library:** Java 1.5 or above, All library of Jena, Jdom.jar, Rome.jar, mysql-connector-java-3.2.0.jar or above.

**Configuration:**

1. Download and install Java 1.5 or above; Copy dt.jar and tools.jar from Java \lib to Tomcat 5.0\common\lib.
2. Download Jena package; Copy all library from Jena\lib to Tomcat 5.0\common\lib.
3. Download Jdom.jar, Rome.jar, mysql-connector-java-3.2.0.jar or above. Copy all these jar files to Tomcat 5.0\common\lib.

4. Copy swm.jar to Tomcat 5.0\common\lib.

#### 7.1.4. Test Installation

Start MySQL and Tomcat service. Try URL <http://localhost:8080/swm/> in your browser. You would see the introduction page, if the system installed correctly.

You can load the swm/RDF/batch.rdf file in Batch Storage page to enter some data to the system. Searching keyword “Textile silk” in Search page and following the link from the results can give you an overview of the search function. You can also enter some items using the example data, which showed on top of each text table, in Physical Object, Activity, Representation or Location Storage pages.

You can install TestComplete and open Semantic Web project in test files to test functions and performance of the system.

### 7.2. User Guide

#### 7.2.1. Structure

The Semantic Web for Museums system is composed of Introduction, Storage, Search, User Guide and Help Section, as seen from the top menu of all the pages.

#### 7.2.2. Introduction

This page gives you an overview of the Semantic Web project. It introduces the semantic web technology, the motivations, background and objectives of the Semantic Web Project.

#### 7.2.3. Storage

There are four types of entity input in Storage section. Physical Objects represent three dimensional objects or substances; Activities concern museum and object history; Representations are Information objects, such as documents, images, videos or audio files, are used to record, depict, design or document the physical objects, activities, locations; Locations mean the places that activities occur or physical objects locate at. Batch input is for saving batch or several entity items with RDF file or text at one time in repository. Batch input increases the interoperability with other semantic systems and provide a convenient transfer way to existing data.

##### 7.2.3.1. Physical Object

The compulsory input items in physical object are identifier and controlled type.

**Identifier:** The unique name or identity of object.

**Controlled Type:** The compulsory classification for all the physical objects.

**Type:** Customised type for classification. It can have multiple values.

**Title:** The name or alternative name for the objects. (multiple values)

**Description:** Description about the object. (multiple values)

**Subject:** The related subject or topic of the object. (multiple values)

**Current Location:** The place that the physical objects currently locates. It can be narrow as individual drawer, such as d8-c2-r7-f6-wn-benaki, which means drawer d8-c2-r7-f6-wn in museum Benaki.

**Creator:** The creator of object. (multiple values)

**Created Date:** The time span or date that the object was created.

**Extent:** The physical size or measure of the object. (multiple values)

**Medium:** The material or composition that the object is made of. (multiple values)

For the items above that can have multiple values, users can press “+” button to add more rows in the text table to enter multiple values, or press “-“ button to delete the rows.

#### 7.2.3.2.Activity

The compulsory input items in physical object are identifier and controlled type.

**Identifier:** The unique name or identity of activity.

**Controlled Type:** The compulsory classification for all the activities.

**Type:** Customised type for classification. It can have multiple values.

**Method:** The methods or techniques used in the activity. (multiple values)

**Description:** Description about the activity. (multiple values)

**Subject:** The related subject or topic of the activity. (multiple values)

**Activity Location:** The place that the activity occurred. (multiple values)

**Creator:** The creator of activity. (multiple values)

**Contributor:** The contributor of the activity, such as participant, assistant, object creator or new keeper. The point need to note is that object creator is different from activity creator. (multiple values)

**Begin Date:** Begin date of the activity.

**End Date:** End date of the activity.

**Related Object:** The activity may transfer, create, load, acquire or exhibit certain object. Object identifier can be input here. (multiple values)

**Object Location:** The former or new location of the transferred, acquired or loaded object. (multiple values)

#### 7.2.3.3.Representation

The items in here are similar to Physical Object, except one additional item:

**Related Object:** The representation may show, document or record the object , activity or location. Entity identifier can be input here. (multiple values)

#### 7.2.3.4.Location

It has the items of Identifier, Controlled Type, Type, Title, Description, Location, which all have the similar meaning with the ones in physical object.

#### 7.2.3.5.RSS

Each time the new item is entered, the identifier, description and link of it would be added to the RSS files. Users can read the notification about the new item and open the resource page through the link using RSS software such as FeedReader. Alternatively, users can read the RSS file directly by clicking the “RSS” menu and opening the RSS file window.

#### 7.2.3.6.Batch

Batch input increases the interoperability with other semantic systems and provide a convenient transfer way to existing data.

RDF file and text input are provided. System will validate the RDF format of the file and text before storing them. The notification of “Successfully saved” or “Invalid RDF File/Text” alert will be given to users after the operation.

### 7.2.4. Search

The functions in search include search keyword, related resource, resource detail, relationships diagram, show RDF and print. Keyword textbox is located at the top of all pages in search function.

#### 7.2.4.1.Search Keyword

It supports multiple keywords and or/and condition. In the keyword textbox users can input multiple keywords separating with one or more space, and choose or/and condition to express the relationship between these keywords.

The search results are classified with resource types, including physical object, activity, representation and location. Not only showing the description of the returning resource, it can provide the relationships between keywords and resource, which give users the helpful relevancy to locate the search target.

#### 7.2.4.2.Search Detail

If users click on the link of returning resource of keyword search, the resource page would display under the keyword textbox replacing the search results. The details in resource page include all the properties of the resource, and image if the resource is

representation.

#### 7.2.4.3. Search Related Resource

Related Resource will be showed on the right hand side of resource page. The contents include arrow expressing relationship direction, relationship title and the identifier of related resource. Each item can be clicked to navigate to the corresponding resource page.

#### 7.2.4.4. Relationships Diagram

If users want to get into more details about the relationships or obtain a visual view of the relationships level, they can click the “1 level” or “2 levels” button to draw the relationships diagram. Function of cleaning is provided to remove the diagram when users do not need it on screen.

The diagram is composed of top resource, relationship direction arrow, relationship title, identifier of related resource and flow branch line to link these items together. The related resource identifier is wrapped by a yellow rectangle, which can be double clicked to navigate to the resource page.

#### 7.2.4.5. Show RDF

Users can click the “Show RDF” button under the resource details to show all the RDF data in the resource page. This RDF data can be transferred to other semantic web system directly. Or you can try to input it to batch storage assuming it is the data from other museums system to experience the data interoperability.

#### 7.2.4.6. Print

It provides the printing function to resource properties and relationship diagram. Related resource window would be hidden when printing. Relationship diagram will be printed if it is showed. Users can clear the diagram if they don’t want to print it.

### 7.2.5. User Guide

You must know it because you are in this section now.

### 7.2.6. Help

Provides the tips or FAQ to users.

### 7.3. Help

The thing you may not know about the system:

1. When you input the RDF data from resource page to Batch Storage, remember to use the right RDF format. RDF data displayed on web browser has additional “-“sign before each resource, it has to be removed before entering the RDF textbox.
2. The yellow rectangle in the relationship diagram can be moved around in the web page to adjust the view and position of the diagram branch or to just have fun.
3. You can get a larger image when you click the image on the representation resource page.
4. You can load the swm/RDF/batch.rdf file in Batch Storage page to enter some data to the system. Searching keyword “Textile silk” in Search page and following the link from the results can give you an overview of the search function. You can also enter some items using the example data, which showed on top of each text table, in Physical Object, Activity, Representation or Location Storage pages.
5. You can install Feed Reader to read the RSS file.
6. You can install TestComplete and open Semantic Web project in test files to test functions and performance of the system.
7. If you are interested in the theory and design methods behind the system, please go to the project home through the link on the top menu of the right hand side or send email to [leijr@hotmail.com](mailto:leijr@hotmail.com).

## 8. Conclusions and Discussion

In this project semantic web technologies have been researched to create intelligent cultural heritage archive systems and promote collaboration and knowledge sharing among museums. There might have many possible ways to implement semantic web system. The base elements of semantic web, metadata, RDF and ontology, are used in the design and implementation of a collection management prototype system. Existing metadata standard Dublin Core, data model and ontology CIDOC CRM are chosen. The SWM vocabulary and schema are designed for better expression and interoperability. In the current prototype system the museums collections could be input and transferred to RDF data for preservation. And data could be presented to users with concrete relationships. The system also provides the convenient way for data exchange with other systems and existing data transfer. The prototype system shows the benefits of semantic web technologies, such as relationships expression, interoperability, information integration and application independence, which could not be provided by text documents or database record systems.

According to C. Daconta, et al (2003), there are four smart levels of data. At the first level, data is proprietary to Text Documents or Database Records, so Data just can be used in a single application; in the second level data is expressed with single domain vocabulary, then it can be moved between applications in a single domain; data can be composed from multiple domains in the third level, since data is documented using mixed vocabularies; at the last level, ontology and rules are used with data, new data can be inferred from existing data by logical rules and data is now smart enough to be described with concrete relationships. (C. Daconta, et al 2003)

If we evaluate the system with the four levels above, it is in the half way of the fourth level. The data in the system is smart enough to be described with concrete relationships, but we cannot infer new data from existing data. The reasons for that are we are using RDFS, the lightweight ontology language, which cannot express enough constraints and rules and using subset of ontology, which has not rich relationships. So there are not enough constraints, rules and relationships to infer new data.

In summary we can see semantic web technologies still has large potential benefits to offer. We should continue to research these technologies and apply them to cultural heritage knowledge preservation for museums community.

## 9. Future Work

The long terms objective of the project is to research the promising semantic technology for creating the knowledge management network among museums. Based on the current limitation of the prototype system, the further development could be redesigning ontology with robust ontology language, which can provides more constraints and rules for inference, applying more model content in the system to increase the capability of relationship expression. The possible research topic could be sub-domain ontology (see also 4.6), web service and taxonomy in semantic web. “Web services are software applications that can be discovered, described, and accessed based on XML and standard Web protocols over intranets, extranets, and the Internet.” (C. Daconta et al 2003) It could provide more application integration and interoperability solutions to semantic web. Although taxonomy has weaker semantics in ontology spectrum (C. Daconta et al 2003), its application is necessary to museums system for the hierarchic classification it provides. The semantic web technology is such an interesting area and there are so many uncertainties and possibilities. The future of it is unknown to us. But the contribution during its development must be valuable.



## Appendix A. Mapping Table

In the mapping table below, if the content in the SWM column exists, then it means Dublin Core element could not express completely the meaning of CRM or using SWM vocabulary is more exercisable for application. In this case, SWM vocabulary would be used instead of Dublin Core element, otherwise Dublin Core element would be chosen.

### A.1.Entity

The entity component is the super-class of all other components, such as object, representation, activity, location and condition assessment. The properties of entity are adoptable to all its sub-class.

Meaning for the application	CRM Vocabulary	Dublin Core elements and qualifiers (DCMI 2005)	SWM Vocabulary
The super-class of object, representation, activity, location and condition assessment.	<i>E1.CRM Entity</i>		<i>SWM: Entity</i>
The unique identity for entity	<i>P47B.identifies, P47F.is_identified_by, E42.Object_Identifier</i>	<i>DC:Identifier</i>	
The name or title for entity	<i>P1B.identifies P1F.is_identified_by E41.Appellation</i>	<i>DC:Title</i>	
The description for entity	<i>P3F.has_note</i>	<i>DC:Description</i>	
The creator of entity if it has	<i>E39.Actor, P108F.has_produced</i>	<i>DC:Creator</i>	
The created date of entity if it has	<i>P82F.at_some_time_within, E52.Time-Span</i>	<i>DCTerms:created</i>	
The subject of entity, eg Lament. It could act as one of the keywords for search.		<i>DC.Subject</i>	

**Table.A.1.Entity Mapping**

### A.2.Physical Object

Physical Object is the sub-class of entity. It inherits all the properties of entity. The mappings for physical object resource and its additional properties are listed below.

Meaning for the application	CRM Vocabulary	Dublin Core elements and qualifiers (DCMI 2005)	SWM Vocabulary
Three dimensional objects or substances, sub-class of entity	<i>E19.Physical_Object</i>		<i>SWM:PhysicalObject</i>
Category of physical object (Controlled Vocabulary: Man-made Object, ... )	<i>P2B.is_type_of</i> , <i>P2F.has_type</i> , <i>E55.Type</i>	<i>DC:Type</i>	
The current location of physical object.	<i>P54F.has_current_permanent_location</i> , <i>P55F.has_current_location</i> <i>E53.Place</i>		<i>SWM:locateAt</i>
The size or duration of entity, such as 815KB, 30 minutes or length 1.5m...	<i>P39B.was_measured_by</i> , <i>E16.Measurement</i> , <i>E54.Dimension</i>	<i>DCTerms:Extent</i>	
The material the physical object is made of; the physical carrier or computer application format, eg .jpeg, .doc...	<i>P45F.consists_of</i> <i>E57.Material</i>	<i>DCTerms:Medium</i>	
To identify the collection which the physical object belongs to.	<i>E78.Collection</i>	<i>DCTerms:isPartOf</i>	

**Table.A.2.Physical Object Mapping**

**RDF Example:**

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

```
<!--
```

*RDF Example for the Semantic Web for Museums Element Set*

*This Version:*

*RDF Example 1 the Semantic Web for Museums Element Set 2006/05/17*

```
-->
```

```
<!DOCTYPE rdf:RDF [<!ENTITY
```

```
swmns "http://escience.anu.edu.au/project/06S1/semanticWeb/SWM/elements/1.0/">
```

```
<!ENTITY basens "http://escience.anu.edu.au/project/06S1/semanticWeb/">
```

```

]>

<rdf:RDF
  xml:lang="en"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:DC="http://purl.org/dc/elements/1.1/"
  xmlns:DCTerms="http://purl.org/dc/terms/"
  xmlns:DCTypes="http://purl.org/dc/dcmitype/"
  xml:base="&basens;"
  xmlns:swm="&swmns;"
>

<swm:PhysicalObject rdf:about="&basens;/object/Epitaphios GE34604">
  <DC:type>Iconographic Object</DC:type>
  <DC:identifier>TA 959a</DC:identifier>
  <DC:identifier>GE 34604</DC:identifier>
  <DC:title>Epitaphios GE34604</DC:title>
  <DC:description>EpitaphiosGE34604 is instance of Iconographic
object:</DC:description>
  <DC:creator>Despoineta</DC:creator>
  <DCTerms:created>1682</DCTerms:created>
  <DC:subject>ecclesiastical embroidery</DC:subject>
  <DC:subject>liturgical cloth</DC:subject>
  <DC:subject>Lament</DC:subject>
  <swm:locateAt rdf:resource="&basens;/location/d8-c2-r7-f6-wn-benaki"/>
  <DCTerms:extent>folio,length 1:5m,width 1:2m</DCTerms:extent>
  <DCTerms:medium>silver thread</DCTerms:medium>
  <DCTerms:medium>gold thread</DCTerms:medium>
  <DCTerms:medium>silk cloth</DCTerms:medium>
  <DCTerms:isPartOf rdf:resource="&basens;/collection/Post-Byzantine"/>
</swm:PhysicalObject>
</rdf:RDF>

```

### A.3.Representation

Representation is the information objects, such as documents, images, videos or audio files, used to record, depict, design or document the physical objects, activities, locations, assessment or collection. The mappings for object description are listed below. The entity properties in object properties are also adoptable for representation. Representation could be seen as a sub-class of physical object, but with different controlled vocabulary and additional properties.

Meaning for the application	CRM Vocabulary	Dublin Core elements and qualifiers (DCMI 2005)	SWM Vocabulary
Information objects, such as documents, images, videos or audio files, are used to record, depict, design or document the physical objects, activities, locations, assessment or collection.	<i>E73.Information Object, E31.Document, E29.Design and Procedure, E33.Linguistic Object, E36.Visual Item, E38.Image</i>		<i>SWM: Representation</i>
Category of representation (Controlled Vocabulary: document, Image, Video, Audio Files)	<i>P2B.is_type_of, P2F.has_type, E55.Type</i>	<i>DC:Type</i>	
To identify the entity that the representation depicts, documents or records.		<i>DCTerms.requires</i>	
The relationship of representation and the related entity. (Controlled Vocabulary: depicts, documents, records, designs, shows)	<i>P70F.documents, P138F.represents, P67F.refers_to, P129F.is_about, P65F.shows_visual_item</i>	<i>DCTerms.requires →DC.Type</i>	

**Table.A.3.Representation Mapping****RDF Example:**

```

<swm:representation
rdf:about="&basens;/representation/EpitaphiosGE34604.png">
  <DC:type>image</DC:type>
  <DC:identifier>EpitaphiosGE34604.png</DC:identifier>
  <DC:description>Produced for CIDOC CRM</DC:description>
  <DC:creator>Martin Doerr</DC:creator>
  <DC:created>10/02/1998</DC:created>
  <DC:subject>Lament Image</DC:subject>
  <DC:extent>76 kb</DC:extent>
  <DC:medium>png</DC:medium>
  <DCTerms:requires rdf:parseType="Resource"/>

```

```

<rdf:value>&basens;/object/Epitaphios GE34604</rdf:value>
<DC:type>Shows</DC:type>
</DCTerms:requires>
< /swm:representation>

```

#### A.4.Activity

The activity components represent all the activities concerning museum and object history, such as artefact creation, production, acquisition, transformation, destruction, dissolution, loan, exhibition and movement. Activity is a sub-class of entity, which means it inherits all the properties of entity. One point need to be noted is that the creator of activity is not equal to the creator of related object.

Meaning for the application	CRM Vocabulary	Dublin Core elements and qualifiers (DCMI 2005)	SWM Vocabulary
Activities concerning museum and object history	<i>E7.Activity, E8.Acquisition, E10.Transfer_of_Custody, E11.Modification_Event, E12.Production_Event, E81.Transformation, E65. Creation Event, E6.Destruction, E68.Dissolution</i>		<i>SWM: Activity</i>
Category of representation (Controlled Vocabulary: creation, production, acquisition, transformation, transfer custody, destruction, dissolution, loan, exhibition and movement)	<i>P2B.is_type_of, P2F.has_type, E55.Type</i>	<i>DC:Type</i>	
The location of activity; former or later location of object movement.	<i>P7F.took_place_at, P8F.took_place_on_or_within, P53F.has_former_or_current_location,</i>		<i>SWM:locateAt</i>

The type of location. (Controlled Vocabulary: activity location, movement former location, movement new location	<i>P2B.is_type_of,</i> <i>P2F.has_type, E55.Type</i>	<i>SWM:locateAt → DC:</i> <i>C:Type</i>	
The date that the activity occurs.	<i>P8F.took_place_on_or_w</i> <i>ithin, E4.Period</i>	<i>DC:Date</i>	
The date that the activity begins.	<i>P8F.took_place_on_or_w</i> <i>ithin, E4.Period</i>	<i>DC:Date</i>	<i>SWM:beginDate</i>
The date that the activity Ends.	<i>P8F.took_place_on_or_w</i> <i>ithin, E4.Period</i>	<i>DC:Date</i>	<i>SWM:endDate</i>
The persons participate in the activities as different roles, such as object creator, object producer, assistant, former owner...	<i>P11B.participated_in,</i> <i>E39.Actor</i>	<i>DC:Contributor</i>	
Role of participant. (Controlled Vocabulary: object creator, object producer, participant, assistance, former curator, new curator, former owner, new owner, donatory, donator, assess)	<i>P2B.is_type_of,</i> <i>P2F.has_type, E55.Type</i>	<i>DC:Contributor →</i> <i>DC:Type</i>	
The relationship of the related object and the activity.	<i>P16F.used_specific_obje</i> <i>ct</i>	<i>DCTerms: requires</i>	
(Controlled Vocabulary: used, created, acquired, transferred, destruct, dissolve, loan)	<i>P2B.is_type_of,</i> <i>P2F.has_type, E55.Type</i>	<i>DCTerms:</i> <i>requires → DC:Type</i>	
The technology used in the activity.	<i>P32F.used_general_tech</i> <i>nique,</i> <i>P33F.used_specific_tech</i> <i>nique</i>	<i>DC:Instructional</i> <i>Method</i>	

**Table.A.4.Activity Mapping****RDF Example:**

*<swm:activity rdf:about="&basens;/activity/ Creation of Epitaphios GE34604" >*

```

<DC:type>production</DC:type>
<DC:identifier>Creation of Epitaphios GE34604</DC:identifier>
<DC:instructionalMethod>handwork</DC:instructionalMethod>
<DC:description> information is derived from the inscription</DC:description>
<swm:locationAt rdf:resource="&basens;/location/Istanbul"/>
<swm:beginDate>1682</ swm:beginDate>
< swm:endDate>1682</ swm:endDate>
<DC:creator>Georgios, son of Kyriazes</DC:creator>
<DC:contributor rdf:parseType="Resource" >
  <DC:type>object creator</DC:type>
  <rdf:value>Despoineta</rdf:value>
</ DC:contributor>
<DCTerms:requires rdf:parseType="Resource" >
  <DC:type> created</DC:type>
  <rdf:value>&basens;/object/Epitaphios GE34604</rdf:value>
</ DCTerms:requires>
</ swm:activity>

<swm:activity rdf:about="&basens;/activity/ Transfer of Epitaphios GE34604
" >
  <DC:type>transfer custody</DC:type>
  <DC:identifier> Transfer of Epitaphios GE34604</DC:identifier>
  <swm:locationAt rdf:resource="&basens;/location/Greece"/>
  <swm:locationAt rdf:parseType="Resource" >
    <DC:type>former location</DC:type>
    <rdf:value>&basens;/location/d0-c4-r8-f6-Metropolitan</rdf:value>
  </ swm:locationAt>
  <swm:locationAt rdf:parseType="Resource" >
    <DC:type>new location</DC:type>
    <rdf:value>&basens;/location/d8-c2-r7-f6-wn-benaki</rdf:value>
  </ swm:locationAt>
  <swm:beginDate>1923</ swm:beginDate>
  < swm:endDate>1928</ swm:endDate>
  <DC:contributor rdf:parseType="Resource" >
    <DC:type>former keeper</DC:type>
    <rdf:value>Metropolitan Church of the Greek Community of Ankara
    </rdf:value>
  </ DC:contributor>
  <DC:contributor rdf:parseType="Resource" >
    <DC:type>new keeper</DC:type>
    <rdf:value>Museum Benaki</rdf:value>
  </ DC:contributor>
  <DCTerms.requires rdf:parseType="Resource" >
    <DC:type>transferred</DC:type>

```

```

    <rdf:value>&basens;/object/Epitaphios GE34604</rdf:value>
  </DCTerms.requires>
</swm:activity>

```

## A.5.Location

Location is a sub-class of entity. It inherits the properties of entity and has an additional property of “locateAt”, which expressing the location level, such as institute falling into city, city being part of province...

Meaning for the application	CRM Vocabulary	Dublin Core elements and qualifiers (DCMI 2005)	SWM Vocabulary
The place that activity occurs or physical object locates at.	<i>E53.Place</i>		<i>SWM:Location</i>
Express the relationship of activity, object and place.	<i>P7F.took_place_at,</i> <i>P8F.took_place_on_or_within,</i> <i>P53F.has_former_or_current_location</i>		<i>SWM:LocateAt</i>

**Table.A.5.Location Mapping**

### RDF Example:

```

<swm:location rdf:about="&basens;/location/d8-c2-r7-f6-wn-benaki">
  <DC:type>drawer</DC:type>
  <DC:identifier>d8-c2-r7-f6-wn-Benaki</DC:identifier>
  <swm:locationAt rdf:resource="&basens;/location/Museum Benaki"/>
</swm:location>

<swm:location rdf:about="&basens;/location/Museum Benaki">
  <DC:type>institute</DC:type>
  <DC:identifier>Museum Benaki</DC:identifier>
</swm:location>

```

## A.6.Condition Assessment

Condition assessment could be regarded as a special case of activity, which having the



additional properties of assessment result, assessed object, issued date, valid date and assessment document.

Meaning for the application	CRM Vocabulary	Dublin Core elements and qualifiers (DCMI 2005)	SWM Vocabulary
The condition assessment of physical object.	<i>E14 Condition Assessment</i>		<i>SWM:condition Assessment</i>
To identify the object that being assessed.	<i>P34F.concerned</i>	<i>DCTerms.requires</i>	
To identify the assessment document	<i>P67F.referred_to</i>	<i>DCTerms.references</i>	
The result of assessment or condition of object (Controlled Vocabulary: good, fair, stability, blemishes, repairs, completeness)	<i>P44F.has_condition</i>		<i>SWM:assessResult</i>
The assessment issued date.		<i>DC.Issued</i>	
The assessment valid date		<i>DC.Valid</i>	

**Table.A.6.Condition Assessment Mapping**

**RDF:Example:**

```

<swm:conditionAssess
rdf:about="&basens;/condition/2005-10-02-Assess-Epitaphios GE34604>
  <DC:identifier>2005-10-02-Assess-Epitaphios</DC:identifier>
  <DC:issued>2005-10-02</DC:issued>
  <DC:valid>2006-10-02</DC:valid>
  <DC:contributor rdf:parseType="Resource">
    <DC:type>assess</DC:type>
    <rdf:value>Martin Doerr</rdf:value>
  <swm:assessResult>stability</swm:assessResult>
  <DCTerms:requires rdf:resource="&basens;/object/Epitaphios GE34604"/>
  <DCTerms:references
rdf:resource="&basens;/representation/2005-10-02-Assess-Epitaphios
GE34604.doc"/>
</swm:conditionAssess>

```

## A.7.Collection

Collection is sub-class of entity. It is used for categorize the physical object.

Meaning for the application	CRM Vocabulary	Dublin Core elements and qualifiers (DCMI 2005)	SWM Vocabulary
Collection used for categorize the physical object or artefact.	<i>E78 Collection</i>		<i>SWM:Collection</i>
Record the current and former curator of the collection	<i>P109F.has_current_or_former_curator</i>	<i>DC:Contributor</i>	
(Controlled Vocabulary: former curator, current curator)		<i>DC:Contributor</i> → <i>DC:Type</i>	
The date the former/current curator begin to act as curator.			<i>DC:Contributor</i> → <i>SWM:beginDate</i>
The date the former/current curator stop acting as curator.			<i>DC:Contributor</i> → <i>SWM:endDate</i>

Table.A.1.Collection Mapping

## Appendix.B. SWM RDF Schema

```
<?xml version="1.0"?>
<!--
RDF Schema declaration for the Semantic Web for Museums Element Set
This Version:
    RDF Schema declaration Draft the Semantic Web for Museums Element Set
    2006/05/10
-->

<!DOCTYPE rdf:RDF [<!ENTITY
swmns "http://escience.anu.edu.au/project/06S1/semanticWeb/SWM/elements/1.0/">
]>

<rdf:RDF
xml:lang="en"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:dctype="http://purl.org/dc/dcmitype/"
xml:base="http://escience.anu.edu.au/project/06S1/semanticWeb/"
xmlns:swm="&swmns;"
>

<!-- Description of this Schema -->
<rdf:Description rdf:about="&swmns;">
    <dc:title>RDF Schema declaration Draft the Semantic Web for Museums Element
Set 2006/05/17</dc:title>
    <dc:publisher/>
    <dc:description/>
    <dc:language>English</dc:language>
</rdf:Description>

<rdfs:Class rdf:about="&swmns;entity">
<rdfs:isDefinedBy rdf:resource="&swmns;"/>
<rdfs:label xml:lang="en">entity</rdfs:label>
<rdfs:comment xml:lang="en">
    The super-class of object, representation, activitiy, location and condition
assessment
</rdfs:comment>
```

</rdfs:Class>

<rdfs:Class rdf:about="&swmns;physicalObject">

<rdfs:isDefinedBy rdf:resource="&swmns;"/>

<rdfs:label xml:lang="en">object</rdfs:label>

<rdfs:comment xml:lang="en">

*Physical objects or artefacts, sub-class of entity*

</rdfs:comment>

<rdfs:subClassOf rdf:resource="&swmns;entity"/>

</rdfs:Class>

<rdfs:Class rdf:about="&swmns;activity">

<rdfs:isDefinedBy rdf:resource="&swmns;"/>

<rdfs:label xml:lang="en">activity</rdfs:label>

<rdfs:comment xml:lang="en">

*The activity components represent all the activities concerning museum and object history, such as artefact creation, production, acquisition, transformation, destruction, dissolution, loan, exhibition and movement.*

</rdfs:comment>

<rdfs:subClassOf rdf:resource="&swmns;entity"/>

</rdfs:Class>

<rdfs:Class rdf:about="&swmns;representation">

<rdfs:isDefinedBy rdf:resource="&swmns;"/>

<rdfs:label xml:lang="en">representation</rdfs:label>

<rdfs:comment xml:lang="en">

*Representation is the information objects, such as documents, images, videos or audio files, used to record, depict, design or document the physical objects, activities, locations, assessment or collection.*

</rdfs:comment>

<rdfs:subClassOf rdf:resource="&swmns;entity"/>

</rdfs:Class>

</rdf:RDF>

<rdfs:Class rdf:about="&swmns;location">

<rdfs:isDefinedBy rdf:resource="&swmns;"/>

<rdfs:label xml:lang="en">location</rdfs:label>

<rdfs:comment xml:lang="en">

*Location is a sub-class of entity. It inherits the properties of entity and has an additional property of "locateAt". Using the "locateAt" property, location level could be expressed, such as institute falling into city, city being part of province...*

</rdfs:comment>

<rdfs:subClassOf rdf:resource="&swmns;entity"/>

</rdfs:Class>

```
<rdfs:Class rdf:about="&swmns;conditionAssess">
<rdfs:isDefinedBy rdf:resource="&swmns;"/>
<rdfs:label xml:lang="en">conditionAssess</rdfs:label>
<rdfs:comment xml:lang="en">
```

*Condition assessment could be regarded as a special case of activity, which having the additional properties of assessment result, assessed object, issued date, valid date and assessment document.*

```
</rdfs:comment>
<rdfs:subClassOf rdf:resource="&swmns;entity"/>
</rdfs:Class>
```

```
<rdfs:Class rdf:about="&swmns;collection">
<rdfs:isDefinedBy rdf:resource="&swmns;"/>
<rdfs:label xml:lang="en">collection</rdfs:label>
<rdfs:comment xml:lang="en">
```

*Collection is sub-class of entity. It is used for categorize the physical object.*

```
</rdfs:comment>
<rdfs:subClassOf rdf:resource="&swmns;entity"/>
</rdfs:Class>
```

```
<rdf:Property rdf:about="&swmns;locateAt">
<rdfs:isDefinedBy rdf:resource="&swmns;"/>
<rdfs:label xml:lang="en">locateAt</rdfs:label>
<rdfs:comment xml:lang="en">
```

*Express the location of activity and physical object.*

```
</rdfs:comment>
<rdfs:range rdf:resource="&swmns;location"/>
<rdfs:domain rdf:resource="&swmns;activity"/>
<rdfs:domain rdf:resource="&swmns;object"/>
</rdf:Property>
```

```
<rdf:Property rdf:about="&swmns;assessResult">
<rdfs:isDefinedBy rdf:resource="&swmns;"/>
<rdfs:label xml:lang="en">assessResult</rdfs:label>
<rdfs:comment xml:lang="en">
```

*The result of assessment or condition of physical object (Controlled Vocabulary: good, fair, stability, blemishes, repairs, completeness)*

```
</rdfs:comment>
<rdfs:range rdf:resource="rdfs:Literal"/>
<rdfs:domain rdf:resource="&swmns;conditionAssess"/>
</rdf:Property>
```

```
<rdf:Property rdf:about="&swmns;beginDate">
```

```
<rdfs:isDefinedBy rdf:resource="&swmns;"/>
<rdfs:label xml:lang="en">beginDate</rdfs:label>
<rdfs:comment xml:lang="en">
  The date that the activity begins
</rdfs:comment>
<rdfs:range rdf:resource="rdfs:Literal"/>
<rdfs:domain rdf:resource="&swmns;Activity"/>
</rdf:Property>

<rdf:Property rdf:about="&swmns;endDate">
<rdfs:isDefinedBy rdf:resource="&swmns;"/>
<rdfs:label xml:lang="en">endDate</rdfs:label>
<rdfs:comment xml:lang="en">
  The date that the activity Ends
</rdfs:comment>
<rdfs:range rdf:resource="rdfs:Literal"/>
<rdfs:domain rdf:resource="&swmns;Activity"/>
</rdf:Property>


























</RDF:RDF>
```

## Appendix.C. RSS Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://purl.org/rss/1.0/"
xmlns:taxo="http://purl.org/rss/1.0/modules/taxonomy/"
xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
xmlns:dc="http://purl.org/dc/elements/1.1/">
<channel rdf:about="http://localhost:8080/swm/Storage/RDF/swm.rss">
<title>Semantic Web for Museums</title>
<link>http://localhost:8080/swm/Search/default.jsp</link>
<description>show result channel</description>
<items>
<rdf:Seq>
<rdf:li resource = "http://localhost:8080/swm/Search/pages/searchRela.jsp? Res =
Transfer of Epitaphios GE3460" />
</rdf:Seq>
</items>
</channel>


























<item rdf:about = " http:// localhost:8080/swm/Search/pages/searchRela.jsp? Res =
Transfer of Epitaphios GE3460">
<title>Transfer of Epitaphios GE3460</title>
<link>http://localhost:8080/swm/Search/pages/searchRela.jsp?Res=Transfer of
Epitaphios GE3460</link>
<description>Transfer custody of Epitaphios GE3460</description>
</item>
</rdf:RDF>
```

## Appendix.D.Timetable - Initial Timetable

ID		Task Name	Start	Finish	Duration
1		<b>Semantic Project</b>	<b>Mon 2/27/06</b>	<b>Fri 6/30/06</b>	<b>126.88 days</b>
2		<b>Requirement</b>	<b>Mon 2/27/06</b>	<b>Sun 3/12/06</b>	<b>14 days</b>
3		Requirement Analysis	Mon 2/27/06	Sun 3/12/06	14 days
4		Project Plan & Report	Sat 3/4/06	Wed 3/8/06	4.5 days
5		Initial Presentation & Report	Thu 3/9/06	Thu 3/9/06	0 days
6		<b>Knowledge Learning and Research</b>	<b>Fri 3/10/06</b>	<b>Mon 4/3/06</b>	<b>24.94 days</b>
7		Development Tools	Fri 3/10/06	Mon 4/3/06	24.94 days
8		RDF	Fri 3/10/06	Mon 4/3/06	24.94 days
9		Ontology	Fri 3/10/06	Mon 4/3/06	24.94 days
10		Repository	Fri 3/10/06	Mon 4/3/06	24.94 days
11		Usable Interface	Fri 3/10/06	Mon 4/3/06	24.94 days
12		Building Development Envirment	Mon 3/13/06	Mon 4/3/06	21.56 days
13		Design	Mon 3/13/06	Sun 4/2/06	21.19 days
14		Design Completed	Sun 4/2/06	Sun 4/2/06	0 days
15		First Version Implementation	Mon 4/3/06	Thu 4/20/06	17.75 days
16		Mid-project results due	Thu 4/20/06	Thu 4/20/06	0 days
17		Second Version Implementation	Fri 4/21/06	Wed 5/17/06	27 days
18		Test & Debug	Thu 5/18/06	Thu 5/25/06	7.56 days
19		Implementation Due	Fri 5/26/06	Fri 5/26/06	0 days
20		Document Organization	Sun 5/28/06	Sun 6/4/06	7.38 days
21		Final Presentation	Thu 6/8/06	Thu 6/8/06	0 days
22		Final Report	Tue 6/6/06	Wed 6/21/06	16.5 days
23		Final Report Due	Thu 6/22/06	Thu 6/22/06	0 days
24		Project Closeout	Fri 6/23/06	Fri 6/30/06	8.06 days



## Appendix.D.Timetable - Actual Timetable

ID		Task Name	Start	Finish	Duration
1		<b>Semantic Project</b>	<b>Mon 2/27/06</b>	<b>Fri 6/30/06</b>	<b>126.88 days</b>
2		<b>Requirement</b>	<b>Mon 2/27/06</b>	<b>Sun 3/12/06</b>	<b>14 days</b>
3		Requirement Analysis	Mon 2/27/06	Sun 3/12/06	14 days
4		Project Plan & Report	Sat 3/4/06	Wed 3/8/06	4.5 days
5		Initial Presentation & Report	Thu 3/9/06	Thu 3/9/06	0 days
6		<b>Knowledge Learning and Research</b>	<b>Fri 3/10/06</b>	<b>Mon 4/3/06</b>	<b>24.94 days</b>
7		Development Tools	Fri 3/10/06	Mon 4/3/06	24.94 days
8		RDF	Fri 3/10/06	Mon 4/3/06	24.94 days
9		Ontology	Fri 3/10/06	Mon 4/3/06	24.94 days
10		Repository	Fri 3/10/06	Mon 4/3/06	24.94 days
11		Usable Interface	Fri 3/10/06	Mon 4/3/06	24.94 days
12		Building Development Envirment	Mon 3/13/06	Mon 4/3/06	21.56 days
13		Design	Mon 3/13/06	Thu 4/13/06	32.13 days
14		Design Completed	Thu 4/13/06	Thu 4/13/06	0 days
15		First Version Implementation	Thu 4/13/06	Thu 4/27/06	14.75 days
16		Mid-project results due	Thu 4/27/06	Thu 4/27/06	0 days
17		Second Version Implementation	Thu 4/27/06	Thu 6/1/06	36.31 days
18		Implementation Due	Thu 6/1/06	Thu 6/1/06	0 days
19		Test & Debug	Thu 6/1/06	Thu 6/8/06	7.56 days
20		Document Organization	Thu 6/8/06	Thu 6/15/06	7.56 days
21		Final Presentation	Thu 6/15/06	Thu 6/15/06	0 days
22		Final Report	Thu 6/15/06	Wed 6/21/06	6.69 days
23		Final Report Due	Wed 6/21/06	Wed 6/21/06	0 days
24		Project Closeout	Wed 6/21/06	Fri 6/30/06	10.25 days

## Reference

Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal 1996, *Pattern-Oriented Software Architecture: A System of Patterns*, Wiley.

Gail Hodge 2001, *METADATA MADE SIMPLER*, NISO Press, National Information Standards Organization

CIMI 2002, *Introduction to CIMI*

Retrieved 05 April 2006 from <http://www.cimi.org/>

CIDOC 2005a, *Definition of the CIDOC Conceptual Reference Model*, Version 4.2

Retrieved 05 April 2006 from [http://cidoc.ics.forth.gr/official\\_release\\_cidoc.html](http://cidoc.ics.forth.gr/official_release_cidoc.html)

CIDOC 2005b, *CIDOC Conceptual Reference Model*, Version 4.2

Retrieved 05 April 2006 from [http://cidoc.ics.forth.gr/official\\_release\\_cidoc.html](http://cidoc.ics.forth.gr/official_release_cidoc.html)

CIDOC 2005c, CIDOC CRM graphical representation,

Retrieved 05 April 2006 from [http://cidoc.ics.forth.gr/cidoc\\_graphical\\_representation/graphical\\_representation.htm](http://cidoc.ics.forth.gr/cidoc_graphical_representation/graphical_representation.htm)

Cover Pages 2002, *ISO Prepares Reference Ontology for Interchange of Cultural Heritage Information*.

Retrieved 12 March 2006 from <http://xml.coverpages.org/ni2002-08-20-c.html>

Michael C. Daconta, Leo J. Obrst and Kevin T. Smith 2003, *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*, John Wiley & Sons

DCMI Dublin Core Metadata Initiative 1999, *Dublin Core Metadata Element Set, Version 1.1: Reference Description*

Retrieved 05 April 2006 from <http://dublincore.org/documents/dces/>

DCMI Dublin Core Metadata Initiative 2005, *Using Dublin Core - Dublin Core Qualifiers*,

Retrieved 05 April 2006 from <http://dublincore.org/documents/dces/>

Martin Doerr 2000, *Mapping of the Dublin Core Metadata Element Set to the CIDOC CRM*,

Retrieved 05 April 2006 from <http://cidoc.ics.forth.gr/reference>

Tony Gill 2003, *Is that a Reference Model in your Pocket...? The CIDOC CRM & the IFLA FRBR*.

Retrieved 05 April 2006 from [http://cidoc.ics.forth.gr/technical\\_papers.html](http://cidoc.ics.forth.gr/technical_papers.html)

Eero Hyvönen, Suvi Kettula, Vilho Raatikka, Samppa Saarela, Kim Viljanen, *Finnish Museums On the Semantic Web*.

Retrieved 12 March 2006 from <http://www.cs.helsinki.fi/group/seco/>

IFLA 2002, *Functional Requirements for Bibliographic Records (FRBR) Review Group*

Retrieved 05 April 2006 from <http://www.ifla.org/VII/s13/wgfrbr/wgfrbr.htm>

Carl Lagoze, Sandy Payette, Edwin Shin, Chris Wilper 2005, *Fedora An Architecture for Complex Objects and their Relationships*, Computing and Information Science, Cornell University

MDA Museum Documentation Association 2005, *SPECTRUM: The UK Museum Documentation Standard*

Retrieved 12 March 2006 from <http://www.mda.org.uk/spectrum.htm>

Michael Morrixon, et al. 2000, *XML Unleashed*, Sams Publishing

Sun Microsystems 2006, *ROME, RSS and atOM utilitiEs for Java*,

Retrieved 05 April 2006 from <https://rome.dev.java.net/>

BERNERS-LEE, TIM, J. HENDLER, O. LASSILA “The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities” Scientific American, 17 May 2001.

Jennifer Trant 2002, *Museums in the Web of Collaboration, the AMICO Model*, Archives Libraries and Museums, Stockholm, Sweden

Retrieved 05 April 2006 from <http://www.amico.org/docs.html>

W3C RDF Primer, Retrieved 6 March 2006 from <http://www.w3.org/TR/rdf-primer/>

W3C 2004, RDF Vocabulary Description Language 1.0: RDF Schema, Retrieved 6 March 2006 from <http://www.w3.org/TR/rdf-schema/>

W3C OWL Guide, Retrieved 6 March 2006 from <http://www.w3.org/TR/owl-guide/>

W3C Semantic Web, Retrieved 6 March 2006 from <http://www.w3.org/2001/sw/>

W3C VML, Retrieved 6 March 2006 from <http://www.w3.org/TR/NOTE-VML>

Tom Worthington 2005, Report on a Workshop on the Use of Technology for Museums of the Pacific Islands Region 2005

Kwok Chung, Yew 2005, Digital Heritage for South Pacific Museums Project Findings and Report