



NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

Priority: IST-2004-2.4.7 – “Semantic-based knowledge and content systems”

D7.5.1 Software architecture for the ontology-based Fisheries Stock Depletion Assessment System (FSDAS)

Deliverable Co-ordinator: Claudio Baldassarre

Deliverable Co-ordinating Institution: Open University

Other Authors: Yves Jaques (FAO), Alejandro Lopez Perez (ATOS)

This document describes the architecture for the ontology-driven Fisheries Stock Depletion Assessment System (FSDAS). The goal is to provide with technical details to support T7.6 implementing activity.

Document Identifier:	NEON/2007/D7.5.1/v1.0	Date due:	August 31, 2007
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	August 31, 2007
Project start date:	March 1, 2006	Version:	v1.0
Project duration:	4 years	State:	Final
		Distribution:	Restricted

NeOn Consortium

This document is a part of the NeOn research project funded by the IST Programme of the Commission of the European Communities by the grant number IST-2005-027595. The following partners are involved in the project:

<p>Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta} @open.ac.uk</p>	<p>Universität Karlsruhe – TH (UKARL) Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 28 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de</p>
<p>Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.upm.es</p>	<p>Software AG (SAG) Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com</p>
<p>Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail address: jcontreras@isoco.com</p>	<p>Institut 'Jožef Stefan' (JSI) Jamova 39 SI-1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si</p>
<p>Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 655 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr</p>	<p>University of Sheffield (USFD) Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk</p>
<p>Universität Koblenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de</p>	<p>Consiglio Nazionale delle Ricerche (CNR) Institute of cognitive sciences and technologies Via S. Martino della Battaglia, 44 - 00185 Roma-Lazio, Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it</p>
<p>Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de</p>	<p>Food and Agriculture Organization of the United Nations (FAO) Viale delle Terme di Caracalla 1 00100 Rome Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org</p>
<p>Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.pariantelobo@atosorigin.com</p>	<p>Laboratorios KIN, S.A. (KIN) C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es</p>

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to the writing of this document or its parts:

FAO

ONTOPRISE

ATOS

Change Log

Version	Date	Amended by	Changes
0.1	12-07-2007	Claudio Baldassarre, Yves Jaques	Deliverable structure and first draft chapters
0.6	10-08-2007	Claudio Baldassarre, Yves Jaques	Rearrangements of Chapters; Completion of documents sections; Figures attachments; Spelling check
0.7	18-08-2007	Claudio Baldassarre, Yves Jaques	Minor documents addition, Document layout, Spelling check
0.8	10-09-2007	Claudio Baldassarre, Yves Jaques	Architecture components final revision, Spelling check
0.9	20-09-2007	Claudio Baldassarre, Yves Jaques	QA comments addressed, Spelling Check
1.0	09-10-2007	Aneta Tumilowicz	Final QA

Executive Summary

This document describes the architecture for the ontology-driven Fisheries Stock Depletion Assessment System (FSDAS) knowledge base. It will be a stand-alone Java Web Start (JWS) application that communicates via web services with a separate server instance of the NeOn core toolkit and an RDF triple store for unstructured data sources.

Users will experience FSDAS as a browsable and queryable application that returns organized, quality-rated, linked results that can be used to make decisions about the state and trends of various fish stocks. Fisheries information resources will be exploited using ontologies to return time-series statistics on capture, production, commodities and fleets by stock, together with direct links to related documents, web pages, news items, images and multi-media.

The document uses a variety of Kruchten's 4+1 views architecture to depict logical, process, deployment, implementation, data and use case views that serve as a blueprint for implementation work on FSDAS by NeOn partners.

The audience for this document are both the NeOn technical work packages that are or may be supplying components, as well as NeOn partner Atos Origin S.A. (ATOS). It is expected that partners will be able to use this document in order to understand which properties and operations their components shall be expected to provide in order to interface with FSDAS. For task T7.6¹, this document provides an architectural design blueprint from which both class-level design can take place and against which the possible use of GNU General Public License components-off-the-shelf (COTS²) can be evaluated.

¹ Task T7.6 will be lead by ATOS and other NeOn partners will participate in the implementation

² Throughout this document the term COTS refers to GNU General Public License (or similar) components that are distributed freely or are open source due to FAO software distribution policy.

Table of Contents

NeOn Consortium	2
Work package participants	3
Change Log	3
Executive Summary	3
Table of Contents.....	5
List of figures	9
1. Introduction	11
1.1 Purpose.....	11
1.2 Scope.....	11
1.3 Overview	12
2. Architectural Representation.....	13
2.1 Design architecture overview	13
2.1.1 Modularity:	15
2.1.2 Symmetry:.....	15
2.1.3 FSDAS iterative design.....	15
2.2 Final design description	16
2.2.1 Ontology related set.....	16
2.2.1.1 <i>Ontology Repository (CT)</i>	16
2.2.1.2 <i>Ontology Repository Access Service (OP)</i>	16
2.2.1.3 <i>Reasoner (CT)</i>	16
2.2.1.4 <i>Query Composition Manager (FSDAS)</i>	17
2.2.1.5 <i>Ontological Resource Query Manager (OP)</i>	17
2.2.1.6 <i>Data Model Mapping Manager (CT)</i>	17
2.2.2 Data Source related set.....	17
2.2.2.1 <i>Query Composition Manager (CT)</i>	17
2.2.2.2 <i>Data Source Query Manager (FSDAS)</i>	17
2.2.2.3 <i>Data Source Repository (FSDAS)</i>	18
2.2.2.4 <i>Data Source Repository Access Service (FSDAS)</i>	18
2.2.2.5 <i>Trend Composer Service (FSDAS)</i>	18
2.2.2.6 <i>Query Wrapping Service (FSDAS)</i>	18
2.2.2.7 <i>DLOs Indexer (FSDAS)</i>	18
2.2.2.8 <i>Summariser Manager (FSDAS)</i>	18
2.2.2.9 <i>Web Site Indexer (FSDAS)</i>	18
2.2.2.10 <i>Crawler (FSDAS)</i>	18
2.2.2.11 <i>Quality Manager (FSDAS)</i>	19
2.2.2.12 <i>Result Pre-processor (FSDAS)</i>	19
2.2.2.13 <i>Ranking Manager (FSDAS)</i>	19
2.2.2.14 <i>Sorting Manager (FSDAS)</i>	19
2.2.3 Visualization components set	19
2.2.3.1 <i>Browser (OP)</i>	19

2.2.3.2	<i>Display Manager (OP)</i>	20
2.2.3.3	<i>Multilinguality Manager (OP)</i>	20
2.2.3.4	<i>Ontological Resource Display Manager (OP)</i>	20
2.2.3.5	<i>Data Source Display Manager (FSDAS)</i>	20
2.2.4	Integration support set	20
2.2.4.1	<i>User management related components (OP)</i>	21
2.2.4.2	<i>User Account Manager (OP)</i>	21
2.2.4.3	<i>User Profile Manager (OP)</i>	21
2.2.4.4	<i>User Group Manager (OP)</i>	21
2.2.4.5	<i>User Session Manager (OP)</i>	21
2.2.4.6	<i>Legacy Data Import Manager (OP)</i>	22
2.2.4.7	<i>User Right Manager (OP)</i>	22
2.2.5	Resource annotation set.....	22
2.2.5.1	<i>Metadata Manager (FSDAS)</i>	22
2.2.5.2	<i>Annotation Manager (FSDAS)</i>	22
2.2.6	User communication set	22
2.2.6.1	<i>Communication Manager (FSDAS)</i>	23
2.3	Overlap with Ontology Life Cycle Management System	23
2.3.1	Mapping #1	23
2.3.2	Mapping #2	23
2.3.3	Mapping #3	24
2.3.4	Mapping #4	24
2.4	Added requirements.....	24
2.4.1	Indexing	24
2.4.2	External system querying	25
2.4.3	Result set interpretation.....	25
3.	Architectural Goals and Constraints	27
3.1	Architecturally significant non-functional software requirements	27
3.1.1	Functionality.....	27
3.1.2	Reliability.....	27
3.1.3	Usability	28
3.1.4	Efficiency.....	28
3.1.5	Maintainability	28
3.1.6	Portability	29
3.2	Design and implementation strategy.....	29
3.2.1	Design strategy.....	30
3.2.1.1	<i>Verify components with project partners</i>	30
3.2.1.2	<i>COTS analysis</i>	30
3.2.1.3	<i>Class design</i>	30
3.2.2	Implementation strategy	30
4.	Use-Case View	31
4.1	Architecturally significant use cases	31
4.2	Use-Case Realizations.....	32
4.3	User Interface mock-ups	36
5.	Logical View	43

5.1 Overview	43
5.2 First iteration design main principles.....	43
5.3 First iteration design partition	45
5.4 Architecturally Significant Design Packages	45
5.4.1 Visual Components.....	46
5.4.1.1 <i>Display Manager</i>	46
5.4.1.2 <i>Ontological Resource Display Manager</i>	47
5.4.1.3 <i>Data Source Display Manager</i>	48
5.4.2 User Components.....	49
5.4.2.1 <i>User Account Manager</i>	49
5.4.2.2 <i>User Profile Manager</i>	50
5.4.3 Integration Support Components.....	51
5.4.3.1 <i>Metadata Manager</i>	52
5.4.3.2 <i>Annotation Manager</i>	52
5.4.3.3 <i>Communication Manager</i>	53
5.4.4 Query Components.....	54
5.4.4.1 <i>Query Composition Manager</i>	54
5.4.5 Rank Components	54
5.4.5.1 <i>Ranking Manager</i>	54
5.4.5.2 <i>Sorting Manager</i>	55
5.4.6 Indexing Components.....	55
5.4.6.1 <i>DLOs indexer</i>	56
5.4.6.2 <i>Summarizer Manager</i>	56
5.5 Interfaces Description	57
5.5.1 User Environment Variables Interface.....	57
5.5.2 Metadata Interface.....	58
5.5.3 Query Interface	58
5.5.4 Query Result Interface.....	59
5.5.5 Ranked Query Result Interface	60
5.5.6 Ontology Model Interface.....	61
5.5.7 Document Instance Interface.....	62
5.5.8 Message Content Interface.....	62
5.5.9 User Contact Detail Interface.....	63
5.5.10 Terminology	63
5.5.11 Data Source Interface.....	64
5.5.12 Index Interface	65
5.5.13 Profile.....	65
5.6 First iteration design concise aspects	66
6. Process View.....	68
6.1 Account context.....	68
6.2 Search context.....	68
6.3 Display context.....	68
6.4 Interaction context.....	69
7. Deployment View	74

7.1 Overview	74
7.2 Configuration description	74
7.2.1 User downloads FSDAS application from Fishery portal	75
7.2.2 User runs FSDAS application	75
7.2.3 User Loads Ontologies	75
7.2.4 User performs a query:	76
7.2.5 User communicates with other users	76
8. Implementation View	78
8.1 Overview	78
9. Data View	80
9.1 RDBMS systems	80
9.1.1 Aquatic Sciences Fisheries Abstracts (ASFA).....	80
9.1.2 Electronic Information Management System (EIMS).....	80
9.1.3 FishBase	80
9.1.4 FIGIS RDBMS fact sheets	81
9.1.5 Fishery Resources Monitoring System (FIRMS)	81
9.2 Flat file systems	81
9.2.1 FIGIS flat-file fact sheets	81
9.2.2 RFB's document repositories.....	81
9.2.3 Globefish document repository	81
9.3 ISIS systems	82
9.3.1 FAOLex.....	82
9.4 Time-series systems	82
9.4.1 FIES commodity, capture, production and fleets databases	82
9.5 GIS systems.....	82
9.5.1 Species distribution maps.....	82
9.6 Ontologies	83
10. Size and Performance	84
10.1 Operating Environment	84
10.2 Implementation Constraints	84
10.3 Assumptions and Dependencies.....	84
Annex A – Updated use cases from Requirements D7.1.1	85
Summary Table of Use Cases and Priorities	115
Annex B - List of Acronyms used in this deliverable	116
References.....	118

List of figures

Figure 1 - FSDAS application design based on D7.1.1 user requirements.....	14
Figure 2 - Conceptual partition of FSDAS application design	15
Figure 3 - User diagram	32
Figure 4 - Search context.....	33
Figure 5 - Display context	34
Figure 6 - Interaction context	35
Figure 7 - Rubber band view.....	37
Figure 8 - Text view	38
Figure 9 - Toolbar detail.....	39
Figure 10 - File menu.....	40
Figure 11 - Edit menu	40
Figure 12 - View menu.....	41
Figure 13 - Annotate menu	41
Figure 14 - Bookmarks menu.....	42
Figure 15 - Help menu	42
Figure 16: FSDAS application first iteration design	44
Figure 17 - Collaboration diagram of account context components.....	70
Figure 18 - Collaboration diagram of search context components	71
Figure 19 - Collaboration diagram of display context components	72
Figure 20 - Collaboration diagram of interaction context components.....	73
Figure 21 - Deployment view of the FSDAS application main activities.....	74
Figure 22 - Layered vision of FSDAS application design.....	78
Figure 23 - FSDAS design conceptual partition against NeOn architecture layered vision	79
Figure 24 - Use case priorities	115

1. Introduction

The ontology-driven Fisheries Stock Depletion Assessment System (FSDAS) knowledge base will be a stand-alone Java Web Start (JWS) application that communicates via web services with a separate application instance of the NeOn core toolkit.

Users will experience FSDAS as a browsable and queryable application that returns organized, quality-rated, linked results that can be used to make decisions about the state and trends of various fish stocks. Fisheries information resources will be exploited using ontologies to return time-series statistics on capture, production, commodities and fleets by stock, together with direct links to related documents, web pages, news items, images and multi-media.

The user interface will support query refinement, assistance on query formulation (e.g. to avoid spelling errors) and multiple languages (e.g. Food and Agriculture Organization of the United Nations (FAO) languages: Arabic, Chinese, English, French and Spanish).

Users will be able to perform ontology browse-based and query-based searches using a single ontology or the union, intersection or complement of various ontologies. They will also be able to navigate associated data instances.

To the extent possible, the FSDAS will directly introduce and/or combine resources in the web page to create dynamic and synthetic views of the state of fish stocks. Users will be able to query and filter results based on their user profile.

1.1 Purpose

This document provides a comprehensive architectural overview of FSDAS. Using a variety of Kruchten's 4+1 views architecture (1995)³ to depict different aspects of the system it captures and conveys the significant architectural decisions which have been made. It is intended to serve as a blueprint for implementation work on FSDAS by NeOn partners, and as such can be placed in relation to several other deliverables. Specifically this document is to be viewed in relation to WP7 requirements document D7.1.1, [User requirements specifications for the Fisheries ontology, knowledge tools and alert system](#), which describes the requirements both for FSDAS as well as for the entire supporting ontology lifecycle management upon which FSDAS depends. This document shall also be considered in relation to ongoing deliverable D7.4.1.a, [Software architecture for managing the fishery ontologies lifecycle](#).

The audience for this document are both the NeOn technical work packages that are or may be supplying components, as well as NeOn partner Atos Origin S.A. (ATOS). It is expected that partners will be able to use this document in order to understand which properties and operations their components shall be expected to provide in order to interface with FSDAS. For task T7.6, this document shall provide an architectural design blueprint from which both class-level design can take place and against which the possible use of GNU General Public License components-off-the-shelf (COTS) can be evaluated.

1.2 Scope

The document applies to the entire expected development cycle of the FSDAS. It represents an architectural view of the requirements defined in *Chapter 5, Requirements for the Fisheries Stock*

³ http://www.inf.ed.ac.uk/teaching/courses/seoc/2006_2007/resources/Mod_5ViewModel.pdf

Depletion Assessment System found in D7.1.1, [User requirements specifications for the Fisheries ontology, knowledge tools and alert system](#).

The document focuses with greater detail on those aspects of the architecture which will be implemented by T7.6, or for which GNU General Public License COTS may be available but have not been identified or selected. A number of components described at high-level are not further described as they are part of the NeOn core toolkit. They will be provided by other NeOn technical work packages and they are described in the deliverables provided by those work packages. References to these documents can be found in [References](#).

The document considers in greater detail a subset of the overall architecture that will be implemented in the first iteration. As the second iteration will primarily involve the introduction of new components provided by partner technical work packages that have at this point not been clearly specified it was considered better to describe those components at high level only.

1.3 Overview

The document is organised by views. Following the introductory section, there is an architectural overview that gives a high-level view of the system. Goals and constraints are then described. Architecturally important use-cases are considered. The document then goes into greater detail covering main component properties and operations. This section also describes other work packages component properties and operations where these are known. This is followed by a deployment view that considers hardware and network. The implementation view then considers the components from a layered perspective. A data view looks at the persistent storage needs, before passing to non-functional descriptions of size and performance.

2. Architectural Representation

2.1 Design architecture overview

Architecture design for FSDAS application is based on the requirements described in deliverable [D7.1.1 chapter 5](#) and [Figure 1](#) which shows the components identified to map the required functionalities. In [Figure 2](#) one can see how the components are grouped together by function into similarly shaded areas:

- Visualization,
- Ontology related,
- Data Source related and,
- Integration support related.

Each of these supervises a functional area and has a communication point with the others in order to create the overall design; the picture also highlights those points of interest useful to a better understanding of the design decisions.

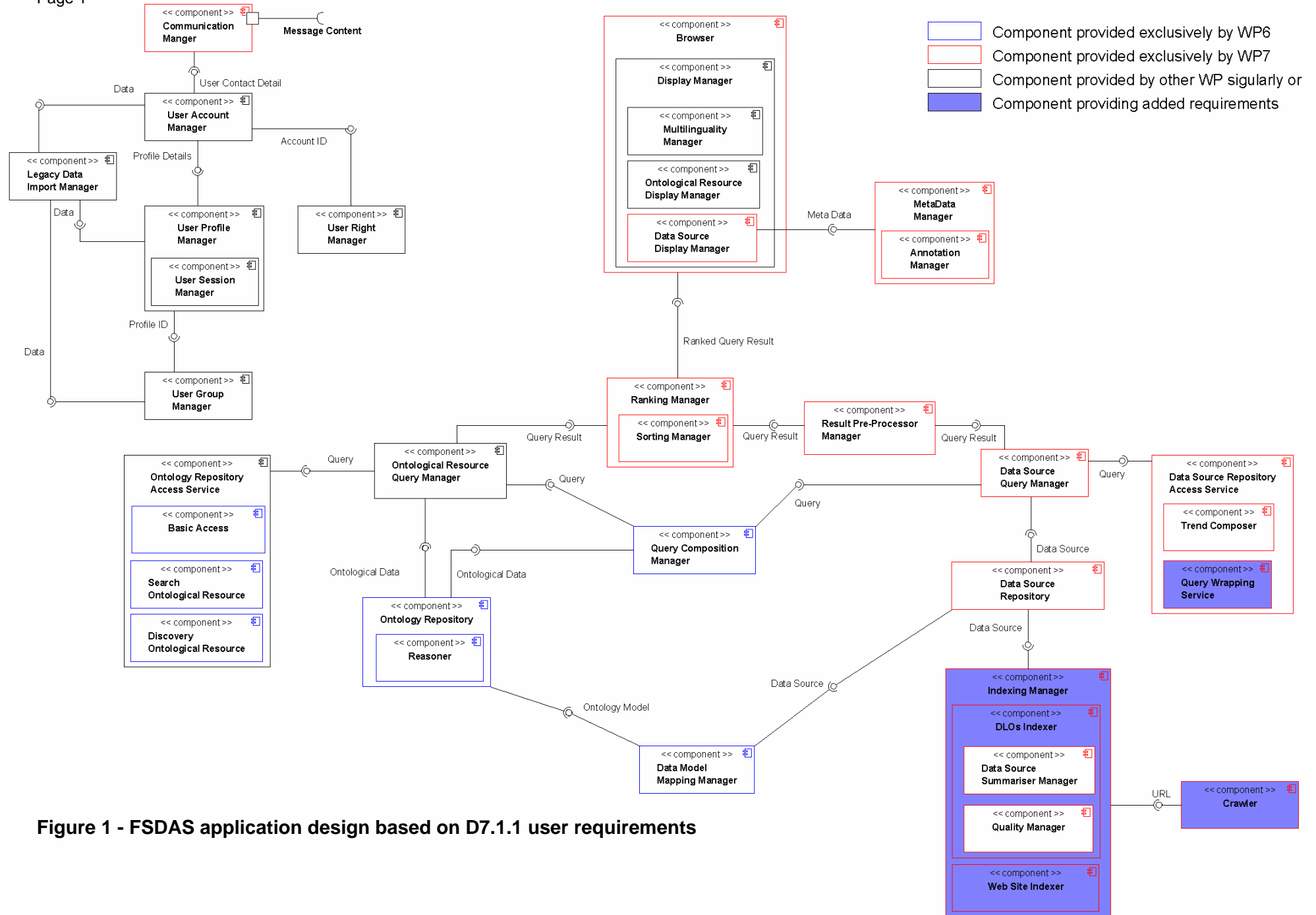


Figure 1 - FSDAS application design based on D7.1.1 user requirements

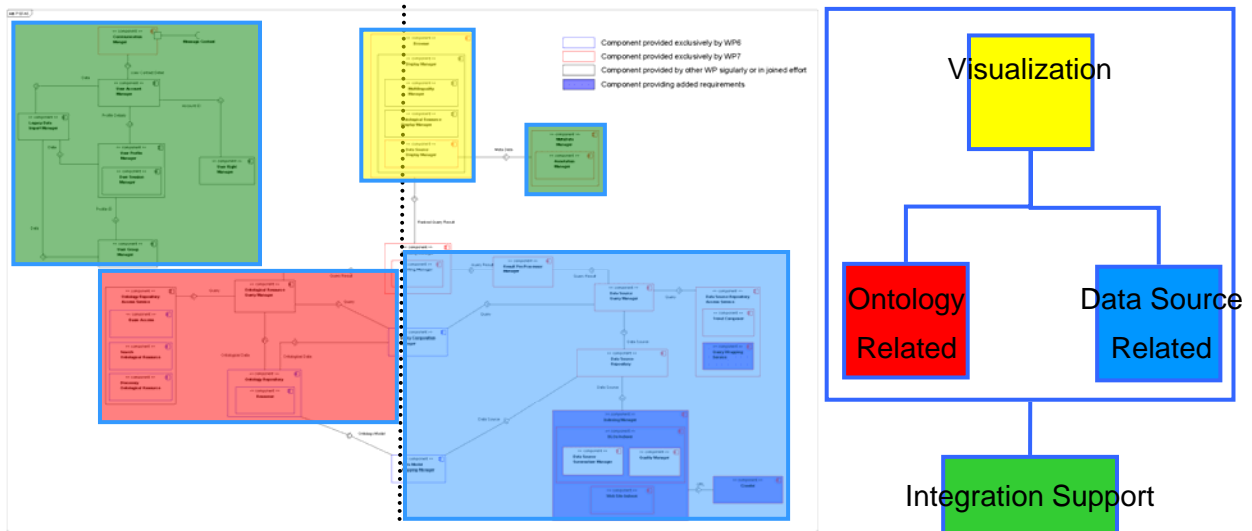


Figure 2 - Conceptual partition of FSDAS application design

2.1.1 Modularity:

As noted above the design is composed of four main sections: the visualization section serves display functionalities, the ontology and data source sections provide critical features related to the final goal of the application, finally, there is a section of components dedicated to integrating new user-generated content.

2.1.2 Symmetry:

There is symmetry in the design for components dealing with ontological resources and those handling data source resources. Both sides require a repository component, query functionality with query services and finally a ranking mechanism. Although functionally these components are the same, there is a deep difference in the type of resources they handle: on one side ontological resources, and on the other data resources that are outside the scope of the ontology. This inevitably affects the set of protocols, languages and formats of communication for the retrieval procedure, as well as for the infrastructure systems hosting the resources. The constraints listed in [3. Architectural Goals](#) are enough to motivate keeping the two wings of the design independent.

The interaction between these components is in the ontological support provided to the data source in the indexing (design time) and retrieval process (run time).

2.1.3 FSDAS iterative design

FSDAS design has gone through several reviews in light of the development point of other WP's contributing to the NeOn architecture in order to arrive at a first iteration design fulfilling what is actually realizable by the due date of the first implementation at month 26. In the following section there is the high-level description of all components belonging to the final design, while in section 5 there is a detailed technical view of just those components belonging to the first iteration design and not part of the general NeOn architecture.

2.2 Final design description

This section presents the description that takes in to consideration the design partition introduced previously and pictured in [Figure 2](#); from a high-level components set view, it then goes in to further detail describing the single components in each set.

It is useful to remember that at the level of architecture design, once a functionality is envisaged as necessary because it satisfies a requirement, its deployment and accessibility are up to the responsible work package to define; it is out of the scope of this document to indicate the final shape of the realization. Functionality needs to be thought of in terms of simple function rather than tightly or loosely coupled components performed by an embedded or remote service.

In the following presentation, a convention will be use to recall from [Figure 1](#) which components are core for the NeOn Toolkit (CT), which ones fulfil a functionality provided by partners other than WP6 (OP) and finally which are developed ad-hoc for the FSDAS application (FSDAS).

Components which are part of the first iteration design link to references in [Chapter 5](#) for further technical description.

2.2.1 Ontology related set

This set of components supervises all the functionalities strictly handling ontologies: storage, reasoning and querying. All these functionalities rely on NeOn infrastructure components, are developed in the responsible WP's and communicate with proprietary FSDAS components through defined interfaces. As FSDAS is a client/server application, some functionality like reasoning will be deployed as a stand-alone web service, together with other second iteration components that will sit on the server.

The following components are part of the ontology component set. They are part of the NeOn toolkit and are included in the FSDAS design for completeness, but they are more deeply described in other proprietary deliverables.

The list of components follows:

2.2.1.1 Ontology Repository (CT)

An infrastructure component that realizes the repository requirement for storing ontologies, this will be installed on a server within the FAO server/network infrastructure. Detailed descriptions of these components are found in [D6.1.1](#).

2.2.1.2 Ontology Repository Access Service (OP)

This component is conceptually representative of the embedded capabilities of the Ontological Resource Query Manager to access the repository of ontologies. It is represented as external to the query manager to highlight the requirements expressed in D7.1 and to leave the design open for future enhancement following the plug-in paradigm.

2.2.1.3 Reasoner (CT)

This is an infrastructure component that realizes reasoning services for ontologies. This feature is deployed as a web service accessible by the application. Detailed description of this component is in [D6.1.1](#).

2.2.1.4 Query Composition Manager (FSDAS)

This is a component that manages the way a user can compose a query which may be the result of either text editing, drag 'n drop action of concepts and/or properties, or natural language description.

The query composed will be a meta-query containing the query itself and added information from the user about preferences, where to search and what format. Based on this the query is addressed to the ontological and/or data source query manager.

Although more than a single way to compose a query is envisaged it is expected that for the first iteration the simpler text editing technique will be implemented.

Technical details for this component are in section 5.

2.2.1.5 Ontological Resource Query Manager (OP)

This component is a query manager specialized in querying ontological resources; it embeds the protocols and query languages necessary to retrieve these kinds of resources. It also is provided with access services to the repository for query execution. For the first iteration, support is expected for keyword search and SPARQL queries.

2.2.1.6 Data Model Mapping Manager (CT)

This component realizes the mapping process of a structured data model repository (relational DB or XML serialized) into an ontology structure such that any stored document is retrieved via an ontological query mechanism. We achieve uniformity of query language and uniformity of retrieved result format since the mapper also embeds languages and protocol interfaces for a number of well known data models.

The tool identified as the ontology mapper is OntoMap running on the server and available as a web service. See [D6.2.1](#) for a detailed description of OntoMap.

2.2.2 Data Source related set

This set of components is strictly related with data source resources, a generic name to identify any source of information contained in a system that consists of an unstructured Fishery document repository. These documents are heterogeneous in format, hosting repository and in their handling of I/O operations.

Two categories of documents are distinguished according to the way they can be processed: DLOs (document-like objects), which are text-based documents for which it is possible to perform text processing like natural language parsing and keyword indexing, and other sources of information such as time-series statistics, GIS maps and hydrographic data. These sources are equally relevant to the FSDAS final goal. To manage both these retrieval scenarios ad-hoc components have been designed to fulfil the requirements.

List of components follows:

2.2.2.1 Query Composition Manager (CT)

See description at [2.2.2.4](#).

2.2.2.2 Data Source Query Manager (FSDAS)

This component is a query manager specialized for querying data source resources. It embeds the protocols and query languages necessary to retrieve these kinds of resources. It also communicates with services used to access these resources and execute queries to the

repositories containing them, e.g. the *query wrapping service* and *trend composer* (used to format data series into a document suitable for human interpretation).

2.2.2.3 Data Source Repository (FSDAS)

This component is representative of all the data repositories in which documents relevant to the retrieval process are contained. These repositories are internal and external to FAO network domain and are heterogeneous in their format, protocols and languages of communication.

2.2.2.4 Data Source Repository Access Service (FSDAS)

This component is conceptually representative of the embedded capabilities of the Data Source Query Manager. This is represented as external to the query manager to highlight the requirements in this context and to supply it with a future way of introducing new features following the plug-in paradigm.

2.2.2.5 Trend Composer Service (FSDAS)

Component for charting numeric data, e.g. Eclipse-based BIRT (Business Intelligence and Reporting Tools) retrieved against a user query over a numeric data store. The trend will be the result of a query composed and executed specifically over numeric data stores and in a second step the query result is interpreted to produce a curve showing the trend.

2.2.2.6 Query Wrapping Service (FSDAS)

This component provides a service of query wrapping for different queried systems. It is invoked each time the Data Source Query Manger receives a meta-query from the Query Composition Manger with the information about the data to retrieve.

2.2.2.7 DLOs Indexer (FSDAS)

This component provides the mechanism by which given relevant domain ontology (ies) and an unstructured container of documents (system folders), an index is created taking the ontology terminology as the one source of indexing terms. In so doing we assure that the process creates indexes aligned with the domain ontology (ies) covering the document subjects.

The index created is stored as a file containing RDF triples that bind document URI's to ontological concept URI's.

2.2.2.8 Summariser Manager (FSDAS)

This component produces an abstract of the text documents contextually with the index generation.

2.2.2.9 Web Site Indexer (FSDAS)

This component manages indexing process of web pages composing the HTML source of information of fishery department portal and other linked sites. The indexing process works as described for the DLOs indexer, where an entry in the index file is represented as an RDF triple binding the document to the term URI which is an ontological concept belonging to feed terminologies.

2.2.2.10 Crawler (FSDAS)

This component performs crawling activity on the server hosting HTML content to discover if new data sources are available for the index process. It collects references (URL) to those possible new contents in order to feed the Web Site Indexer.

2.2.2.11 Quality Manager (FSDAS)

This is a component for assessing the quality of retrieved Data Sources. The assessing functionality can be realized in different ways according to the user's favourite perspective.

2.2.2.12 Result Pre-processor (FSDAS)

This is a component that manages the result set elements conversion from a proprietary format particular to a queried system into a uniform format available for easier visualization inside the application.

2.2.2.13 Ranking Manager (FSDAS)

This component supervises the ranking strategy for presenting results to the user. The rank mechanism takes into account the rank of the document either as an overall value, a local user value or both in combination. Semantic distance between the query and the returned result may be used as an alternative ranking approach.

2.2.2.14 Sorting Manager (FSDAS)

Component for managing rearrangement of the result set according to any aspect but the one considered for ranking. The aspects envisaged could be timestamp, alphabetical order, concept clustering, or role in the RDF triple (Subject, Object) when SPARQL is used to formalize the query.

2.2.3 Visualization components set

FSDAS application is heavily dependent on the graphic interface to facilitate user exploitation of its functionalities. In the original requirements it was asked that FSDAS be a browser-based application, since many of the existing systems actually used by Fishery department have content distributed through a web interface. The interface is mainly divided in:

- General purpose part consisting of all the panels needed for classical input such as login activity, user details update or content annotation;
- Ontological resource display panels specialized for visualising ontological elements and
- Data source display panels specialized for visualizing document instances.

In the review process of the full architecture design it has been agreed with FAO that the FSDAS application will not be a web application, due partly to complex visualisation requirements and partly because the technology used by contributor WP's to develop their components does not well integrate with thin-client technologies suitable for web browser environments. Java Web Start technology has been chosen since it offers the strength and flexibility of Java, easier reuse of work provided by contributor WP's and a simple distribution mechanism for end users.

List of components follows:

2.2.3.1 Browser⁴ (OP)

The Browser component is representative of the web environment and encapsulates all those graphic panels that provide input and display functionalities useful for the user to interact with the application.

⁴ Although this component is included in order to map the architecture to the initial requirements, it will not be part of the final system architecture as following the review process it has been determined to be of low utility.

Since after review of the final design it has been agreed that FSDAS will not be a web application anymore, this component is excluded because it does not fulfil the requirement.

2.2.3.2 Display Manager (OP)

This component encapsulates the graphical object formerly embedded in the web browser.

Since the agreement to use Java as the implementation language, this component is representative of the main window of the application maintaining the characteristic of encapsulating all the panels providing input and display functionalities useful for the user to interact with the application.

Technical details for this component are in section 5.

2.2.3.3 Multilinguality Manager⁴ (OP)

This component manages multilingual descriptions of the ontological resources and metadata associated with ontological and data source resources.

In the review process it has been clarified that the functionality to have natural language descriptions in the other five FAO official languages is not related to a run time translator but is realized by the attempt to map a description expressed in a certain language with the same expressed in another available translation. This is out of the scope of FSDAS application hence this component is not included in the design.

2.2.3.4 Ontological Resource Display Manager (OP)

This component displays ontological resources. Ontological resources can be entire ontology (ies) gathered by the user from the repository, a piece of the ontology model, or a single ontology element given the focus by navigating or performing queries.

Panels to display single ontological elements (Class, Property, and Individual) and their attributes directly or their inferences are also part of the functionalities provided by this component.

Technical details for this component are in section 5.

2.2.3.5 Data Source Display Manager (FSDAS)

This component displays document instances gathered in response to navigating the focused ontology or by performing queries. Document instances are those text documents, web pages, news items considered.

Technical details for this component are in section 5.

2.2.4 Integration support set

This set of components provides functionalities not strictly related to the FSDAS final version, but at the same time integrates necessary services an interactive, multi-user application must have. Further subdividing this set, 3 main groups are identifiable by topic activity:

- User management related components;
- Resources annotation related components and
- User communication related components.

List of components follows:

2.2.4.1 User management related components (OP)

As in any multi-user application, managing registered users at different levels is a mandatory requirement. For this purpose FSDAS design considers generation of a semi-blank profile contextually with the creation of a new user account, and then associates the profile with existing user groups.

The possibility to manage account details at three different levels is achieved by adopting:

- Account level: cancellation, enabling/disabling user, rights assignation;
- Profile level: environment customization, preferred resources assignation;
- Group level: group policies (execute the same action on a group of profiles connected with single users), group rights.

2.2.4.2 User Account Manager (OP)

This component fulfils the need of managing a single user registered to the application to gain access to the resources. A user creates his own account by filling in most common personal details, at the same time a general profile is generated that can be customized at a later stage.

Each user must have an account for reasons related to the requirements and to the application environment: an identified user is tracked during his normal activity and the information is fed to the profile manager to enrich its content. The user identity is necessary when he produces new contents about the resources he handles. The user's right to view retrieved resources are strictly related to his identity. Finally, the account is needed for any eventual communication by other users.

Technical details for this component are in section 5.

2.2.4.3 User Profile Manager (OP)

This component manages a single user in terms of environment adjustment to usual activities in the application;

The user profile is built by collecting a few pieces of general information during the registration process and then by tracking user actions and saved preferences.

The general details collected at registration time to fill in the user profile can be: expertise, area of interest, resources of interest and relevant sources of information. Mainly, these variables are subject to automatic augmentation while the user acts in the application over a reasonably long period of time.

Technical details for this component are in section 5.

2.2.4.4 User Group Manager (OP)

This component manages groups of registered users who use the application.

As described in the requirements, FSDAS users are fisheries experts within some domain of interest; therefore managing groups of users to apply group policy is a must within such a scenario. When we want to modify some domain, it is recommended to do this by acting at group level instead of at single user level.

2.2.4.5 User Session Manager (OP)

This component manages the session that the user either wants to save or that is automatically saved by the application. A session is a set of information such as last open documents, used

ontologies, workspace layout, preferences and settings. Saving the session persists this information to the profile manager as part of its regular activity of user behaviour tracking.

The technical details for this component are included in section 5.

2.2.4.6 Legacy Data Import Manager (OP)

This component manages legacy data in terms of existing users FAO may want to import from existing systems. The import process is better seen as a mapping process from existing data model to future data model. Another view of the import process can be interfacing existing storage with future repositories.

2.2.4.7 User Right Manager (OP)

This component supervises user access to any resources involved in the application. These can be ontological resources (ontology modules, entire ontology) or data source resources through ontology access restriction.

2.2.5 Resource annotation set

As an interactive application it is important the user can associate personal annotations to retrieved resources whether ontological or data instances. The annotation about any resource in the application will be constructed by declaring an RDF statement over the resource URI and storing in a local repository. The content produced will be valid only within the scope of the user meaning that it will be loaded at login time.

The list of components follows:

2.2.5.1 Metadata Manager (FSDAS)

This component realizes the functionality of retrieving all annotation properties declared over the focused resource and passes them to the graphic interface component (Display Manager) to be visualized.

Technical details for this component are in section 5.

2.2.5.2 Annotation Manager (FSDAS)

This component realizes annotation functionality over both ontological and data source resources. Annotation is a general term which can take the form of comments in natural language, tags, resource ratings or text abstracts. Annotations are created either by the user: comments, resource ratings, tags or automatically as the result of data source processing: tags or text abstracts. In both cases annotations are stored on the server-side in the RDF triple store.

Technical details for this component are in section 5.

2.2.6 User communication set

Being a multi-user application it is important to support user communication and object exchange. FSDAS implements a mechanism providing the users with capabilities of message exchange, with other registered users, as well as object exchange produced in each individual session e.g. result data set, annotation, user query and resources exported as RDF syntax, which don't have global validity but are proper to each user's session.

List of components follows:

2.2.6.1 Communication Manager (FSDAS)

This component manages communication among users registered to the application. Communication includes text messages as well as object exchange, e.g. result data set, annotation, user query and resources exported as RDF format.

Technical details for this component are in section 5.

2.3 Overlap with Ontology Life Cycle Management System

In the process of requirements analysis many functionalities of the FSDAS application have been discovered to be in common with the Ontology Life Cycle Management System described in D7.1.1 at chapter 4.

As confirmation of overlap between the two systems, use cases showing shared functionality have been identified across the two applications:

Notes:

The pairing is not ONE to ONE due to a different level of description granularity between the two deliverables. For these examples the single use case covers multiple finer-grained use case(s).

Although functionally equivalent, components developed for FSDAS cannot be shared with D7.4.1 depending on the different developing and running environment: FSDAS is deployed as JWS application while D7.4.1 is an architecture deployed in Eclipse platform. FAO distribution constraints make JWS as the best solution for FSDAS, but anyway a level of functionality sharing is preserved for all those engineering and infrastructure components deployed as web services which both the architecture can access remotely.

2.3.1 Mapping #1

UC-6 Search ontological resource in ontology

Maps D7.4.1:

- UC-1.1: Search Using Free Text
- UC-1.2: Search Using Advanced Input

2.3.2 Mapping #2

UC-11 Query composition

Maps D7.4.1:

- UC-2.1: Answer Standard Query (not applicable to subject experts)
- UC-2.2: Answer Domain Expert Query
- UC-2.3: Answer Structural Query

2.3.3 Mapping #3

UC-8 Browse Taxonomy

Maps D7.4.1:

- UC-7.1: Visualize Ontology
- UC-7.2: Visualize Mappings and Relations between Ontologies
- UC-7.3: Browse Ontology

2.3.4 Mapping #4

UC-18 Propose ontology modification

Maps D7.4.1:

- UC-12.6: Send to “To Be Approved”

2.4 Added requirements

In the process of design review some new issues have been raised concerning data source retrieval techniques. These are deeply affected by the heterogeneity of data sources format, repository data models and query API's.

Following the [Fisheries Systems Inventory T7.2.1](#) it is clear that a portion of the information systems designed to be exploited by FSDAS contain data types that are difficult to index (e.g. time-series statistics, hydro-graphic data and GIS maps) using standard indexing components such as Lucene. Thus there is a need for query mechanisms that can exploit both pre-made indexes (more efficient) and remote system query APIs.

2.4.1 Indexing

Various retrieval strategies have been discussed and a clear need has emerged to be able to index systems containing document-like objects (DLOs). The case study application when searching for related data instances should use indexes of these underlying systems instead of the system's query API. An example would be: <http://www.fao.org/documents/>

Such a method carries with it a set of implications for the case study design:

- Components for index building need to be included. These components should be open source such as Sesame or Lucene.
- These components must be able to index not just web pages but other DLOs such as MS-Word documents and Acrobat documents.
- A component that manages indexing tasks must also be considered, such that indexes can be rebuilt periodically, automatically.
- The indexing component needs to be able to use URL's for indexing and cannot rely on a pool of documents manually placed in a local folder.

- A crawling component would be desirable for web-site indexing, in order to generate URL's for the indexer. An example of such a site would be: <http://www.globefish.org/>

2.4.2 External system querying

Some of the systems considered for exploitation within the FSDAS do not contain DLOs and cannot be indexed using available indexing technologies. These systems must be queried and their response either presented within the application or within an associated application. An example would be:

http://www.fao.org/figis/servlet/TabLandArea?tb_ds=Capture&tb_mode=TABLE&tb_act=SELECT&tb_grp=COUNTRY

Again this requirement implies a set of required functionalities:

- A query wrapping component capable of taking a query within the FSDAS and rewriting it according to the needs of the system being queried.
- The previous point implies a set of properties maintained for each such system describing the query service and parameters, including the indexing system used for DLOs
- It should be possible to make queries using several methods, such as SQL via JDBC, or via URL using either standard frameworks such as SOAP, or non-standard URL's that return possibly RDF, but more likely XML, HTML or CSV not conformant to any standard data representation.
- SOAP technology for distributed objects should be the preferred method.

2.4.3 Result set interpretation

Closely related to indexing and querying is the interpretation of the remote system's response. Using a remote system's query API will in many cases imply a non-standard response which may come in a variety of formats and models. FSDAS will need a component capable of interpreting and reformatting such responses into a standard result. An example is:

[http://www.fao.org/figis/website/SearchActionXML.do?kv\[0\]=oyster](http://www.fao.org/figis/website/SearchActionXML.do?kv[0]=oyster)

The component shall:

- Maintain a set of properties for remote systems describing the format and model of the system.
- Be able to automatically convert the remote system response using the aforementioned properties into a standard result that can be displayed within the FSDAS application or an associated application, e.g. Acrobat or Word.
- Be able to merge results from several remote systems into a single result set.

As a solution to these additional requirements the design has been augmented with components like:

- DLOs Indexer
- Web Site Indexer
- Crawler

- Query Wrapping Service
- Result Pre-processor

3. Architectural Goals and Constraints

This section uses a *subset* of the categories from ISO9126 to describe the non-functional software requirements and objectives that have some significant impact on the architecture. It also captures NeOn project-related constraints to the design and implementation strategy and schedule.

3.1 Architecturally significant non-functional software requirements

3.1.1 Functionality

Interoperability

FSDAS shall be interoperable with the NeOn core toolkit or at least with the instance of the NeOn toolkit installed in FAO customized to the fisheries domain. This implies that all external interfaces will communicate via web services. This will guarantee a continuous technological support as long as the NeOn toolkit is maintained.

Due to FAO network security constraints, the server-side of FSDAS must reside within the FAO firewall in order to be able to connect with local databases.

Compliance

The software code for the application shall be General Public Licence (GPL), Version 2. It is policy of the FAO to distribute software freely to any member state entity involved in project or common activities.

Security

Access to the system shall require a user account approved by an administrator of the NeOn toolkit or at least with the instance of the NeOn toolkit installed in FAO customized to the fisheries domain, with which the FSDAS communicates. Centralization of identification is a necessary condition to manage sensible data owned by FAO; any user of the application must provide minimum identification details for gaining grant to the resources.

3.1.2 Reliability

Maturity

The software shall be subjected to unit, component application and validation testing to insure it is sufficiently bug free to permit normal operation. It shall not be subject to uncaught errors that cause it to freeze or be otherwise unusable.

An XUNIT style code-driven testing framework such as JUNIT shall be used for unit testing.

Fault Tolerance

The software shall be designed to recover gracefully from common error situations such as sudden lack of network connectivity and/or unavailability of component services. The system shall log all such error situations and provide end users with recovery information.

3.1.3 Usability

Learnability

It shall be possible for the end user group identified in the requirements to be able to use the software as intended with a minimum of effort. E.g. a half day training course shall be able to suffice for a fisheries scientist to be able to operate the system.

The system shall have at least a simple help system component linked to underlying functionality such that future expansion of the system can be easily accompanied by expansion of the help system.

The system shall have at least a simple operation manual.

Understandability

The system shall be structured where possible to avoid too many new ways of working. Menus, icons and labels shall exploit where possible known paradigms and working practices. Semantic web jargon and ontological jargon shall be kept to the minimum possible. Areas that are likely to be unfamiliar to the average audience as defined in the requirements shall present additional help or clarification to the user to enhance application clarity, and self-descriptiveness.

Users shall be able to navigate intuitively the ontologies and their associated instances without having more than a very basic understanding of the underlying metamodel.

Operability

The system shall be easily operable using standard computer input devices, e.g. keyboard and mouse. Users shall not be required to understand the exact meaning of OWL or RDF triples, but rather be able to navigate ontologies graphically by browsing concepts and following their associated relations.

3.1.4 Efficiency

Time Behaviour

The system should respond in real time to user ontology navigation and not require a full screen refresh for changes. Average query return time should be less than five seconds.

Resource Behaviour

The client-side application shall be able to operate on a normal Wintel desktop computer, e.g. Pentium IV 3 GHz, 1 GB Ram, 128Meg video card.

3.1.5 Maintainability

Stability

As FSDAS is being developed according to an iterative process and given that it needs to interact with a number of components provided by other partners, stability is an important issue. Object classes should show low coupling and high cohesion within components. External interfaces in particular should be considered to be extensible only.

Analysability

It should be easy to diagnose deficiencies or causes of failures in the software. Logging design should be considered as an ongoing activity during class design and implementation, and a logging library such as Log4J should be considered.

Application programming interface (API) documentation shall be created for all classes, attributes, variables, constructors and methods using the JavaDoc code documentation tool.

Changeability

Architecture of object classes should follow Model-View-Controller (MVC) and standard object-oriented design patterns, e.g. observer, decorator, etc. to implement low coupling and high cohesion within components, allowing for an application that is as changeable as possible, particularly given the iterative nature of the case study within the NeOn project.

3.1.6 Portability

Installability

The software should be installable via web URL using Java WebStart technology.

Replaceability

New versions should be installable over older versions without the loss of user profile or annotation data.

Adaptability

The application should be generic enough that it can be used for any domain, not just fisheries. This implies that all labels, domains, user groups, etc. be easily configurable.

3.2 Design and implementation strategy

This deliverable defines needed components for the vision of FSDAS as defined in the requirements document D7.1.1. As many of these components are provided by other project partners and have not been specified, these components have been left necessarily vague.

Described in greater detail are the components needed for a first iteration based on a subset of the D7.1.1 requirements.

In particular, those components needed for the first iteration that will be provided solely by the T7.6 leader have been described in the greatest detail.

The design remains at component level and does not go to class level. Only external component interfaces are described. Class-level design and implementation is left to task T7.6, thus allowing the implementer to select GNU General Public License or similar COTS which they or others may have available rather than being constrained by a prior specification.

3.2.1 Design strategy

3.2.1.1 Verify components with project partners

For toolkit components with which FSDAS interfaces, task T7.6 should investigate with the project partners creating those components documentation on the external interfaces, i.e. operations, properties and exceptions that are supported, as well as an indication of the expected delivery date.

3.2.1.2 COTS analysis

For FSDAS-specific components for which no provider has been identified, task T7.6 should perform a COTS analysis, both for components they may have in house, as well as components created by other groups doing ontological applications. A suggested COTS analysis framework is EPIC, by Carnegie Mellon, though T.7.6 leader may have an in-house method; COTS components must be distributed following GNU General Public License or similar, i.e. intended for use as free or open source due to FAO software distribution constraints stated [here](#).

3.2.1.3 Class design

Following COTS analysis it should be possible to create a class-level design for the remaining components.

3.2.2 Implementation strategy

Implementation should begin by satisfying the most [architecturally significant use cases](#) that are a part of the [first iteration](#). Exact order is left to the implementer.

4. Use-Case View

This section lists use cases or scenarios from the use-case model that represent significant, central functionality of the final system, have a large architectural coverage, or stress or illustrate a specific, delicate point of the architecture.

4.1 Architecturally significant use cases

NOTE: See [Appendix A](#) for full list of use cases.

The following subset of FSDAS use cases together exercise virtually every component of the system.

- UC1.3 Register
- UC1.6 Search ontological resource in ontology
- UC1.7 Search for related ontological resources
- UC1.8 Browse Taxonomy
- UC1.11 Query Composition
- UC1.12 Query for Data related to individual
- UC1.20 Save session
- UC1.21 Generate RSS feed from current query
- UC1.24 Annotate retrieved document with comments

4.2 Use-Case Realizations

This section illustrates how the software actually works by giving a few selected use-case realizations, and explains how the various design model elements contribute to their functionality.

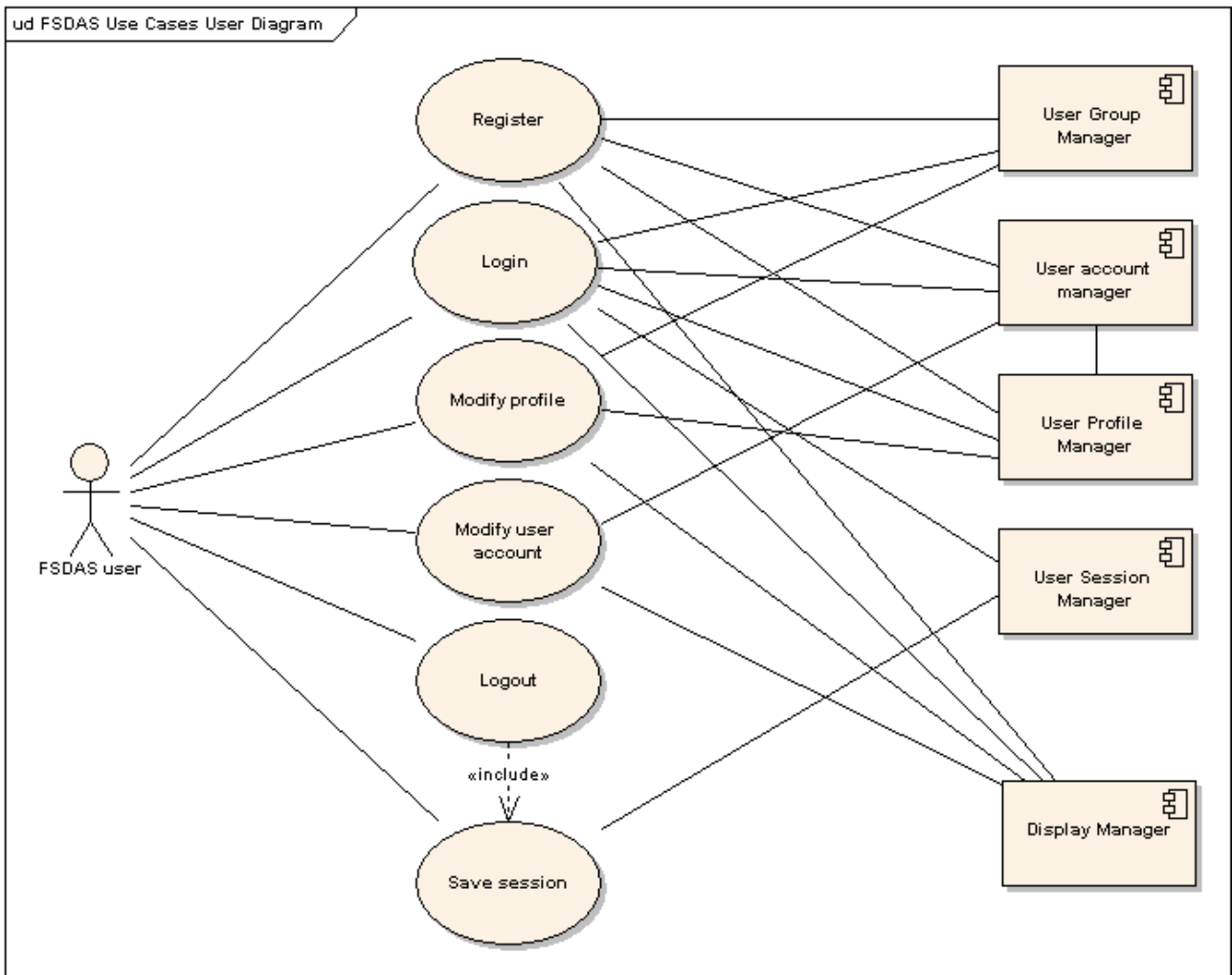


Figure 3 - User diagram

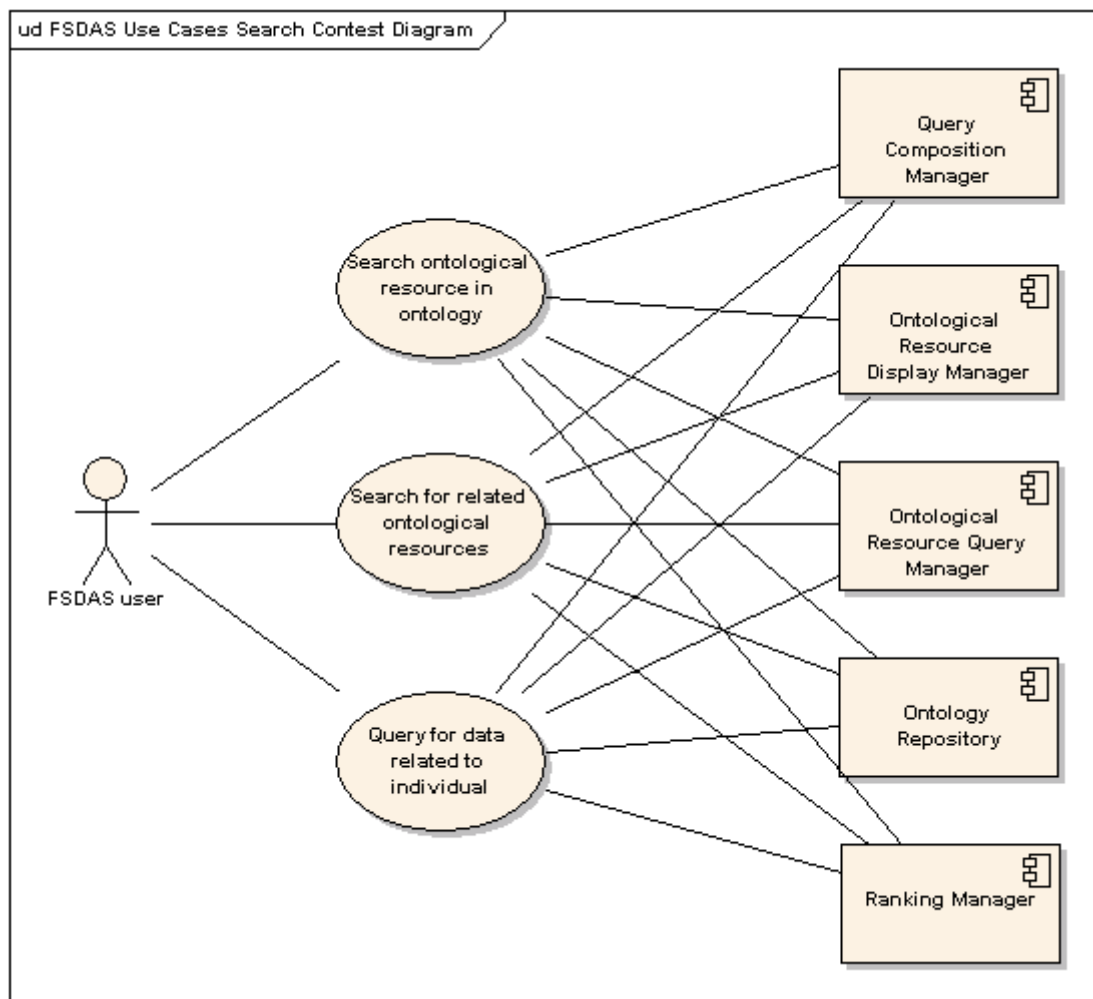


Figure 4 - Search context

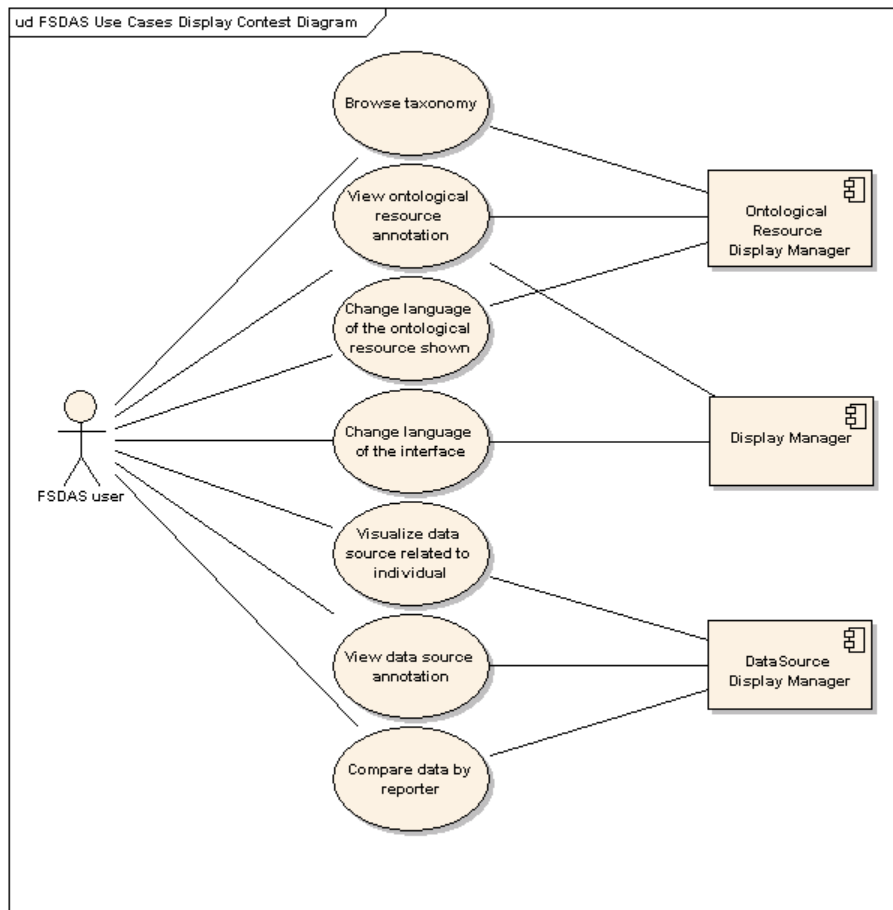


Figure 5 - Display context

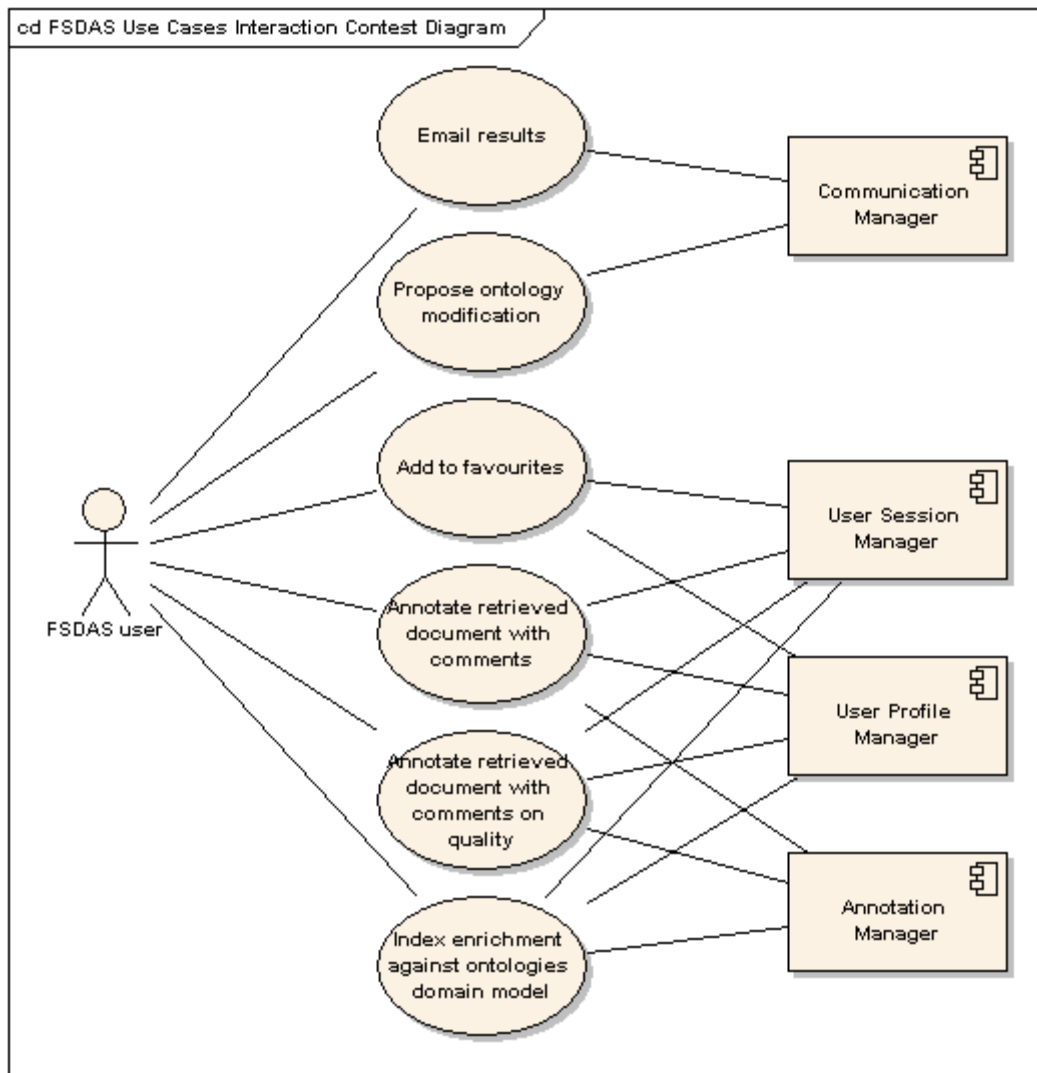


Figure 6 - Interaction context

4.3 User Interface mock-ups

The user interface mock-ups take the requirements and use cases and attempt to express them visually to give some guidance to the software implementers. They need not be taken as a literal expression of what must be realised, but more as guidance in interpreting the use case requirements.

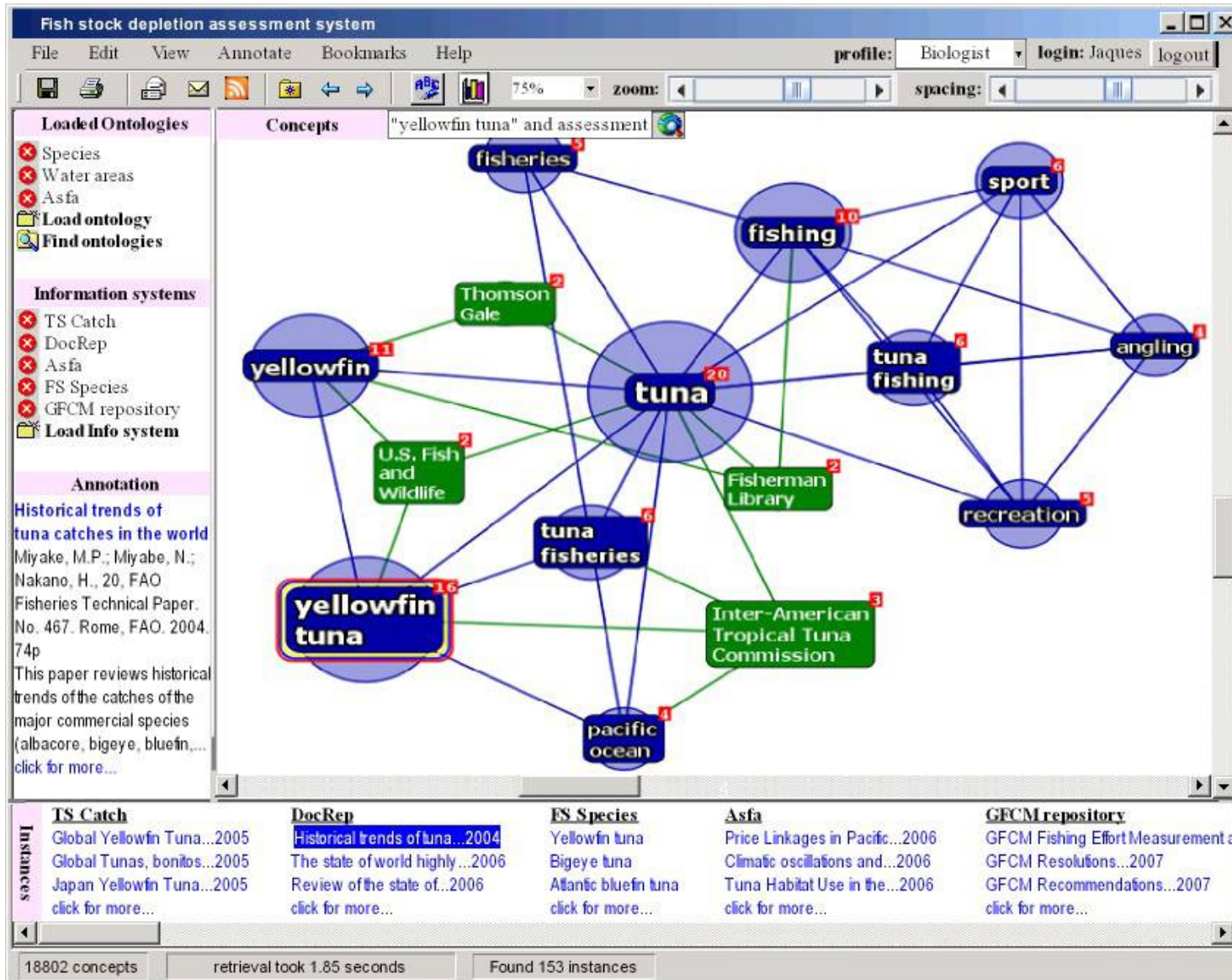


Figure 7 - Rubber band view

The screenshot shows a web application window titled "Fish stock depletion assessment system". The interface includes a menu bar (File, Edit, View, Annotate, Bookmarks, Help), a toolbar with various icons, and a search bar containing "yellowfin tuna". The main content area is divided into several sections:

- Loaded Ontologies:** A list of ontologies with checkboxes, including Species, Water areas, Asfa, and Information systems.
- Information systems:** A list of information systems with checkboxes, including TS Catch, DocRep, Asfa, FS Species, and GFCM repository.
- Annotation:** A section titled "Historical trends of tuna catches in the world" with a brief description and a "click for more..." link.
- Concepts:** The central focus is "Yellowfin Tuna". It lists:
 - Properties:** Scientific Name: Thunnus albacares, Taxonomic code: 1750102610, FAO 3-Alpha code: YFT.
 - Narrower terms:** none.
 - Broader terms:** Taxonomy (+Family, +Order, +Major group), Commercial (+ISSCAAP group, +ISSCAAP division, +FAOSTAT), and Commodity (+ISSCFC classification, +EU harmonized, +FAO-FIDI yearbook, +Harmonized).
 - Relations:** A list of relationships such as isFoundIn, hasPopulation, isFishedBy, isLandedIn, isExploitedBy, isMonitoredBy, isManagedBy, and isSubjectTo, each with associated terms like +Water area, +Stock, +Resource, etc.
- Instances:** A table at the bottom showing instances from different ontologies:

Instances	TS Catch	DocRep	ES Species	Asfa	GFCM repository
	Global Yellowfin Tuna...2005	Historical trends of tuna...2004	Yellowfin tuna	Price Linkages in Pacific...2006	GFCM Fishing Effort Measurement a
	Global Tunas, bonitos...2005	The state of world highly ...2006	Bigeye tuna	Climatic oscillations and...2006	GFCM Resolutions...2007
	Japan Yellowfin Tuna...2005	Review of the state of...2006	Atlantic bluefin tuna	Tuna Habitat Use in the...2006	GFCM Recommendations...2007
	click for more...	click for more...	click for more...	click for more...	click for more...

At the bottom of the window, a status bar indicates: "18802 concepts", "retrieval took 1.85 seconds", and "Found 153 instances".

Figure 8 - Text view

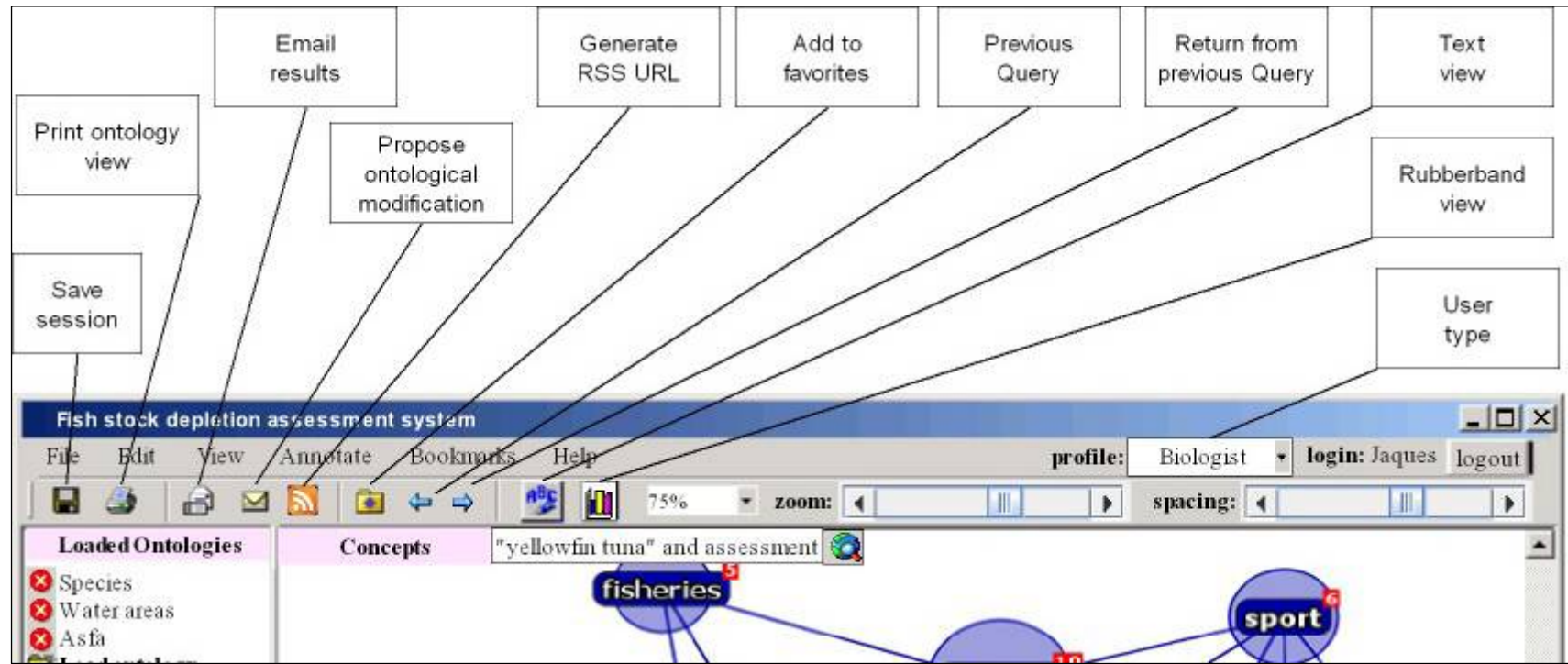


Figure 9 - Toolbar detail

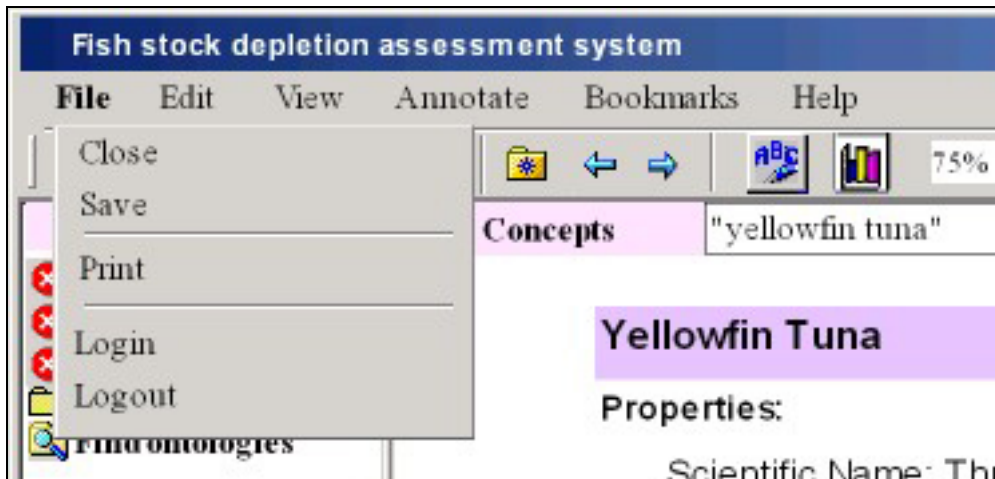


Figure 10 - File menu

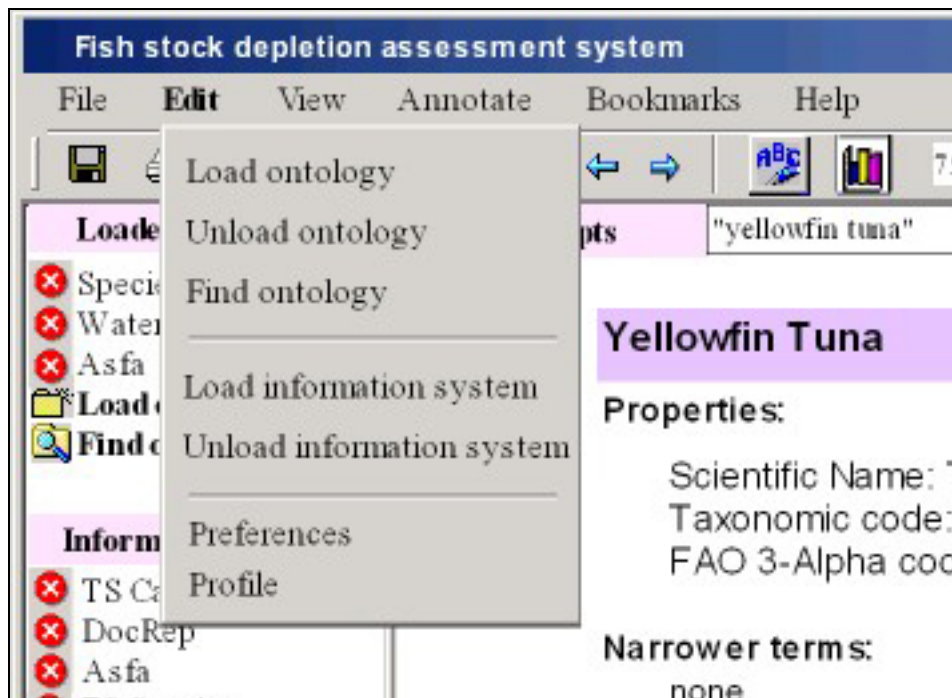


Figure 11 - Edit menu

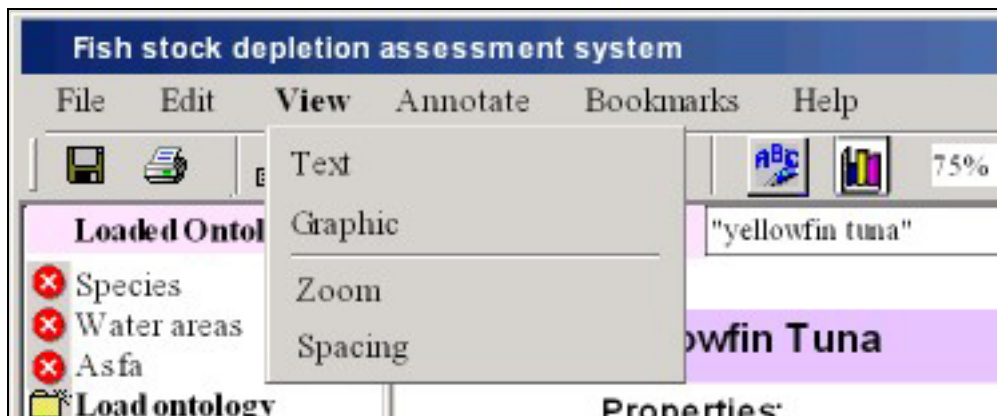


Figure 12 - View menu

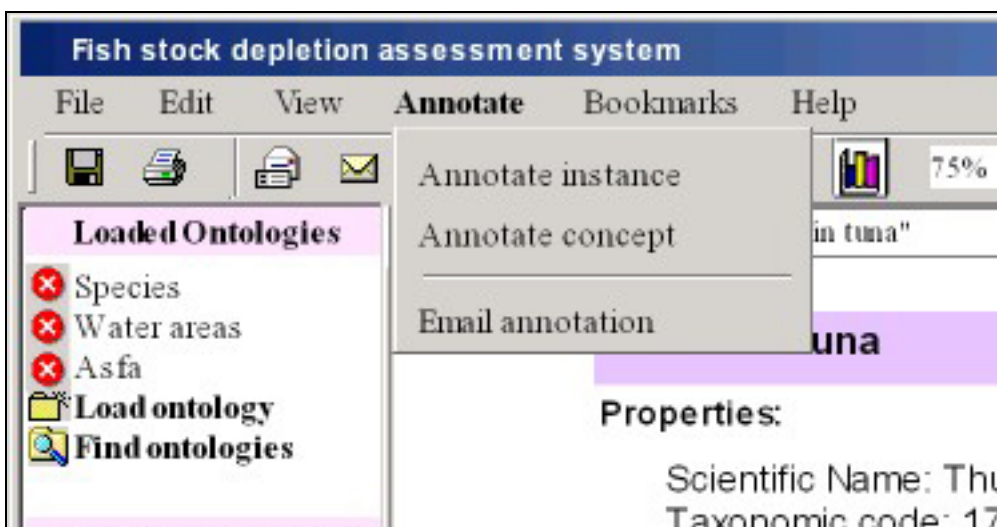


Figure 13 - Annotate menu

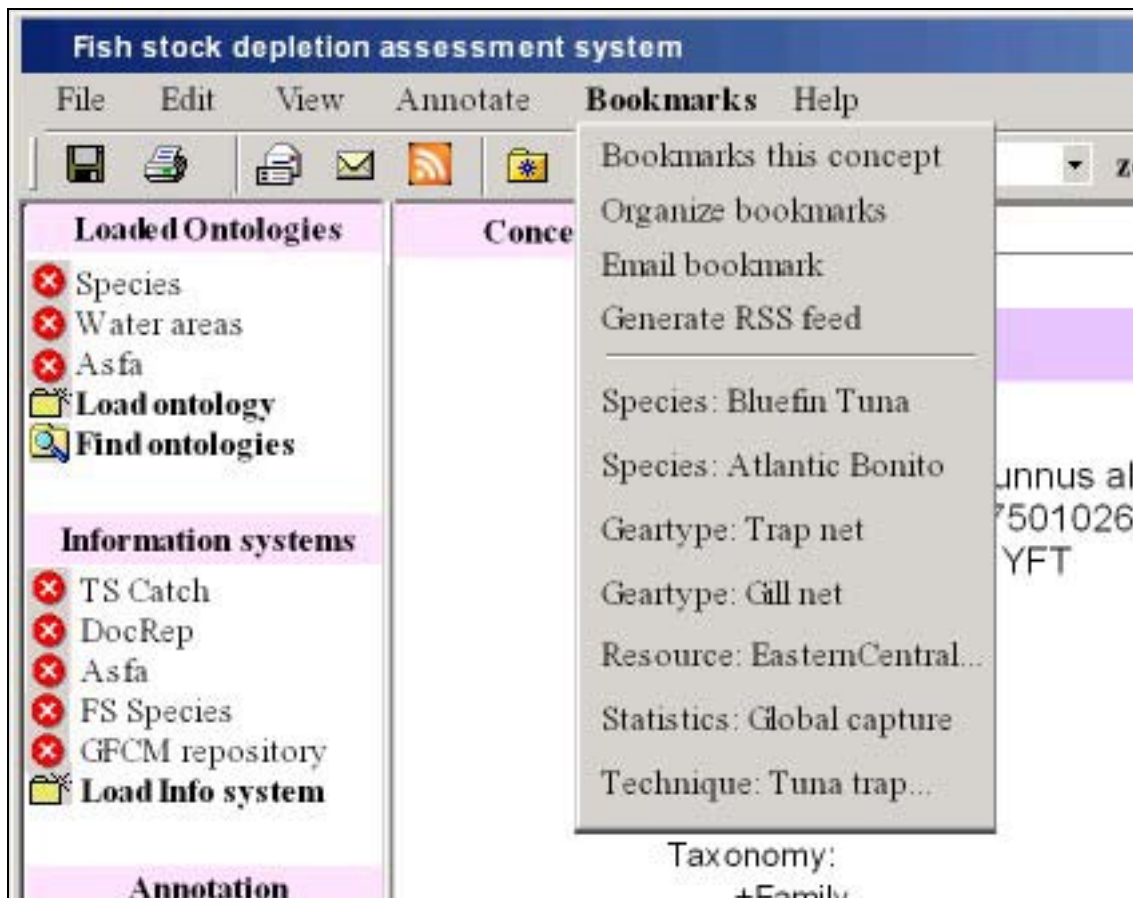


Figure 14 - Bookmarks menu

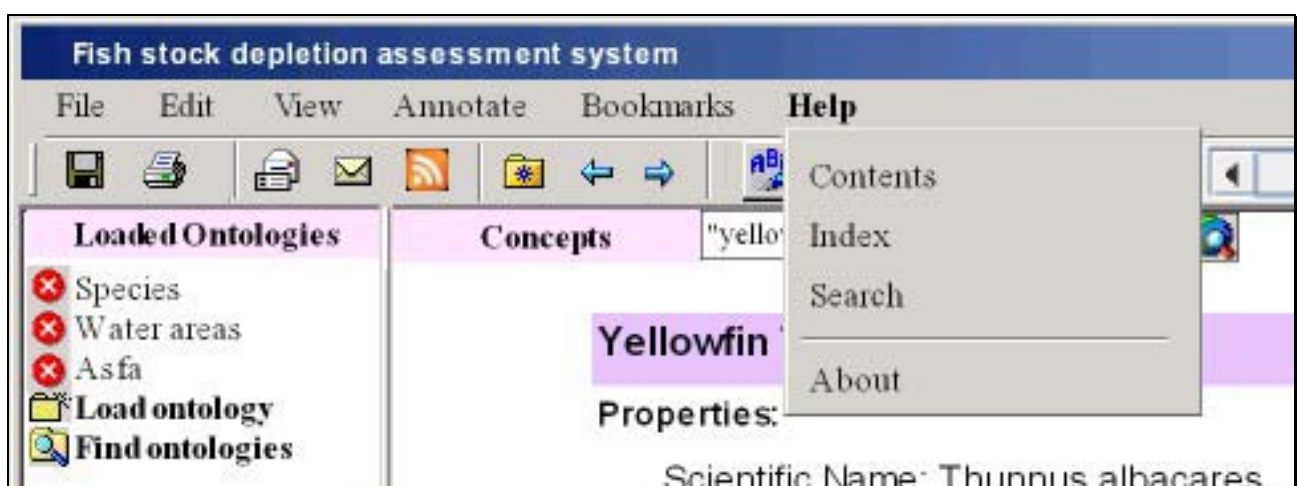


Figure 15 - Help menu

5. Logical View

5.1 Overview

The Design architecture introduced in chapter 2 is referred to as final design architecture compared with the first iteration architecture because it covers all the requirements described in the deliverable D7.1.1.

Because some of the technologies implied by that design are not mature enough to provide the required functionality (ies), a subset has been prioritized according to the availability at the time of the writing of this deliverable. With this approach, the first iteration architecture is excerpted and technically described in this section. This is also the architecture to implement by M26.

5.2 First iteration design main principles

In the first iteration component diagram in [Figure 16](#) only components useful to provide the functionalities described in the first iteration use cases are depicted. The principles according to which this prototype design of the FSDAS application has been realized are:

- Focus on retrieving Document-like Objects, i.e. text documents for which it is possible to perform text processing with the aim of indexing generation and reasoning, as opposed to processing statistical or GIS data.
- Documents stored in structured data models; i.e. DB model or XML serialized structures;
- Documents stored in file system directory structures; i.e. file system folders.

Adopting these principles gives 3 considerations:

1. We don't need to consider components treating other than DLO's.
2. Only the components dealing with ontological resources remain in the design.
3. We need a mechanism to map the above described document repositories into ontological shape to exploit the query facilities to retrieve the documents as this is the main goal of the FSDAS application.

The first two considerations lead to the component diagram as pictured in [Figure 16](#) while the third is achieved by using a mapping tool which maps structured data sources to ontology structures.

Applying this mechanism we can count on the same functionalities designed for querying the pool of the ontologies composing the fishery background knowledge to also refer to the information contained in the document repositories.

As concerns the documents stored in a directory structure, the approach considered is to instantiate existing domain ontologies with these documents by indexing them according to the same relevant terminologies found in the ontologies.

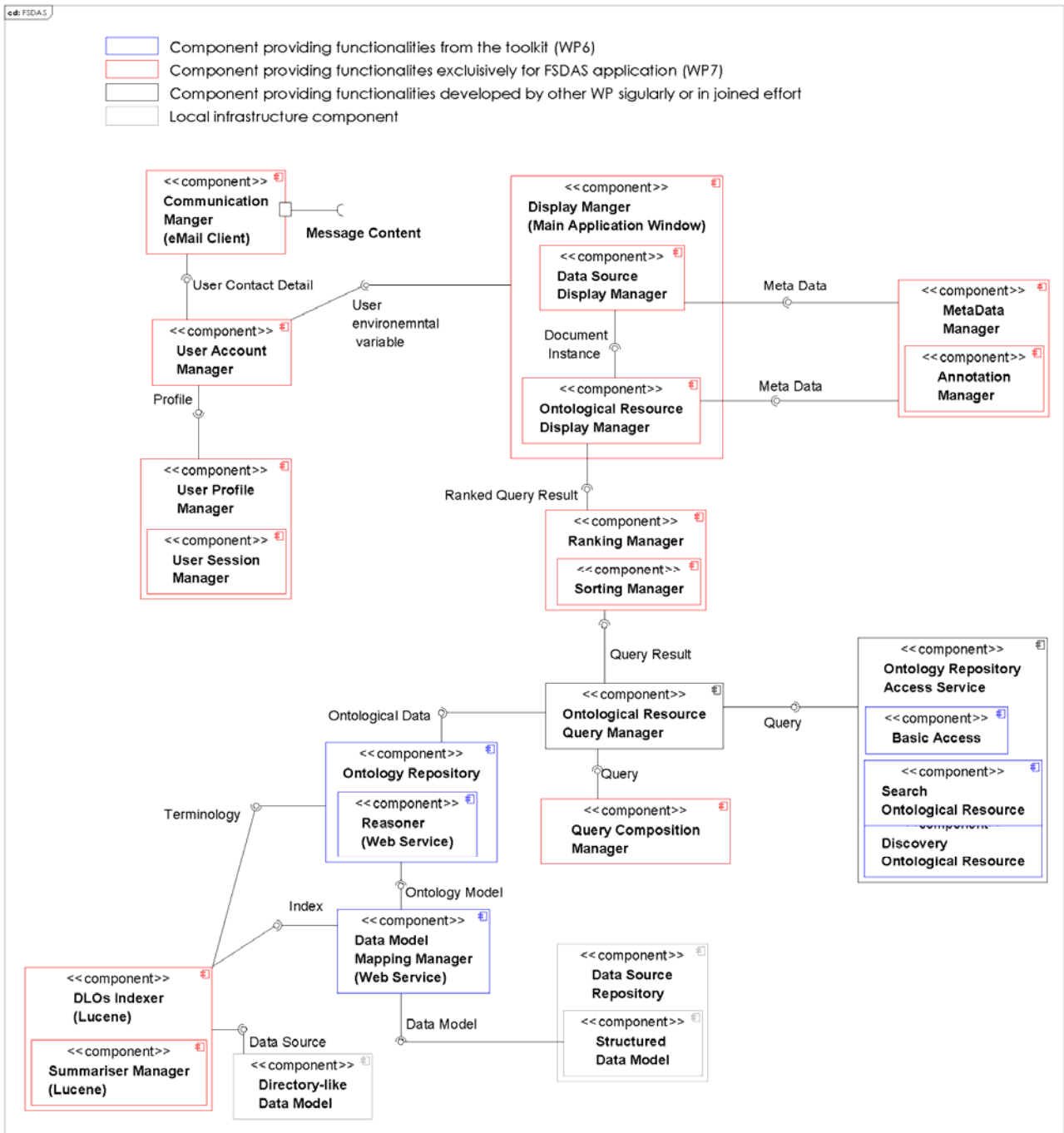


Figure 16: FSDAS application first iteration design

In this scenario, once the structured data model is mapped to an ontology and the directory-like data model is processed to populate the existing ontology, from the query point of view there will be no difference concerning heterogeneity of the underlying data models, DB's or file systems, and results will be seen as an ontology. Thus the result will be in a single ontological format.

A document instance in this vision is an individual of concept(s) in the queried ontology, and when the user decides to visualize that document it will be considered as an element of the file system to be opened.

5.3 First iteration design partition

Within this section, even if the terminology between ontological resource and data source is kept it must be remembered that from the data model point of view there is no difference, queries to retrieve fisheries division documents are addressed as ontological queries.

The components in the diagram [Figure 16](#) have been grouped into 4 main groups according to the WP which will work in providing the functionalities:

- **Components providing functionalities from the NeOn tool kit (WP6)**

These components supply the functional core of the NeOn architecture, especially from the web service accessible infrastructure layer.

- **Components providing functionalities exclusively designed for FSDAS application (WP7)**

These components supply functionalities designed ad hoc to fulfil FSDAS requirements as described in D7.1.1 and to reflect use cases description. These are the component features for which T7.6 will take care of the technical development.

- **Components providing functionalities developed by other WP singularly or in joined effort**

These components supply functionalities as a result of single or joined effort of WP's involved in development of components that can be used by T7.5. These WP's are actually developing strategies and means to achieve their final goal in the project. Part of these achievements will be used as functional features in FSDAS application.

- **Local Infrastructure components**

These components are representative of local databases, structured documents (fact sheets) or unstructured documents repository (folders of the file system).

Among these 4 groups this document describes those for which FSDAS is explicitly the target application since the other components will be developed in response to the proprietary WP goals and interact through interfaces which have yet to be explicitly defined.

5.4 Architecturally Significant Design Packages

The picture in [Figure 16](#) shows conceptual components originating from the conversion of D7.1.1 user requirements for T7.5. Each component is representative of one or a group of functionalities provided by that component. The fact that they are represented as separate is not a constraint of separation for the implementation work.

Each component exposes inwards or outwards interface(s) specifying the data type and operations needed for internal functionalities. The description of these interfaces is in the next sections.

It should be noted that this component diagram is a sub component diagram excerpted for the first FSDAS iteration from the more complete diagram shown [Figure 1](#).

As displayed in [Figure 1](#) there are a number of components which are to be developed only for FSDAS purposes and hence are out of the NeOn toolkit scope. These are the components described in more detail in this section.

As in [Chapter 2](#) this section lists the component sets related with the first iteration design describing them in greater technical detail. The exposure approach analyzes:

- Visual Components;
- User management Components;
- Integration Support Components;
- Query Components;
- Ranking Components;
- Indexing Components.

For each described component a list is given of:

- Associated Use Cases;
- Available operations;
- Inward Interface(s);
- Outward Interface(s).

In the next section technical descriptions of the interfaces are also provided.

5.4.1 Visual Components

This category groups together three components whose aim is to display resources to the user and to give an interaction point for keyboard input and for all those features that need graphic elements.

5.4.1.1 Display Manager

This is the highest-level graphic component that behaves as a container in which several graphical access points to functionalities coexist: Ontological Resource Display Manager, Data Source Display Manager plus the graphic end point for all the functionalities in the application that need a graphic element or input from the keyboard. This can be seen as the main window of the FSDAS application presenting sections of other GUI parts.

Use Cases associated to this component for the first iteration:

Change Language of the Interface;

Operations:

Graphic User Interface

- `DisplayManager.showLoginPanel();`
- `DisplayManager.showRegistrationPanel();`
- `DisplayManager.showUserAccountPanel();`
- `DisplayManager.showUserSessionPanel();`
- `DisplayManager.showQueryPanel();`
- `DisplayManager.showOntologicalDisplayManager();`
- `DisplayManager.showDataSourceDisplayManager ();`

Inwards Interfaces:

N/A

Outwards Interfaces:

User Environment Variables: complex of Environment information about the user: account details, profile details, session details.

5.4.1.2 Ontological Resource Display Manager

This component displays Ontological resources. Ontological resources can be entire ontology (ies) gathered by the user from the repository, a part of the ontology model, or can be a single ontology element (Class, Property, Individual) given the focus by navigating or performing queries.

Use Cases associated to this component for the first iteration:

Browse Taxonomy;

Change Language of the Ontological Resource Shown;

View Ontological Resource Annotation;

Operations:

Visualize ontology (ies) for navigation

- `OntologicalDisplayManager.showOntology(ontologyModel);`
- `OntologicalDisplayManager.showOntologyMetadata(ontologyModel);`

Visualize ontological resource

- `OntologicalDisplayManager.showOntologicalResourceMetadata(resource);`
- `OntologicalDisplayManager.showListSuperClass (classResource);`
- `OntologicalDisplayManager.showListSubClass (classResource);`
- `OntologicalDisplayManager.showListSuperProperty(propertyResource);`
- `OntologicalDisplayManager.showListSubProperty(propertyResource);`
- `OntologicalDisplayManager.showPropertyRange(propertyResource);`
- `OntologicalDisplayManager.showPropertyDomain(propertyResource);`
- `OntologicalDisplayManager.showIndividualClass(individualResource);`
- `OntologicalDisplayManager.showSameAsIndividual(individualResource);`

Visualize query result

- `OntologicalDisplayManager.showQueryResult(rankedResultSet);`

Inwards Interfaces:

Metadata: Information generated about the ontological or data resource loaded in the application environment. This is the equivalent of annotation as named in previous chapters.

Ranked Query Result Set: set of ontological resources returned from a user query execution after the ranking mechanism has been performed.

Ontology Model: one or more ontologies gathered from the repository and loaded in the application environment.

Outwards Interfaces:

Document Instance: an instance of an ontology concept representing a data source element.

5.4.1.3 Data Source Display Manager

This component displays document instances gathered in response to navigation of the focused ontology or by performing queries. Document instances are those text documents, web pages, news items considered an instance of the ontological concept selected by the user.

Use Cases associated to this component for the first iteration:

Visualize Data Source related to individual;

View Data Source Annotation;

Visually compare data by Reporter;

Operations:

Visualize the object document.

- DataSourceDisplayManager.openDocument(documentInstance);

Visualize metadata if available about the document object.

- DataSourceDisplayManager.showDocumentMetadata(documentInstance);

Visualize more than a document at a time to visually compare them.

Inwards Interfaces:

Metadata: Information generated about the ontological or data resource loaded in the application environment. This is the equivalent of annotation as named in previous chapters.

Document Instance: an instance of an ontological concept representing a data source element.

Outwards Interfaces:

N/A

5.4.2 User Components

This category groups together three components who collaborate with the user in the operations concerning user account management and application environment customization.

5.4.2.1 User Account Manager

This component fulfils the need to manage a single user registered to the application to gain access to the resources. A user creates an account by filling in common personal details, at the same time a general profile is generated that can be customized at a later stage.

Use Cases associated to this component for the first iteration:

Login;

Logout;

Register;

Modify User Account;

Operations:

User registers an account:

- `UserAccountManager.registerUser(userInput);`

User enter the application

- `UserAccountManager.login(userID);`
- `UserAccountManager.logout(userID);`

User modifies details

- `UserAccountManager.modifyAccount(userAccountDetails);`
- `UserAccountManager.modifyProfile(userProfileDetails);`

Inwards Interfaces:

N/A

Outwards Interfaces:

User Contact Detail: User contact as declared in the account detail.

Profile: User profile detail as declared in the User Environment Variable.

5.4.2.2 User Profile Manager

This component manages a single user in terms of environment adjustment to his usual activities in the application. User profile is built by collecting general information from the user during the registration process and then by tracking his actions and saved preferences.

Use Cases associated to this component for the first iteration:

Modify Profile

Operations:

User modify details in the profile

- `UserProfileManager.modifyProfile(userInput);`

Inwards Interfaces:

Profile: User profile detail as declared in the User Environment Variable.

Outwards Interfaces:

N/A

Session Manager

This component manages a session the user either wants to save or is automatically saved. A session is a set of information such as last open documents, ontologies, workspace layout, preferences and settings.

Use Cases associated to this component for the first iteration:

Add to Favourites;

Save Session;

Operations:

User can save session

- `SessionManager.saveWorkSpace();`
- `SessionManager.saveOpenOntologiesList();`
- `SessionManager.savePreferences();`
- `SessionManager.saveSettings();`

Inwards Interfaces:

N/A

Outwards Interfaces:

N/A

5.4.3 Integration Support Components

This category groups together three components whose aim is to supply the user with the possibility to manipulate the resources loaded in the application environment. Manipulating a resource includes annotation, tagging and rating, as well as communicating and sharing with other FSDAS user.

5.4.3.1 Metadata Manager

This component covers the functionalities of retrieving all annotation properties declared over the focused resource and passing them to the graphic interface component (Display Manager) to be visualized.

Operations:

Retrieve and aggregate metadata about the focused resource

- List<Metadata> : MetadataAggregator.listMetaData(resource);

Inwards Interfaces:

N/A

Outwards Interfaces:

Metadata: Information generated about the ontological or data resource loaded in the application environment. This is the equivalent of annotation as named in previous chapters.

5.4.3.2 Annotation Manager

This component realizes annotation functionality over the resources. Annotation is a general term that includes forms of natural language comment, tags, ratings or text abstracts. Annotations are created either by the user and stored in the format of RDF triples in a local file loaded at login time, e.g. comments, ratings, tags or they are created automatically as part of text processing, e.g. text abstract, tags. The RDF triple format is well-aligned with NeOn as a format for content exchange and presentation.

Use Cases associated to this component for the first iteration:

Annotate Retrieved Document with comments;

Annotate Retrieved Document with comments on quality;

Index Enrichment against Ontology (ies) domain model;

Operations:

Annotate Retrieved Document;

- AnnotationManager.setAnnotation(userID, resourceURI, annotationProperty, comment);
- AnnotationManager.setAnnotation(userID, resourceURI, annotationProperty, rate);
- AnnotationManager.setAnnotation(userID, resourceURI, annotationProperty, tag);
- AnnotationManager.setAnnotation(userID, resourceURI, annotationProperty, abstract);

Inwards Interfaces:

N/A

Outwards Interfaces:

N/A

5.4.3.3 Communication Manager

This component manages communication among users registered to the application. Communication includes text messages as well as object exchange, e.g. result data sets, annotations, user queries and resources exported as RDF format.

The user will use his main mail client application for the realization of the communication functionality; hence the best form of object exchange is the attachment.

Use Cases associated to this component for the first iteration:

Email result

Propose Ontology Modification

Operations:

Send message to other registered user (with other than simple text content.)

- `CommunicationManager.sendMessage(textMessage, attachment, List<userContact>);`

Propose a modification of ontological resource (particular communication instantiation between the user and the ontology expert)

- `CommunicationManager.sendModificationProposal(textMessage,resourceURI,List<expertContact>);`

Inwards Interfaces:

Message Content: Content of the message.

User Contact Detail: Users contact as declared in the account detail.

Outwards Interfaces:

N/A

5.4.4 Query Components

This category includes 1 component whose aim is to support the user in query composition.

5.4.4.1 Query Composition Manager

This component manages the way a user can compose a query which can be the result of either text editing, drag 'n drop action of resources or natural language description.

The query composed will be a meta-query containing the query itself and added information from the user about preferences, where to search and in what format. Based on this the query is addressed to the ontological or data source query manager.

Although more than a single way to compose a query is envisaged, it is expected that for the first iteration the text editing technique will be implemented.

Use Cases associated to this component for the first iteration:

Query Composition

Operations:

Provide input functionality for composing a user query

- QueryCompositionManager.KeywordQuery(List<Keyword>);
- QueryCompositionManager.sprqlQuery(Subject, Predicate, Object);

Inwards Interfaces:

N/A

Outwards Interfaces:

Query: user query formatted according to supported keywords and SPARQL syntax.

5.4.5 Rank Components

This category groups together 2 components whose aim is to order results returned from a user query execution.

5.4.5.1 Ranking Manager

This component supervises the ranking strategy for presenting results to the user. The ranking mechanism takes into account the rate of the document either as an overall value, local value or a combination of the two. Semantic distance between the query and the returned result can be used as an alternative ranking approach.

Operations:

Apply a mechanism to rank the result of a user query

- RankingManager.ontologicalDistanceRank(resultSet);
- RankingManager.profileBasedRank(resultSet);

Inwards Interfaces:

Query result: Unranked result set.

Outwards Interfaces:

Ranked Query result: Ranked result set.

5.4.5.2 Sorting Manager

This component manages the rearrangement of a result set according to any aspect but the one considered for ranking. The aspect envisaged for this process can be timestamp, alphabetical order, concept clustering or role in the RDF triple (Subject, Object) when SPARQL is used to formalize the query.

Operations:

Arrange the ranked result set according to the aspect upon which the user decides to focus

- SortingManager.sortAlphabetically(resultSet);
- SortingManager.sortByTime(resultSet);
- SortingManager.sortByConcept(resultSet);
- SortingManager.sortBySubject(resultSet);
- SortingManager.sortByObject(resultSet);

Inwards Interfaces:

N/A

Outwards Interfaces:

N/A

5.4.6 Indexing Components

This category groups together 2 components whose aim is to create the index for text documents stored in a file system directory structure according to relevant terminologies of the domain fishery ontologies. The purpose of the indexes will be to later instantiate the correspondent ontologies with the indexed documents.

5.4.6.1 DLOs indexer

This component provides the mechanism according to which given relevant domain ontology (ies) and an unstructured container of documents (system folders), an index is created taking the ontology terminology as the only source of allowable indexing terms. In so doing we assure that the process creates indexes aligned with the domain ontology (ies) covering the document subjects.

An index entry is an RDF triple binding the document URI to the ontological concept URI. We thus achieve a direct connection between ontologies covering the document content, a set of tagging terms related within the ontologies and an easy way to relate concepts of different ontologies through the documents.

More generally, producing RDF triples is highly compatible with all the techniques and technologies within NeON.

The realization of the mechanism can be done by using tools such as Sesame or Lucene.

Operations:

Produces index terms according to selected terminology (ies) that are part of the fisheries background knowledge

- `DLOsIndexer.indexDocument(List<Terminology>, dataSource);`

Inwards Interfaces:

Terminology: ontology (ies) set of named concepts relevant to the documents according to which the index terms will be generated.

Data source: document contained in the directory structure.

Outwards Interfaces:

Index: representation of document corpora through terms of domain fishery terminology.

5.4.6.2 Summarizer Manager

This component produces an abstract of the text documents contextually with the index generation. The realization of this feature can be done by exploiting the indexing functionality of tools such as Lucene.

Operations:

Produces text abstract of the processed document

- SummarizerManager.produceAbstract(dataSource);

Inwards Interfaces:

- N/A

Outwards Interfaces:

- N/A

5.5 Interfaces Description

5.5.1 User Environment Variables Interface

User Environment Variables Description:

- Complex of Environment information about the user: account details, profile details, session details.

User Environment Variables Attributes:

- Account_Details;
- Profile_Details;
- Session_Details;

User Environment Variables Operations:

- void : setAccountDetails(accountDetails);
- void : setProfileDetails(profileDetails);
- void: setSessionDetails(sessionDetails);
- Account_Details : getAccountDetails();
- Profile_Details : getProfileDetails();
- Session_Details : getSessionDetails();

User Environment Variables Exceptions:

- DetailsNotFoundException() : thrown if details are not found.

5.5.2 Metadata Interface

Metadata Description:

- Metadata: Information generated about the ontological or data resource loaded in the application environment. This is the equivalent of annotation as named in previous chapters.

Metadata Attributes:

- Authority;
- Resource_URI;
- Annotation_Property;
- Meta_Data_Value;

Metadata Operations:

- void : setMetaDataAuthor(authorID);
- void: setMetadataSubject(resourceURI);
- void: setMetaDataProperty(property);
- void: setMetaDataObject(value);
- Author_ID : getMetaDataAuthor();
- Resource_URI : getMetadataSubject();
- Annotation_Property : getMetaDataProperty();
- Meta_Data_Value : getMetaDataObject();

Metadata Exceptions:

- InvalidAuthorException() : thrown when the user identifier is not declared or not valid.

5.5.3 Query Interface

Query Result Description:

- Result set of a user query before the ranking mechanism is applied

Query Result Attributes:

- User_ID;

- List<Keywords>;
- SPRQL_Triple;
- Query_Syntax;

Ranked Query Result Set Operations:

- void : setUserID(user_ID);
- void : setKeywords(List<Keywords>);
- void : setSPRQLtriple(SPRQL_Triple);
- void : setQuerySyntax (Query_Syntax);
- UserID : getUserID();
- List<Keywords>: listKeywords();
- SPRQL_Triple : getSPRQLtriple();
- Query_Syntax : getQuerySyntax ();

Query Result Set Exceptions:

- InvalidAuthorException() : thrown when the user identifier is not declared or not valid.

5.5.4 Query Result Interface

Query Result Description:

- Result set of a user query before the ranking mechanism is applied

Query Result Attributes:

- UserID;
- User_Query;
- List<Ontology>;
- List<Class>;
- List<Property>;
- List<Individual>;

Ranked Query Result Set Operations:

- void : setUserID(userID);
- void : setUserQuery();
- void : setOntologies(List<Ontology>);

- void : setClasses(List<Class>);
- void : setProperties(List<Property>);
- void : setIndividuals(List<Individual>);
- UserID : getUserID();
- User_Query : getUserQuery();
- List<Ontology>: listOntologies();
- List<Class>: listClasses();
- List<Property>: listProperties();
- List<Individual>: listIndividuals();

Query Result Set Exceptions:

- InvalidAuthorException() : thrown when the user identifier is not declared or not valid.

5.5.5 Ranked Query Result Interface

Ranked Query Result Description:

- Set of ontological resources returned from a user query execution after the ranking mechanism has been performed

Ranked Query Result Attributes:

- UserID;
- User_Query;
- List<Ontology>;
- List<Class>;
- List<Property>;
- List<Individual>;
- Resource_Rank;

Ranked Query Result Set Operations:

- void : setUserID(userID);
- void :setUserQuery();
- void :setOntologies(List<Ontology>);
- void : listClasses(List<Class>);
- void :listProperties(List<Property>);

- void : listIndividuals(List<Individual>);
- UserID : getUserID();
- User_Query :getUserQuery();
- List<Ontology>:listOntologies();
- List<Class>: listClasses();
- List<Property> :listProperties();
- List<Individual. : listIndividuals();
- Resource_Rank : getResourceRank(resource)

Ranked Query Result Set Exceptions:

- InvalidAuthorException() : thrown when the user identifier is not declared or not valid.

5.5.6 Ontology Model Interface

Ontology Model Description:

- Ontology gathered from the repository and loaded in the application environment.

Ontology Model Attributes:

- List<Metadata>;
- Repository_Provenance;

Ontology Model Operations:

- void : setMetaData(List<MetaData>);
- void : setProvenance(Repository_Provenance);
- List<MetaData> : listMetaData();
- Repository_Provenance : getProvenance();

Ontology Model Exceptions:

- NullProvenanceException() : thrown when the provenance is not retrievable.

5.5.7 Document Instance Interface

Document Instance Description:

- An instance of an ontological concept representing a data source element.

Document Instance Attributes:

- List<MetaData>;
- Repository_Provenance;

Document Instance Operations:

- void : setMetaData(List<MetaData>);
- void : setProvenance(Repository_Provenance);
- List<MetaData> : listMetaData();
- Repository_Provenance : getProvenance();

Document Instance Exceptions:

- NullProvenanceException() : thrown when the provenance is not retrievable.

5.5.8 Message Content Interface

Message Content Description:

- Content of the message

Message Contact Attributes:

- Message_body;
- Attachment;
- List<User_Receipient_Contact>;

Message Content Operations:

- void : setMessageBody (messageBody);
- void : setAttachment(attachment);
- void : setReceipientsContacts(List<User_Receipient_Contact>);
- Message_body : getMessageBody ();
- Attachment : getAttachment();

- List<User_Contact> : listContacts();

Message Content Exception:

- NullReceipientException() : thrown if no recipient has been selected.

5.5.9 User Contact Detail Interface

User Contact Detail Description:

- Users contact as declared in the account detail.

User Contact Detail Attributes:

- Email;
- UserID;

User Contact Detail Operations:

- void : setUserContact(email);
- void : setUserID();
- Email : getUserContact();
- UserID : getUserID();

5.5.10 Terminology

Terminology Description:

- Ontology(ies) set of named concepts relevant to the documents according to which the index terms will be generated.

Terminology Attributes:

- Ontology_Provenance;
- List<Concepts Names>;

Terminology Operations:

- void : setOntologyProvenance(Ontology_Provenance);
- void : setConceptNames(List<Concepts Names>);
- Ontology_Provenance : getOntologyProvenance();
- List<Concepts Names> : ListConceptNames();

Terminology Exceptions:

- NullProvenanceExceptions();

5.5.11 Data Source Interface

Data source Description:

- Document contained in the directory structure.

Data Source Attributes:

- File;
- Data_Source_Name;
- Data_Source_Path;

Data Source Operations:

- void : setFile(documentIFile);
- void : setDataSourceName(Data_Source_Name);
- void: setDataSourcePath(Data_Source_Path);
- File : getFile();
- Data_Source_Name : getDataSourceName();
- Data_Source_Path : getDataSourcePath();

Data Source Exceptions:

- FileNotFoundException() : thrown when the Data Source is not retrievable.

5.5.12 Index Interface

Index Description:

- Generated terms representing the document according to its content expressed in terms of the selected ontology(ies).

Index Attributes:

- Data_Source_Path;
- List<Terminology>;
- List<Index_Terms>;

Index Operations:

- void : setDataSourcePath(Data_Source_Path);
- void : setTerminolgy(List<Terminology>);
- void : setIndexTerms(List<Index_Terms>);
- Data_Source_Path : getDataSourcePath();
- List<Terminology> : listTerminolgy();
- List<Index_Terms> : listIndexTerms();

5.5.13 Profile

Profile Description:

- User profile detail as declared in the User Environment Variable.

Profile Attributes:

- User_ID;
- Expertise;
- List<LastUsedResources>;
- Session;

Profile Operations:

- void : setUserID(userID);
- void : setExprtise(expertise);

- void : setLastUsedResources(List<LastUsedResources>);
- void : setSession(session);
- User_ID : getUserID();
- Expertise : getExprtise();
- List<LastUsedResources> : listLastUsedResources();
- Session : getSession();

Profile Exception:

- NullUserIDException: thrown when the user identifier is not declared or not valid.

5.6 First iteration design concise aspects

Adopts Structured Data Model Mapper:

- Structured Data Model and Directory-like repository of data sources are mapped to ontologies.

Indexes Documents:

- Directory-like structures are navigated and contained documents are indexed from the set of terms composing the taxonomies of identified ontologies covering the domain to which the documents are related. Uses Lucene or other open source tool to this purpose.

Everything is an ontological resource:

- Once Structured Data Models are mapped to ontologies and documents in the directories have populated the indexing ontologies, the system only deals with ontological resources as far as querying features are concerned.

Uses RDF statements:

- Since all the resources in the application can be referenced by URI, every time the user adds an annotation, tag, abstract or rating, a new statement about a URI resource is added to a local RDF document loaded in the system at login time.

Networked Oriented:

- All users will have their local files of statements kept separate from official fisheries ontologies but interlinked by semantic mechanisms.

Web 2.0 oriented

- Producing information in interchangeable format, e.g. RDF/XML, to be compliant with Web 2.0 spread technologies such as mashing up different context results to obtain an optimized answer to complex user queries. Using RDF syntax provides a uniform means for producing new content and at the same time allows storing it independently from the location of the axiomatized resources. It also means that exchanging objects among users

can be realized by sharing/passing the same set of axioms about targeted resources; RDF is also a good fit for the visualization mechanisms envisaged for FSDAS, the application in fact embeds a mechanism to process RDF data and applies different visualization paradigms.

6. Process View

The activities a user can perform within FSDAS are centred on 4 major contexts. Each of these include a number of use cases restricted to the first iteration design, as they are described in D7.1.1:

6.1 Account context

This is closely related with all the use cases involving operations to manage user accounts entering, exiting and using the application:

- Login
- Logout
- Register
- Modify profile
- Modify user account

6.2 Search context

This is closely related with all the use cases involving search operations without regard to resource type (ontological or data source):

- Search ontological resource in ontology
- Search for related ontological resources
- Query for data related to individual

6.3 Display context

This is closely related with all the use cases involving GUI capabilities of the application needed to give an interaction end point to the user and a way to visualize retrieved resources:

- Browse taxonomy
- Change language of the interface
- Change language of the ontological resource
- Visualize data source related to individual
- View ontological resource annotation
- View data source annotation
- Compare data by reporter

6.4 Interaction context

This is closely related with all the use cases involving operations the user performs as complementary activity to the main purpose of the FSDAS application:

- Email results
- Add to favourites
- Save session
- Annotate retrieved document with comments
- Annotate retrieved document with comments on quality
- Index enrichment against ontology(ies) domain model

Because each use case is associated with a process flow that involves many components of the design, it is useful to have a picture of how the collaboration among multiple objects finalizes in a use case process.

In the following pages there are pictures of collaboration diagrams reflecting the contexts mentioned above.

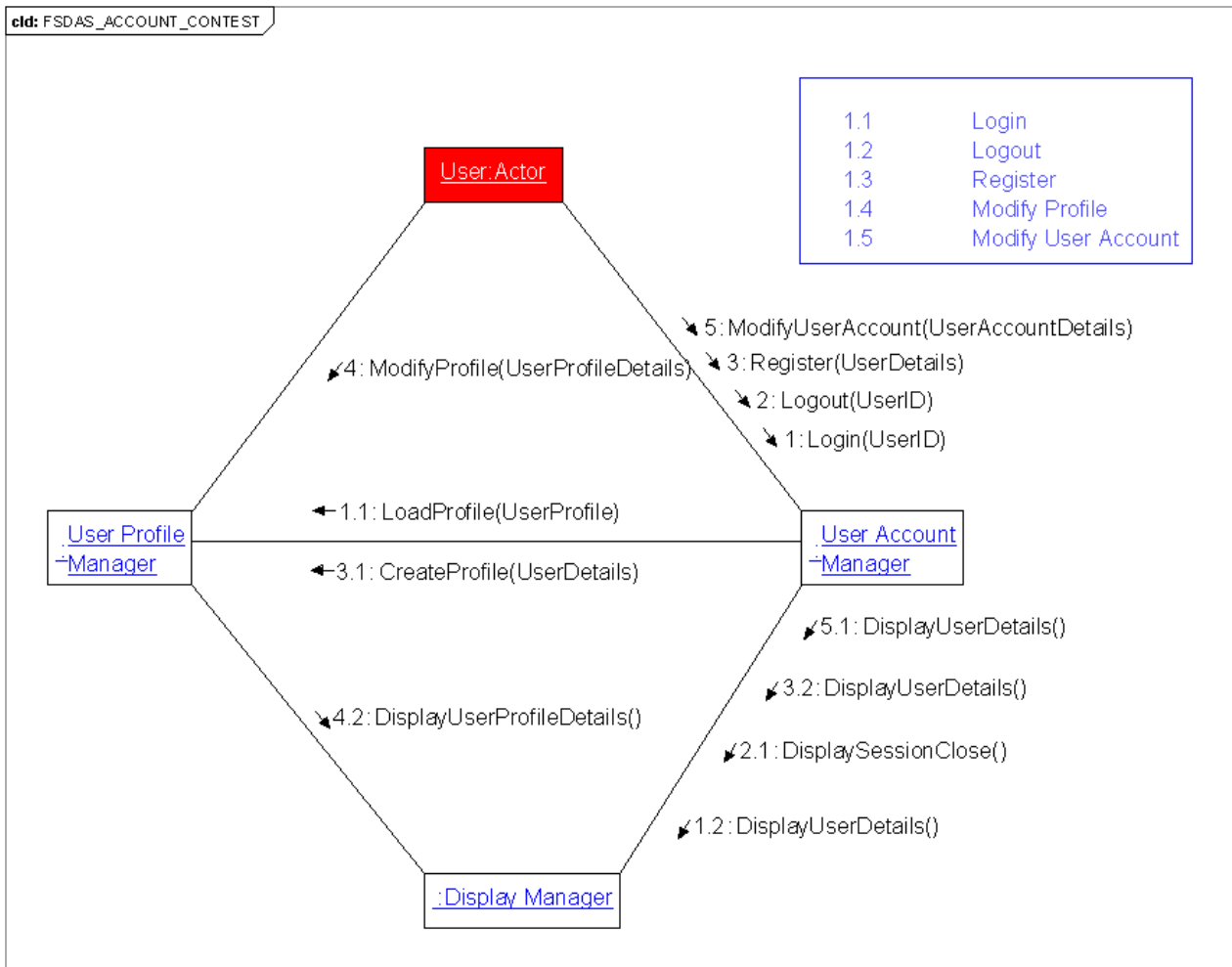


Figure 17 - Collaboration diagram of account context components

Only first iteration use cases are considered

The above diagram shows how the user can manage his account and what processes this involves.

When Login or Registration processes are triggered, User Account Manager internally asks for the user profile or creates one if it doesn't exist. Granting access, loading or creating the profile creates a user account object within the session that is finally displayed through the Display Manager.

When Logout or Modify User Account processes are triggered, they involve the User Account Manager straight away which performs logout action or update action and finally shows a confirmation message through the Display Manager.

When Modify User Profile process is triggered, User Profile performs the update.

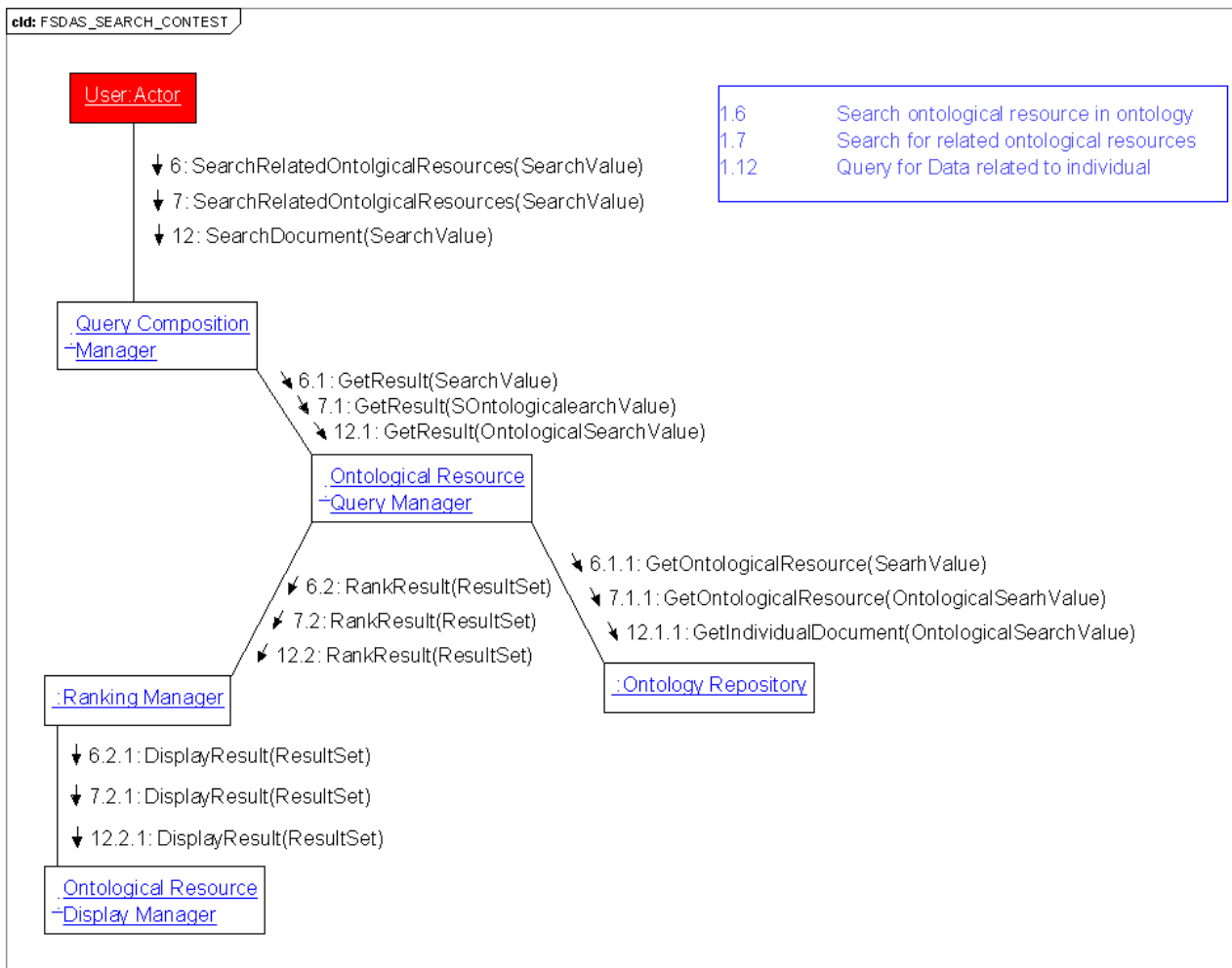


Figure 18 - Collaboration diagram of search context components

Only first iteration use cases are considered

The above diagram shows how the user can achieve the FSDAS application main goal of document retrieval and what this involves.

When any search process is triggered, the Query composition Manager is involved for query building. The Query Manager then executes the retrieval process from the Ontology Repository and transfers the result to the Ranking Manager which will make them available for display through the display manager specialized for visualization of ontological resources.

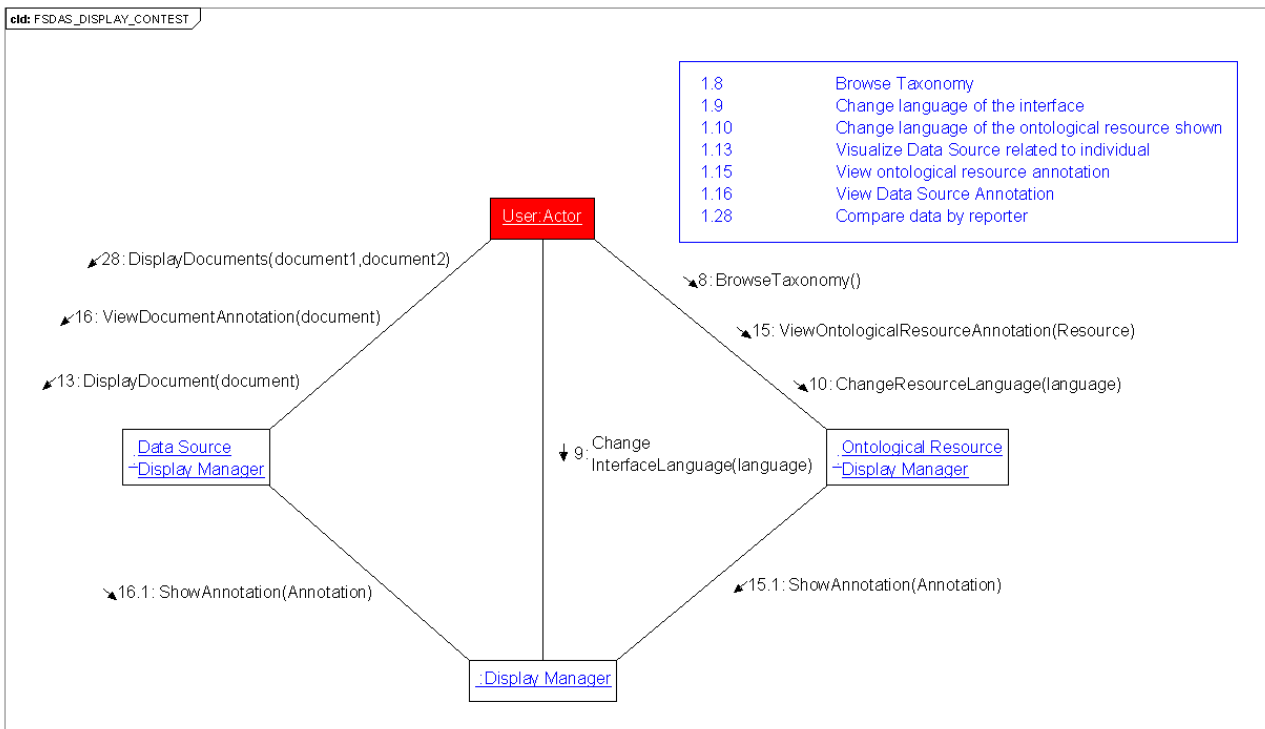


Figure 19 - Collaboration diagram of display context components

Only first iteration use cases are considered

The above diagram shows how the user can visualize generated content in FSDAS application and what this involves.

When Browse Taxonomy, View Ontological Resource Annotation or Change Resource Language processes are triggered, Ontological Resource Display Manager is component invoked; it is specialized for visualizing ontology or ontological elements as classes, properties or individuals.

Showing annotations for the focused resource also involves the display manager which is the general purpose visualizer.

When Display Document, Compare Data by Reporter or View Data Source Annotation processes are triggered, Data Source Display manager is the component invoked.

Showing annotations for the focused document instance also involves the Display Manager which is the general purpose visualizer.

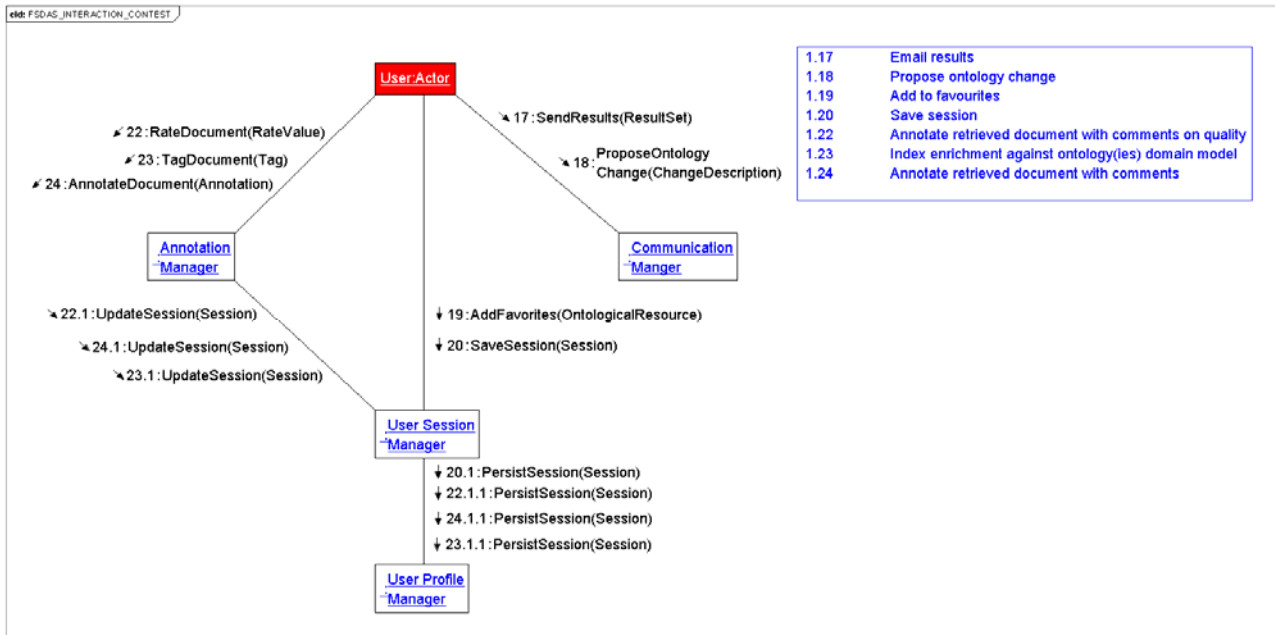


Figure 20 - Collaboration diagram of interaction context components

Only first iteration use cases are considered

The above diagram shows how the user can perform complementary operations to the main purpose of the application and what this involves.

When Email Result or Propose Ontology Change processes are triggered, the only component in charge of executing both these activities is the Communication Manager which embeds capabilities of message exchange.

When Annotate Retrieved document with comments (comments on quality), or Index Enrichment against Ontology Domain Model processes are triggered, the Annotation Manager component collects the data and updates the session with the newly generated content, persisting this within the User Profile.

When Save Session or Add to Favourites processes are triggered, User Session Manager will update the session and then persist within the User Profile.

7. Deployment View

7.1 Overview

FSDAS application is meant to be used by any user involved in working with Fishery department resources, not necessarily within any FAO site.

In [Figure 21](#) is depicted the connection of a single user computer with all the parts reached by the application once it has been downloaded from the Fishery Department portal. It is restricted to the first iteration design.

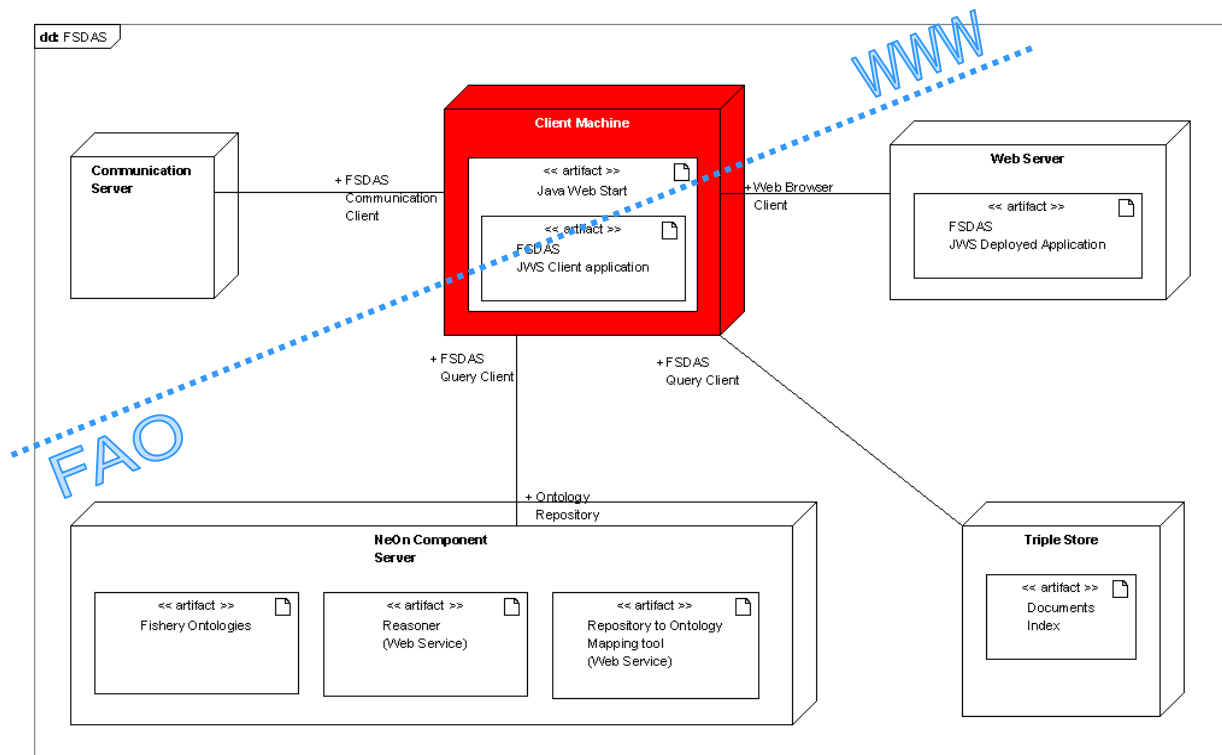


Figure 21 - Deployment view of the FSDAS application main activities

In the above view is depicted the separation between resources belonging to the FAO network domain and to those reachable over the World Wide Web. Typically a user computer is assumed to be anywhere in the world together with its mail client communication software. The server containing the downloadable Java Web Start deployment package of the application is internal to the FAO domain, as are those infrastructure and engineering components capable of being run in server mode that are proper to the NeOn toolkit.

7.2 Configuration description

Considering physical nodes in [Figure 21](#), 5 main processes representative of FSDAS capabilities can be described in terms of the hardware configuration involved.

7.2.1 User downloads FSDAS application from Fishery portal

Physical nodes:

- User computer: any computer internal or external to FAO network domain.
- Web Server: this is the computer where the Fishery portal is hosted.

Process:

- User computer runs a web browser application and connects to Fishery portal. User clicks on link to download FSDAS application distributed through JWS technology. If JWS is not installed the user is pointed to download it then the FSDAS download starts.

Communication:

- Communication is established via HTTP and is transported over TCP/IP

7.2.2 User runs FSDAS application

Physical nodes:

- User computer: any computer internal or external to FAO network domain.

Process:

- User computer has installed the JWS, user runs FSDAS application.

Communication:

N/A

7.2.3 User Loads Ontologies

Physical nodes:

- User computer: any computer internal or external to FAO network domain.
- NeOn Component Server: this is the computer where NeOn infrastructure and engineering components are deployed

Process:

- User computer runs FSDAS application and loads ontologies from ontology repository hosted on NeOn Components Server, internal to FAO network domain.

Communication:

- Communication is transported over TCP/IP

7.2.4 User performs a query:**Physical nodes:**

- User computer: any computer internal or external to FAO network domain.
- NeOn Component Server: this is the computer where NeOn infrastructure and engineering components are deployed.
- Web Server: this is the computer hosting an RDF triple store of the indexed documents.

Process:

- User computer runs FSDAS application. User performs a query which returns result from ontology repository in NeOn Components Server and from web service invocation of the RDF triple store. Reasoning web service can be also invoked for reasoning support.

Communication:

- Communication for web services is established via SOAP and is transported over TCP/IP.

7.2.5 User communicates with other users**Physical nodes:**

- User computer: any computer internal or external to FAO network domain.
- Communication Server: computer providing communication gateway capabilities.

Process:

- User machine runs FSDAS application; user selects to communicate with other user registered to the application and prepares a message and/or object to send.

Communication:

For the first iteration design communication is established via the user's mail client that in general is already configured to communicate with a mail server.

8. Implementation View

8.1 Overview

This section describes how the components envisaged for the FSDAS design fit in to a layered structure reflecting the NeOn architecture layers.

In [Figure 22](#) are depicted the design components divided by architecture such that one can see the support that the NeOn toolkit can provide to the FSDAS application in its final design version.

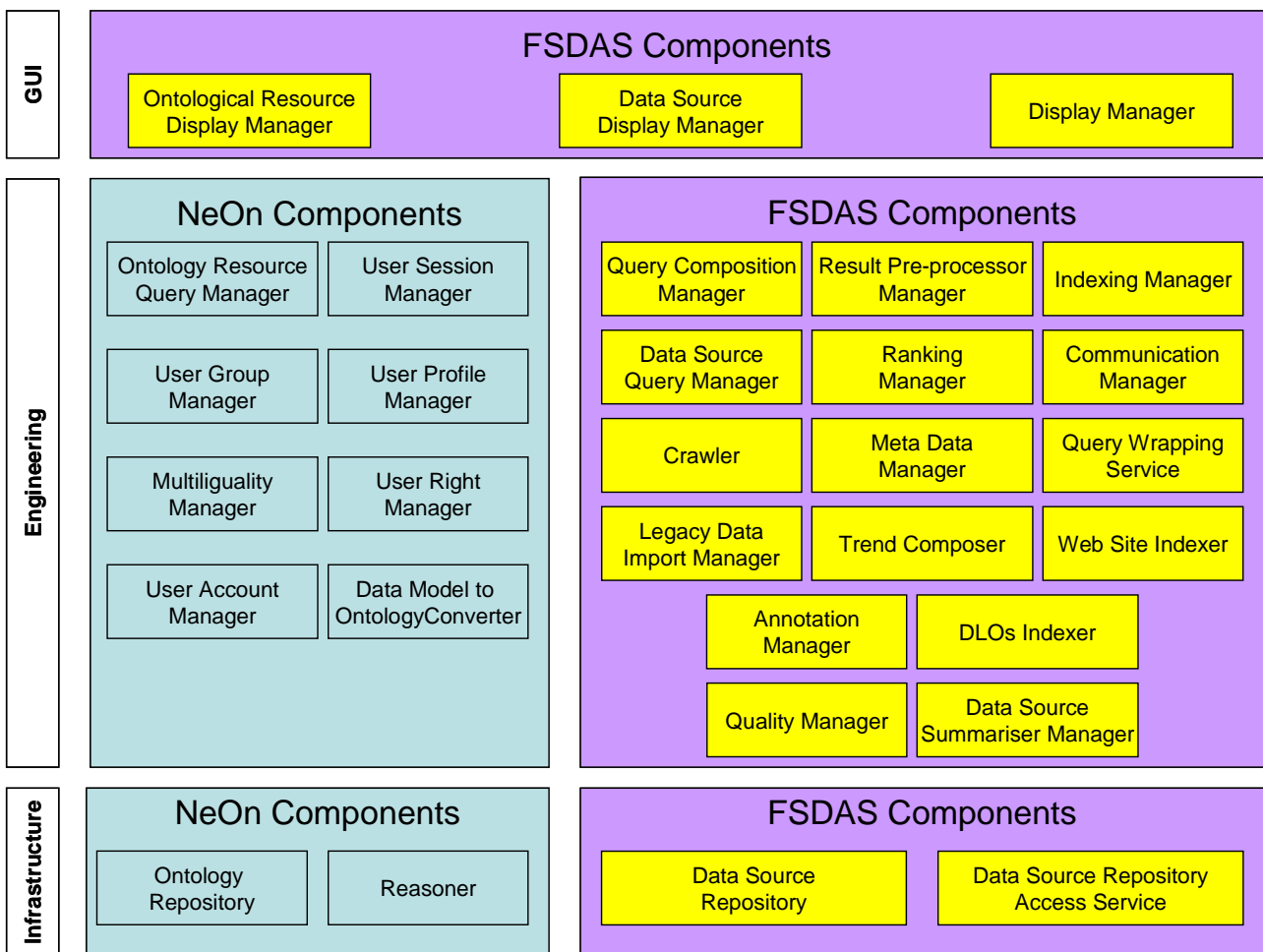


Figure 22 - Layered vision of FSDAS application design

The layers are based on the proposed layering definitions promoted by work package six and found in deliverable [D6.2.1](#).

A component is grouped in the NeOn set if its functionalities are planned to be covered by the NeOn toolkit, otherwise a component is in the FSDAS set if its functionalities are ad-hoc for the application.

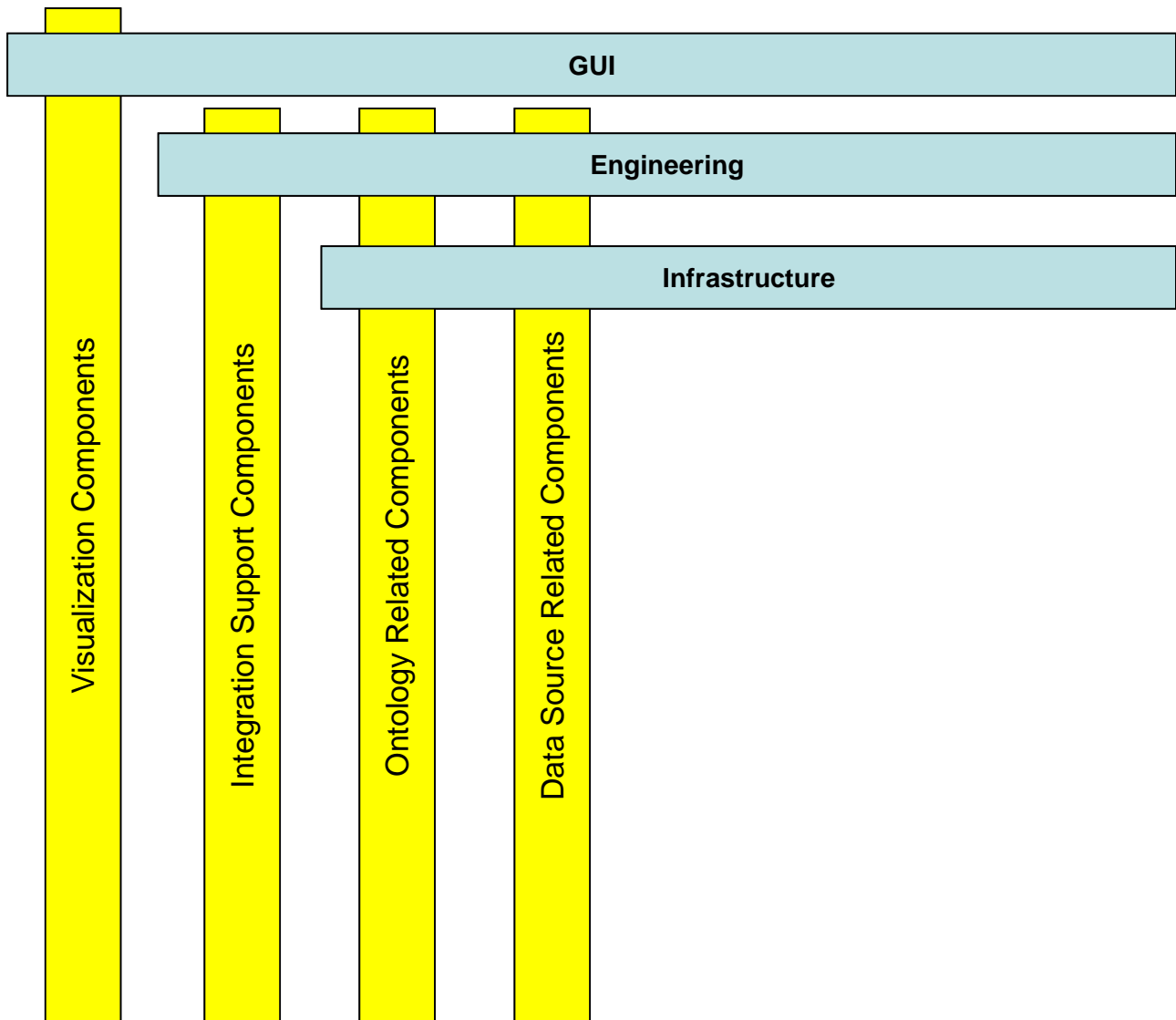


Figure 23 - FSDAS design conceptual partition against NeOn architecture layered vision

A complementary view of FSDAS design comes from considering the conceptual partition introduced in [Chapter 2](#) compared against the layered vision of the NeOn architecture.

What is highlighted by this cross comparison is the architecture domain each partition involves. Most components are centred in the engineering layers, confirming a high dependency on functionalities developed in the toolkit. It is also natural to expect that the GUI layers provide little dependent functionality as the toolkit is mostly focused on developing infrastructure and engineering layers.

9. Data View

FSDAS will rely on the OntoMap/Semantic Miner component capabilities of the NeOn toolkit which can be run in stand-alone server mode.

The creation of ontological mappings between end user domain ontologies and database schema ontologies will be handled within the NeOn toolkit as part of ontology lifecycle management.

Data instances returned will be transparent to FSDAS and will be merely the result of a semantic query towards the reasoner.

Documentation on Semantic Miner can be found in [Annex B](#).

9.1 RDBMS systems

9.1.1 Aquatic Sciences Fisheries Abstracts (ASFA)

The Fisheries and Aquaculture department is currently awaiting the full set of ASFA abstracts from the commercial publisher in a format that will allow importing them into a database. The database will in all likelihood be Oracle 10G, as that is the FAO standard, and what is used within the Fisheries department.

The database schema will be extracted as an ontology and mapped to the ASFA thesaurus ontology (currently being produced by ASFA in OWL) and/or fisheries domain ontologies via OntoMap.

It is important to note that this database will be fire walled and any server accessing it will need to be within the firewall.

9.1.2 Electronic Information Management System (EIMS)

EIMS is a FAO-wide document repository of official publications. Metadata including abstracts is stored for each publication in a queryable RDBMS (Oracle).

The database schema will be extracted as an ontology and mapped to the AGROVOC ontology and/or fisheries domain ontologies via OntoMap.

It is important to note that this database will be fire walled and any server accessing it will need to be within the firewall.

9.1.3 FishBase

Hosted by World Fish Center in Penang, Malaysia, FishBase is an encyclopaedic database of species data. A partner with FAO on various projects, final approval is yet to be obtained on connecting directly to the WorldFish database.

The database schema will be extracted as an ontology and mapped to the aquatic species ontology via OntoMap.

9.1.4 FIGIS RDBMS fact sheets

A series of fishery fact sheets divided by domain and stored as XML in Oracle, and connected to classification schemes that are also stored in Oracle.

The database schema will be extracted as an ontology and mapped to fisheries domain schemas via OntoMap.

9.1.5 Fishery Resources Monitoring System (FIRMS)

FIRMS is a consortium of over a dozen regional fishery bodies that pools, harmonizes and publishes resource assessments. They are stored as XML in an Oracle database and connected to classification schemes that are also stored in Oracle.

The database schema will be extracted as an ontology and mapped to fisheries domain schemas via OntoMap.

9.2 Flat file systems

There are some repositories for which we there is no database access. There may be a queryable index, but the API's vary.

This document describes an RDF indexing component that will be used to index these resources using one or more fisheries ontologies to limit the indexing terms to those concepts found in the ontologies. The index is created at engineering time, and queried at run time simultaneously with the query to the reasoner. This will give a stable, centralized access with a single API.

9.2.1 FIGIS flat-file fact sheets

One branch of fact sheets is stored in directories on a server. A Lucene index is created from these fact sheets to render them queryable.

Either they will be indexed by the RDF indexing component or OntoMap can be connected to the Lucene index.

9.2.2 RFB's document repositories

The document repositories of the approximately fifty regional fishery bodies constitute an important source of information for resource assessments. Agreement can likely be reached to index at least some of these using the RDF indexing component.

9.2.3 Globefish document repository

GlobeFish represents an important, frequently updated source of commodities data which can among other things indicate trends of relative species abundance. Hosted by FAO, this repository can be indexed using the RDF indexing component.

9.3 ISIS systems

The UNESCO ISIS system is a database system in production since the 1980's. It does not have any standard ODBC/JDBC connectors. It can however produce an output in ISO2709, a standard format for exchange of bibliographic data, as well as data in XML.

9.3.1 FAOLex

FAOLex is a database of agricultural regulation which contains a large amount of fisheries and marine legislation.

Further investigation is needed. It may be simplest to output the entire database periodically to ISO2709 and import it into a standard RDBMS with JDBC connectors.

9.4 Time-series systems

This refers to systems containing numeric data as a sequence of data points, measured typically at successive times, spaced at (often uniform) time intervals. Within fisheries these sets typically regard capture and production for commercially important aquatic species, collected annually, and expressed in units of metric tonnes by country, year and species.

9.4.1 FIES commodity, capture, production and fleets databases

Fisheries and Aquaculture Information and Statistics Service maintains statistical datasets stretching back to the 1950's. Stored in an Oracle RDBMS, data are linked to reference data in a separate set of tables.

For these time-series, there is no URI that points to a stored "resource" per se, but rather a flexible dynamic tabular, chart or graph result based on a wide variety of criteria, e.g. a range of years, a set of water areas, groups of species, catch data, display type, etc.

It is not clear how a semantic query can retrieve a useful table, chart or graph of data from these databases, yet the results are vital to a knowledge base on fishery resource assessments.

The preferred option at this point is to generate at engineering time URI's related to ontological concepts that return relevant formatted statistics from the time-series databases. These URI's could be stored as instances in the RDF triple store index discussed in the previous section.

9.5 GIS systems

A geographic information system (GIS) is a system for capturing, storing, analyzing and managing data and associated attributes which are spatially referenced to the earth. For the purposes of FSDAS, it is a computer system capable of integrating, storing, editing, analyzing, sharing, and displaying geographically-referenced information.

9.5.1 Species distribution maps

Fisheries and Aquaculture Information and Statistics Service maintain species distribution maps created dynamically from an in-house GIS system known as KIMS/KIDS. Shape files are stored for land areas, water areas and species. Most shape files also have an associated XML file of

metadata about the shape file that conforms to ISO tc211. Images based on the shape files are generated dynamically on request.

It should be possible to map the KIMS/KIDS data model to a GIS ontology, thus linking the shape file metadata to it and giving the possibility for such resources to be returned with a semantic query.

9.6 Ontologies

The ontologies developed or in development for FAO are in OWL format. They represent either thesauri or domain-specific ontologies that will be mapped and loaded together on the reasoner side of FSDAS.

At least the following ontologies will be used in FSDAS, though the list will likely grow following the first iteration:

- biological species http://www.fao.org/aims/aos/fi/species_v1.0.owl
- land areas http://www.fao.org/aims/aos/fi/lands_v1.0.owl
- water areas http://www.fao.org/aims/aos/fi/fishing_areas_v1.0.owl
- fishery commodities http://www.fao.org/aims/aos/fi/commodities_v1.0.owl
- vessel types http://www.fao.org/aims/aos/fi/vessels_v1.0.owl
- gear types http://www.fao.org/aims/aos/fi/gears_v1.0.owl
- ASFA thesaurus
- AGROVOC thesaurus

10. Size and Performance

10.1 Operating Environment

Server-side must operate on an Intel/Linux platform.

Client must function on Windows/Linux/Mac OS-X operating systems.

10.2 Implementation Constraints

Reasoner, OntoMap and triple store server should be able to coexist on a single hardware platform.

Due to FAO network security constraints, the server-side of FSDAS must reside within the FAO firewall in order to be able to connect with local databases.

Memory requirements should be less than four gigabytes.

Any need for RDBMS support must use either ORACLE 10G or MySQL5.

Application must be written in Java (min ver. 1.5) to take advantage of developer knowledge at FAO.

Application must be able to work natively with UTF-8 in order to support FAO languages (Arabic, Chinese, English, French and Spanish)

Results should not take more than two seconds to display.

10.3 Assumptions and Dependencies

FSDAS depends on the NeOn toolkit reasoner and OntoMap running in server mode for reasoning capabilities and mapping to relational databases.

FSDAS depends on the creation of a data services API that can run in server mode and return pieces of the ontology model in a standard format. This API has not yet been defined. Partners involved in T7.6 need to collaborate with WP6 on this aspect of the design and implementation of FSDAS.

FSDAS depends on the creation of a set of ontologies describing the fisheries domain.

FSDAS depends on the cooperation of a group of partner information repositories from which to draw its concept instances.

All users of the system will need to be authenticated as some concept instances and/or ontologies may not be part of the public domain.

Annex A – Updated use cases from Requirements D7.1.1

UC1 Login [page 62 in D7.1.1]

ACTORS

- Fisheries scientist

DESCRIPTION

- User logs into the system

PRECONDITIONS

- Application launch

TRIGGERING EVENT(s)

- Application launch

POST CONDITIONS

- User is logged in

FLOW OF EVENTS

a. BASIC FLOW

- User inputs name and password.
- User is authenticated
- System loads default profile and user rights

b. ALTERNATIVE FLOW

- User not found, is invited to try again
- User not found, is invited to register

RELATED USE CASES

- Register

NOTES / ISSUES

- N/A

UC2 Logout [page 63 in D7.1.1]**ACTORS**

- Fisheries scientist

DESCRIPTION

- User logs out

PRECONDITIONS

- User is logged in

TRIGGERING EVENT(s)

- User selects logout

POST CONDITIONS

- User is logged out, system returns to login screen

FLOW OF EVENTS**a. BASIC FLOW**

- User selects logout
- System asks if user wants to save session
- System updates user profile
- System logs user out
- System returns to login screen

b. ALTERNATIVE FLOW**RELATED USE CASES**

- Register
- Save session

NOTES / ISSUES

- N/A

UC3 Register [page 64 in D7.1.1]

ACTORS

- Fisheries scientist

DESCRIPTION

- Register

PRECONDITIONS

- Application launch

TRIGGERING EVENT(s)

- User selection of “register” link from login page

POST CONDITIONS

- Creation of generic user
- Registration request is sent to admin for role specification
- System displays a success page to the user

FLOW OF EVENTS

a. BASIC FLOW

- User inputs: name, email (twice), desired password (twice)
- User inputs profile: user type, organization, ontologies preferred language, etc.
- User clicks “submit”
- Registration request is sent to the system administrator with details
- System displays a success page to the user

b. ALTERNATIVE FLOW: email or password double entries do not match

- System invites user to repair input

c. ALTERNATIVE FLOW: mandatory field(s) is missing

- System invites user to repair input

d. ALTERNATIVE FLOW: mandatory field(s) is missing

- System invites user to repair input

RELATED USE CASES

- Login
- Logout

NOTES / ISSUES

- Registration is sent for action by system administrator

- Defaults access rights should be “read only”
- Administrator will establish more detailed privileges

UC4 Modify Profile (medium priority) [page 66 in D7.1.1]

ACTORS

- Fisheries scientist

DESCRIPTION

- User modifies her profile

PRECONDITIONS

- User is logged in

TRIGGERING EVENT(s)

- User clicks “modify profile” from main page

POST CONDITIONS

- Profile is modified
- User is shown main page

FLOW OF EVENTS

a. BASIC FLOW

- User clicks “modify profile” from main page
- System loads profile page
- User modifies profile details
- User validates changes
- System returns to main page

b. ALTERNATIVE FLOW

- User abandons action by clicking “cancel”
- System returns to main page, no changes are made

c. ALTERNATIVE FLOW

- User makes some kind of invalid input
- System invites user to try again

RELATED USE CASES

- Save session

NOTES / ISSUES

- The user profile contains information about the user preferred options including list of ontologies and collections of documents and statistics commonly used for search, browsing and reading.

UC5 Modify User Account [Formerly not included in D7.1.1]**ACTORS**

- Fisheries scientist

DESCRIPTION

- User modifies his account

PRECONDITIONS

- User is logged in

TRIGGERING EVENT(s)

- User clicks “modify account” from main page

POST CONDITIONS

- Account is modified
- Registration request is sent to admin for role specification
- User is shown main page

FLOW OF EVENTS**a. BASIC FLOW**

- User clicks “modify account” from main page
- System loads account page
- User modifies account details
- User validates changes
- Modify request is sent to the system administrator with details
- System returns to main page

b. ALTERNATIVE FLOW

- User abandons action by clicking “cancel”
- System returns to main page, no changes are made

c. ALTERNATIVE FLOW

- User makes some kind of invalid input
- System invites user to try again

RELATED USE CASES

- Register

NOTES / ISSUES

- The user account contains information about the user (name, organization, email, editing rights, ...)

UC6 Search ontological resource in ontology [Update of UC 5.4.6 in D7.1.1]**ACTORS**

- Fisheries scientist

DESCRIPTION

- Search ontological resources within an ontology

PRECONDITIONS

- At least one ontology is loaded
- User is logged in

TRIGGERING EVENT(s)

- User enters text in search box and clicks “search ontological resource”

POST CONDITIONS

- System displays retrieved ontological resources

FLOW OF EVENTS**a. BASIC FLOW**

- User enters text in search box
- User clicks “search ontological resources”
- System consults loaded ontologies
- System displays results

b. ALTERNATIVE FLOW: no ontological resources found

- System returns message that no ontological resources were found
- User invited to try another query

c. ALTERNATIVE FLOW: closely matching ontological resources found

- System displays result with a warning that ontological resources are closely matching

d. ALTERNATIVE FLOW: search box is empty when user clicks “search ontological resource”

- System shows alerts message

RELATED USE CASES

- Browse ontological resources
- Search for related ontological resources

NOTES / ISSUES

- N/a

UC7 Search for related ontological resources [Update of UC 5.4.7 in D71.1]

ACTORS

- Fisheries scientist

DESCRIPTION

- User searches for ontological resources related to the one at hand

PRECONDITIONS

- User has selected a ontological resource (after browsing or searching)

TRIGGERING EVENT(s)

- User clicks “search for related ontological resources”

POST CONDITIONS

- Related ontological resources are displayed

FLOW OF EVENTS

a. BASIC FLOW

- User selects a ontological resource
- User specifies relation(s)
- User selects direct/indirect relation (reasoning)
- User clicks “search for related ontological resources”
- System queries loaded ontologies
- Related ontological resources are displayed

b. ALTERNATIVE FLOW

- Related ontological resources are displayed
- User is invited to try another query

RELATED USE CASES

- Browse ontological resources
- Search ontological resources in ontology

NOTES / ISSUES

The option “search for related ontological resources” is only available if an ontological resource has been selected

UC8 Browse Taxonomy [Update of UC 5.4.8 in D71.1]

ACTORS

- Fisheries scientist

DESCRIPTION

- User browses concepts in an ontology

PRECONDITIONS

- User is logged in
- Taxonomy panel is visible

TRIGGERING EVENT(s)

- User clicks “browse concepts” from main page
- Taxonomy panel is in focus

POST CONDITIONS

- Browsable list of concepts displayed

FLOW OF EVENTS

a. BASIC FLOW

- User clicks “browse concepts” from main page
- Systems queries ontology server for concepts
- Browsable list of concepts displayed

RELATED USE CASES

- Search concepts
- Search for related concepts

NOTES / ISSUES

- n/a

UC9 Change language of the interface [page 72 – section 5.4 in D7.1.1]

ACTORS

- Fisheries scientist

DESCRIPTION

- Change language

PRECONDITIONS

- User is logged in

TRIGGERING EVENT(s)

- User clicks a language icon in the toolbar

POST CONDITIONS

- User interface has changed to selected language

FLOW OF EVENTS

a. BASIC FLOW

- User clicks a language icon in the toolbar
- System asks the user whether to extend the choice to ontological resources
- User interface changes to selected language

RELATED USE CASES

- Change language of the ontological resource shown

NOTES / ISSUES

- The languages at disposal of the user for the interface are the five languages of FAO: Arabic, Chinese, English, French, and Spanish.
- The languages at disposal of the user for the content depend on the availability
- Object annotation should also switch if users selected to extend the choice to ontological resources

UC10 Change language of the ontological resource shown [page 74 – section 5.4 in D7.1.1]**ACTORS**

- Fisheries scientist

DESCRIPTION

- User changes language of the resource being displayed

PRECONDITIONS

- User is logged in

TRIGGERING EVENT(s)

- User clicks a language icon among those associated with the resource being displayed

POST CONDITIONS

- The resource is displayed in the selected language

FLOW OF EVENTS**a. BASIC FLOW**

- User clicks a language among those associated to the resource being displayed
- The resource is displayed in the selected language

RELATED USE CASES

- Change language of the interface

NOTES / ISSUES

- The languages at disposal of the user for the interface are the five languages of FAO: Arabic, Chinese, English, French, and Spanish.
- The languages at disposal of the user for the resource depend on the availability
- The possibility of setting preferences for associating language of interface with language of resource

UC11 Query Composition [Formerly not included in D71.1]

ACTORS

- Fisheries scientist

DESCRIPTION

- Add ontological resource to query

PRECONDITIONS

- User has selected a ontological resource

TRIGGERING EVENT(s)

- User drags the selected ontological resource to the query area

POST CONDITIONS

- Ontological resource displayed in query area

FLOW OF EVENTS

a. BASIC FLOW

- User drags a ontological resource to the query area
- Ontological resource is displayed in query area
- System adds the ontological resource to the query being formulated

RELATED USE CASES

- Search ontological resource in ontology
- Browse Taxonomy
- Modify set of ontologies

NOTES / ISSUES

- How many concepts will need to be added to the query area depends on the type of query being formulated, which depends, in turn, on the type of resource to query. For example, in order to query a statistical database, the user needs to specify at least *three* variables, i.e., time, space, and subject.

UC12 Query for Data related to individual [Update of UC 5.4.11 in D71.1]**ACTORS**

- Fisheries scientist

DESCRIPTION

- User queries available instances of concepts

PRECONDITIONS

- User composed a query
- At least one concept is in the query area

TRIGGERING EVENT(s)

- User clicks “search Data”

POST CONDITIONS

- System has displayed results as clickable titles linked to their parent concept classes
- System displays related concept instances, if available

FLOW OF EVENTS**a. BASIC FLOW**

- User clicks “search data”
- System consider user tags (local/remote)
- System displays results as clickable titles linked to their parent concept classes
- System displays related cluster of document, if any

b. ALTERNATIVE FLOW: no results found

- System returns to previous page with message

RELATED USE CASES

- Refine query
- Visualize Data Source related to individual
- Visualize Data Source metadata
- Query Composition

NOTES / ISSUES

- N/a

UC13 Visualize Data Source related to individual [Update of UC 5.4.12]

ACTORS

- Fisheries scientist

DESCRIPTION

- Visualize Data Source instance

PRECONDITIONS

- At least one Data Source is listed in the query result

TRIGGERING EVENT(s)

- User clicks on the title of the Data Source

POST CONDITIONS

- System has opened resource in associated application, e.g. browser, word, Acrobat, image viewer, etc.

FLOW OF EVENTS

a. BASIC FLOW

- User clicks on the title of the Data Source
- System opens resource in associated application, e.g. browser, word, Acrobat, image viewer, etc.

b. ALTERNATIVE FLOW: there is no associated application

- System invites user to locate another application or to install needed application

RELATED USE CASES

- Query for Data related to individual
- Refine query

NOTES / ISSUES

- N/a

UC14 Refine query [page 77 in D71.1]**ACTORS**

- Fisheries scientist

DESCRIPTION

- Refine query

PRECONDITIONS

- At least one query has been performed

TRIGGERING EVENT(s)

- User clicks “refine query”

POST CONDITIONS

- System has returned to concept page with previously selected concepts still in the query area, and last searched or selected concept in centre of screen with related concepts

FLOW OF EVENTS**a. BASIC FLOW**

- User clicks “refine query”
- System returns to concept page with previously selected concepts still in the query area, and last searched or selected concept in centre of screen with related concepts

RELATED USE CASES

- Visualize Data Source related to individual
- Query for Data related to individual

NOTES / ISSUES

- N/a

UC15 View ontological resource annotation [Update of UC 5.4.15 in D71.1]

ACTORS

- Fisheries scientist

DESCRIPTION

- View ontological resource annotations

PRECONDITIONS

- At least one ontological resource is displayed

TRIGGERING EVENT(S)

- User selects an ontological resource and clicks “view annotation”

POST CONDITIONS

- System displays ontological resource together with associated annotations

FLOW OF EVENTS

a. BASIC FLOW

- User selects an ontological resource and clicks “view annotation”
- System queries loaded ontologies and displays ontological resource together with associated annotations

RELATED USE CASES

- View Data Source Annotation
- Visualize data related to individual

NOTES / ISSUES

- It is especially important that the visualized metadata include the provenance of the concept, i.e. who created/revised it

UC16 View Data Source Annotation (low priority) [Update of UC 5.4.16 in D71.1]**ACTORS**

- Fisheries scientist

DESCRIPTION

- View Data Source metadata

PRECONDITIONS

- At least one Data Source is displayed

TRIGGERING EVENT(s)

- User selects a Data Source and clicks “view provenance”

POST CONDITIONS

- System displays Data Source metadata, e.g. title, author, subject, last modified, etc.

FLOW OF EVENTS**a. BASIC FLOW**

- User selects a Data Source and clicks “view provenance”
- System queries ontology server and displays concept together with associated metadata

RELATED USE CASES

- View ontological resource annotation
- Visualize data related to individual

NOTES / ISSUES

- N/a

UC17 Email results (low priority) [page 80 in D71.1]

ACTORS

- Fisheries scientist

DESCRIPTION

- Email results

PRECONDITIONS

- At least one Data Source link is displayed

TRIGGERING EVENT(s)

- From a results page user clicks “email results”

POST CONDITIONS

- System has emailed results

FLOW OF EVENTS

a. BASIC FLOW

- From a results page user clicks “email results”
- A form is displayed to allow the user to write and send an email message
- User inputs email and clicks “send”
- System sends formatted results, informs user of success and returns to previous result page

b. ALTERNATIVE FLOW: email is malformed

- System invites user to retry

RELATED USE CASES

- Browse Taxonomy
- Search ontological resources
- Query data related to individual

NOTES / ISSUES

- N/a

UC18 Propose ontology modification (medium priority) [page 81 in D71.1]

ACTORS

- Fisheries scientist

DESCRIPTION

- Propose to modify ontology

PRECONDITIONS

- User is logged in
- At least one ontology is loaded

TRIGGERING EVENT(s)

- User selects “propose modification to ontology”

POST CONDITIONS

- Proposal sent to ontology editors/reviewer

FLOW OF EVENTS

a. BASIC FLOW

- User selects “propose modification to ontology”
- System queries ontology server and opens a page showing the currently loaded ontologies together with any contexts, unions or intersections modifying the overall ontology set and any available unloaded ontologies
- User selects the element(s) to modify (i.e. concepts, instances, entire branches, context), and writes the reason for changes
- User clicks “propose modification to ontology”
- The modification proposal is sent to the ontology editors/reviewer

RELATED USE CASES

- Browse Taxonomy
- Search ontological resources
- Search ontological resource

NOTES / ISSUES

- N/a

UC19 Add to favourites (medium priority) [page 82 in D71.1]

ACTORS

- Fisheries scientist

DESCRIPTION

- Add to favourites

PRECONDITIONS

- A results page

TRIGGERING EVENT(s)

- User selects “add to favourites”

POST CONDITIONS

- System has added current set of query concepts to a list of favourites against the user profile
- The list appears under a heading of “favourites”

FLOW OF EVENTS

a. BASIC FLOW

- User selects “add to favourites”
- System durably writes parameters for current set of concepts and underlying ontology set to a favourites list
- The item is added to a list that appears under a heading of “favourites”

RELATED USE CASES

- Save session
- Modify profile

NOTES / ISSUES

- N/a

UC20 Save session (medium priority) [page 83 in D71.1]**ACTORS**

- Fisheries scientist

DESCRIPTION

- Save session

PRECONDITIONS

- User is logged in

TRIGGERING EVENT(s)

- User selects “save session”
- User Log out

POST CONDITIONS

- System has saved the session parameters against the user profile
- The list appears under a heading of “favourites”

FLOW OF EVENTS**a. BASIC FLOW**

- User selects “save session”
- System durably writes parameters for current set of concepts and underlying ontology set against the user profile
- The user is made aware that the save was successful

RELATED USE CASES

- Add to favourites

NOTES / ISSUES

- N/a

UC21 Generate RSS feed from current query (iteration 2) [page 84 in D71.1]

ACTORS

- Fisheries scientist

DESCRIPTION

- Generate RSS feed

PRECONDITIONS

- A results page is displayed

TRIGGERING EVENT(s)

- User selects “generate RSS feed”

POST CONDITIONS

- System has opened a browser with the URL for the RSS feed based on the current concept query set

FLOW OF EVENTS

a. BASIC FLOW

- User selects “generate RSS feed”
- System opens a browser with the URL for the RSS feed based on the current concept query set

RELATED USE CASES

- Add to favourites
- Modify profile

NOTES / ISSUES

- N/a

UC22 Annotate retrieved document with comments on quality (low priority) **[page 85 in D71.1]**

ACTORS

- Fisheries scientist

DESCRIPTION

- After reading a retrieved document, a user annotates it with a judgement and a comment about its usefulness

PRECONDITIONS

- User is logged in
- A document is opened
- User has annotation rights

TRIGGERING EVENT(s)

- User selects “assess this document”

POST CONDITIONS

- System has attached an assessment to the document, together with relevant metadata concerning the user who issued the assessment.

FLOW OF EVENTS

a. BASIC FLOW

- User selects “assess this document”
- System requests user to rank the document according to some criterion (e.g., reliability)
- User ranks the document
- System requests user to add a comment on the ranking

RELATED USE CASES

- View data related to individual

NOTES / ISSUES

- Document generically refers to any type of annotatable resource, such as textual documents, graphics, statistical tables, ...
- The model of interaction with the system to produce the information requested in Basic Flow is to be defined. For example, the user could be requested to fill in a form, or follow a step-by-step process (first the ranking, then the comment)
- Annotation cannot be conflicting, as they express the personal view of the user who made them

UC23 Index enrichment against ontology(ies) domain model (low priority)
[Update of UC 5.4.23 in D71.1]

ACTORS

- Fisheries scientist

DESCRIPTION

- After reading a retrieved document, user annotates it (or part of it) with keywords

PRECONDITIONS

- User has annotation rights
- A document is opened

TRIGGERING EVENT(S)

- User selects “assign keywords to this document”

POST CONDITIONS

- System has attached keywords to the document

FLOW OF EVENTS

a. BASIC FLOW

- User selects “assign keywords to this document”
- User inputs keyword(s) and finishes the operation
- System annotates document with relevant keyword(s)

RELATED USE CASES

- Annotate retrieved document with comments on quality
- Annotate retrieved documents with comments

NOTES / ISSUES

- The possibility of adding keywords not taken from any resources could be discussed
- The possibility of letting the system suggest which keywords to assign to the document should also be investigated: for example, the system could show keywords already assigned to the document, or compare the document with a selected thesaurus or classification schema

UC24 Annotate retrieved document with comments (low priority) [page 88 in D71.1]**ACTORS**

- Fisheries scientist

DESCRIPTION

- After reading a retrieved document, user annotates it

PRECONDITIONS

- A document is opened

TRIGGERING EVENT(s)

- User selects “add your comment”

POST CONDITIONS

- System has attached user comment to the document

FLOW OF EVENTS**a. BASIC FLOW**

- User selects “add your comment to this document”
- User writes comment

RELATED USE CASES

- Annotate retrieved document with comments on quality
- Index enrichment against ontology(ies) domain model

NOTES / ISSUES

N/A

**UC25 Select ontologies to use for browsing documents or web pages
(iteration 2) [page 89 in D71.1]**

ACTORS

- All

DESCRIPTION

- User selects ontologies to support browsing and reading of documents and web pages

PRECONDITIONS

- User is logged in

TRIGGERING EVENT(s)

- User selects “add ontologies for browsing”

POST CONDITIONS

- Ontologies are selected and can be activated to be used to support browsing and reading

FLOW OF EVENTS

a. BASIC FLOW

- User selects “add ontology for browsing”
- System asks user to select an ontology from file system
- User selects ontology from file system
- System asks whether to select the entire ontology or a fragment of it
- User selects “entire ontology”
- System saves the selected ontology in the user profile

b. ALTERNATIVE FLOW: user selects fragment of ontology

- System asks whether to select the entire ontology or a fragment of it
- User select “fragment”
- System opens the ontology
- User selects the desired fragment and finishes the operation
- System saves the selected fragment in the user profile

RELATED USE CASES

- Include a selection from an existing ontology (see Chapter 4)

NOTES / ISSUES

- (MAGPIE FUNCTIONALITY)

UC26 Use ontology to support browsing (iteration 2) [page 90 in D71.1]

ACTORS

- All

DESCRIPTION

- User uses ontologies to support browsing and reading of documents and web pages

PRECONDITIONS

- User is logged in
- Ontologies are selected to support browsing

TRIGGERING EVENT(s)

- User selects “activate ontologies for browsing”

POST CONDITIONS

- Ontologies are activated to be used to support browsing and reading of ontologies

FLOW OF EVENTS

a. BASIC FLOW

- User selects “activate ontologies for browsing”
- System loads the selected ontologies
- Words occurring both in the opened document or webpage and the ontologies are highlighted, and
- Information or services associated with the ontologies are shown to the user

RELATED USE CASES

- Select ontologies to use for browsing documents or web pages

NOTES / ISSUES

- Examples of pieces of information associated to an ontology are: available relations to other terms or documents, translation of a highlighted word in other languages, available definitions
- Examples of services associated with the ontology are: search for documents relevant to the highlighted term, search for documents annotated with the highlighted term (i.e. the term user as keyword).
- (MAGPIE FUNCTIONALITY)

UC27 Identify trend (iteration 2) [page 91 in D71.1]

ACTORS

- All

DESCRIPTION

- System presents the trend of one or more selected parameters

PRECONDITIONS

- User is logged in
- At least one concept instance has to be selected

TRIGGERING EVENT(s)

- User selects “identify trend”

POST CONDITIONS

- User is shown the trend of the selected parameter

FLOW OF EVENTS

a. BASIC FLOW

- User selects one or more parameter to compute a trend
- User clicks “identify trend”
- System shows trends of the selected parameter

a. ALTERNATIVE FLOW

- System has no data about the trend
- System returns warning and suggestions about available “similar” trends

RELATED USE CASES

- N/a

NOTES / ISSUES

- Possible parameters of which to compute the trend include: marine areas, land areas, national fishing fleets, species, species groups, fisheries commodities, vessel types, time ranges

UC28 Compare data by reporter (iteration 2) [page 92 in D71.1]**ACTORS**

- All

DESCRIPTION

- User compares data

PRECONDITIONS

- User is logged in
- At least two data source results have been returned

TRIGGERING EVENT(s)

- User clicks “compare”

POST CONDITIONS

- Two documents are shown side by side for comparison

FLOW OF EVENTS**a. BASIC FLOW**

- User makes a query that returns a result set of at least two documents
- User selects two items from the list and clicks “compare”
- System opens the two selected items, and shows them side by side

RELATED USE CASES

- N/a

NOTES / ISSUES

- By data source we mean the organization or body (national ministries, regional fisheries bodies, institutions, etc.) that released the data

Summary Table of Use Cases and Priorities

Use Case	High Priority	Medium Priority	Low Priority	Second Iteration
1.1 Login	X			
1.2 Logout	X			
1.3 Register	X			
1.4 Modify Profile		X		
1.5 Modify User Account	X			
1.6 Search ontological resource in ontology	X			
1.7 Search for related ontological resources	X			
1.8 Browse Taxonomy	X			
1.9 Change language of the interface	X			
1.10 Change language of the ontological resource shown	X			
1.11 Query Composition	X			
1.12 Query for Data related to individual	X			
1.13 Visualize Data Source related to individual	X			
1.14 Refine query	X			
1.15 View ontological resource annotation	X			
1.16 View Data Source Annotation			X	
1.17 Email results			X	
1.18 Propose ontology modification		X		
1.19 Add to favourites		X		
1.20 Save session		X		
1.21 Generate RSS feed from current query				X
1.22 Annotate retrieved document with comments on quality			X	
1.23 Index enrichment against ontology(ies) domain model			X	
1.24 Annotate retrieved document with comments			X	
1.25 Select ontologies to use for browsing documents or web pages				X
1.26 Use ontology to support browsing				X
1.27 Identify trend				X
1.28 Compare data by reporter				X

Figure 24 - Use case priorities

Annex B - List of Acronyms used in this deliverable

AGROVOC	FAO-Multilingual Agricultural Thesaurus
API	Application Programming Interface
ASFA	Aquatic Sciences Fisheries Abstracts
CITES	Convention on International Trade in Endangered Species
CORBA	Common Object Remote Broker Architecture
COTS	Commercial off the Shell
CPUE	Fishery Basic Catch Per Unit Effort
DLO	Document-Like Object
CSV	Comma Separated Value files
EIMS	Electronic Information Management System
FAOLEX	FAO-Legislative database
FI	Fisheries Department, FAO of the UN
FIDI	Fishery Information, Data and Statistics Unit, FAO of the UN
FIGIS	Fisheries Global Information System
FIRMS	Fishery Resources Monitoring System
FSDAS	Fisheries Stock Depletion Assessment System
GIS	Geographical Information Systems
HTML	HyperText Markup Language
ISO	International Organization for Standardization
ITIS	Integrated Taxonomic Information System
JWS	Java Web Start
KIMS/KIDS	Key Indicator Mapping System/Key Indicator Data System
MIME	Multipurpose Internet Mail Extensions
MIT	Massachusetts Institute of Technology
NMFS	National marine Fisheries Service
NOAA	National Oceanic and Atmospheric Administration
OBIS	Ocean Bio-geographic Information System
OO	Object Oriented

OWL	Web Ontology Language
PDF	Portable Document Format
RDF	Resource Description Framework
RMI	Remote Method Invocation
RFB	Regional Fishery Body
RSS	Really Simple Syndication
RTMS	Fisheries Reference Table Management System
SGML	Standard Generalized Markup Language
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
UTF8	8 bit Unicode Transformation Format
WP	Work package
XML	Extensible Markup Language
XHTML	EXtensible HyperText Markup Language
XSL	EXtensible Stylesheet Language

References

NeOn WP6:

- [6.2.1 - NeOn API and Architecture](#) (05/04/2007)

NeOn WP7:

- [D7.2.2 - Revised and enhanced Fisheries ontologies](#) (ongoing due 15/08/2007)
- [D7.4.1 - Software architecture for managing the fishery ontologies lifecycle](#) (ongoing, due 15/08/2007)
- [D7.1.1 - User requirements specifications for the Fisheries ontology, knowledge tools and alert system](#) (15/11/2006)
- [Fisheries Systems Inventory T7.2.1](#) (05/04/2007)

ISO standards:

- [ISO 9126-1](#) Non-functional requirements (2006-09-15)
 - See also: <http://www.cse.dcu.ie/essiscope/sm2/9126ref.html>
- [ISO 2709](#) bibliographic exchange format
- [ISO tc211](#) GIS metadata standard

Design analysis methods:

- [Evolutionary Process for Integrating COTS-Based Systems](#) (EPIC, 2002)
- [The 4+1 View Model of Architecture](#) (Kruchten, 1995)

Licenses:

- [GNU GENERAL PUBLIC LICENSE](#) (June 1991)

Software:

- [Log4J](#) Logging software (N.D.)
- [JUNIT](#) code-driven testing software (N.D.)
- [JavaDoc](#) code documentation tool
- [UNESCO ISIS](#) database
- [BIRT Project](#) (Business Intelligence and Reporting Tools)

