Grant Agreement No. ICT-2009-270082

Project Acronym PATHS

Project full title **Personalised Access To cultural Heritage Spaces**

D 3.1 Specification of System Architecture

Authors: **Stein Runar Bergheim (AVINET);**
Idar Thoresen Kvam (AVINET)

Contributors: **Phil Archer (iSieve)**
Kate Fernie (MDR)
Paul Clough (USFD)
Tor Gunnar Øverli (AVINET)
Mark Stevenson (USFD)

Project funded under FP7-ICT-2009-6 Challenge 4 – "Digital Libraries and Content"	
Status	Final
Distribution level	Public
Date of delivery	10/10/2011
Type	Report
Project website	http://www.paths-project.eu
Project Coordinator	Dr. Mark Stevenson University of Sheffield

Change Log

Version	Date	Amended by	Changes
0.1	08/06/2011	Stein Runar Bergheim	TOC
0.8	30/08/2011	Idar Thoresen Kvam	First complete draft
0.92	28/04/2011	Stein Runar Bergheim	Added section on methodology
0.93	19/09/2011	Stein Runar Bergheim	Added sections on operational considerations and security, incorporated comments from partners.
1.0	03/10/2011	Stein Runar Bergheim	Made changes following second round of input from project partners.
1.0	10/10/2011	Kate Fernie	Added executive summary

1 EXECUTIVE SUMMARY

The purpose of PATHS is to provide users with innovative ways to access and utilize the contents of digital libraries that enrich their experiences of these resources.

This deliverable describes in detail the PATHS system architecture which contains modules for (1) storage and management of data, (2) user profile management, Path authoring, content recommendation, information retrieval and presentation – all accessible through (3) a Web API invoked through lightweight client-side libraries which may be used for application development in HTML5 and JavaScript.

The system architecture was developed in accordance with the principles of the Systems Development Life Cycle management (SDLC) following a profile which allows for agile, iterative development. This document is the result of these phases:

- Project planning and feasibility study
- Systems analysis and technical requirements definition
- Systems design

The future phases of the system design include implementation, integration and testing, acceptance, installation and deployment.

This system architecture specification follows the IEEE System Design Description (SDD) and uses UML as the primary modelling language, expressing the system architecture as use-case diagrams, class diagrams and deployment diagrams.

This deliverable consists of:

Chapter 2: Introduction. This gives an overview of the system architecture and an introduction to the specification structure.

Chapter 3: Methodology. This chapter describes the approach used to develop the PATHS system architecture in terms of the methodology, the evaluation of different alternatives for the PATHS sub-systems, risks management related to the system development and conformance to functional requirements specified in PATHS deliverable D1.3.

Several design alternatives were evaluated towards five key criteria for the overall system:

1 Robustness
2 Scalability
3 Performance
4 Flexibility

In cases where sub-systems of the PATHS architecture have dependencies on existing software which might be satisfied by one or more product, several products have been tested and/or benchmarked against each other to determine which is best suited for PATHS. The key aspects which were evaluated were:

- Database
- Triple store
- Spatial visualisation engine
- Development platform

The development of the System Design Description identified a possible risk of performance problems for RDF stores with large numbers of triples. To mitigate this risk a hybrid mechanism is proposed using both a high-performance relational database management system in conjunction with a triple store. This strategy will allow applications to use the triple store to look up related concepts and then retrieve the actual data from the RDBMS.

Section 3.4 provides a cross-reference between the list of functional requirements described in PATHS deliverable D1.3 and the system architecture.

Chapter 4: Roles and responsibilities. This chapter describes the roles and responsibilities of the development resources available in the PATHS consortium, and which sub-system each partner will contribute to.

Chapter 5: System description. This chapter provides an overview of the PATHS system architecture in terms of: (1) a high-level software view; (2) a technical architecture view which shows how the software is deployed to the logical server infrastructure; (3) an overview of the physical server and network infrastructure of the system; and (4) description of communication interfaces which are employed in integrating the different layers of the architecture which may also be used as external interfaces for development of 3rd party applications on top of the PATHS application layer.

Three logical servers are required to support the system architecture:

1. A **database server** to support the operation of the PATHS system
2. A **file server** to store non-database files such as images.
3. An **application server** to provide a triple store, spatial visualisations, web server and search indexer

The hardware infrastructure is envisaged to consist of three physical servers, a hardware fire-wall and an internet router

Chapter 6: Sub-system specifications. This chapter provides detailed technical specifications of the modules of each of the sub-systems of the PATHS application layer. The sub-systems are:

- **Data integration** - which provides the interfaces needed to communicate with the data stores. Four different interfaces are offered to support the flexibility required by the system
- **User profile** – allows the creation and maintenance of user profiles, and logging of user behaviour
- **Presentation** – enables client applications to request visualisations of Paths and other items
- **Path authoring** – enables the creation of Paths (a set of items connected by narratives)
- **Information retrieval** – secures efficient indexing and searching through Paths and items using text, time and spatial search mechanisms
- **Recommender** – provides a user browsing an item with suggestions of other items to visit
- **Web Service API** - exposes the PATHS sub-systems to the Internet for integration into end-user applications

Chapter 7: Data architecture. This chapter describes the PATHS data model and data storage strategy. The three modules of the virtual data repository sub-system consist of:

- Non-database file storage of images, audio, video and enriched XML files
- RDBMS data storage for items and Paths
- RDF data store (or triple store) for vocabularies (persons, geographical names, time periods, topics, subjects etc).

Chapter 8: Security. This chapter describes how security concerns have been addressed in the system architecture including:

- User level permissions
- Control points (authentication, firewall, local user management)
- Vulnerabilities (logins and user rights permitting write access)

Chapter 9: Operational considerations. This chapter describes the procedures, validations and tests required to operate the system at the end of the Software Development Life Cycle (SDLC). It covers:

- The Audit trail put in place to trace the cause of system incidents
- Recoverability in the event of system failure
- Data retention
    - Application and web server logs are retained for two months before archiving
    - Management of 'deletions' by setting of a data type
    - Retention of 'deleted' user accounts for three months
- Conventions and standards deployed through the development and implementation of the PATHS system architecture.

References to other relevant PATHS project deliverables.

Appendix A provides an acronym list and glossary

In conclusion, this deliverable describes the PATHS technical system architecture intended to support systems to enrich the experience of users who are accessing content from very large digital libraries.

Its scope is the technical system architecture and not the end-user requirements, client-layer applications, end-user interfaces and processing of raw data which are the subject of separate deliverables.

2 INTRODUCTION

The purpose of PATHS is to provide users with innovative ways to access and utilize the contents of digital libraries that enrich their experiences of these resources.

In the first instance achieving this objective means identifying the state-of-the-art of current applied technology in the domain of digital libraries systems. This will establish the baseline from which the PATHS system must evolve. Second user requirements must be gathered and analysed in order to project which new functionality should be supported by PATHS user interfaces and third, a system must be implemented which supports all necessary entry, indexing, retrieval and display mechanisms required to drive the user interfaces sought by the users.

This System Design Documentation describes in detail the PATHS system architecture. A system which contains modules for (1) storage and management of data, (2) modules for user profile management, Path authoring, content recommendation, information retrieval and presentation – all accessible through (3) a Web API which may be invoked through lightweight client-side libraries which may be used for application development in HTML5 and JavaScript.

2.1 Overview

This system architecture specification follows the IEEE System Design Description (SDD) and uses UML as the primary modelling language, expressing the system architecture as use-case diagrams, class diagrams and deployment diagrams.

The different elements of the system are described as a hierarchy:

- The PATHS system consists of several sub-systems which are responsible for distinctive parts of the system. Systems and sub-systems are abstract and only used to logically group the functionality.
- A sub-system consists of one or more modules which interact to provide the functionality required from the sub-system. Modules are physical pieces of software.
- A module, if complex, may be broken down into components for the ease of description.

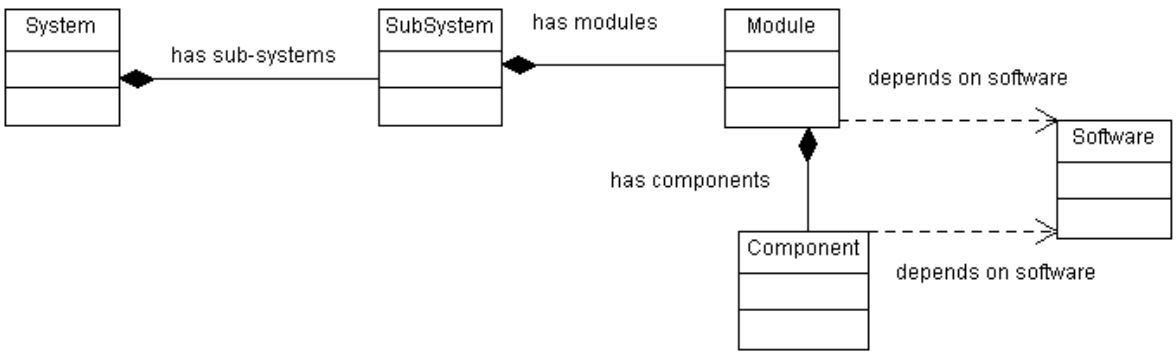

Figure 2.1-1 Hierarchy of elements described in the SDD document.

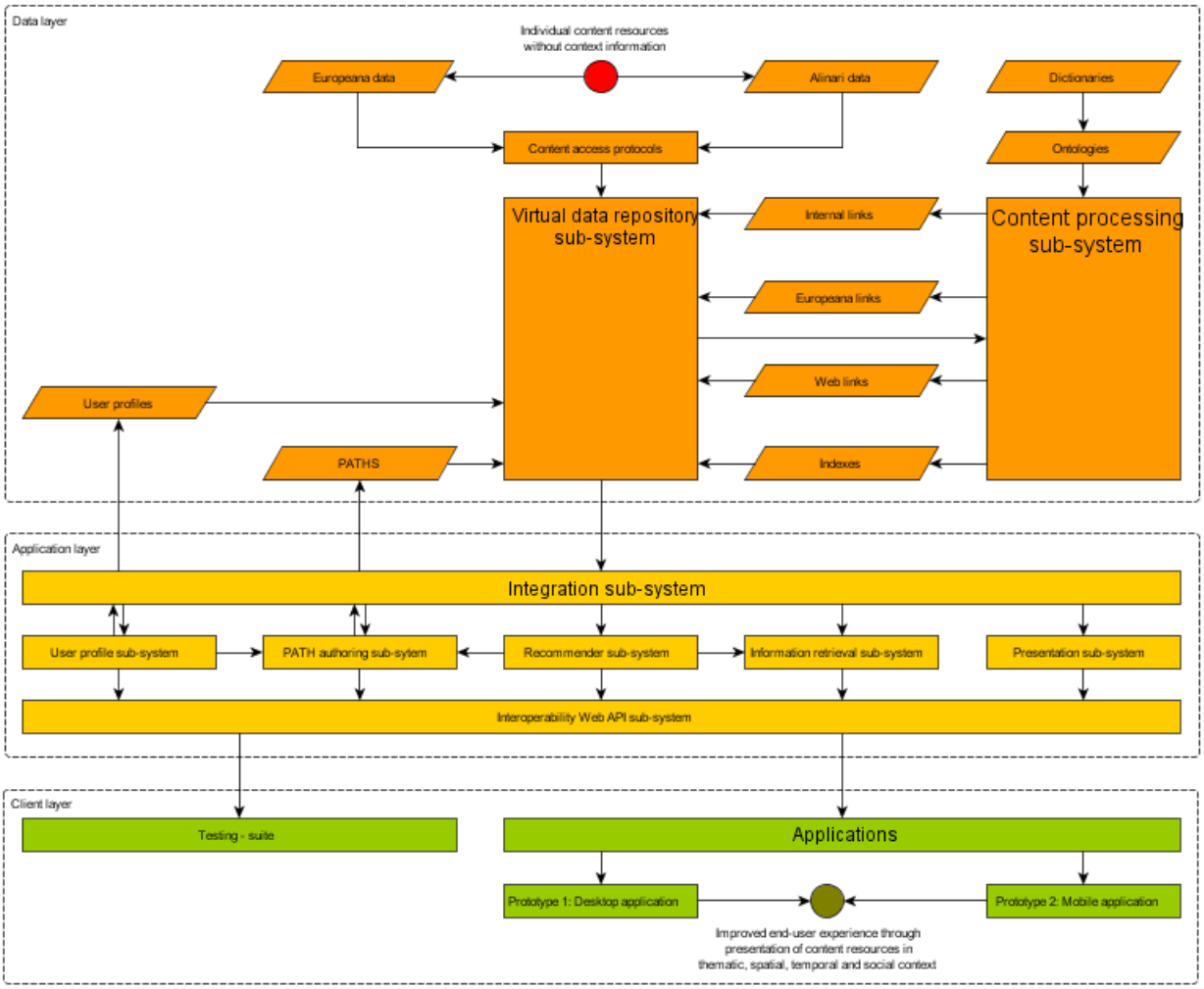

Figure 2.1-2 Preliminary system architecture as outlined in PATHS description of work.

2.2 Scope

The scope of this document is the technical system architecture for the application and data layers of the PATHS system.

The System Design Description does not cover the following:

- End-user requirements
- Client layer applications
- End-user interfaces
- Processing of raw data from PATHS data providers Europeana and Alinari

These elements are described in detail in deliverables D1.3, D2.1 and D4.1. See **Error! Reference source not found.**

2.3 Background

The PATHS system is built as part of a FP7 ICT research project and is meant to result in technically sophisticated prototypes which validate innovative paradigms for indexing, enrichment, retrieval and presentation. This allows for some flexibility in terms of the system architecture which will be geared more towards enabling demonstration of potential than towards making a deployment ready solution which effortlessly should integrate with any data source. It is however a sustainability requirement that the project results in a solution which may be deployed to additional cultural heritage platforms, including Europeana, towards the end of the project.

The main challenge which the PATHS system addresses is the ability to enrich the experience of users who are accessing content form large digital libraries like Europeana. In the latter digital library, the content is sourced from hundreds of cultural heritage institutions across Europe and provides detailed descriptions of individual cultural heritage objects held in these collections in the form of Dublin Core (baseline) or more semantically rich metadata.

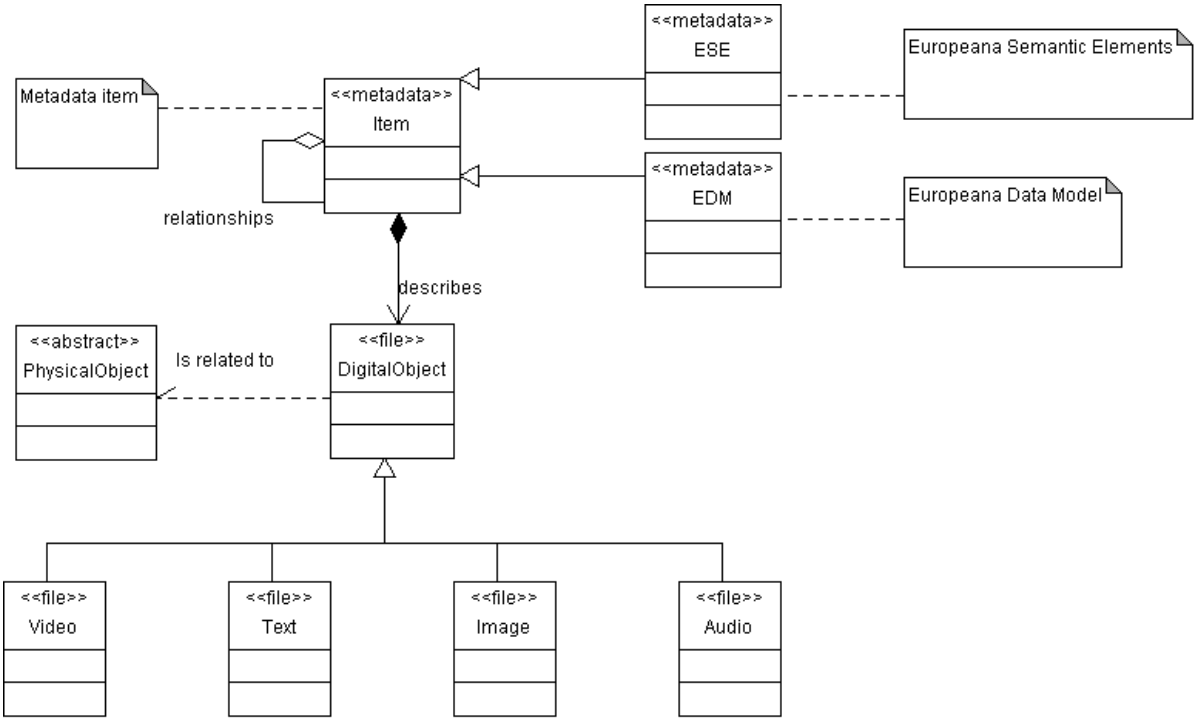

Figure 2.3-1 Europeana stores metadata "items" which describe digital objects held by institutions all over Europe. Digital objects in many cases represent one or more characteristics of a physical object but may also be original, born-digital content.

For users with a traditional search engine problem which needs solving, Europeana is currently able to provide the answers if the user is able to phrase the question. For users who approach Europeana wishing to explore the content in a personalized way, the situation is different as there are few relationships between items in Europeana allowing users to move from one to another – also, where relationships are present; there is no information which may aid or inspire the user in determining if he would like to browse the related content. In short, the data used as a starting point for the PATHS project are a very large digital library consisting of "atomic" facts with a less well-developed relational model connecting them.

PATHS provides mechanisms to enrich Europeana and Alinari content with links to third-party content (Wikipedia) and other content items through a content processing process which adds *where, what, when and who* annotation to the original data. The PATHS system architecture is geared towards exploiting the enriched content to provide a personalized user experience when exploring large volume digital libraries.

2.4 Assumptions and Constraints

This section identifies preconditions in the form of assumptions and constraints for the technical work to be carried out within PATHS WP3.

2.4.1 Assumptions

The system architecture assumes the following:

- Semantic processing and annotation will be done as bulk pre-process. The system may therefore not be deployed automatically on top of a compliant information volume without manual pre-processing of the content

2.4.2 Technical constraints

- Fixed time-frame and funding requires a strict adherence to a phased project schedule. The flexibility to add new user requirements towards the end of the project will decrease.
- The budget does not permit the use of high-end commercial software in any modules of the system architecture. This constraint is not envisaged to have a significant impact on the project as it is very much in alignment with the organizational policies of the partners. However, when it comes to semantic non-SQL databases such as triple-stores there are currently no open source technologies that perform equally well as their commercial counterparts. The system architecture takes this into consideration by employing a hybrid approach, combining the query capabilities of SPARQL with the efficiency of SQL.

2.4.3 Non-technical constraints

- For the personalization of the system there are legal constraints as to which aspects of user behaviour may be captured and stored. The system may therefore not store behaviour which may be traced back to identifiable users.
- Resulting end-user applications must conform to the Web Accessibility Initiative's (WAI) Web Content Accessibility Guidelines (WACG), a W3C recommendation which describes for developers and content authors how to make web content accessible for users.

3 METHODOLOGY

This chapter describes the approach used to develop the PATHS system architecture in terms of methodological framework, development and evaluation of different alternatives for the PATHS system architecture sub-systems, risks management related to the system development and conformance to functional requirements specified in PATHS deliverable D1.3.

3.1 System Design Framework

The system architecture was developed in accordance with the principles of the Systems Development Life Cycle management (SDLC). This methodology has been around for a long time and is not without its deficiencies, but is suitably abstract and may easily be adapted to the specific requirements of the PATHS project. The custom SDLC profile applied by the project allows agile, iterative development and involves the following phases:

- Project planning and feasibility study: Establishing a high-level view of the PATHS system and determines its goals and objectives.
- Systems analysis and technical requirements definition: Transforms PATHS objectives into defined sub-systems and modules, providing functionality for the planned end-user applications. Analyses end-user information needs.
- Systems design: Describes desired features and operations in detail, including BPMN business processes, UML use case diagrams, UML class diagrams and other documentation.

The methodology is iterative and the result of the first iteration of the three phases described above is version 1.0 of this System Design Description document. Subsequent iterations of these three phases may be required prior to physical implementation of the system. Further phases include:

- Implementation: Physical implementation of the sub-systems and modules defined in the specification into executable code.
- Integration and testing: Combines a test-suite of system level, module level, unit level, performance and integration tests aiming to identify errors, bugs and interoperability issues.
- Acceptance, installation, deployment: The final stage of initial development, where the software is put into production and runs actual business.
- Maintenance: What happens during the rest of the system life-cycle: changes; correction; additions; moves to different computing platforms and more.

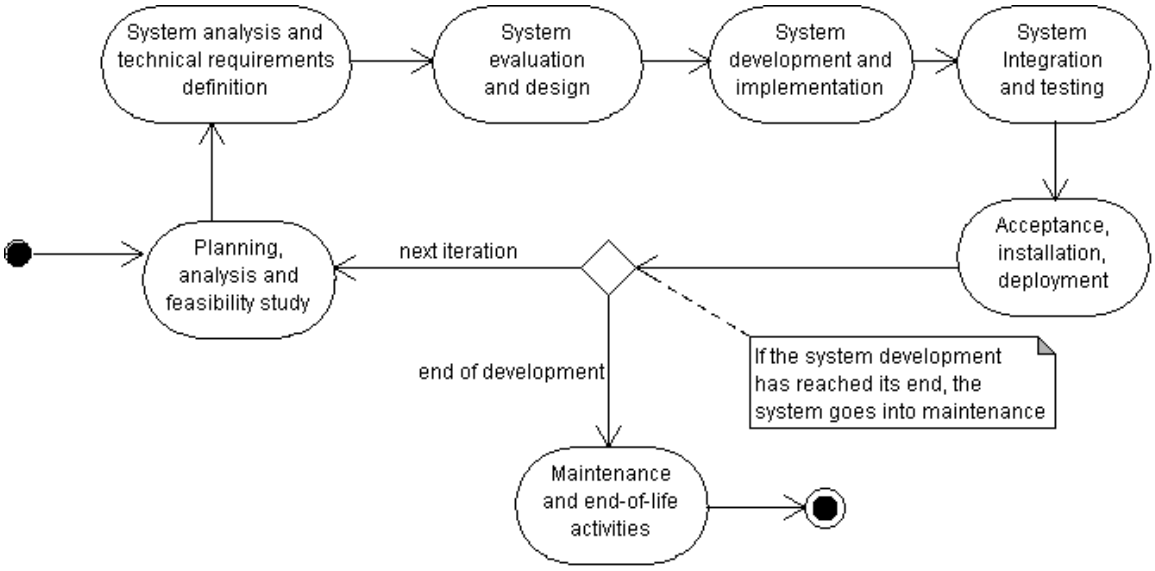

Figure 3.1-1 Simplified overview of the System Development Life-Cycle process

For ease of working across a development team consisting of members from four different organizations with heterogeneous working methodologies, a formal structure has been chosen for the development of design documentation. The IEEE System Design Document (SDD) standard, IEEE 1016, secures a 360 degree view of the system architecture and formalizes the semantics used to describe the system. This way, the risk of ambiguity or interpretation issues is significantly reduced throughout the decentralized development team.

Furthermore, for all technical illustrations, the Unified Modelling Language (UML) v1.4 from the Object Modelling Group has been used. In the same way as the SDD standard improves the readability of the design document, UML defines formal semantics which makes it easy to understand what is meant by a specific drawing. Free-form drawings may often be more visually appealing than the strict UML diagrams but are often difficult to interpret due to the freedom of semantics which leaves them open to different interpretation based on the thematic and technical skills and experience of the reader.

3.2 System Design Alternatives

Throughout the development of the system architecture, several design alternatives have been evaluated towards five key criteria for the overall system, these are:

- Robustness
- Scalability
- Performance
- Flexibility

Additionally, several of the sub-systems of the PATHS system architecture have dependencies towards existing software. In cases where more than one product has proven able to satisfy the technical requirements, several products have been tested and/or benchmarked against each other to determine which is best suited for PATHS. Key options which have been evaluated are listed below.

3.2.1 Choice of database

Product	Evaluation
PostgreSQL	PostgreSQL is a RDMBS which has always been focussing on features and standards compliance, aiming to become an open source alternative, or replacement, to the main commercial RDBMS such as Oracle, SQL Server and DB2. It has been around since 1985 and was after some years in academia, made into an open source effort. The core strength of PostgreSQL is to safely and securely hold the data it manages and a flexible architecture allows for core extensions to be developed by the entire community – this has resulted in successful modules such as PostGIS and PgRoute which provides spatial functionality and network problem resolver capabilities to the PostgreSQL core respectively. The PostgreSQL development community is the largest and oldest Open Source database community. Released under a BSD License.
MySQL	MySQL was started in Sweden in 1994 out of a need to have a high-speed database behind Websites. It was released in the open-source world a few years later. MySQL was designed to be a fast indexed sequential access method (ISAM) data store for Websites. This type of work load is geared mostly towards read operations with many small queries. The extensibility is secured through an API for development of extensions. Development community large but partly controlled by Sun and later Oracle. Released under a GPL license.

Conclusion: While the performance of the two evaluated systems is similar, the security features and scalability of PostgreSQL as well as its ease of integrating spatial data through the PostGIS and PgRoute modules have made this the preferred choice.

3.2.2 Choice of triple store

Product	Evaluation
AllegroGraph RDF Store (free edition)	Although released under a proprietary license, AllegroGraph comes with a free edition which does not conflict with the usage envisioned within the PATHS research project. The system has a wide range of query APIs and has demonstrated good scalability.
Mulgara Semantic Store	Mulgara is a native RDF triple store written in Java. It provides a Connection API that can be used to connect to the Mulgara store. Being a native triple store it has a “load” script which can be used to load RDF data into the triple store. In addition to supporting SPARQL queries through the connection API, these can be performed through a dedicated TQL shell client application.
SESAME	Sesame is an open source framework for storage, inferencing and querying of RDF data. It offers a

	connection API, inferencing support, availability of a web server and SPARQL endpoints and Jena it provides support for multiple back ends like MySQL and PostsgreSQL.
Jena	Jena is a java framework for building semantic web applications. Jena implements APIs for dealing with Semantic Web building blocks such as RDF and OWL. RDF models can be created from the file system or from remote files, accessible over the Internet. Using JDBC, it can also be tied to an existing RDBMS such as MySQL or PostgreSQL.
Virtuoso	Virtuoso, is a native triple store available in both open source and commercial licenses. It provides command line loaders, a connection API, support for SPARQL and a built-in web server for performing SPARQL queries and uploading of data over HTTP. Independent testing has found Virtuoso to be scalable to the region of 1B+ triples

Conclusion: The performance of FOSS triple-stores is still inferior to that of its commercial counterparts but due to the constraints of the PATHS project, it is an absolute requirement that the technology must be open source or free software. The system architecture will be implemented using Virtuoso with Mulgara as a fall-back strategy should the former not prove sufficient to meet the robustness requirements.

3.2.3 Choice of spatial visualization engine

Product	Evaluation
MapServer	A simple map server written in C which has been around since the mid-90s. Does not aspire to be a full-fledged GIS processing engine but is compliant with main OGC standards, is highly efficient when it comes to generating map images and supports a large variety of input and output formats. Available APIs include PHP, .NET and Java
GeoServer	A comprehensive set of geo-manipulation tools written in Java and packed into a map server suite. Highly sophisticated spatial data manipulation functionality and high performance.
OpenLayers	A client-side map visualization framework written entirely in JavaScript allowing easy embedding of cartographic and stylized maps into web pages without the need for client-side plugins. Allows seamless integration of PATHS data with global map service providers such as Google, Yahoo!, Microsoft Live Local etc.

Conclusion: the most suitable technology for generation of maps is MapServer on account of its superior performance. For client-side applications consuming map visualizations, OpenLayers will be used to access MapServer from client-side applications.

3.2.4 Choice of development platform

Product	Evaluation
Java	The preferred language in academic circles. Highly object oriented and platform neutral but not as widely used in the mainstream end-user application development community as VB and VB.NET. Many libraries and software components which PATHS sub-systems and modules depend on are written and only available in Java. Java is suitable for both executable programs and just-in-time compiled scripts.
PHP	A programming language originally geared towards creating personal homepages which in later times have come to power industry grade web applications such as Facebook.com, WordPress.com and other web giants. Easy to work with and object oriented with many programming libraries available to facilitate rapid application development.
.NET (C#, VB)	Due to the vast popularity of the Windows platform and the once near monopoly of VBA as a macro and extension programming language for desktop applications, .NET is currently the most widely used platform for commercial development of services on the web. Syntactically, C# is very close to Java and promotes the same good practices in coding. Has to some extent taken up competition with Java in the academic community – many libraries are available.
HTML5 (HTML + CSS + JavaScript)	The key technologies which together form HTML5 have all been around for a long time but have only recently come to fame, first with Web 2.0 and Ajax – then with HTML5 where the programmability of the new elements permits development of rich client-side applications.
C	High-level interpreter languages which require compilation are easy to write code in but invariably slower than its lower level machine code counterparts. For these purposes, Java and especially C are superior to the interpreted languages. A number of high-performance applications including the spatial visualization engine, MapServer, are written in C.

Conclusion: Due to the heterogeneity of the platforms used in the digital libraries communities it is envisaged that the development platform must permit applications to be built using several common technologies. This requires the PATHS systems to communicate with the client applications through web service interfaces. Due to dependency on existing software and in order to draw on the experiences of PATHS partners, it is also envisaged that the sub-systems of the PATHS system architecture may be implemented in several languages. Java will for an example be used for the indexing and information retrieval sub-systems as this module depends on Apache Lucene (see chapter 6.5 below), whereas C will be used for the map generation module of the presentation sub-system which will invoke MapServer. HTML5 may be used in developing client applications and will communicate with the PATHS system application layer via web services.

3.3 Risks

This section describes risks identified throughout the development of the System Design Description and proposes mitigation strategies.

Risk	Mitigation strategy
Performance problems for RDF stores with large amounts of triples	The PATHS system architecture does not rely on all data being stored in an RDF-store. Rather a hybrid mechanism is proposed whereby most data are being retrieved using traditional high-performance RDBMS technology at the same time as what, where, when and who vocabularies used to annotate the content resources are loaded into a triple store. This allows applications to use the triple store to look up related concepts in a graph of limited size and then retrieve the actual data related to the concepts from the RDBMS.

3.4 Requirements Compliance Matrix

This System Design Description takes as a starting point the comprehensive list of functional requirements described in PATHS deliverable D1.3.

Priority	Req. Identifier	Requirement	Description
MUST	1.1.1.	Registration	See section 6.2 below
	1.1.2.	Profile	See section 6.2 below
	1.1.3.	Edit Profile	See section 6.2 below
	1.1.4.	Visibility of profile	See section 6.2 below
	1.1.5.	Search the collection	See section 6.5 below
	1.1.6.	Primary object	See section 6.5 below
	1.1.7.	Collect Objects	See section 6.5 below
	1.1.8.	Search workspace	See section 6.5 below
	1.1.9.	Search Paths by topic	See section 6.5 below
	1.1.10.	Save searches	See section 6.2 below
	1.1.11.	Find existing Paths	See section 6.2 below
	1.1.12.	Links to related content	See section 6.6 below
	1.1.13.	Create Paths	See section 6.4 below
	1.1.14.	Edit Paths	See section 6.4 below
	1.1.15.	Identity	See section 6.4 below
	1.1.16.	Search engine friendly	

	1.1.17.	Add content	
	1.1.18.	Describe themes and sub-themes	
	1.1.19.	Add content tied to objects	
	1.1.20.	User comments on Paths	See section 6.2 below
	1.1.21.	Attribution	
	1.1.22.	Grant access to specific users & users	See section 6.2 below
	1.1.23.	Permission to clone	
	1.1.24.	User identity	
	1.1.25.	Multiple platforms	
	1.1.26.	Zoom	
	1.1.27.	Sense of discovery	
	1.1.28.	Delete user profiles and user-generated content	
SHOULD	1.2.1.	Familiarity	
	1.2.2.	Organise Personal Collection	
	1.2.3.	Flexible design	
	1.2.4.	No restriction on object type	
	1.2.5.	Create Paths across multiple sessions	
	1.2.6.	Grant access to specific groups	
	1.2.7.	Communication with Path creator	
	1.2.8.	Activity description	
	1.2.9.	Tagging objects	
	1.2.10.	Aggregate tags	
	1.2.11.	Search via tags	
	1.2.12.	Show/hide annotations	
	1.2.13.	Clone Paths	
	1.2.14.	Time factor	
	1.2.15.	User comments on items in a Path	
COULD	1.3.1.	Add any resource to holding space	
	1.3.2.	Rate Paths	

	1.3.3.	Receive private comments
	1.3.4.	Tag rewards
	1.3.5.	Geo-location data
	1.3.6.	Matching Paths and objects to locations
	1.3.7.	User content
	1.3.8.	Web content as object

4 ROLES AND RESPONSIBILITIES MATRIX

This section describes the roles and responsibilities of the development resources available in the PATHS consortium. The table below shows which sub-system each partner with resources committed towards work package 3 will contribute to.

Sub-system	AVINET	iSieve	USFD	UPV/EHU
Content processing sub-system			X	X
Sentiment analysis sub-system		X		
Virtual data repository sub-system	X		X	
Data integration sub-system	X			
User profile sub-system	X	X		
Path authoring sub-system	X	X		
Recommender sub-system			X	X
Information retrieval sub-system	X		X	
Presentation sub-system	X	X	X	X
Web API sub-system	X	X		
Test suite		X		

5 SYSTEM DESCRIPTION

This chapter provides an overview of the PATHS system architecture in terms of: (1) a high-level software view; (2) a technical architecture view which shows how the software is deployed to the logical server infrastructure; (3) an overview of the physical server and network infrastructure of the system; and (4) description of communication interfaces which are employed in integrating the different layers of the architecture which may also be used as external interfaces for development of 3rd party applications on top of the PATHS application layer.

5.1 System Software Architecture

The PATHS system architecture is a Service Oriented Architecture (SOA) with three logical layers:

1. A **data layer**
2. An **application layer**
3. A **client layer**

The data layer consists of two sub-systems: (1) a virtual data repository sub-system; and (2) a content processing sub-system. Only the former is described in this SDD document as the latter is dealt with in great detail in PATHS deliverable D2.1 Processing and Representation of Content for First Prototype.

The application layer consists of seven sub-systems: (1) a data integration sub-system; (2) a user profile sub-system; (3) a Path authoring sub-system; (4) a recommender sub-system; (5) an information retrieval sub-system; (6) a presentation sub-system and; (7) a Web-API sub-system.

The content of the client layer is not defined in detail in this SDD document as the specification of end-user applications is the topic of PATHS deliverables D1.3 and D1.5. The functionality which will be present in the end-user applications is however dictated by the capabilities exposed by the application layer through the Web-API sub-system mentioned above.

Together, the components of the three logical layers forms a system capable of implementing all functionality which has been identified as part of the user requirements phase and which has subsequently been translated into a functional specification in D1.3.

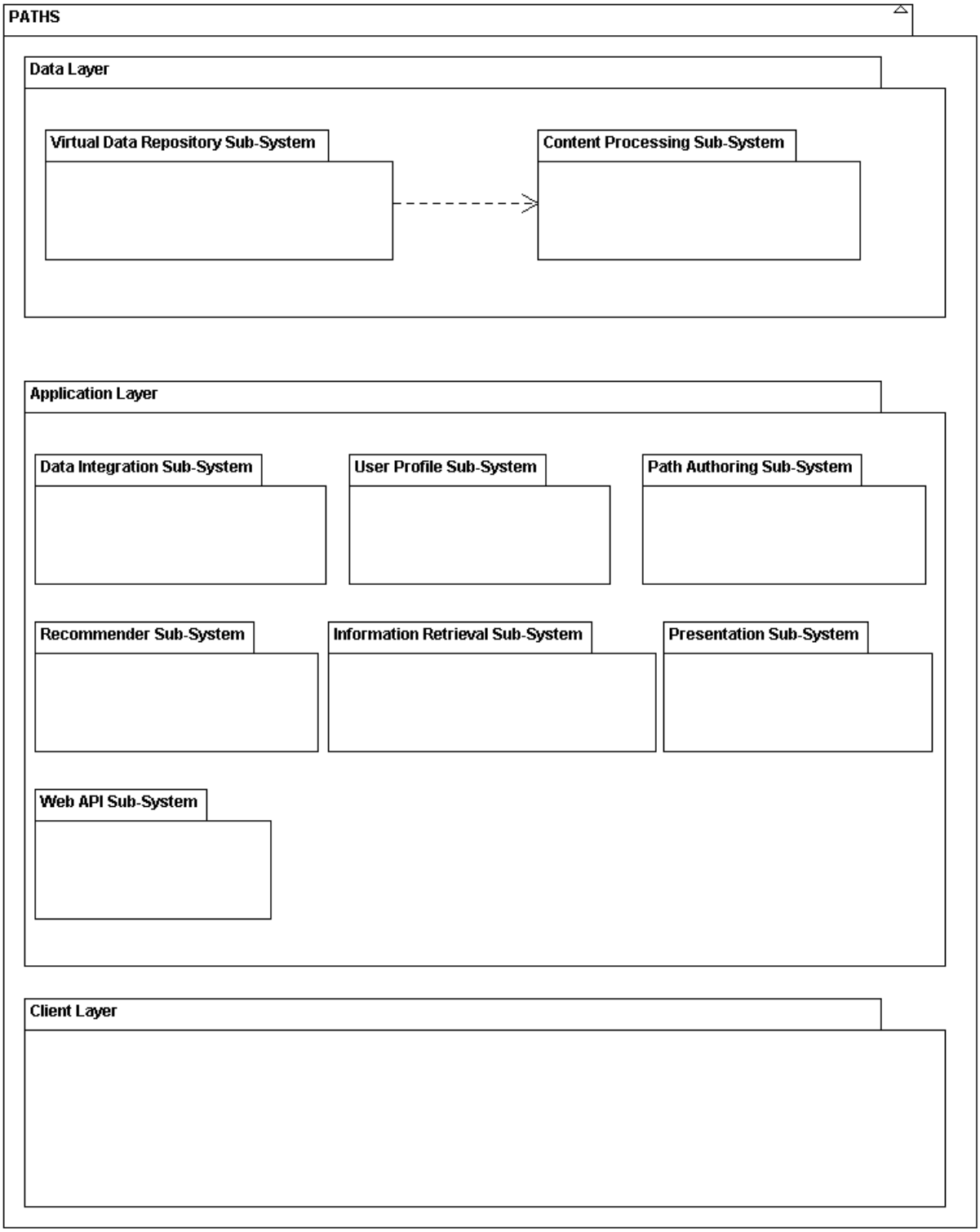

Figure 5.1-1 High level software architecture UML class diagram

5.2 System Technical Architecture

The PATHS system software architecture requires the establishment of a technical architecture consisting of logical servers and software capable of supporting the functionality to be provided by the sub-systems of the PATHS Service Oriented Architecture (SOA) data and application layers.

Three logical servers are required to support the system architecture:

4. A **database server**
5. A **file server**
6. An **application server**

The database server will be equipped with PostgreSQL and PostGIS and a data definition script will be installed and executed in order to establish the relational database structured required to support the operation of the PATHS system.

The file server will be used to store non-database binary and text files which form part of the data such as imagery from Alinari, XML-files from Europeana etc. No additional software or functions is envisaged for this server.

Finally, the application server will provide a triple store in the form of Virtuoso, a spatial rendering engine in the form of MapServer, a Java servlet engine in the form of Tomcat, an HTTP server in the form of Internet Information Server (IIS) and a search indexer in the form of Apache Lucene.

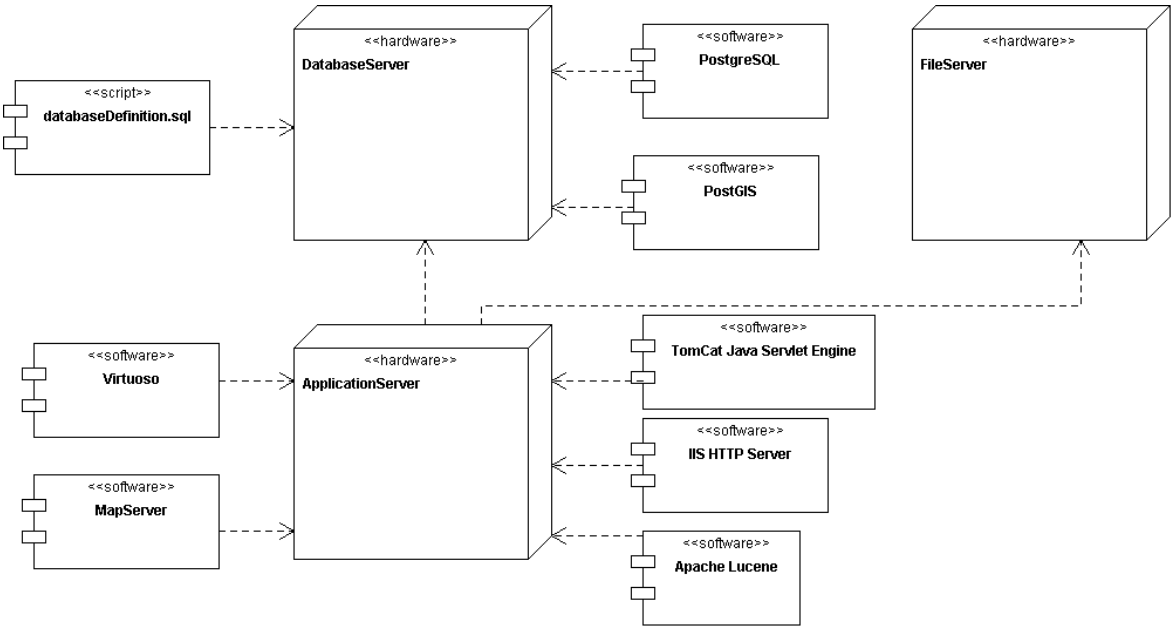

Figure 5.2-1 PATHS technical architecture UML deployment diagram

5.3 System Hardware Architecture

The logical infrastructure consisting of software and logical servers requires a solid hardware platform to provide the robustness and scalability required to handle the significant quantities of data available through the PATHS system. The hardware infrastructure is envisaged to consist of three physical servers, a hardware fire-wall and an internet router.

1. The file server has mid-range hardware requirements (>= 1 x 2.2 GHz quad-core processor, >= 4 GB DDR3 RAM and gigabit or optical connection to the application server, 500 GB storage in RAID3 or better.
2. The database server has high-end hardware requirements (>= 2 x 3 GHz quad core processor, >= 8 GB DDR3 RAM, 500GB of storage in RAID3 or better.
3. The application server has high-end hardware requirements (>= 2 x 3 GHz quad core processor, >= 8 GB DDR3 RAM, 500GB of storage in RAID3 or better.
4. The firewall has must permit a large number of concurrent connections reserved for the PATHS system as the potential usage volume is quite high. All communication through the firewall towards this Internet will go over HTTP on port 80.
5. There are no specific requirements for the Internet router except that the bandwidth available to the PATHS system must be >= 30 Mbit/sec in order to guarantee a satisfactory user experience.

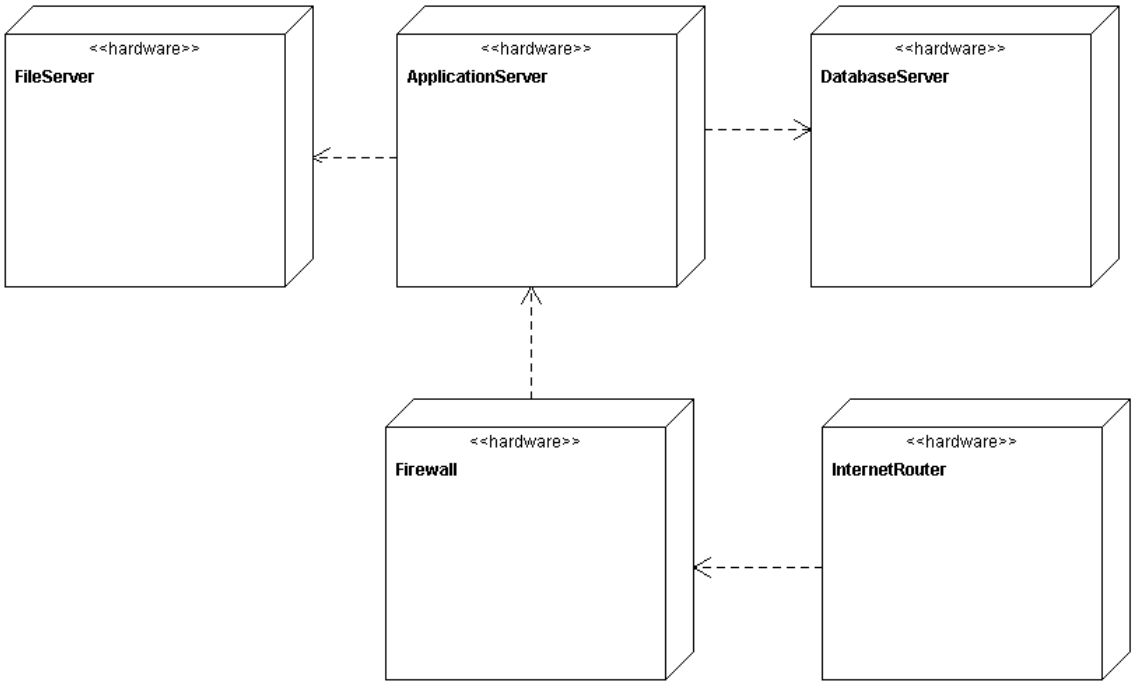

Figure 5.3-1 PATHS hardware architecture UML deployment diagram

5.4 External Interfaces

The PATHS system architecture is self-contained in that it does not have any external extension points beyond the capability of the application layer to communicate with the end-user applications in the client layer via web services. The interfaces between the different system architecture layers are:

- Communication Between data layer and application layer
  - JDBC over TCP/IP
  - ODBC over TCP/IP
  - SMB over TCP/IP
  - SPARQL/HTTP over TCP/IP
- Between the application layer and the client layer
  - RESTful XML and JSON Web Services

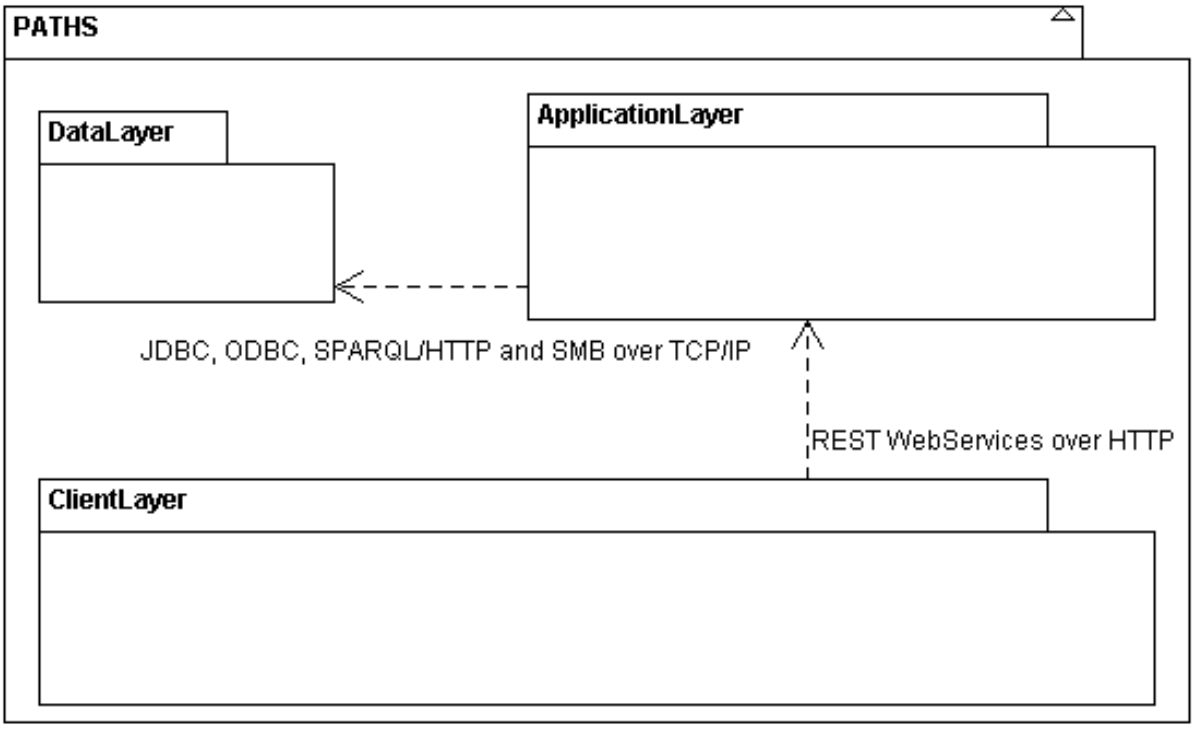

Figure 5.4-1 PATHS SOA (external) interfaces UML class diagram

6 SUBSYSTEM SPECIFICATIONS

While chapter 5 above is concerned with providing a logical overview of the system, this chapter provides detailed technical specifications of the modules of each of the sub-systems of the PATHS SOA application layer. The sub-systems of the data layer are described in chapter 7 below.

6.1 Data integration sub-system

The fundamental part of the PATHS application layer is the data integration sub-system. This provides the interfaces required to communicate between PATHS code and the data stores.

Due to the flexibility requirements of the system, the sub-system will offer four different interfaces for communication between application code and the virtual data repository:

1. Open DataBase Connectivity (ODBC) for connecting to PostgreSQL from .NET, C and PHP applications.
2. Java DataBase Connectivity (JDBC) for connecting to PostgreSQL from Java applications
3. Simple Protocol And RDF Query Language (SPARQL) over HTTP for issuing queries towards the Virtuoso RDF store from any application.
4. Server Message Block (SMB) protocol for read-write access to file server from any application.

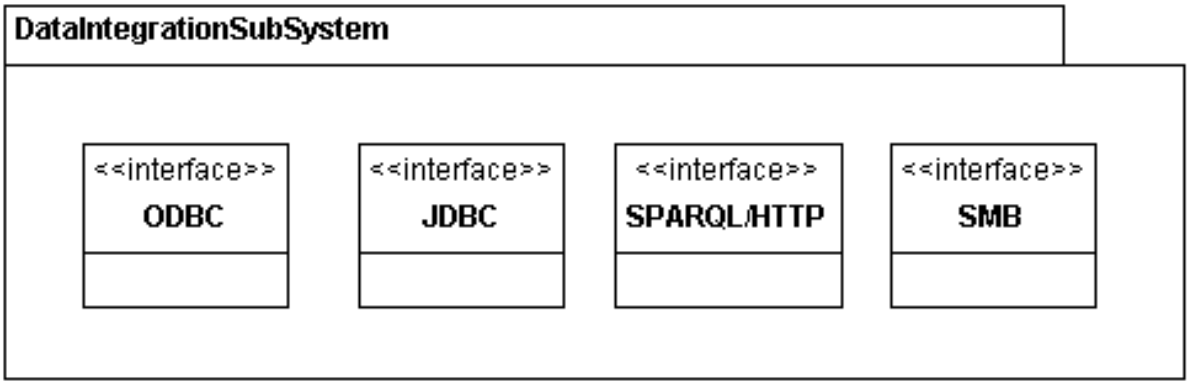

Figure 6.1-1 UML class diagram showing communication interfaces in the PATHS data integration sub-system

6.2 User profile sub-system

The user profile sub-system allows the creation and maintenance of user profiles as well as logging of user behaviour for authenticated and non-authenticated PATHS users.

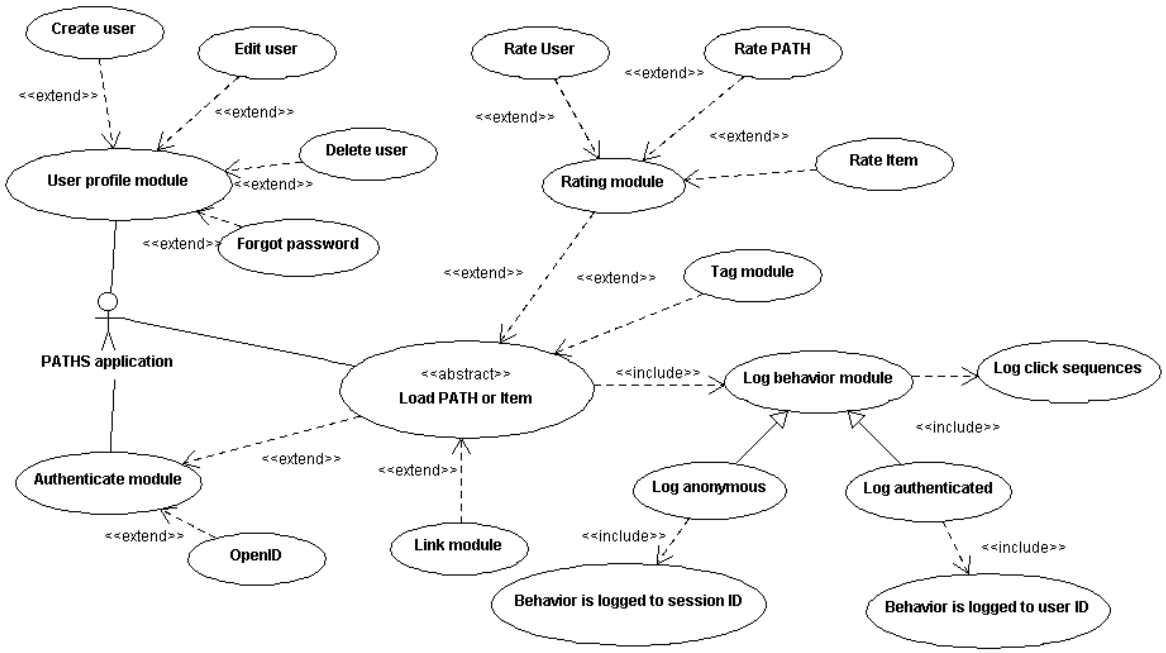

Figure 6.2-1 PATHS user profile sub-system UML use case diagram

6.2.1 User profile sub-system modules

This section provides a short description of each of the modules in the sub-system, their functions and dependencies.

6.2.1.1 User profile module

The user profile module allows creation, editing and deletion of user profiles. User profiles will capture an absolute minimum of information and will not store any form of sensitive data which makes it possible to resolve a user id to a legal or physical person unless the username discloses this information. A user ID may be linked to an existing OpenID account such as Facebook, Google or other OpenID providers. This is likely to be a catalyst, or at least the removal of a barrier to, gaining registered users for the PATHS system.

6.2.1.2 Authentication module

The authentication module allows authentication of users. If the user is linked to an OpenID account, the password verification will be conducted remotely and a session will be created in which the user may enjoy privileges which require authentication.

6.2.1.3 Log behaviour module

The log behaviour module captures the patterns in which users move from one PATHS item to another. Behaviour will be recorded for any movement regardless whether an item is part of a PATH or the user is merely browsing collections freely. Whenever a user jumps from one item to another within a user session, the movement is recorded and a “weight” is added to the route. Once sufficient data has been collected, the behaviour log may be used as a data source for the recommender system (see chapter 6.6 below) to suggest alternative ways forward whenever a user is browsing a collection, following or authoring a path.

6.2.1.4 Rating module

The rating module allows authenticated users to rate: (1) other users; (2) Items and (3) Paths she or he is browsing. This information will also be used as input to the recommender system

in order to provide suggested “reading” in the context of viewing a specific item. A rating may be applied to any item which has a URI.

6.2.1.5 Tag module

It is possible to assign a tag to any item identified by a URI including Items, Paths and Users.

6.2.1.6 Link module

It is possible to create links between any two items as long as they are identified by a URI including Items, Paths and Users.

6.3 Presentation sub-system

The presentation sub-system enables client applications to request visualizations and representations of Paths and Items for presentation in an end-user interface. Information returned by the presentation sub-system will be either XML, JSON or “pure” HTML which may be styled using CSS from the end-user application.

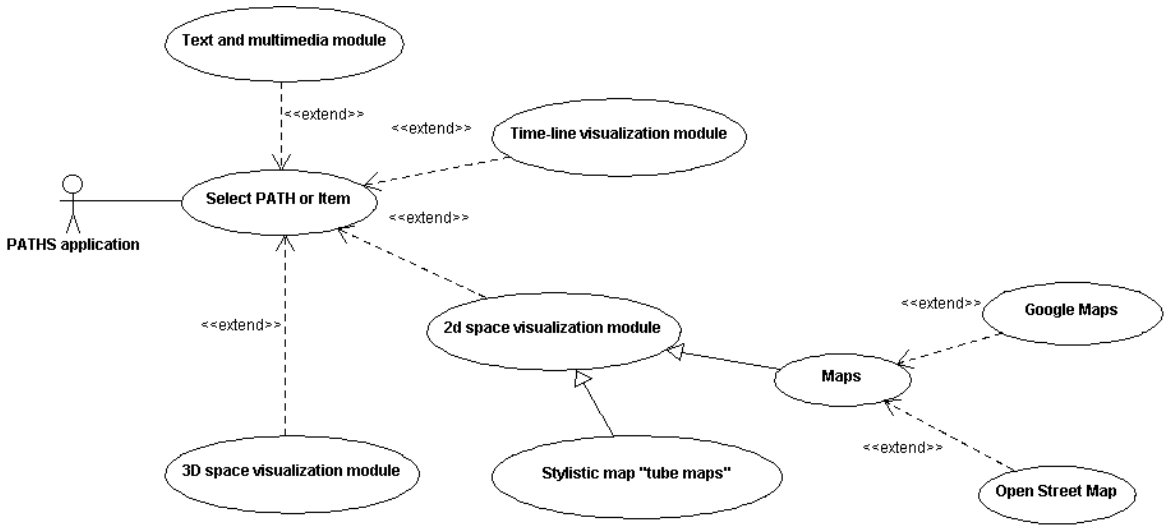

Figure 6.3-1 PATHS presentation sub-system UML use case diagram

6.3.1 Modules

This section provides a short description of each of the modules in the sub-system, their functions and dependencies.

6.3.1.1 Text and multimedia module

The text and multimedia module takes care of presentation of “traditional” multimedia article content such as text, images, audio and video or combinations of these. The module must accept common formats which are widely deployed. The content will be pure HTML or XML which may be styled and presented in a flexible manner using CSS or XSLT respectively.

6.3.1.2 3D space visualization module

No user requirement has yet been identified which requires 3D visualization but the module is left in the SDD document as a potential future development for the second prototype.

6.3.1.3 2D space visualization module

Spatial visualization in the form of traditional topographic maps as well as stylized maps has come up as user requirements during the initial stage of the project. The module will allow visualization of individual items in a geographic map or stylized 2d space . Items may also be visualized in their spatial context, showing other items in the proximity. The user may filter which categories/types of items are shown in the map.

Base maps may be sourced from e.g. OpenStreetMap (OSM) or GoogleMaps in order to gain global coverage. The module will be implemented using MapServer and the services exposed thorugh the WebAPI will be conformant to the OGC WMS standard. Maps will be returned to the client application as an interactive widget.

6.3.1.4 Time-line visualization module

This module will provide temporal visualization in the form of presentation of a content item in the context of other content items on a time axis will be returned to the client application as an interactive widget.

6.4 Path authoring sub-system

The Path authoring sub-system enables the creation of Paths consisting of a set of items chained together by narratives which subjectively describe the relationships between them as seen by the author.

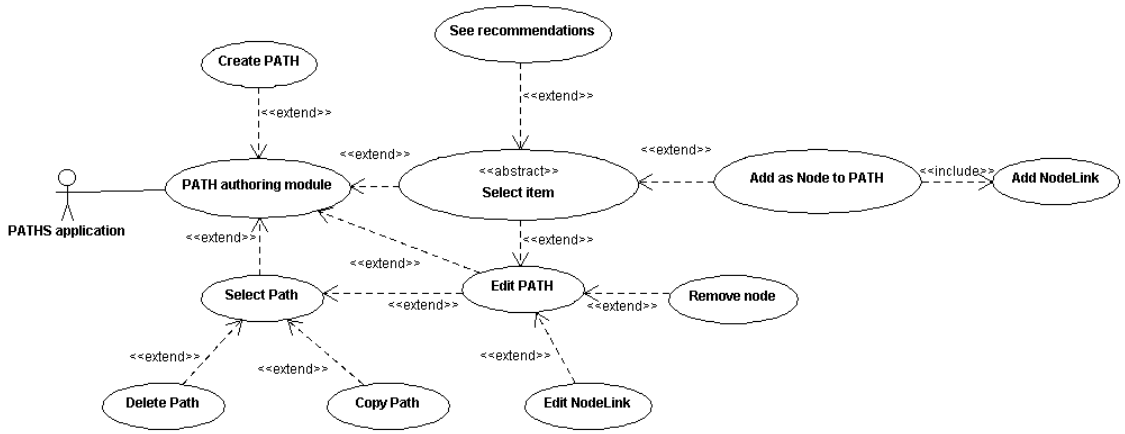

Figure 6.4-1 PATHS authoring sub-system UML use case diagram

6.4.1 Modules

This section provides a short description of each of the modules in the sub-system, their functions and dependencies.

6.4.1.1 PATH authoring module

The Path authoring module allows authenticated users to create new Paths as well as edit clone and delete existing Paths. Paths created by a user will be available to her or him throughout the authenticated user session.

A Path has a direction which is implied by the sequence of the items. Each item is added to the Path through a proxy class called a Node which stores the position of the item in the Path as well as a reference to the concerned Item.

Whenever the author is looking at an item presentation, she or he may add it to the end or beginning of any existing Path – or add the item to a new Path in which case the Item will become the first node of the Path.

The Path itself has a narrative which describes the content or purpose of the Path. Each Node may have a description and all NodeLinks, the “glue” which keeps the Nodes in a Path together, have a narrative which describes the relationship between two adjacent nodes. The narrative is a free text. If some typology is desirable, this may be implemented on the Client side, allowing the user to select from a number of pre-defined relationships as an alternative to supplying a comprehensive narrative.

6.5 Information retrieval sub-system

The PATHS information retrieval sub-system will secure efficient indexing of and searching throughout Paths and Items using a variety of textual, temporal and spatial search mechanisms.

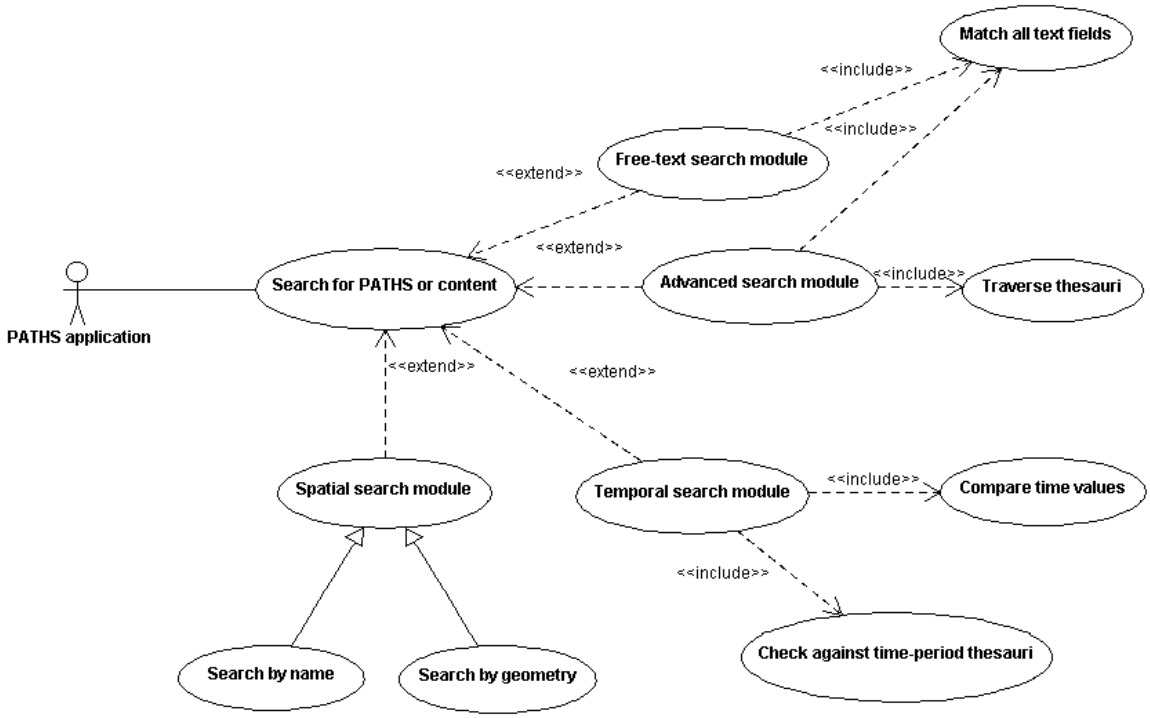

Figure 6.5-1 PATHS information retrieval sub-system UML use case diagram

6.5.1 Modules

This section provides a short description of each of the modules in the sub-system, their functions and dependencies.

6.5.1.1 Free-text and advanced search module

The free-text search module will utilize the industry proven Apache Lucene indexing and search engine in order to provide “lightning fast” free text queries towards the entire PATHS data volume including both user authored Paths and Items sourced from Europeana and Alinari. Lucene supports certain simple semantics which allows users to specify “Google style” inclusion and negation operators in the queries to further refine them. This will enable users to search for entire phrases, omit results which contain a certain word etc.

The advanced search module works and is based on the same technical components as the free-text module but will allow querying against specific metadata items for higher precision. The advanced search module will also optionally perform query expansion in order to retrieve not only exact matching but related items.

6.5.1.2 Temporal search module

The temporal search module will enable users to search for Items by a single date, by time intervals and time period designations which will be attempted resolved to dates or time intervals as defined in the time period vocabulary of choice. (The appropriate vocabulary for the PATHS information domain has yet to be decided upon).

6.5.1.3 Spatial search module

The spatial search modules will allow searching towards Paths and Items using spatial search criteria, namely points, lines or polygons according to the OGC Simple Features Specification (SFS) or place names according to the global geographical names web service, GeoNames. This module anticipates user requirements which are envisaged for the second prototype.

6.6 Recommender sub-system

The recommender sub-system will provide a user browsing an Item with suggestions on which Item to visit next. The recommender system will for this purpose employ a set of components looking for related Paths, related locally held Items, related Items as implied by user behaviour and related Cloud content.

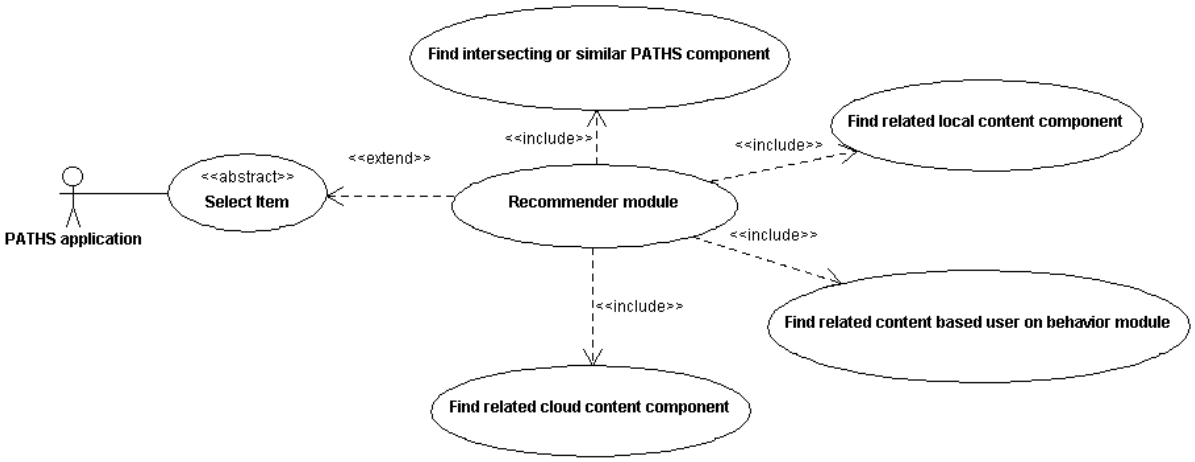

Figure 6.6-1PATHS recommender sub-system UML use case diagram

6.6.1 Modules

This section provides a short description of each of the modules in the sub-system, their functions and dependencies.

6.6.1.1 Recommender module

The recommender module takes as a starting point the item currently being browsed by a user and tries to determine relevant related content by a number of different methods, invoked by specialized components as shown in the use case diagram above.

The queries towards the different components are executed in parallel, and the priority of the requests is shown in the UML sequence diagram below.

1. First a component tries to identify other items which belong to Nodes in the currently browsed Path or in Paths which are intersecting the currently browsed Path.
2. Second, a component identifies items which are relevant based on their spatial, temporal and topical metadata compared to the current Item
3. Third, a component proposes relevant items based on recorded user behaviour such as frequently traversed nodes etc.
4. Forth, a query is issued towards links generated during the PATHS content pre-processing in WP2 to provide external context for the Item being browsed from e.g. Wikipedia, Panoramio or other, similar service providers.

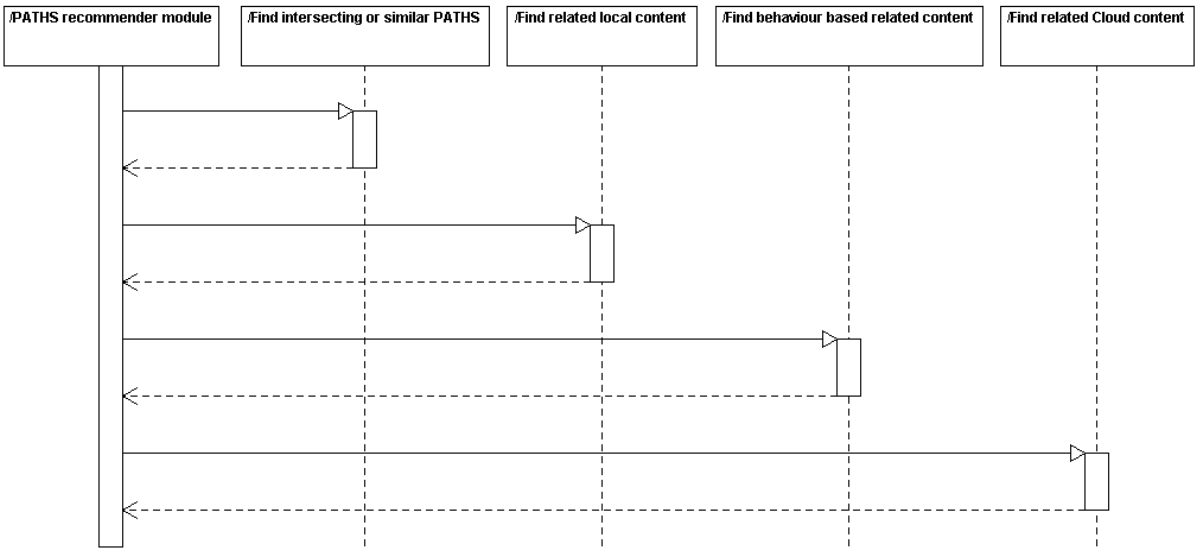

Figure 6.6-2 The sequence of priority for the query components which identifies related Paths and Items.

6.7 Web Service API sub-system

The Web Service API sub-system will expose the functionality of all other sub-systems in the PATHS system architecture to the Internet from where they may accessed and integrated into full-fledged end-user applications as specified in PATHS deliverable D1.3 and subsequently D1.5.

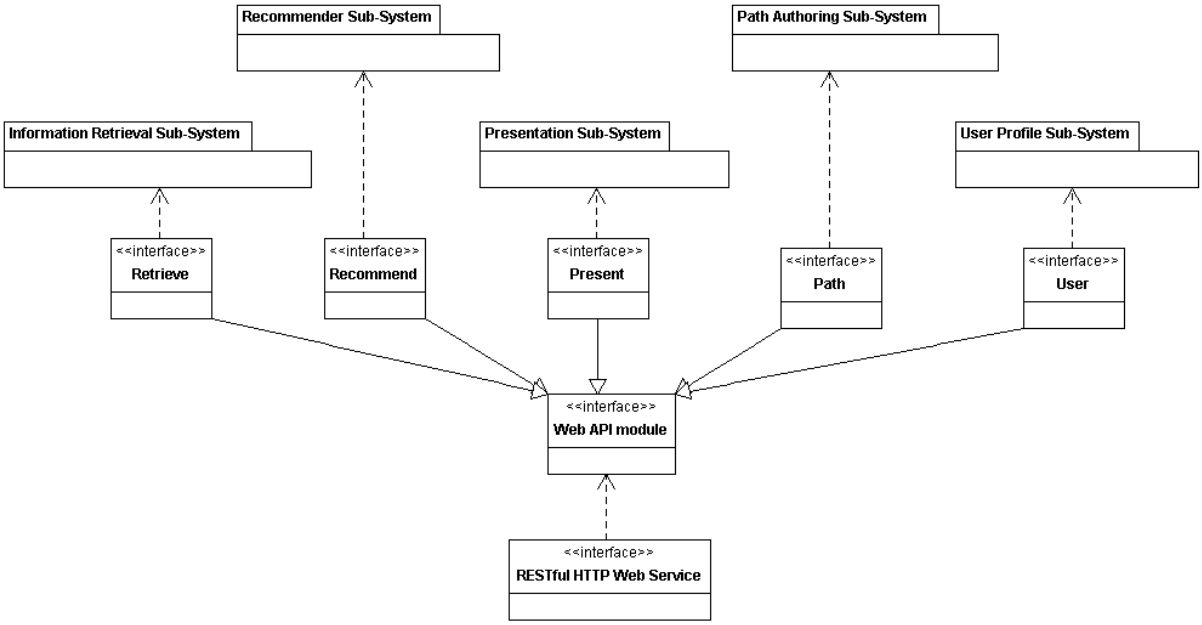

Figure 6.7-1 PATHS web service API UML class diagram

6.7.1 Modules

This section provides a short description of each of the modules in the sub-system, their functions and dependencies.

6.7.1.1 Web API module

The Web API module will provide web service interfaces for accessing all the functionality in the PATHS SOA application layer. The interface will consist of two parts: (1) a web service API; and (2) a set of client libraries supporting rapid/easy client application development. The Web API will as far as possible re-use existing code and libraries such as jQuery, OpenLayers and similar.

7 DATA ARCHITECTURE

This chapter describes the content of the PATHS system architecture data layer and the virtual data repository consisting of a file store, an RDBMS and an RDF store.

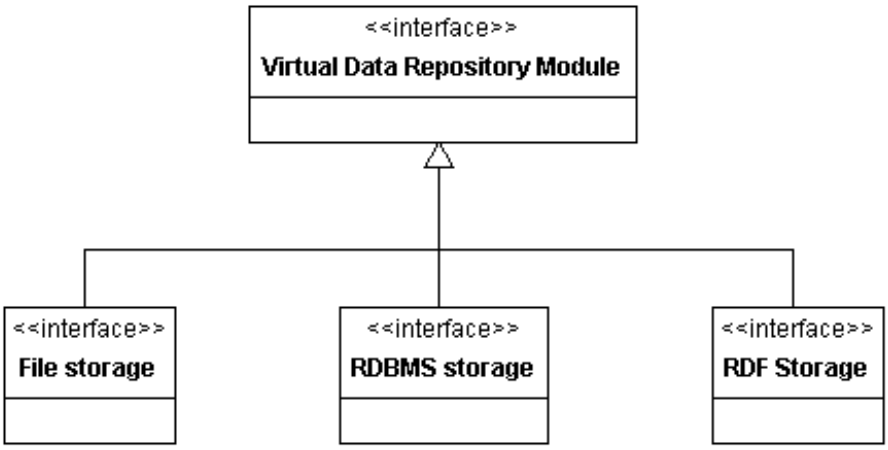

Figure 6.7-1 PATHS Virtual Data Repository sub-system UML class diagram

7.1 Non-database file storage

The most fundamental part of the PATHS virtual data repository sub-system is the file storage where all non-database files such as binary image, audio and video files as well as enriched/annotated XML files are stored.

7.2 RDBMS data storage

In order to achieve sufficient performance with a potentially large number of users and a relatively large quantity of Items, the primary data storage for Items and Paths is an RDBMS. The selected product, PostgreSQL provides a comprehensive set of standard compliant as well as product specific functions for data creation and maintenance. The built-in mechanisms for insertion, updates and deletions will not compromise data integrity and comprehensive operations may be wrapped within transactions which may be rolled back upon failure.

7.2.1 PATHS RDBMS data model

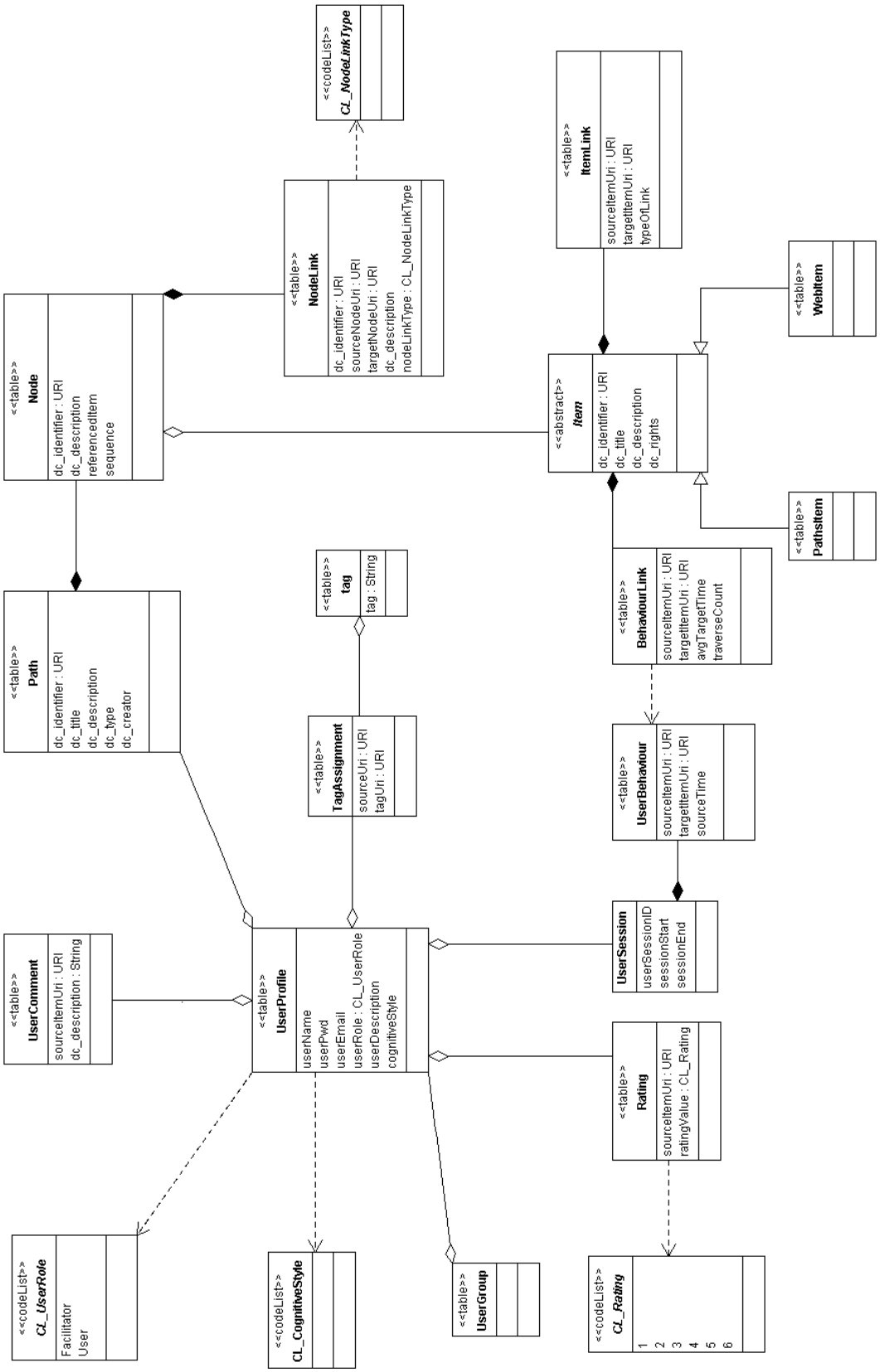

Figure 7.2-1: PATHS data model

7.3 RDF data storage

The third module of the Virtual Data Repository sub-system is the RDF data store, a native RDF graph database which allows the storage of information as triplets. The RDF data store will be used for RDF vocabularies expressing named entities (persons, geographical names), time periods, topics/subjects etc.

The RDF data store will expose a SPARQL end-point which may be used to issue queries and retrieve data from the data store. RDF data stores, or triple-stores as they are commonly called, are far from as efficient as relational databases yet. For this reason, PATHS applies a hybrid solution whereby the bulk of the data will be stored in a relational database whereas indexes, vocabularies, annotations to and relationships between items will be placed in the RDF store. This provides the flexibility, timeliness and power of a contemporary Semantic Web solution without exhausting the performance and capacity of the system architecture.

8 SECURITY

The PATHS system architecture implements a series of measures targeted at securing the system against unauthorized or unintentional access which may compromise the integrity of PATHS data.

8.1 User Level Permissions

The PATHS system architecture implements the following three user levels with the purpose of mitigating the risk of unauthorized and/or unintentional access to the system which may compromise data integrity.

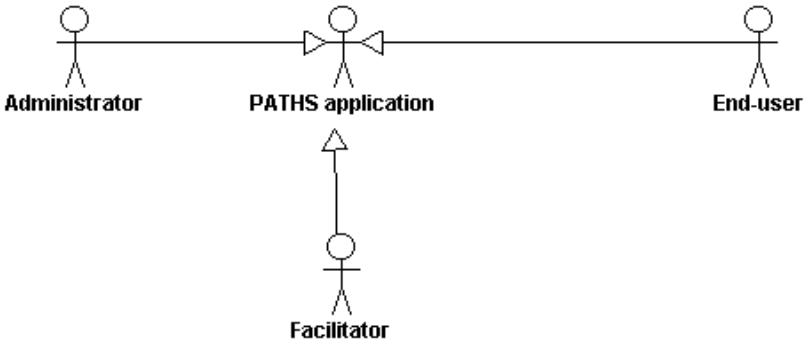

Figure 8.1-1 User levels UML use case diagram

- A system administrator level is available only to trusted individuals in the organization providing hosting for the PATHS system.
- An application user is created for each specific application and is given only such permissions to the file-system and database as are required to run the application.
- A read-only user is created for volume queries by non-authenticated user which should be tuned towards efficiency.

8.2 Control Points

The control points in the PATHS system architecture consists of:

- the authentication module described in detail in section 6.2.1.2 above;
- the firewall described in section 5.3 above and;
- local and network user management on the three physical servers described in section 5.35.3 above

The authentication module determines which users have access to which application functionality, Paths and Items.

The firewall controls network traffic access and prevents unauthorized or unintentional access to the system over TCP/IP.

The local user management secures that applications are executed under certain user privileges which secure that data will not be accidentally overwritten or have their integrity compromised by concurrent access or operations.

8.3 Vulnerabilities

Vulnerable points in the PATHS system architecture include:

- The authentication module accepts logins over standard HTTP which renders user credentials visible in the case of IP address spoofing.
- Login screens are vulnerable to brute-force password attacks
- Elevated user rights permitting read/write to the virtual file repository sub-system may cause accidental deletion of PATHS non-database files.
- Elevated user rights permitting table level update and deletion of records may compromise data integrity for PATHS RBDMS data.

9 OPERATIONAL CONSIDERATIONS

Previous chapters having specified all the technical aspects of the system, this chapter elaborates the operational considerations related to the development, running and maintenance of the PATHS system architecture.

9.1 Audit Trail

In order to be able to trace the cause of system incidents such as application and data errors, the PATHS system architecture employs the following mechanism:

- All code is developed towards a Concurrent Versioning System (CVS) repository that allows the current code base to be reverted to any previous build upon detection of errors. Each developer has a distinctive user identity for the CSV, allowing any change in the code to be traced to a specific commit.
- All tables in the RDBMS are equipped with a “last update” column and the username of the database user who most recently made a change.
- All operating system logs are retained for the purpose of auditing the error trail for any application or system incidents which may occur throughout the system life cycle.

9.2 Recoverability

The system architecture is designed bearing recoverability of the system in mind in the event of system failure. The system may be recovered from a complete crash by the following steps:

1. System restore for operating systems.
2. Re-applying any non-file based application configuration.
3. Reloading of backups for application server, file server and database
4. Execute test-suite and rectify any issues identified

9.3 Data Retention

The PATHS system covers applications in the “edutainment” segment and do not contain any form of sensitive information or records which are legally obliged to be retained. However, a number of data retention measures are put in place to secure the audit trail, see section 9.1 above

- All application and web server logs are retained for a grace period of two months before they are archived, allowing for incident tracking of internal application communication issues as well as external “attacks” on the infrastructure, unintentional failures due to proxies etc.
- In order to prevent wrongful deletion of Paths or Items, these objects are never physically deleted from the database but the Boolean data type column with the name “deleted” is set to true for records which are disabled.
- A similar mechanism is put in place to avoid unintentional deletion of user accounts. User accounts which have been deleted through the user interface will initially be marked as deleted by setting the Boolean data type column with the name “deleted” to true. A column with the timestamp of the last update of the user information will also be updated and upon the completion of three months in this status, the user

account will be permanently deleted and its “owned” information elements such as Paths, comments, etc attached to the system archive user.

9.4 Conventions/Standards

The following conventions and standards are employed throughout the development and subsequent implementation of the PATHS system architecture:

Standard	Description
IEEE SDD	IEEE 1016-1998, also known as the Recommended Practice for Software Design Descriptions, is an IEEE standard that specifies an organizational structure for a software design description (SDD). An SDD is a document used to specify system architecture and application design in a software related project.
OGC WMS	Web Map Service (WMS) is a standard protocol for serving geo-referenced map images over the Internet that are generated by a map server using data from a GIS database. The specification was developed and first published by the Open Geospatial Consortium in 1999.
Google KML	Keyhole Markup Language (KML) is an XML notation for expressing geographic annotation and visualization within Internet-based, two-dimensional maps and three-dimensional Earth browsers. KML was developed for use with Google Earth, which was originally named Keyhole Earth Viewer
W3C XML	Extensible Markup Language (XML) is a set of rules for encoding documents in machine-readable form. It is defined in the XML 1.0 Specification produced by the W3C.
W3C HTML5	HTML5 is a language for structuring and presenting content for the World Wide Web, a core technology of the Internet. Its core aims have been to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices. HTML5 is intended to subsume HTML4, XHTML1 and DOM2HTML (especially JavaScript).
ECMAScript	ECMAScript is the scripting language standardized by Ecma International in the ECMA-262 specification and ISO/IEC 16262. The language is widely used for client-side scripting on the web, in the form of several well-known dialects such as JavaScript, JScript, and ActionScript.
W3C RDF	Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model. It has come to be used as a general method for conceptual description or modeling of information and as the fundamental building block of the Semantic Web.
SDLC	Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering, information systems and software engineering, is a process of creating

	or altering information systems, and the models and methodologies that people use to develop these systems.
OMG UML	Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of object-oriented software engineering. UML includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems.
W3C SKOS	Simple Knowledge Organization System (SKOS) is a family of formal languages designed for representation of thesauri, classification schemes, taxonomies, subject-heading systems, or any other type of structured controlled vocabulary. SKOS is built upon RDF and RDFS, and its main objective is to enable easy publication of controlled structured vocabularies for the Semantic Web.
W3C OWL	The Web Ontology Language (OWL) is a family of knowledge representation languages for authoring ontologies. The languages are characterized by formal semantics and RDF/XML-based serializations for the Semantic Web.
Europeana ESE	The Europeana Semantic Elements (ESE) data model is an XML format consisting of elements from Dublin Core, DC terms and a few Europeana specific extensions for the purpose of providing end-user functionality in the current Europeana portal.
Europeana EDM	The Europeana Data Model (EDM) is a semantically rich data model which supersedes ESE and seeks to overcome the limitations of expressing relationships through Dublin Core.
HTTP	The Hypertext Transfer Protocol (HTTP) is a networking protocol for distributed, collaborative, hypermedia information systems.[1] HTTP is the foundation of data communication for the World Wide Web

The definitions of the above are adapted from their corresponding articles on Wikipedia

REFERENCES

Other important documents from the PATHS deliverables include:

D1.1 User Requirements Analysis

D1.3 Functional Specification of First Prototype

D2.1 Processing and Representation of Content for Second Prototype

D4.1 Initial prototype interface design

Project deliverables will be published on the project website at: http://www.paths-project.eu/eng/Resources

APPENDIX A – Acronym List and Glossary

Term	Description
API	Application Programming Interface
FOSS	Free Open Source Software
HTML	Hyper-Text Mark-up Language
HTTP	Hyper-Text Transfer Protocol
IP	Internet Protocol
JavaScript	See: ECMA Script
JDBC	Java DataBase Connectivity
JSON	JavaScript Object Notation
KML	Keyhole Mark-up Language
ODBC	Open DataBase Connectivity
OGC	Open Geospatial Consortium
OMG	Object Modelling Group
RDBMS	Relational Database Management System
REST	REpresentational State Transfer
SDLC	System Development Life Cycle
SMB	Server Message Block. A protocol for file sharing on Windows and Unix based systems
SOA	Service-Oriented Architecture
SPARQL	Simple Protocol And RDF Query Language
SQL	Structured Query Language
TCP	Transmission Control Protocol
UML	Unified Modelling Language
WFS	Web Feature Server. A protocol for on-the-fly generation of map images using http requests.
WMS	Web Map Server. A protocol for query and download of vector maps using http requests.
WP	Work Package
WS	Web Service
WSDL	Web Service Description Language

XML	eXtensible Mark-up Language
SFS	Simple Features Specification
CVS	Concurrent Versioning System
WAI	Web Accessibility Initiative
WCAG	Web Content Accessibility Guidelines