Deliverable D.4.2

Knowledge Lenses and Process Support Tools

Authors: Victoria Uren, Open University, v.s.uren@open.ac.uk
Sam Chapman, University of Sheffield, sam@dcs.shef.ac.uk
Aba-Sah Dadzie, University of Sheffield, a.dadzie@dcs.shef.ac.uk
Piercarlo, Slavazza, Quinary S.p.a., x-media@quinary.com
Thomas Franz, University of Koblenz-Landau, franz@uni-koblenz.de
Yuangui Lei, Open University, y.lei@open.ac.uk
: Johannes Busse, Ontoprise, busse@ontoprise.de
Daniel Rabus, Ontoprise, rabus@ontoprise.de
Roman Korf, Ontoprise, korf@ontoprise.de
Tran duc Thanh, University of Karlsruhe, dtr@uni-karlsruhe.de
Till Christopher Lech, CognIT, Till.Christopher.Lech@cognit.no

Work-package: WP4 Knowledge sharing and process support

Type: Technical Report

Distribution: Public

Status: Preliminary

Date: 21.12.2007

Deliverable Coordinator: Victoria Uren, Open University

Reviewers: Steve Fullerton/Andrew Maisey, Solcara, Christine Preisach, University of Hildesheim

Area Coordinator: Steffen Staab, Univerity of Koblenz-Landau

Project Coordinator: Fabio Ciravegna, University of Sheffield

EU Project Officer: Ralph Feirgolla

ABSTRACT

The X-Media consortium has responded to the requirements for knowledge sharing and process support with a range of task centric tools that are individually innovative while working within the constraints of the X-Media architecture to facilitate interoperability. The next tasks are to build the use case test beds and to evaluate the tools in context. This report presents technical descriptions of the knowledge sharing and process support prototype tools that will be used in phase 1 test beds. It describes tools for annotation, semantic search and browsing, presentation of results and analysis and sharing of results. For each tool its research aims and its contribution to achieving X-Media requirements are detailed as well as a technical description of its operation.

TABLE OF CONTENT

1 Introduction

Workpackage 4 is responsible for producing end user tools for knowledge sharing and process support which exploit the semantic metadata extracted by Area 2 partners and which operate with the X-Media kernel. The document has three aims,

1. To present the research contributions of each tool.
2. To detail what each tool does tool does, its inputs, outputs, user interaction etc..
3. To specify how each tool meets the project requirements.

Concerning point 3, extensive requirements work has been undertaken in the project. This document especially references the *top level requirements* of X-Media knowledge sharing defined in D4.1 [D4.1] and the specific *needs of the use cases* described in D12.2 [D12.2] and D13.2 [D13.2]. The top level requirements in D4.1 concern interoperability, performance, handling heterogeneous knowledge, fundamental functions and innovative X-Media functions. Each tool reported is planned to be included in at least one use case as detailed in table 1. The use case needs are expressed in the scenario description, particularly Technical Insertion Points (TIPs).

Use case	**WP4 tools involved**
R-R Experimental Vibration	Interactive Annotation, K-Search, K-Views
R-R Issue Resolution	Summarisation tool, Interactive annotation, K-Search, K-Views
FIAT Noise Curves	SemSearch, LENA, semantic email
FIAT Competitor Analysis	XXploreKnow!, Ontoprise, semantic scratchpad

Table 1 Planned use of tools per use case

2 Subtask 4.2.1 Annotation

In task 4.2.1 several prototypes of interactive annotation tools are developed. During the first phase of the project Ontoprise has worked on two different classes of tools:

- Three flavours of office documents annotation tool allow for annotating regions of Open Document Format (ODF) text, presentation and spread sheet documents.
- An image annotation tool allows for annotating and amending annotations of regions within images

Scope of the annotation tools: Annotating text or images is narrowly related to three typical other tasks:

- Identifying the media which are to be annotated interactively.
- Triggering IE and text annotation engines of area2 in order to get pre-annotated media.
- Calling the interactive annotating component (i.e., run it), possibly by telling it which image or text there has to be annotated.

These related tasks have to be performed by a workflow tool, a search tool or other task supporting components. Several of such tools will be provided within wp4 and to be integrated into the area 4 interaction system. It is the role of this X-Media user interaction paradigm to relate knowledge search and knowledge browsing components with the semantic scratchpad and various (interactive) annotation components.

2.1 The Open Document Format (ODF) Annotation Toolbox

2.1.1 Technical Description

Until summer 2007, three related text annotation tools have been produced. In order to explore the needs of the users and the technical challenges of annotation interactions it was decided to realize different interaction paradigms first. In a second step it has to be decided which paradigm eventually should be the common one for all wp4 interactive components.

Text (ODT): Because of restrictions of the ODP API for the time being the tools interactively can annotate only the whole document.

- Select (highlight) text segment (typically a word or phrase)

- Assign concept: navigate to concept or instance from ontology browser; then mouse click to the „annotate“ button
- All (!) occurrences of the text within the document will be highlighted
- TBD: selecting an image should open the image annotation tool; interaction with X-Media kernel required

Presentation (ODP): Because of restrictions of the ODP API for the time being the tool interactively can annotate only whole slides.

- navigate to a single slide
- assign concept or instance from ontology browser to this slide.
- no highlighting so far
- PROBLEM: OpenOffice API shortcomings

Spreadsheets (ODS):

- select interactively a cell range (on a specific sheet)
- highlighting is supported, though not recommended:
- allowing highlighting would allow also changing spreadsheet content.

Since it is assumed that there is no write access to the documents to be annotated this is it not feasible. However, OpenOffice does not allow to open spread sheets for highlighting only in read only mode. This behaviour is considered a design flaw of the API

2.1.2 Status of the Document Annotation Tool Integration

Ontoprise has presented it’s prototypes of the image and the ODF annotation tool in spring and autumn 2007 (Athens, Milton Keynes). For the time being these tools are narrowly integrated with interaction components of the commercial version of OntoStudio®. The respective developing tasks ground on a non-obfuscatable closed source head version of the company’s software stack. It is regrettable that it is not possible to disseminate this version to the public. Ontoprise currently works on implementing the X-Media components in a more lightweight and open source version.

A note on the user interaction (UI) integration platform: The X-Media WP4 partners committed to ground the first integration of the various tools (annotation, knowledge browsing, knowledge lenses) on the Eclipse rich client platform (RCP).

This decision will allow a comparatively easy and – at least from an interaction point of view – sound integration of the partner's components.

However, because it can be forseen now that a fat client might not be acceptable for R-R and CRF there the challenges of a lightweight and browser based client communication where also considered. Accordingly the image annotation tool was crafted with Swing components (as opposed to Eclipse SWT) in order to become independent from the Eclipse RCP. While the image tool can now be run stand alone as a Java component this approach cannot be recommended. The problems and challenges to be solved when someone does not want to rely on a certain platform are too tedious and annoying compared to the research focus of X-Media.
Concerning the second phase of the X-Media project there should be a discussion whether either (1) grounding completely on a sophisticated and highly tailorable integration platform (e.g., the Eclipse RCP together with a public domain semantic integration platform like the Neon Toolkit) during the lifetime of the X-Media project or (2) moving to a common web based knowledge integration paradigm, ideally a social web platform like a semantic wiki for user interaction.

2.2 Image Annotation

Beside the ODF annotation components an image annotation component was developed. This component differs from the former components in that it can be run as a java stand alone application.

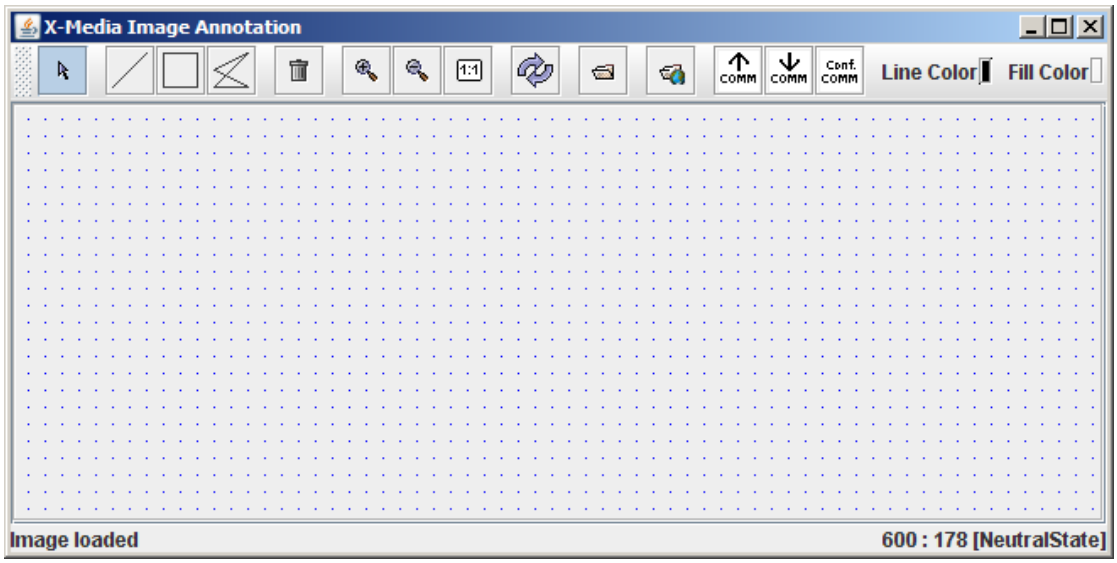
2.2.1 Technical Description

Interaction with the knowledge browser for the time being is simply done by drag and drop. (This however does not mean that a more sophisticated interaction according to a common interaction paradigm is not urgently feasible.)The interaction steps are:
(1) Select a still region within an image. Draw a line, rectangle or polygon.Optionally colourize a selection. Optionally add / remove / rearrange nodes. Optionally flex the edges to 2nd order polynomials or Bezier curves (as requested from WP7 raw data).

(2) Select an arbitrary annotation string (ideally an URI) in a knowledge browser (e.g. Neon Toolkit class or instance browser; Firefox Web browser; Office Document). "Assign" the selected concept or instance from ontology browser to a segment. TBD: Define the interaction / HCI paradigm of „assigning" an annotation.

(3) Drag 'n drop the annotation string onto an still region. Moving the mouse over a still region shows all annotation strings.

The annotation tool is deployed as a java stand alone application together with the source code. Change directory to the directory COMM_Interface > imageannotation. There you should find the batch file start-imageannotation.bat. Execute it. You should get a window similar to this one:



As long the image annotation tool is not integrated into a larger tool suite some additional interaction elements are needed.

Configure a Sesame repository: First you have to connect to an RDF repository: Press the button “Conf. COMM” in order to configure a Sesame2 repository. Choose between four possibilities:

(1) „Memory-Repository“ resides temporarily in the main memory. Allocating an in memory repository automatically loads the COMM Ontologies from the directory „ProtegeProjects“. You can use an in memory repository easily for testing. However, all changes will be lost after the application stops running.

(2) „Persistent Memory-Repository“: Data will be kept persistently on your disc. Navigate to an existing directory. If it is empty a new store will be allocated. If it is not empty an existing store will be read.

(3) „Remote Repository“: Connect to a Sesame repository on the net. Give the Sesame URI and the ID of the repository.

(4) „Memory-Repository End-Point Test“.Similarly to (1). In addition a COMM Image Object will be allocated in the repository: Copy the endpoint reference of the allocated image! (This is a unique id, similar to http://test.x-media.org/id-bcb66d01-

75e1-4911-bb1e-5f5f2d638ecc)You will need it in the next step to read in the allocated object.

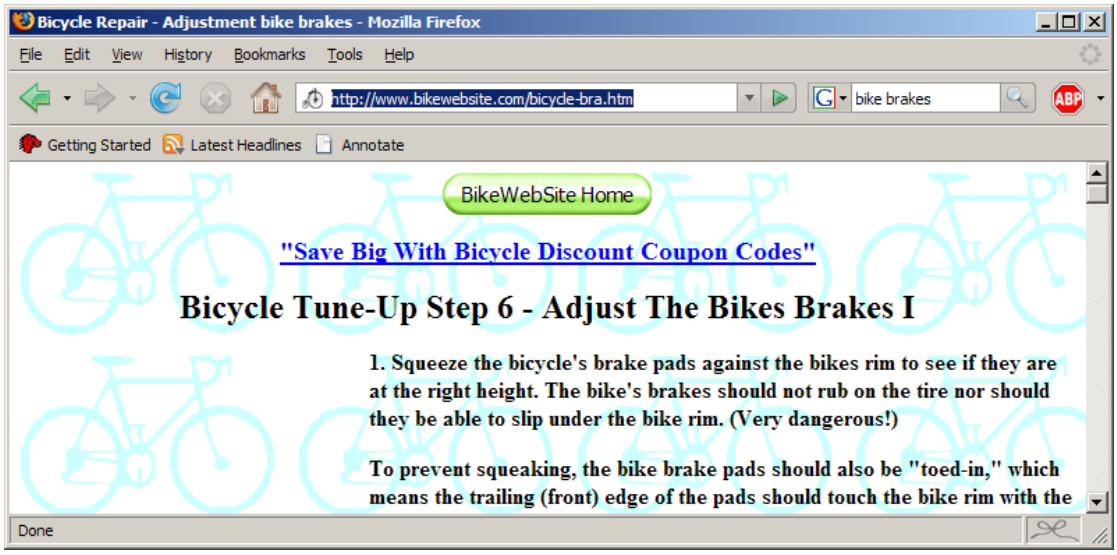
Load an image and it's annotations: Once you have started the image annotation tool successfully you should load an image. (Note that selecting the image which is to be annotated preferably should be done by another X-Media application like the search tool or the knowledge browser).

Path 1: Load an image from an URI: Press the button "load image from URL" and give the URI of an image (e.g. http://upload.wikimedia.org/wikipedia/commons/4/4f/Scheibenbremse-magura.jpg). Then press the button "read annotations from COMM". The tool looks up all endpoint references available for the give URL and loads the respective annotations. All annotations are all shown in parallel.

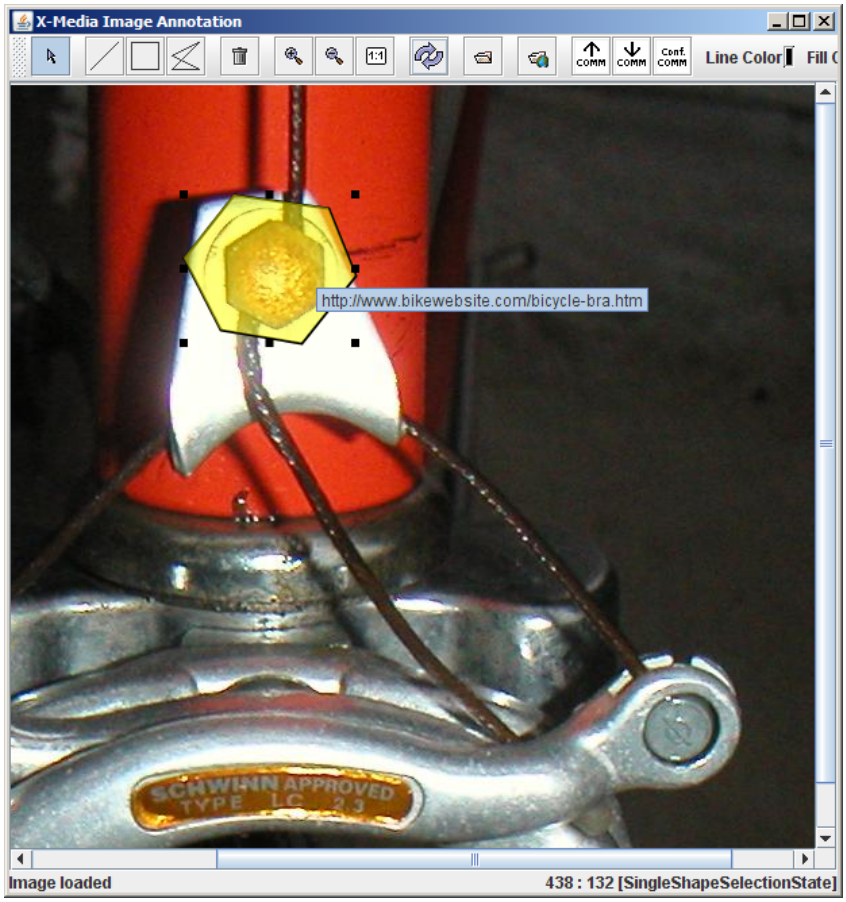
Path 2: If you have chosen to work with (4) "Memory-Repository End-Point Test" you do not have to load an image first. You directly can press the button "read annotations from COMM". For there is no image loaded you will directly be asked for an end point reference. You should paste here the endpoint reference you have copied before while configuring the repository. The endpoint test loads the image from http://upload.wikimedia.org/wikipedia/commons/4/4b/Brake.agr-edit.jpg . There are two annotations which will we shown.

Remark: In a more integrated UI it is up to other components to select which image (given by URI or by endpoint reference) should be annotated. In fact the "load image" buttons do not belong to the core of the annotation tool.

Annotate a region: Annotating a region comprises, in principle, three steps: (1) Select the region to be annotated. Simply draw a polygon or select an existing one. (2) Select a resource you will use in order to annotate the region. In general selecting this instance is *not* up to the annotation tool itself. Instead it is, e.g. the X-Media knowledge browser or search tool, which allows identifying the instance we want to use for the annotation. (3) Instantiate the annotation, i.e. instantiate a link between a region and a resource. Note: For the time being this instantiation of an image annotation is done merely by **drag and drop:** Select the URI you might want to use for annotating the region, e.g. with Firefox, and drag it to the image region.



Here the user navigated to a page about bike brake adjustment. She highlights the URL, grabs it with the mouse and drops it to the region previously selected in the annotation tool.



Drag and drop might look somewhat tedious from a usability point of view, at least for the current stand alone application. But once we have defined the platform for integrating the various X-Media user interface components (Ontoprise suggests to

commit for the Eclipse Rich Client Platform) we will develop many more and much nicer workflows to establish this annotation link between resource and data region interactively. Our text annotation tool, which is more narrowly integrated in OntoStudio (head version, not suitable for roll out for the time being), allows annotating a text by selecting the appropriate instances and concepts with standard ontology browsing plugins and then simply pushing an annotate button.

How to test the tool: Start the tool. Connect it to a repository. “Load image from URL”, e.g. http://upload.wikimedia.org/wikipedia/en/8/8f/Center_Pull_Brakes.JPG . Define the still region to be selected, i.e. draw a polygon or a rectangle. Annotate the still region with a resource, e.g. an URI you have browsed to with an ordinary web browser. Write the annotations to the store. Select each still region and remove it (you have an image without annotations now). Read annotations from the store. The regions and it’s annotations you defined before are displayed.

2.3 Meeting the X-Media requirements

The primary addressees of the interactive annotation tools are the end users described in the use cases of R-R and CRF. The main purpose of the tool is to visualize specific aspects of image and text annotations and allow for refining or amending existing annotations and adding new annotations.
While there seems to be evidence from the use case descriptions that none of the end users will have the resources to identify and annotate segments manually in detail basic functionality is provided for that. This is because secondary addressees are experts producing the text, image or raw data IE components within area 2. While the user might find the image segmentation and annotation functionality useful she will recognize that the annotation tools do not (and cannot) allow editing and amending at each detail the COMM Ontology and API allows to state. The interactive annotation tool is not intended to be an interactive front end for researchers which need to amend annotation details in order to e.g. improve IE machine learning algorithms.

Concerning the top level requirements, the annotation tool addresses the following (numbers refer to sections in Deliverable 4.1 [D4.1]):

3.1.1 Interoperability between Tools and Tool Components: The current prototype of the image annotation tool is implemented as a stand alone java application. However, Ontoprise strongly advises to integrate it with other tools by means of standard UI integration platform, i.e. the Eclipse Rich Content Platform (Eclipse RCP)

or a social web platform. This should make it compatible with most of the other WP4 knowledge sharing components.

3.1.2 Interoperability with the X-Media Architecture: All interactive annotation tools use the COMM API to interact with the X-Media kernel.

3.1.3 Interoperability with Relevant Standards: Compliance with the X-Media semantic annotation model is provided by using the COMM API.

3.2 Performance: As only one document can be annotated at a time performance is not expected to be an issue.

3.3 Handling Heterogeneous Knowledge: For the time being provenance information is not shown to the user. However, if it is possible to integrate the annotation tools within the Eclipse RCP framework with other semantic components (e.g. from the NeOn toolkit) an elegant and flexible way to allow all sorts of provenance annotation editing can then be provided.

3.3.2 Images, Text and Raw Data: For the time being a separate tool for text, presentation, spread sheet and image annotation is provided. Integrating theses tools can easily be done i.e. with the Eclipse RCP.

3.3.3 Working concurrently with multiple domain models is fully supported if we can use the NeOn toolkit.

3.4.1 The tools support a **broad diversity of users** by means of multi domain models. Integration with other wp4 components under a common UI additionally allows a fine grained user tailoring.

3.5 Innovative X-Media Functions: The interactive annotation tools realize Cross-Media Annotation by focusing on a parallel, integrated use of single media annotation combined with other standardized semantic tools like knowledge (A-Box) browser or the T-box browser from the NeOn Toolkit.

3 Subtask 4.2.2 Knowledge Lenses – Search and Select

Semantic search/select is a functionality which allows end users or software agents, such as a semantic scratchpad service, to specify their information seeking needs by means of constructing complex queries about the underlying knowledge in the

application and returns precise results to the queries. It provides a way to filter and select the required knowledge (i.e. semantic metadata) from the underlying repositories which are often heterogeneous and large scale.

A search and select lens feature is central in all of the use cases because filtering the knowledge repository is an essential first step in all knowledge sharing activities. Without some form of selection the user would simply be overwhelmed by the volume of metadata. In phase one, we present four different search and select tools which will be tested in at least one use case. The tools are: LENA a semantic browser that can browse over semantic email, SemSearch a keyword search engine with refinement facilities, K-Search a combined metadata and text search engine, and XXploreKnow! a combined keyword search tool and visual knowledge browser.

3.1 LENA

LENA[1] stands for **LE**ns based **NA**vigator. A lens represents a particular view onto RDF data and is described by the Fresnel Display Vocabulary [Pietriga et al. 2006]. LENA supports viewing RDF data in a web browser, rendered according to the lens descriptions provided. The use of multiple lenses is supported and these are indicated when available for a resource. The design goal is to make different views onto the same data easily accessible for the user. As shown in Figure 1, LENA lists available lenses in a box (named *Default Lenses*) on the left. The box below (named *Included Classes*) lists RDF classes referenced in the underlying RDF repository and the number of instances for each class. Furthermore, a snowball-like icon indicates that lenses are defined at least for some of the instances.

1 http://isweb.uni-koblenz.de/Research/lena

Figure 1: Screenshot of LENA

The biggest portion of the screen (the right side) displays instance data. Figure 1 shows the *standard* view for instance, i.e. the *RDF-lens* where all triples are shown where the resource shown in the header is the subject. That resource is accompanied by multiple icons. Again, if a snowball-icon is shown that indicates that a lens is available for a resource. Clicking on the snowball-icon leads to a view according to another lens. Figure 2 is an example of a lens for a person, representing similar information shown as in Figure 1 in a more human-readable way.

Figure 2: Screenshot of LENA, Application of a lens for persons

3.1.1 Technical Description

LENA is implemented as a web-based application based on Java Servlet technology. Lenses are written in RDF using the N3 format. To write lenses for complex RDF structures created through sophisticated ontology frameworks like COMM [Arndt et al. 2007] or X-COSIM [Franz et al. 2007] as they are employed in X-Media, LENA supports SPARQL[2] selectors. As a comprehensive query language for RDF, SPARQL complies with the requirements needed to select from these complex structured RDF graphs. As an extension to the Fresnel engine developed as part of the Simile project[3], LENA adds Fresnel SPARQL selectors.

3.1.2 Meeting the X-Media requirements

Knowledge lenses are one of the contributions of the X-Media project planned for WP4. LENA implements one interpretation of a knowledge lens based on an interpretation of RDF data towards different visual representations. Such different views are required in several of the X-Media use cases, e.g. in the *Issue Resolution* use case at R-R where a document-centric view or issue-centric view onto resources can be considered beneficial (cf. to TIP 3 and TIP 5, D12.2). As a generic mechanism

2 http://www.w3.org/TR/rdf-sparql-query/

3 http://simile.mit.edu

LENA-based knowledge lenses, can also be applied in Fiat use cases (cf. to TIP 1, 2a, 9).

3.2 SemSearch

As discussed in our review paper [Uren et al.], the requirements space for semantic search systems can be viewed as having four dimensions (see figure 3): the kinds of queries that can be made, the search environment, problems intrinsic to semantic searching, and supporting the user in formulating iterative and exploratory searches.

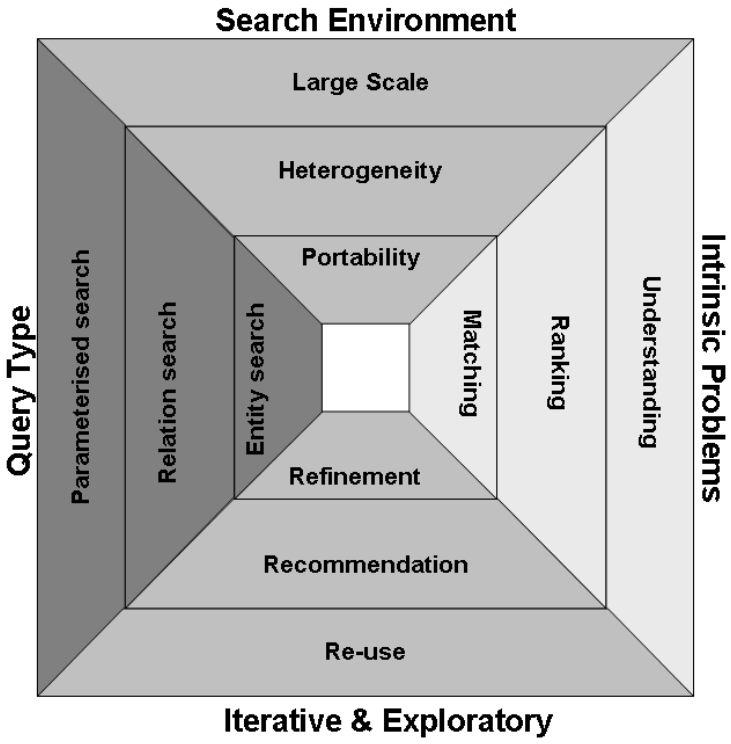


Figure 3 A pyramid representing the requirements space for semantic search systems.

One key contribution of the current SemSearch prototype is its mechanism for keyword based querying of semantic metadata, which makes semantic search intuitive for ordinary end users. Particularly, it allows users to make searches without having to know either the structure of the ontology, tackling the issue of *understanding* the semantic space, or a formal search language. The SemSearch search mechanism starts by interpreting the user query and *matching* senses of the keyword terms, from the ontology, that could be used in the search. Taking a components-based approach [Uren 2006] it then generates a collection of formal search statements and selects the

best to formulate the search. This process is described in detail below and in our recent papers [Lei 2006] [Lei et al.].

Another important contribution made by SemSearch is the methodology designed for supporting multi-keyword semantic searching, which is computing formal queries from user queries. Since the publication of the SemSearch approach in 2006, at least two other works have been published based upon this methodology. The first is about SPARK [Zhou 2007], which builds a query tree from keyword input and then produces a ranking of the multiple possible queries (SemSearch currently takes a heuristic approach to selecting from the multiple possible queries). The second concerns an alternative graph-based approach to query translation being explored by the University of Karlsruhe (see section 3.4 and [Tran 2007]). The combination of the efforts of the Open and Karlsruhe Universities puts X-Media at the forefront of international research in the emerging field of semantic query interpretation.

3.2.1 Technical Description

As shown in Figure 4, SemSearch comprises i) a query interface, which supports the specification of multi-keyword queries; ii) a keyword search engine, which makes sense of user queries by exploiting the domain ontology and the extracted metadata; iii) a query translation engine, which derives appropriate formal queries from the user query; iv) a query refinement engine, which allows the user to reformulate their query towards their information seeking needs; v) a ranking engine, which presents the search results in an order that indicates their degree of satisfaction for the user query; vi) a user interface component, which supports all the user interaction required for query specification and refinement and results presentation. The system also provides an index engine, which indexes semantic entities contained in the domain ontology and the gathered metadata repositories.

Although it is iterative, a search in SemSearch often takes five steps, namely *interpreting user queries, generating formal queries, query refinement, querying*, and *ranking*. Prior to the search process, there is an important step called *indexing*. Now let us get a closer look at each step to understand how the search tool works.

Step0 – Indexing: Like any other search system, SemSearch needs to pre-index the search targets, which includes the specified semantic data repositories and the underlying ontologies in the context of data oriented semantic search. Briefly, we use Lucene[4] to build the indexes. The parts of the RDF content that are indexed are the

[4] http://lucene.apache.org/

local names and labels of semantic entities (including concepts defined in the ontologies and the instances and relations contained in the data content) and their short literal values. The rationale behind this is that these descriptions tend to be able to reasonably reflect the meaning of the indexed entities.

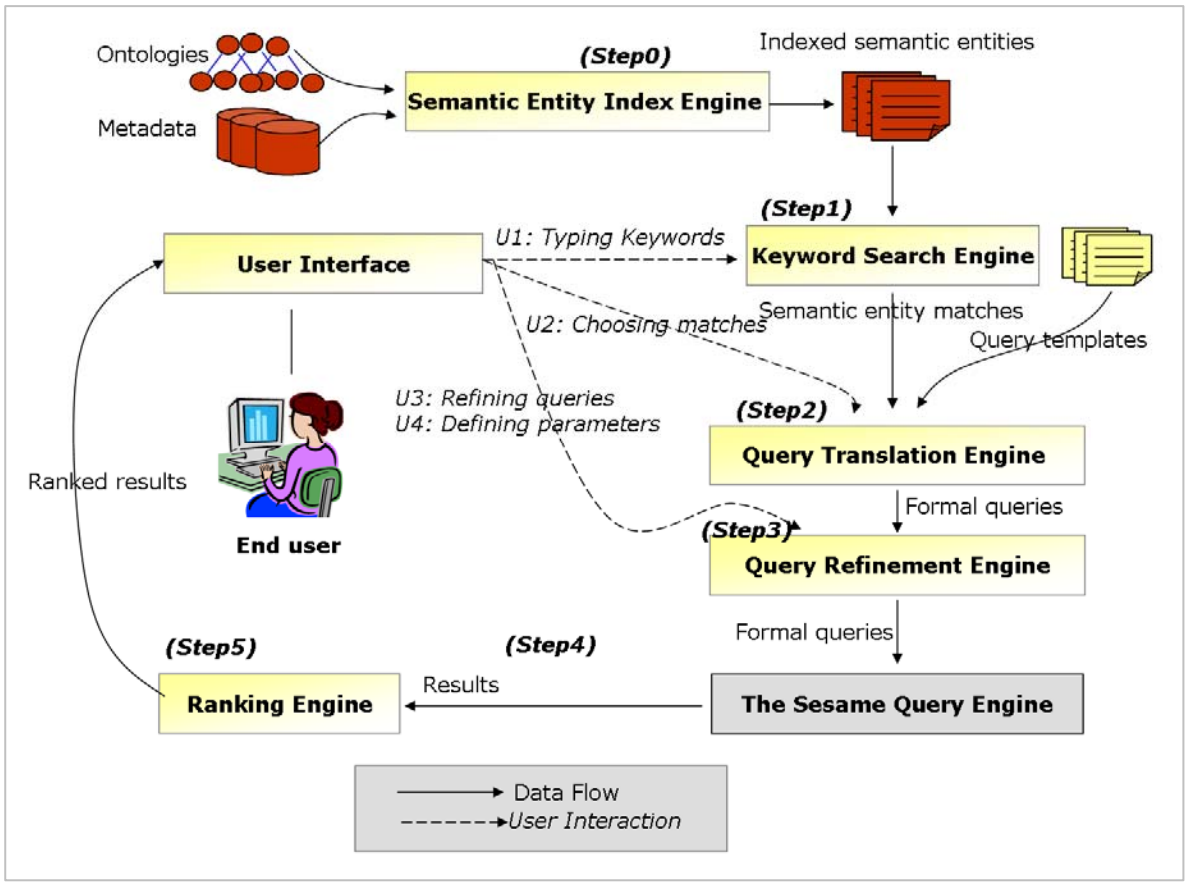


Figure. 4. An Overview of the SemSearch Architecture

Step 1 - Interpreting User Queries: The task of this step is to find out the semantic meanings of the keywords specified in user queries so that the search engine knows what the user is looking for and how to satisfy the user query. From the semantic point of view, one keyword may match i) general concepts (e.g., the keyword “car seats” which matches the concept Car-Seat), ii) semantic relations between concepts, (e.g. the keyword “make” matches the relation has-maker), or iii) instance entities (e.g., the keyword “Fiat” which matches the instance Fiat-Company). The system achieves this task by looking up the indexes built in the indexing step. As described earlier, we used the local names, labels, and short literal vales as the bases for finding matches.

Step 2 - Generating Formal Queries. In this step, the search engine takes as input the semantic matches of user search terms and outputs appropriate formal queries. We

classify user queries into two types: i) simple queries, which only comprise two keywords; and ii) complex queries, where more than two keywords are involved.

Dealing with simple user queries. As the types of semantic entity match combinations are fixed in simple user queries, we developed a set of templates to describe how to retrieve relations between two semantic entities. Among all the combinations, there are three most possible types between two keywords. This is because we had made the assumption that the subject keyword matches a class concept, in the proposed query interface. Figure 5 shows the templates for these combinations.

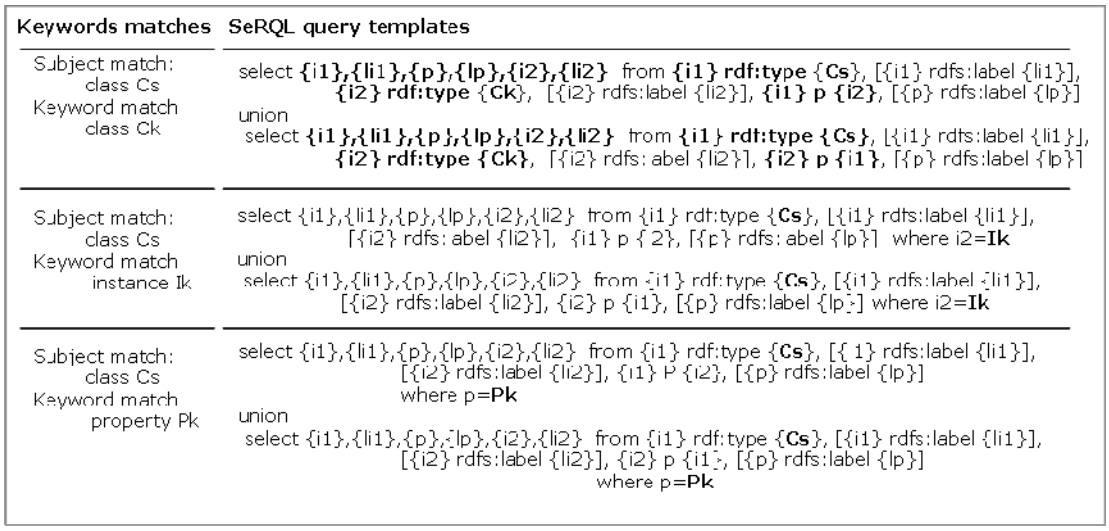


Figure. 5. Selected SeRQL Query Templates for Simple User Queries.

Now let us investigate the first combination where both keywords in a query match classes. The search results are expected to be the instances of the class Cs (i.e. the match of the subject keyword) which have explicitly specified relations with the instances of the class Ck (i.e. the match of the other keyword). For example, when querying for “cars: student”, the expected results are those instances of the class Car that have some kind of relation with the instance *Student*, e.g., best selling.

Please note that there are situations where no class matches could be found for the subject keyword. The focus of user query in such situations varies according to the type of the semantic matches of keywords. We have also developed templates for such queries. Due to the lack of space, please refer to [Lei’s KMi technical report] for details.

Dealing with complex user queries. For complex queries (which involve more than two keywords), the search engine needs to combine the semantic matches of each keyword together and construct queries for each of the combinations. A key operational problem is that in real world situations there can be a large number of matches and hence much more combinations. Rules are therefore needed to reduce the number of matches for each keyword. We used several heuristic rules, including, i) the subject keyword always matches class entities when there are more than two keywords involved in the user query, ii) choosing the closest entity matches to the keyword as possible, and iii) choosing the most specific class match among the class matches. These rules can significantly reduce the number of entity matches.

Step 3 – Refining User Queries. The system allows the user to reformulate his/her queries at a basic level and an advanced level. At the basic level, the tool allows the user to view the entities found and select those which should be used in the search, as shown in Figure 6. At the advanced level, the system allows the user to exploit the structure of the associated ontology to narrow down or broaden the search scope. We call it a knowledge-lens based refinement. Figure 7 shows the user interface. Hence, the system has developed a multi-mode interaction routine, which supports end users in making semantic queries by combining keyword-based search and view-based search.

Step 4 - Querying the Data Repositories. In this step, the system queries the underlying data repositories using the selected formal queries derived from the user query in the previous step. The search results are semantic data entities that satisfy those formal queries.

Step 5 – Ranking the Search Results. We have examined three factors that may reflect the relevance of the results to the user query, including the similarity, the domain context, and the query context factor. For the sake of simplicity, we treat all three factors as equally important in ranking at the moment.

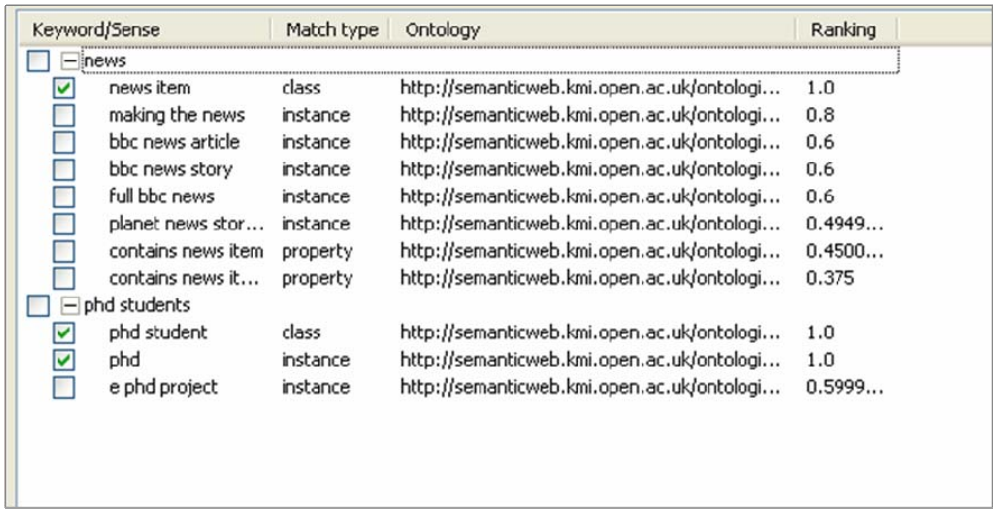


Figure. 6. Basic Refinement Support Offered by SemSearch

- The *similarity* factor measures the closeness of a match to the keyword syntactically (i.e., string similarity), which is derived from the text search engine employed. In future, we will add a mechanism to compute the semantic similarity of the match to the keyword.
- The *domain context* factor helps decide the closeness of the matches to the keyword from the specific domain point of view. For example, with the keyword "John", is the user more likely to mean the person John-Domingue than the person John-Smith? The system computes such ranking by looking into the popularity of the entity in the specified repositories.
- The *query context* factor takes the position of a query term and its expected match type

(i.e., class, instance, and property) into account when ranking the match results. Several heuristic rules are used to rank the matches. For instance, the subject keyword (i.e., the first keyword in a user query) is assumed to refer more often to a class than an individual entity. Thus, for a subject keyword, class matches would get higher ranking.

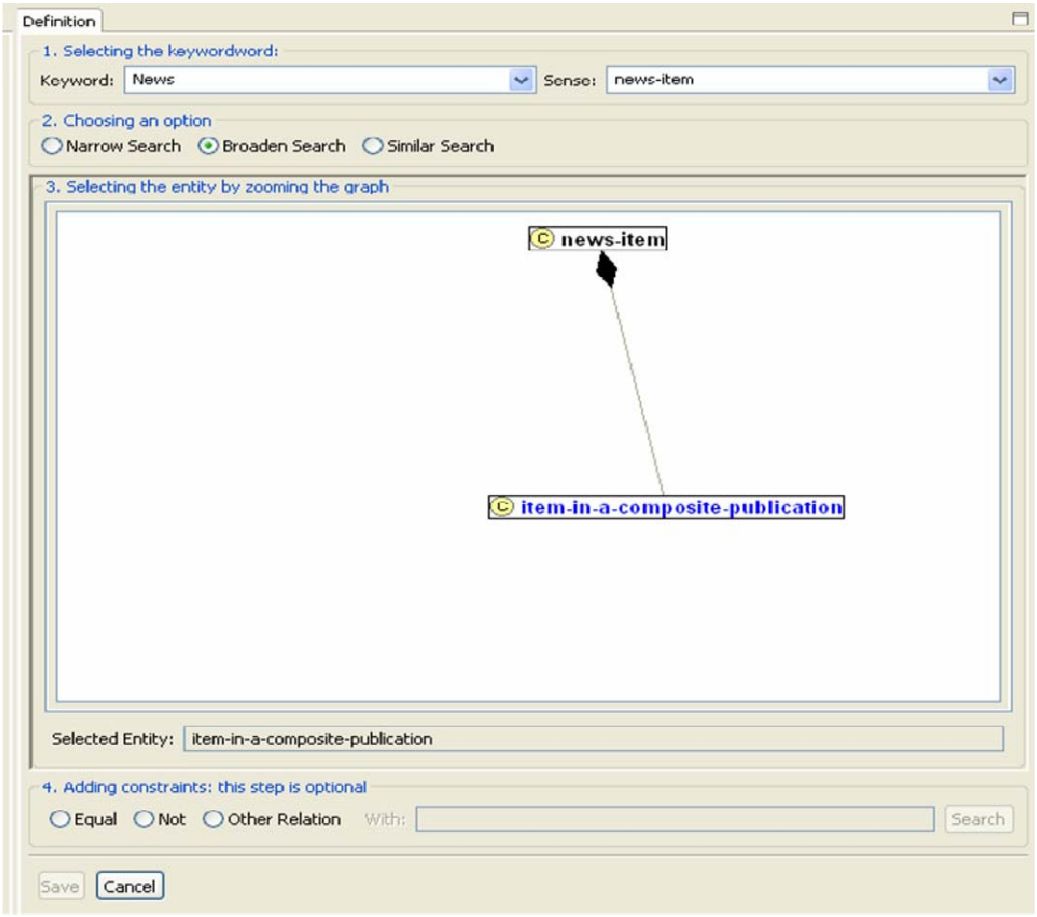


Figure. 7. Advanced Refinement Support Offered by SemSearch

End User Support. SemSearch offers a user interface component, which supports all the interactions required for formulating and refining user queries and presenting search results. Four major facilities are provided by the main user interface, including

i) allowing the specification of keyword queries;

ii) displaying the matches of the keywords;

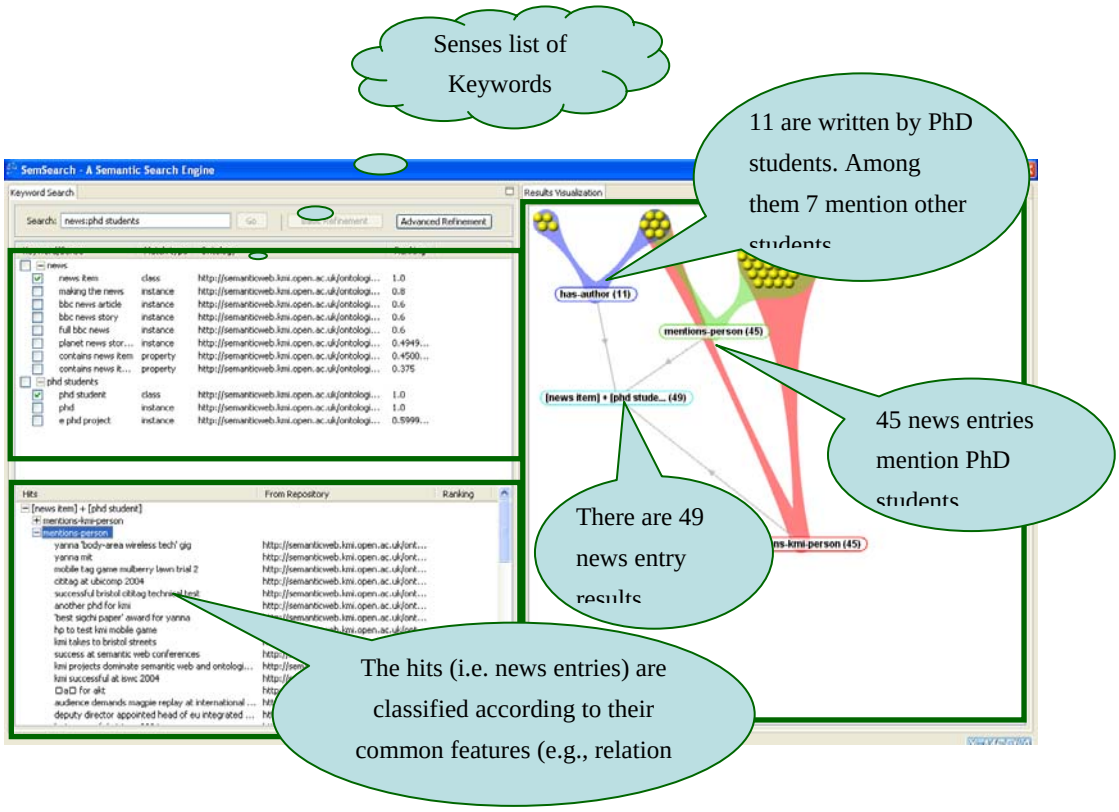
iii) allowing the ticking on/off keyword matches to refine user queries;

iv) displaying the final search results returned by the Sesame Query Engine.

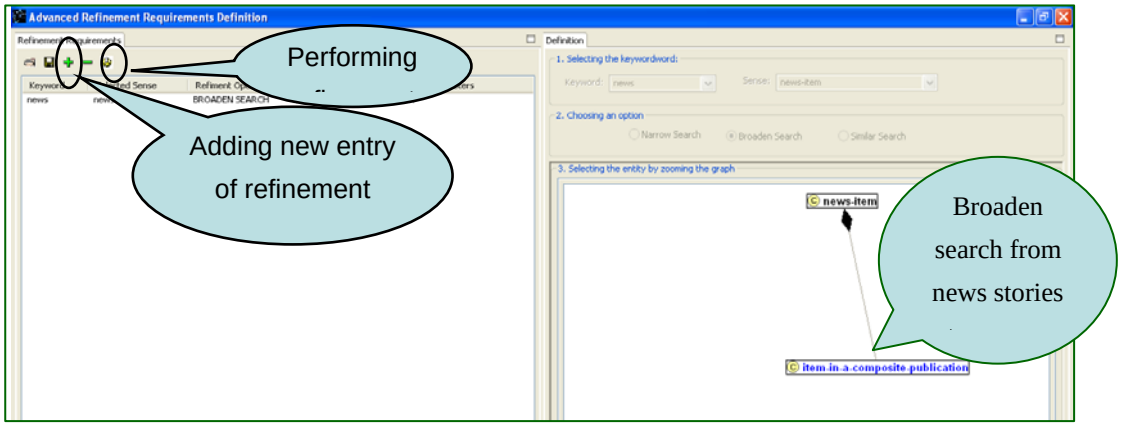
Tree-based views and graph-based views are generated to help users understand the search results. More advanced query refinement is supported by the concept of knowledge lenses, which supports a zoom-like facility in exploring the search space. It supports narrowing the search focus by going down to more specific classes (or topics) or broadening the search focus by going up to more generic classes. Below we present a walk through of a typical user session with refinement.

- Task1: Typing in query "News:phd students"

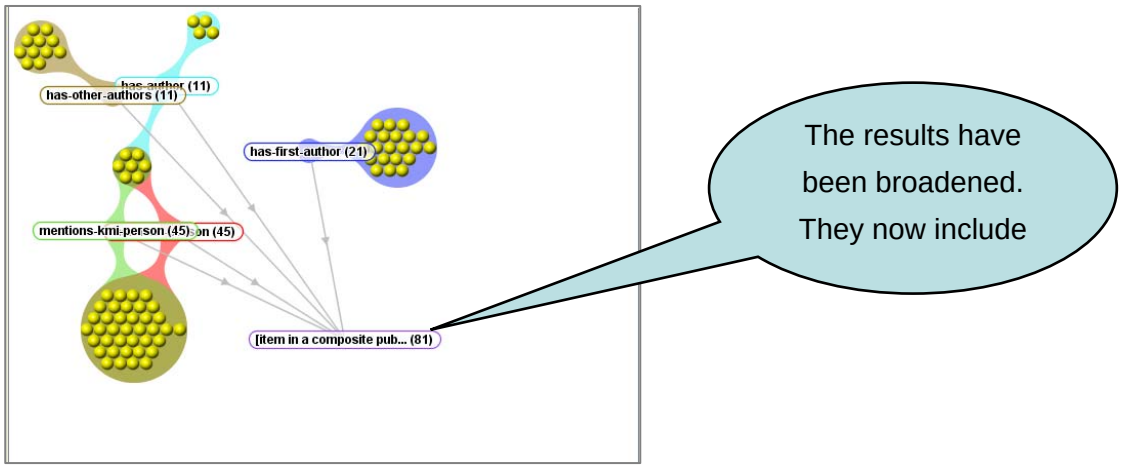
- Task2: Picking up the right senses found from the ontologies and repositories
  - ❖ The class *news-item* for the keyword news
  - ❖ The class *phd-student* for the keyword "phd students"
- Task3: Viewing the results of the basic search



- Task 4: Refining search results by broadening the search target (i.e. news) to a more generic class



- Task 5: Viewing the refined results



3.2.2 Meeting the X-Media requirements

In this section, we discuss how well SemSearch meets the X-Media requirements outlined in D4.1.

3.1 Interoperability

3.1.1 Interoperability between Tools and Tool Components - The current prototype of SemSearch is implemented in Eclipse and Sesame2, using the API provided by the X-Media Kernel. As such, it should be compatible with most of the other WP4 knowledge sharing components that operate on the X-Media Kernel.

3.1.2 Interoperability with the X-Media Architecture – SemSearch uses the AccessKnow API to interact with the X-Media kernel.

3.1.3 Interoperability with Relevant Standards – SemSearch searches RDF-based documents. Hence it works with RDF, RDFS, and OWL.

3.2 Performance

3.2.1 Response Times for Large Scale, Cross Media Search – SemSearch has not yet been tested on a large scale testbed. It is known that the query interpretation process can sometimes produce very large numbers of queries and that this can be problematic for performance speeds. One of the key issues that will be investigated in the future is how to deal with large-scale search environments.

3.2.2 Quality of Results – This needs to be tested in the evaluation phase.

3.3 Handling Heterogeneous Knowledge

SemSearch has not yet been tested with heterogeneous data. It searches OWL, RDF only. A multi ontology version is under development.

3.4 Fundamental Functions

3.4.1 Diversity of Users – SemSearch has a very simple mode for naïve users who can start by just typing one keyword but also supports the formulation of quite complex multi-keyword searches. The technology that underlies this flexible approach is the semantic interpretation of keyword queries. SemSearch is designed with general semantic search in mind rather than one particular user group.

3.4.2 Search Refinement – As described in the section above, SemSearch currently has two search refinement methods. One presents the user with a list of semantic matches from which they can pick the ones they wish to use in the search. The other allows "zooming" by allowing the user to browse up and down fragments of the ontology to select terms.

3.4.3 State of the Art Functions – of the functions listed in D4.1, SemSearch is most focused on developing ranking methods. As described earlier in the section above, we have explored three factors to rank the search results, namely similarity, domain context, and query context. In future, user profile will be included in the ranking loop in order to present the results better.

3.5 Innovative X-Media Functions – The X-Media functions that SemSearch aims to provide come in the category of Knowledge Lenses. It provides mechanisms for selecting and viewing facts. Facilities such as the visualization of results clusters help the user to gain an overview of the contents of the knowledge base to help users refine their search and reset the focus until it meets their needs.

Phase 1 deployment - Within phase 1, SemSearch will be deployed in the Noise Curves Analysis and Evaluation test bed described in section 5 of D13.2 [D13.2 2007]. It will be used, for example to search for facts using FIAT model name, segment, competitor's basket, performances etc. It is anticipated that this will require some small extensions of the tools. Specifically, because the use case is strongly

focussed on the noise curve data, the presentation of tables and graphs will have to be included (possibly presenting graphs as images). We will also need to investigate whether SemSearch needs to integrate directly with specialist services produced by the University of Llubjana, particularly the service for similarity search.

3.3 K-Search

K-Search is a semantic search interface capable of bringing together the worlds of semantic conceptual search and also keyword search. These search modalities can be utilised independently as well as together in a hybrid modality. These novel interaction methods allow flexible searching of knowledge which may only be partially represented in the domain ontology. Following is a technical description and then a discussion of how this helps aid the developmental aims of X-Media and its workpackages.

3.3.1 Technical Description

Storing Knowledge for reuse has two distinct types:

- **Structured**: knowledge is formally organised into a predefined rigid structure used for direct knowledge access. In the case of X-Media this structured store uses RDF and ontologies. Concepts in any initial document can be related through logical statements, e.g., *headsup_decision* for an application for a *technical_variance* on *engine_familyA still under investigation*, to an ontology, this allows the retrieval of all instances where *headsup_decision*s have been made for instances of *engine_familyA* have the vale *under_investigation*. Such storage enables: quantitative analysis, reasoning, automatic manipulation and direct access to contained knowledge. All these are central aspects of most X-Media technology; however, structured data storage has some limitations. Structured data storage can be inflexible regarding access to it is constrained to that of the storage mechanism itself; for instance knowledge within a document that is not represented in structured data is lost. Further to this, designing structured resources for holding knowledge is costly, and although covering the typical cases, may exclude specialist usage. Even the best ontology is unlikely to cover all user information needs, as these may change with time and in ways unexpected during the design.
- **Unstructured**: data is not coded into a predefined structure, but instead is indexed to allow free-text retrieval on a token basis. An example of unstructured data access is provided by search engines that index and retrieve knowledge from enterprise archives or the whole web. A user typically expresses their knowledge needs by

entering a query; the search engine uses its index to retrieve potentially relevant documents for the input query terms. While this approach provides a high degree of flexibility, as the data retrieved matches the query, there is no guarantee the retrieved result set contains all and only relevant data intended by the user. As a consequence, quantitative analysis is highly unreliable. Much of the effectiveness of this technique depends on the text source; engineers' reports are generally extremely succinct and the language very specialised, two conditions highlighted as critical for the success of traditional unstructured, keyword-based retrieval. Another limitation is in the retrieval of entire documents; manual reading of all documents' content is required to retrieve the specific knowledge and make optimal use of the results. It must be noted that although unreliable, with contextual information missing and issues of conceptual ambiguity, unstructured knowledge repositories still have a single advantage over structured approaches, that they can encompass everything within documents and not just pre-assigned concepts.

Structured and unstructured data both have advantages and disadvantages; one providing order and precision, the other flexibility of use. To take advantage of both, K-Search uses a dual store (K-Store) that simultaneously holds structured and unstructured knowledge. For unstructured information a normal keyword index is used, in this case SOLR, alongside a semantic index storing structured triples. K-Search differs from CSail, an approach where conceptual instances are keyword indexed; in our approach entire documents are indexed thus completing system coverage of documents content.

The semantic store facilitates higher level inference as well as additions to the underlying data without impairing functional effect. In particular stores such as 3-store[5], Allegrograph[6] and Sesame (used in this case) facilitate flexible query through the SPARQL[7] query language. This freedom along with the improved coverage of keyword approaches allows users a more flexible knowledge access than that offered by a single mechanism alone. In order to combine structured and unstructured querying K-Search performs three offline steps:

- indexing documents using keywords,

5http://sourceforge.net/projects/threestore/

6http://agraph.franz.com/allegrograph/

7http://www.w3.org/TR/rdf-sparql-query/

- defining a domain ontology,
- gathering structured knowledge using an ontology.

The dual storage K-Store supports more flexible interaction as the user is able to choose the most appropriate search mode for the task in hand, as described in the next section.

3.3.2 K-Search Functionalities

To take advantage of the double storage facility in K-Store, K-Search combines the flexibility of unstructured retrieval with semantic structure, making synergistic use of the strengths of both techniques, and supporting users in focusing on relevant issues with faster retrieval and more accurate results.

From a user point of view, structured queries must be formulated in a logical language that has to be learned and remembered. Conversely, unstructured retrieval has the advantage of being all encompassing - any term can be searched for independently of previous processing - and straightforward to use, - terms are simply entered into a (keyword) query. The interface of K-Search supports the user in formulating queries in whichever way suits their skills and information requirements.

The hybrid approach (HS) adopted in K-Search uses and fuses keyword search (KS) and ontology search (OS). The user interface of K-Search has been designed to support the composition of Hybrid queries as well as quick changes from one mode to another, KS only or OS only (see Fig.8).

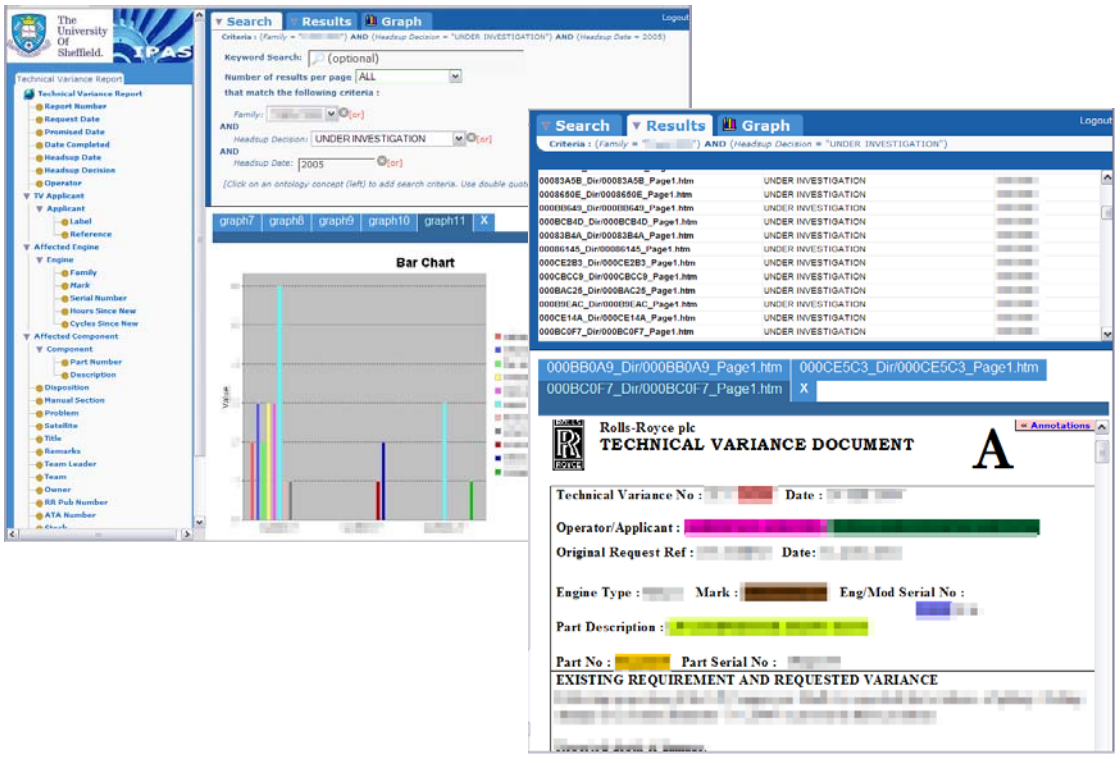


Fig. 8. K-Search querying interface and visualisation of results. Please note that portions of the text have been deliberately obscured for reasons of confidentiality.

When a query is performed, the result set contains the reports where the concepts and the keywords in the query co-occur. The set is displayed as a list in the mid-right panel of the interface; each item in the list shows the name of the document and the values of the fields used for OS. Individual reports are displayed on the bottom right on request (by clicking on the file name for a list item). Multiple documents may be opened simultaneously, each displayed in a different tab. The original layouts of the documents are maintained (see Fig.8). Annotations are made evident through colour highlighting, and are the means to advanced features or services: for example, clicking on a concept results in a query expansion with the selected term. One of the advantages of structured data is the support for quantitative analysis of the retrieved result set using graphs and charts. K-Search allows users to selected concepts from the ontology to display the retrieved set with respect to the selected parameters. The graph in the centre of Fig.8 first refines the query by restricting results returned to *request_date* equal to *2005* only, then draws a plot showing the *operators* for each (concept) *engine_mark*. Each plotted *operator* (each bar in the example) is active and may be clicked on to focus on the sub-set of documents that contains that specific occurrence.

3.3.3 Meeting the X-Media requirements

In this section, we discuss how well K-Search meets the X-Media requirements outlined in D4.1 (numbering refers to the section headings in D4.1).

3.1 Interoperability

3.1.1 Interoperability between Tools and Tool Components: K-Search is Java-based, and uses JavaScript and CSS for the user interface. It is therefore platform-independent, and should be compatible with other Java-based tools for WP4. Further, the web-based interface means that it is easily used over a network, with the client requiring only a relatively recent version of a web-browser.

3.1.2 Interoperability with the X-Media Architecture: Although K-Search was developed independently of the X-Media project it should still be able to communicate with the X-Media kernel. K-Search **uses RDF to store the knowledge;** query results are retrieved from its triple store **using SPARQL and** the actual documents from which each triple is retrieved is what is displayed using K-Search's user interface (shown in Fig. 8). In order to interface with other X-Media tools K-Search will read from and write to the dedicated triple stores for X-Media via the kernel. The extract below shows part of the result set for the query described in section 3.3.1, using a simple XML document.

```
<?xml version="1.0"?>
<search_results>
    <search_details id="0">
        <search_type>concept_search</search_type>
        <hits>28</hits>
        <output_concepts>
            <concept>has_file_location</concept>
            <concept>has_heads_up_decision</concept>
            <concept>has_heads_up_date</concept>
            <concept>has_family</concept>
        </output_concepts>
    </search_details>
    <documents>
        <document>
            <has_file_location>path_to_file</has_file_location>
            <has_heads_up_decision>UNDER INVESTIGATION</has_heads_up_decision>
            <has_heads_up_date>xx/xx/2005</has_heads_up_date>
            <has_family>engine_familyA </has_family>
        </document>
...
```

3.1.3 Interoperability with Relevant Standards: K-Search searches RDF-based documents. Hence it works with RDF, RDFS, and OWL. K-Search outputs results using RDF/XML triples.

3.2 Performance

3.2.1 Response Time for Search: K-Search has been tested on a corpus of 18,097 Event Reports provided by Rolls-Royce. A hybrid search that retrieves 20 matching documents over one corpus, for example, displays the result list to the interface in approximately 2 seconds, running over a network with up to 12 simultaneous users. A keyword search returning 161 and 4181 documents take approximately 2s and 57s respectively. K-Search was evaluated by Rolls-Royce users: it was judged fast or very fast in executing the query allowing a quick task completion by 98% of users.

3.2.2 Quality of Results: The results of in-situ user evaluations to date show that users understand the concept of hybrid search implemented in K-Search. There is a significant decrease in user time and effort required to retrieve knowledge from the corpuses used for the current implementations, compared to existing methods for searching over these corpora. Results of user and system evaluations for K-Search are, overall, very positive, showing very high user acceptance; [Bhagdev et al., 2007, Lanfranchi et al., 2007] provide more detail on the application and the evaluations carried out. K-Search user evaluation allowed to asses the system reliability in retrieving relevant documents; the reliability was high with 82% judging K-Search reliable or highly reliable. A more in depth in vitro evaluation conducted, with the aim of showing that Hybrid Search can provide better results than the pure keyword based or semantic search, by combining their reciprocal strengths. The evaluation was done considering a set of 21 topics generated on the basis of observed tasks, sequences of user queries recorded in the event corporate database or as elaboration of direct input from users (i.e. examples of their recent searches). Hybrid Search reports very high precision (same as ontology-based search, +51% with respect to keyword-based search), and the highest recall (+46% with respect to keywords and +109% with respect to ontology-based search). F-Measure is +49% with respect to keywords and +55% with respect to ontology-based.

3.3 Handling Heterogeneous Knowledge

3.3.2 Images, Text and Raw Data: K-Search has been built to focus upon textual data, however, it may be used to search images and raw data for which metadata exists in textual format. K-Search currently has an implementation that retrieves images for which metadata stored in textual format satisfies query criteria.

3.3.3 Multiple domain models: K-Search performs semantic searches by making use of ontologies (in OWL/RDF format). The current implementations load a single domain ontology each; the ability to make use of multiple ontologies within a single implementation is currently being considered, in order to support searching across different corpuses or domains using a single front-end.

3.4 Fundamental Functions

3.4.1 Diversity of Users

K-Search supports different users and tasks by supporting both document and knowledge retrieval and by providing an interface to perform three different types of queries: : (i) pure semantic search via unique identification of concepts/relations/instances (e.g. via *URIs* or unique identifiers); (ii) keyword-based search performed on the whole document and (iii) keyword-in-context search.

Different implementations of K-Search are being evaluated using independent domain ontologies, with users from multiple departments in R-R, who have differences in knowledge requirements, based on the roles performed and the tasks being performed. K-Search may be extended to other user types and domains, by importing ontologies to match each domain as required.

3.4.2 Search Refinement: K-Search has two main methods of refining the queries. First of all the user can use the search tab to insert more conditions, both selecting additional concepts from the ontology or inserting new keywords, constraining further the result set. Moreover when a search has been fired and the results returned, the user can refine the results by clicking on one of the highlighted annotations or by producing a graph of the results.

3.4.3 State of the Art Functions: Concerning the functions listed in D4.1, K-Search is mostly focusing on the Advanced Search features.

Security: K-Search is accessed using a log-in screen. Security based on user profiles is currently implemented such that access is provided to the entire corpus in an implementation. A stricter security model would require the development of an

additional module to filter results within a corpus based on user profiles attached to log-in information.

Relevance ranking: Traditional keyword indexing and document ranking is done in parallel with ontology-based annotation, implemented within K-Search using Nutch, because of the high quality indexing it provides, and the ability of Nutch to exploit the strategies used by search engines. Semantic search in K-Search retrieves exact matches, so that all results are equally ranked. The final result set contains the intersection of the results of the (separate) keyword and semantic searches.

Internal and external database searching: K-Search provides support for querying using SPARQL and SERQL, to retrieve knowledge currently from 3-store and Sesame. K-Search is however able to make use of any suitable (RDF) triple store.

Wildcard searching: K-Search provides support for wildcard searching in the keyword-based search, using the * character.

Advanced search features: K-Search supports the execution of Hybrid Search queries (as described above) using a form-based approach. K-Search aims to provide a mixed approach to searching based on a combination of keyword-based and ontology-based search. The method is designed to overcome some of the limitations in the pure semantic search that may suffer from unavailability of metadata

3.5 Innovative X-Media Functions

K-Search aims to provide the X-Media functions of Knowledge Lenses and Perspective Building.

3.5.2 Knowledge Lenses: It provides mechanisms for querying RDF facts and selecting and viewing query results. The visualization model allows to present the query results according to a number of dimensions: as a list of ranked documents, as aggregated metadata (e.g. via graphs or charts) with associated provenance, etc. depending on a tasks goal (i.e. document retrieval Vs knowledge retrieval), thus providing different knowledge lenses on the facts stored in the knowledge base. The users can use the graph building functionality to switch knowledge lenses, choosing to view the results plotted accordingly to their needs.

3.5.5 Perspective Building: The ontologies used for semantic search, combined with a terminology recogniser, help to provide context and perspective for especially

inexperienced users, by providing structure to the search and aiding users in identifying appropriate input for building and refining queries.

3.4 XXploreKnow!

XXploreKnow! is an ontology-based application, which supports the exploration and retrieval of documents and facts contained in knowledge bases. In summary, research contributions are reflected mainly in four distinctive features. Firstly, it is capable of translating an unrestricted number of keywords to a ranked list of interpretations, of which the user can choose to obtain the intended formal query. Secondly, it combines factual search with document retrieval. This document retrieval is based on resource descriptions with a degree of expressiveness not available in state-of-the-art tools. They can be exploited to address complex IR tasks, where the user can pose expressive queries which might involve general document metadata, document structure as well as expressive description of the document's content. Thirdly, the implemented data filtering mechanism allows the user to create customized views on the KB, i.e. knowledge lenses on the KB. This mechanism can be declaratively specified in the form of policies, which incorporate the different provenance-related knowledge as well as the process knowledge available in X-Media. Finally, XXploreKnow! features personalization functionalities, which are based on a rich ontology-based model of the context.

3.4.1 Technical Description

XXploreKnow! is an ontology-based application that supports exploration and retrieval of ontologies, semantic data contained in ontologies, as well as annotated documents. Besides, it features personalization functionalities and a data filtering mechanism to provide customized knowledge lenses to the semantic data. In particular, semantic data are elements of RDF(S) /OWL ontologies such as concepts, properties and individuals. They are manually added to the system using the annotation tool discussed previously or automatically extracted from documents of various formats and web pages by knowledge extractors. We will continue with a brief presentation of the main UI components featured by XXploreKnow!.

- *Concept Hierarchy View*: A tree-based visualization of concept hierarchy
- *Schema View*: Visualizes the T-Box of an ontology. The user can explore the T-Box by expanding and/or collapsing concept nodes in order to add neighbours or filter neighbours from the view, respectively. This view is linked to the concept hierarchy view such that in case a user clicks on a

concept in one view, the same concept is highlighted in the other view. This way, the user always sees the position of the concept both in the hierarchy and in the schema respectively.

- *Concept Outline View*: Shows the facets of the concept currently selected on the schema and concept hierarchy view. In particular, it shows all properties (as implied specified by domain / range restrictions and complex concept descriptions) and all individuals of the concept.
- *Query Definition View*: Shows a multi-tree based representation of a SPARQL query (which might represent a graph pattern). It facilitates the specification of the formal SPARQL query by enabling the user to enter keywords or to drag-and-drop ontology elements shown in the views discussed previously. In the case of keywords, an interpretation process is triggered which makes use of domain knowledge in the underlying ontology to translate the entered keywords to a ranked list of SPARQL queries (the possible interpretations). From this ranked list, the user can choose the intended meaning. This way, the user does not have to cope with the syntax of the formal query. Tran et al. discuss further details on the interpretation of keywords in [Tran07int].
- *Fact Result View*: This view shows the results of the SPARQL query. Whereas the constituent table view shows all the bindings to variables of the query, the corresponding tree view shows property values for each of these bindings.

Figure 9 below illustrates these UI components and shows a possible interaction between the user and the system. At the beginning (STEP 1), the user enters keywords which are processed by the interpretation service. The result is a list of possible queries. The user then selects the intended query. Subsequently, the system visualizes which portion of the schema the query corresponds to. Depending on the action performed by the user, i.e., either he activates the “search" or the “xxplore" button (STEP 2), subsequent interactions are either further exploration of the schema or inspection of the search results. With “xxplore" (STEP 3a), the user can expand nodes shown in the visualization of the query to traverse to neighbouring elements. During this exploration, the user can drag and drop elements from the schema view and the concept outline view to the query definition view (STEP 4) to further refine the query. With “search" (STEP 3b), the SPARQL query is sent to the underlying

query engine. The results, which may contain also inferred facts, are then finally shown to the user in the fact result view (STEP 5).

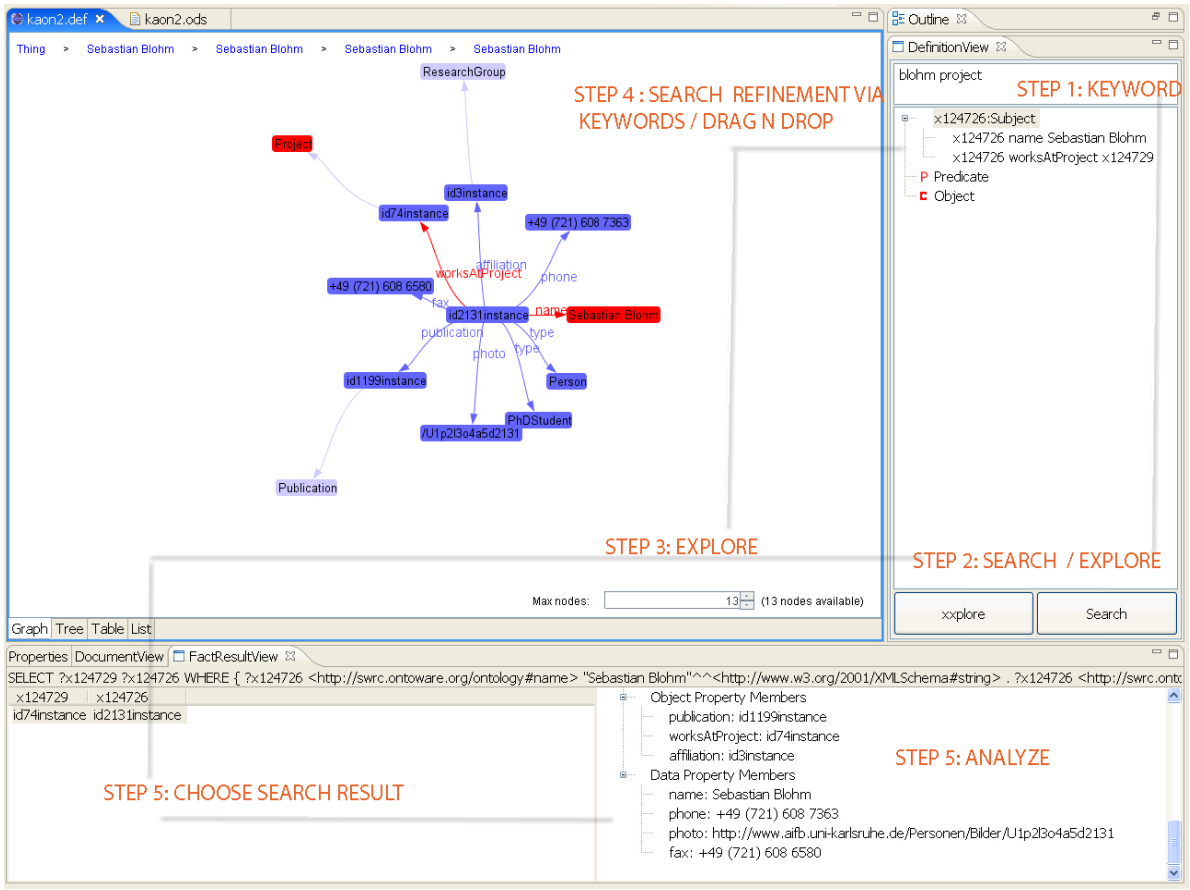


Fig. 9 XXploreKnow!

Besides this factual search, a combination of semantic search and classical index-based search is also supported. This means that besides the exploration and retrieval of facts contained in the KB as discussed above, the user can also retrieve documents based on keywords and document annotations. The latter refers to expressive descriptions of documents that capture general metadata, structure as well as the content of the document. These descriptions are also stored in the KB in the form of facts (A-Box assertions). Whereas document retrieval with keywords runs against an inverted index (Lucene[8]), document retrieval based on resource descriptions are specified in terms of SPARQL queries, which run against the KB. Further details on document retrieval based on expressive resource descriptions can be found in [Tran07res]. The next development of document retrieval in XXploreKnow! will be a

[8] http://lucene.apache.org/

combination of classical index-based techniques with expressive resource descriptions.
A further extension will extend XXPloreKnow! to retrieve multimedia documents, in particular images.
In particular, UI components that support document retrieval are the following:

- *Resource Result View*: Show the list of ranked documents that are result either of the keywords query or the SPARQL query based on resource descriptions.
- *Resource View*: Present the document to the user that is selected in the resource result view.

In order to provide personalized knowledge lenses, XXploreKnow! features the enactment of data filters, i.e., declarative policies that are defined with respect to provenance-related information as discussed in WP1 deliverables. In particular, these policies are based on the notions of trust and tasks. They are defined as concepts of a policy ontology. These concepts build on the notions already defined in the provenance ontology, i.e., meta-information to the facts such as the agent, the source, the confidence degree and the process. Note that the notion of process is used to represent tasks and bears information as to what agent (e.g. knowledge extractors, employees) are involved in the task and what sources (information providers) are relevant for the task. As an example, a data filtering policy for Fiat can be specified, which specifically says which employees, which information sources and which knowledge extractors are trustworthy / reliable. An extension of this policy captures the notion of task, e.g. competitive analysis policy, which further restricts the extractors, the employees, the sources of the Fiat policy to the ones involved in (relevant for) the competitive analysis. Given a specific user and task, these policies can be used by the underlying reasoner (e.g. KAON2) to infer which sources and which agents are relevant, thus filtering the data accordingly. As a result, the user can browse and search a virtual KB that contains only facts that are relevant for the competitor analysis task**.** Further details on how to specify data filtering policies based on meta-information can be found in [Tran08Fil]. UI components in XXPloreKnow! that support this data filtering are:

- *Task View*: Support the definition of task policy that extends an organization policy such as a Fiat policy.

- *Task List View*: An activation of a task triggers the data filtering mechanism which computes a virtual ontology for the activated task, i.e., provide a knowledge lens to the KB.

Currently, personalization functionalities are implemented for XXploreKnow!. They are realized on the basis of an ontology-based context model which encompasses several dimensions such as the user model, the resource model, the task model and the environment model. For instance, interactions between the user and the system are tracked to develop the user model. Information contained in the context model is exploited by adaptation rules. For instance, when a user is working with a document, an adaptation rule might be triggered, resulting in a list of recommendations consisting of further documents which are possibly relevant w.r.t. to the current context. Further information on such adaptation rules are personalization functionalities as discussed in previous work [see Tran07adap].

3.4.2 Meeting the X-Media requirements

XXploreKnow! is developed to address the following top level requirements in WP4:

- *Interoperability with the X-Media Architecture:* It is based on the X-Media Kernel. In particular, AccessKnow! of the Kernel is used to access the underlying KB.
- *Diversity of Users:* Simple keyword search interface is provided for the casual user and formal queries can be used by an expert. Also, different user policies for data filtering can be defined to meet the information needs of different user groups.
- *Search Refinement:* As discussed above, query refinement is supported either through the addition of further keywords or via drag-n-drop.
- *Knowledge lens:* Supported through the discussed policy-based data filtering.
- *Process Support*: Process is considered as a task, which is used to filter information not relevant to the task.
- *Perspective Building:* Both the data filtering and the personalization functionalities can be used to adapt information provision to the context.

Currently, XXploreKnow! is customized to target the Competitors Scenario Forecast Use Case. With respect to D13.2, the relevant insertion points are TIP1, TIP3 and TIP8A and TIP8B, TIP9.

4 Subtask 4.2.3 Knowledge Lenses - Presentation of Results

Knowledge lenses in the X-Media context, for the presentation of the results of semantic search, encompass tools that support customisation based on users' profiles and specific knowledge and task requirements. In practical terms, this means the provision of intuitive filters that present knowledge retrieved from different perspectives, providing context that enriches the user experience.

The Issue Resolution use case may be used to illustrate how knowledge lenses can be used to support the user in knowledge management: during the evidence collection stage clustering methods may be used to categorise data, grouping assertions retrieved based on potential causes of an issue defined, for instance. Table or tree lenses in this case provide alternative methods for grouping data; a lens may be applied to the overall data set to allow users to focus on a sub-set of the data, filtering out data based on the level of confidence attached to each assertion retrieved, for instance. An alternative lens could filter data based on media type.

Similarly, a knowledge lens in the Competitor Analysis use case could filter data based on the type of the feature of current interest, e.g., cars with air bags for both passenger and driver doors.

4.1 K-Views

A number of options for the presentation of search results were identified following the analysis of user requirements for the Issue Resolution use case. The X-Media vision demonstrator [refer D4.1 and D12.2] illustrates the options explored for the presentation of results for this scenario, starting with tables and trees, and expanding to include network graphs, parallel co-ordinates, a facetted browser, timelines and a geographical visualisation metaphor. A focus session guided by the version of the demonstrator described in D12.2, with end users at R-R from multiple departments who work on different aspects of Issue Resolution cases allowed further refinement of users' knowledge management requirements, and helped to pinpoint aspects of each option that mapped to users' methods for problem solving. We discuss here the

analysis of user requirements leading to the option developed: the network graphs used to visualise the results of searching.

The evidence collection and analysis stages in the vision demonstrator use two main methods for clustering/categorising data:

1. The visual representation of classes derived from the causes ontology, using an interactive method to drag assertions into each box representing a class, in the version of the demonstrator illustrated in D4.1. This led to the idea of a knowledge cloud in Fig. 10, the visual representation of all elements in a class, using summaries of the assertions retrieved,

2. A *causes* trees to cluster evidence on each node of the tree, based on the contribution of an assertion for or against the hypothesis that the node represents the root cause of the issue identified.

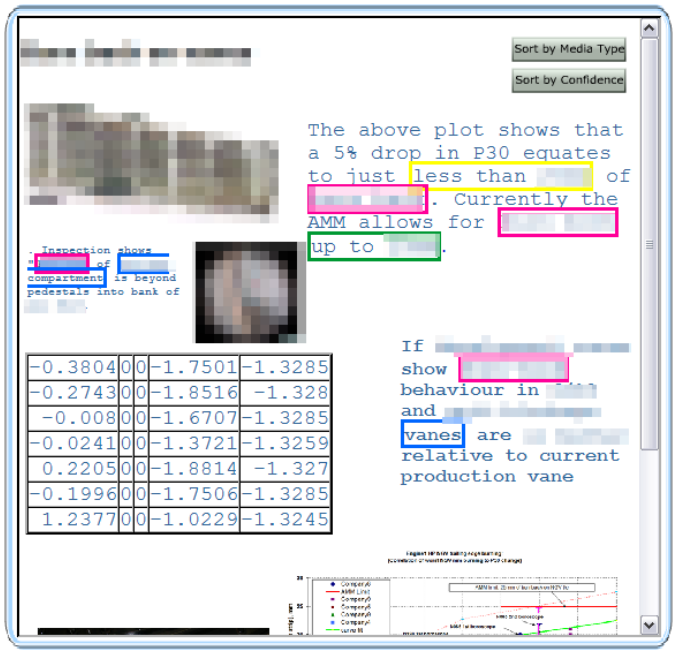


Fig. 10: A *knowledge cloud* displaying previews of the contents of assertions contributing to the assertion that the root cause of the issue being investigated is "Cause IJK". Please note that the collection of documents displayd do not map to a specific use case. Further, portions of the image have been deliberately obscured for reasons of confidentiality.

Both methods specified, for each assertion retrieved, whether it provided positive or negative evidence for the assertion concerned. Further discussions with target users recognised the limitations posed by the use of the absolute values *PLUS* and *MINUS* only ; using a (percentage) scale to display the degree to which each assertion

contributes to (51-100%) or detracts from (0-49%) a root cause provides a richer representation of the knowledge contained in each assertion.

A rooted, top-down tree was used to represent the causes tree in the preliminary versions of the vision demonstrator. Exploring this metaphor further led to a representation using a rooted graph, but in network form, in order to display, in addition to the structure of the causes ontology used to categorise knowledge, the relationships between assertions across multiple nodes in the tree. As is done for data annotation in tools such as AKTiveMedia[9] [Chakravarthy et al. 2006] colour highlighting is used to provide an intuitive method for mapping nodes in the network graph to concepts in the ontology. To aid users in determining areas to focus on during data analysis (relative) node size was then mapped to the confidence that a node represents the root cause of an issue being investigated, weighted against other nodes for which evidence has been found that contributes to the hypothesis that these nodes, rather, represent the root cause of the issue.

Fig. 11: The screen shot on the right (from the vision demonstrator) shows the contents of an assertion on request. The graph on the left uses relative size to indicate the probability that the "Cause QRP" (with the dark tan background) is the root cause of the issue being investigated. Please note that portions of the image have been deliberately obscured for reasons of confidentiality.

Close collaboration with end users ensured the design of the prototype for presenting the results of semantic search maps to the ways in which users model information. The next section provides a description of the prototype being developed based on the analysis of user requirements and the user-centred design process followed.

[9] http://www.dcs.shef.ac.uk/~ajay/html/cresearch.html

4.1.1 Technical Description

The presentation tool is built using the *prefuse* visualisation toolkit [Heer 2005]. *Prefuse* provides support for data input using readers for GraphML (an XML format), *prefuse* data tables built from csv files, for instance, and from databases such as MySQL, with inbuilt support for querying using SQL.

A more obvious choice for a graph visualisation library would be TouchGraph[10] (TG). However, in addition to the benefits of *prefuse* (detailed in [Heer 2005]), TG is now being developed on a commercial basis, so that there is significantly lower support for development using (older opensource) TG libraries than for *prefuse*, which is currently fully open source and continues to be updated on a regular basis.

Fig. 12 shows a network graph drawn using *prefuse,* with central nodes colour-coded to match the causes ontology shown on the far left in Fig. 11. Concept nodes, which encompass knowledge clouds (as in Fig. 1**Error! Reference source not found.**0), have node size weighted based on the relative probability that a concept represents the root cause of the issue under investigation. *Cause QRP* can be seen in the graph in Fig. 12 to have the strongest probability of being the root cause of the issue under investigation, based on the knowledge retrieved to date and user interaction with this knowledge (that brings expertise to data analysis). Two additional filters are currently available; based on feedback from end users at RR on the data attributes commonly studied when searching for information during Issue Resolution. These include, but are not limited to, the confidence that a specified assertion contributes to or detracts from a hypothesis formulated on the root cause of an issue, and the recency of a report.

Each leaf node in the graph is displayed using a thumbprint that provides a preview of the contents of the assertion it represents. Hovering the mouse over a node brings the focus to it: this changes the background of a concept node to red, or paints a red border round a leaf node (representing assertions), in addition to highlighting (in pale orange) the links to and the focus node's immediate neighbours. A leaf node for the concept *Cause DEF* is highlighted in Fig. 12 this pops up extra information, showing the confidence that this assertion contributes to this potential root cause to be 69%, i.e., it provides evidence that the knowledge the assertion contains contributes positively to this hypothesis.

Using dynamic slide queries as described in [Ahlberg 1994] filters for confidence level and report (generation) date are used to highlight each leaf node meeting the

[10] http://touchgraph.com/

current filter criteria using a green ring. An alternative implementation could hide or fade out nodes that do not meet query criteria; this however removes important contextual information that is still relevant to data analysis, and is therefore not a preferred option. Allowing users to set preferences for either method for visual feedback during querying will be explored during the user evaluations to follow.

Finally, double-clicking on a leaf node retrieves the source document from which an assertion is retrieved, using external applications, e.g., Adobe Reader for PDF documents and a web browser for HTML.

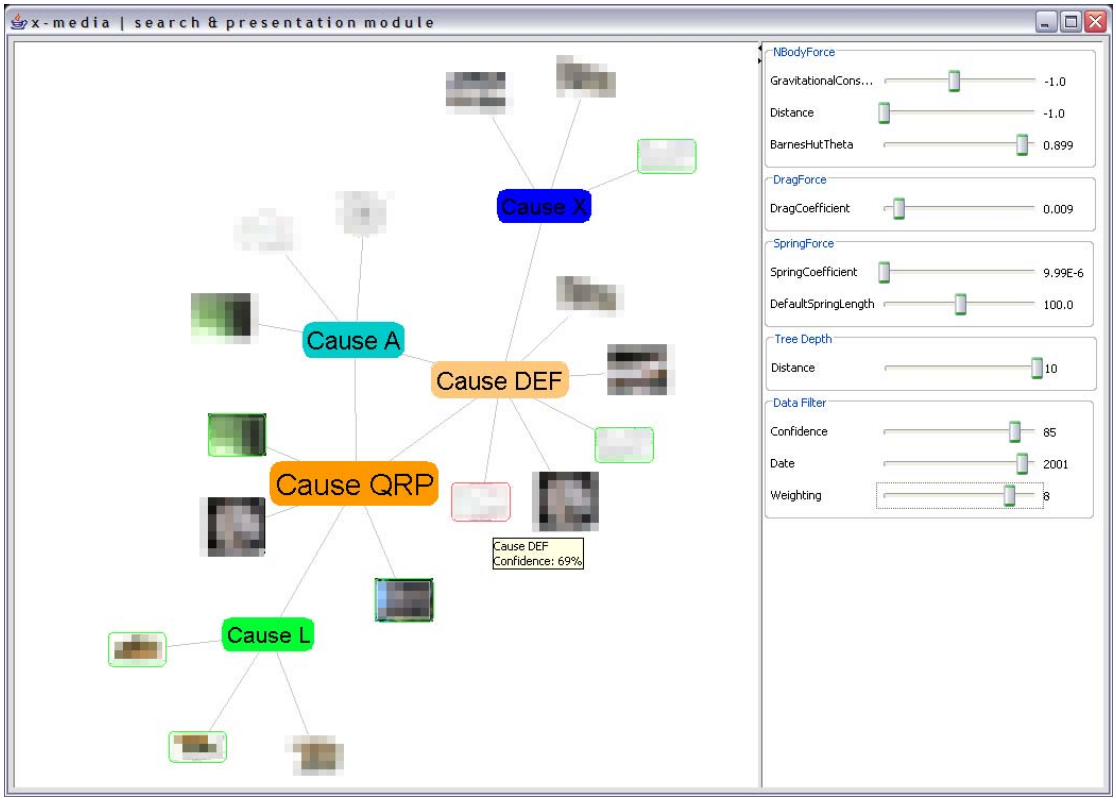


Fig. 12: A network graph drawn using the *prefuse* visualisation library, showing the relationships between concepts in a causes ontology, and thumbprints providing a preview of the assertions retrieved that contribute to the hypothesis for each concept being the root cause of the issue under investigation. Please note that portions of the image have been deliberately obscured for reasons of confidentiality.

4.1.2 Meeting the X-Media requirements

TIPs 3 and 5 for the Issue Resolution use case involve data analysis for the formulation and verification of hypotheses for the root cause of the issue being investigated. Both stages require the use of knowledge lenses to provide alternative perspectives on search results, customised based on user profiles and information requirements. Users require support for exploring the evidence collected from legacy

data, in addition to new evidence obtained due to the creation of new documents (e.g., test results). During hypothesis formulation it may also be necessary to (manually) categorise data into classes defined in the domain ontology, where automated classification has been unsuccessful, and/or to reorganise the evidence collected to reflect users' understanding of the knowledge it contains, in addition to setting confidence values for the assertions retrieved.

The knowledge presentation tool being developed by Sheffield, *K-Views*, provides a simple, graphical method for exploring data. The previews of the assertions retrieved, organised according to the ontology loaded, allow users to quickly obtain an overview of the evidence collected, in addition to the relationships between data. That a single assertion may be assigned to multiple classes is obvious from the graph in Fig.12, suggesting to users the possibility of relationships between the potential causes of the issue under investigation. The supplementary information provided by the confidence levels aids users in determining to which degree each assertion contributes to the strength of a particular hypothesis. The concept of knowledge lenses, implemented using dynamic query sliders, allows users to quickly filter data to highlight assertions that meet criteria specified.

The following focuses specifically on the X-Media requirements (the numbering comes from D4.1) for:

3.1 Interoperability

3.1.1 Interoperability between Tools and Tool Components: K-Views is developed using Java, and the open-source *prefuse* visualisation toolkit, which uses Java2D. It is therefore platform-independent, and should be compatible with other Java-based tools for WP4.

3.1.2 Interoperability with the X-Media Architecture: K-Views is still under development; in its final form it will read in search results returned using RDF triples via the X-Media kernel, using Fresnel lenses as in LENA to interpret the content and visualise the results, customised based on requirements.

3.1.3 Interoperability with Relevant Standards: Input to K-View is in RDF/XML format.

3.2 Performance

3.2.1 Response Times for Large Scale, Cross Media Search: Tests for system response for the visual presentation of the results of search have not yet been performed; these require the results of search, to be retrieved via the X-Media kernel, which will be available closer to the end of the integration phase. Although K-Views is expected to read in large amounts of data (from search results), for usability reasons (good system response and low user cognitive load) the amount of data that will be displayed on the screen at a time is restricted (by folding sub-trees or filtering out less relevant data, based on users' information requirements).

3.2.2 Quality of Results: a useful measure of quality for the visualisation tool requires user evaluation, as this is dependent on users' ability to interpret the presentations effectively.

3.3 Handling Heterogeneous Knowledge

3.3.1 Dynamic and Uncertain Data: K-Views does not handle dynamic and uncertain data directly. However search results returned should include confidence levels set for each assertion retrieved, and this information is displayed along with other data properties using the network graph. Further development of the tool may consider interactive editing of confidence levels, feeding this information back to the X-Media data stores via the kernel.

3.3.2 Images, Text and Raw Data: K-Views provides previews of assertions retrieved using thumbprints. However the actual data represented by each assertion is handled by external applications registered to handle the source documents from which the assertions are retrieved.

3.3.3 Multiple Domain Models: K-Views is to model graphs based on domain or other relevant ontologies (e.g., a causes ontology for Issue Resolution), input in RDF/XML format.

3.4 Fundamental Functions

3.4.1 Diversity of Users: K-Views is being developed with a focus on the requirements of end users gathered for the Issue Resolution use case, to support knowledge seekers in exploring query results. It is envisaged that it will be customisable and extendable to knowledge exploration in other use cases.

3.4.2 Search Refinement: K-Views provides simple methods for filtering search results based on data properties, and provides multiple perspectives on data using knowledge lenses. It is however not a search tool and does not modify the underlying result set.

3.4.3 State of the Art Functions:

Security: K-Views reads in search results, which are expected to have already had security restrictions applied to data retrieved. Therefore security restrictions are not directly handled.

3.5 Innovative X-Media Functions

3.5.2 Knowledge Lenses: Users will be provided with mechanisms for visualising search results from different perspectives, by applying filters to the data, focusing on sub-sets as required, and highlighting different properties of data, in order to weight hypotheses based on the properties most relevant to issue resolution.

4.2 The CORPORUM Summarizer

Text Summarisation denotes the automated production of summaries by a computer program. In environments where both the production and consumption of large amounts of text documents is essential, Text Summarisation can be a means of achieving a swift overview of the content in a certain electronic document without having to download/open and read it. As with manual summaries, automated summarisation is in most cases not a replacement for actually reading a document, but should rather be regarded as a decision help whether to read a document in question or not, and thus can ease the load on knowledge workers.

According to [Hovy and Lin 1997], there are two ways to view text summarisation either as text extraction or as text abstraction. Text extraction means to extract pieces of an original text on a statistical basis or with heuristic methods and put together it to a new shorter text with the same information content. Text abstraction is to parse the original text in a linguistic way, interpret the text and find new concepts to describe the text and then generate a new shorter text with the same information content.

The main focus in X-Media will be the presentation of summaries along a timeline in order to reflect the development in, e.g., an issue resolution case. The main challenge in this context will be to identify new versus given information across multiple

documents in order not to present identical text parts in different summaries that are presented to the user.

The scientific evaluation of the quality of automatically generated summaries is not straight forward, as quality in this context is rather subjective and to a high degree dependent on the recipient's background knowledge of the matter in question. However, an evaluation methodology, based on reference summaries has been proposed by [Hassel and Dalianis 2005] and evaluated in for example [deSmedt et al. 2005]. Here, a number of test users manufacture an extraction based summary each by choosing a certain number of sentences to be included into the summary. Based on the user-generated summaries, a *reference summary* is generated, containing the sentences chosen by the most users. This reference summary then serves as gold standard when evaluating the automatically generated summaries. In X-Media, such *intrinsic* measures (focusing on the output texts only) of the summarisation tool will be used in Phase 1 of the project, whereas *extrinsic* evaluation methods (focusing on the degree of assistance in user tasks) will need to be devised once the system is put to use in the end user environments

4.2.1 Technical Description

The summarisation tool proposed X-Media is based on CognIT's CORPORUM Summarizer [Bremdal 2000]. This tool is an extraction-based summariser, developed within the OnToKnowledge project, building on CognIT's CORPORUM natural language processing tool OntoExtract [Engels and Lech 2003]. OntoExtract is an NLP tool, which analyses documents and extracts the core concepts and named entities as well as associations between them, thus establishing a lightweight concept graph. This semantic representation is then used as a basis for the choice of sentences to be included in the summary.

As mentioned above, the main focus of the X-Media Summariser will be the presentation of extracts of document on a given incident (i.e., issue resolution case) along a timeline, thus providing an overview of the development in the case, as illustrated in the mock-up below (Figure 13).

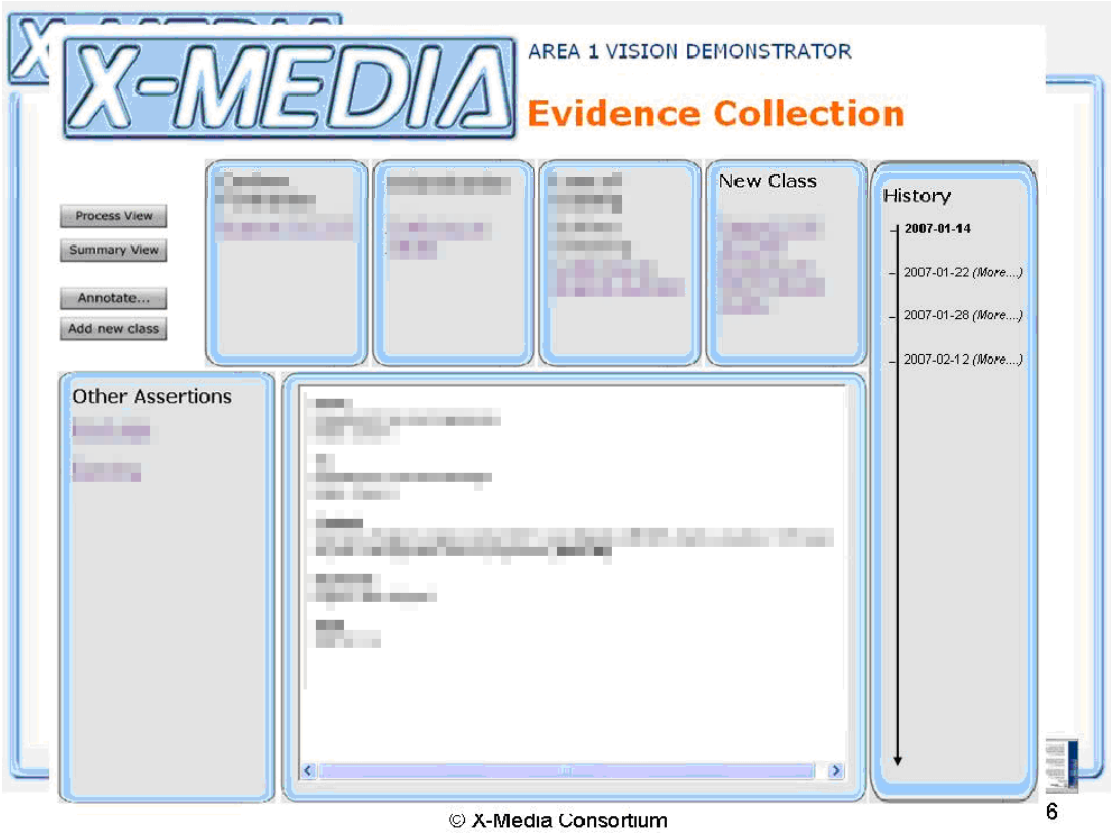


Figure 13. The summariser interface (vision demonstrator) . Please note that portions of the text have been deliberately obscured for reasons of confidentiality.

The CORPORUM Summarizer is currently implemented as a Microsoft Component Object Model (COM) Component, thus relying on a MS Windows platform to run. In order to make the Summarizer interact with the X-Media environment, there will be implemented an X-Media JAVA-wrapper for the interaction with other components. This wrapper will enable the interaction with, e.g., the Semantic Scratchpad (see Chapter 5) in order to import a set of documents that is of interest to the user, the imported documents will then be enhanced with information from the X-Media Knowledge base, such as information on provenance, in order to ensure the correct presentation along the publication timeline (see Figure 14 below). The X-Media Wrapper will provide an API that can be used by other X-Media presentation tools for inclusion of the summary presentation.

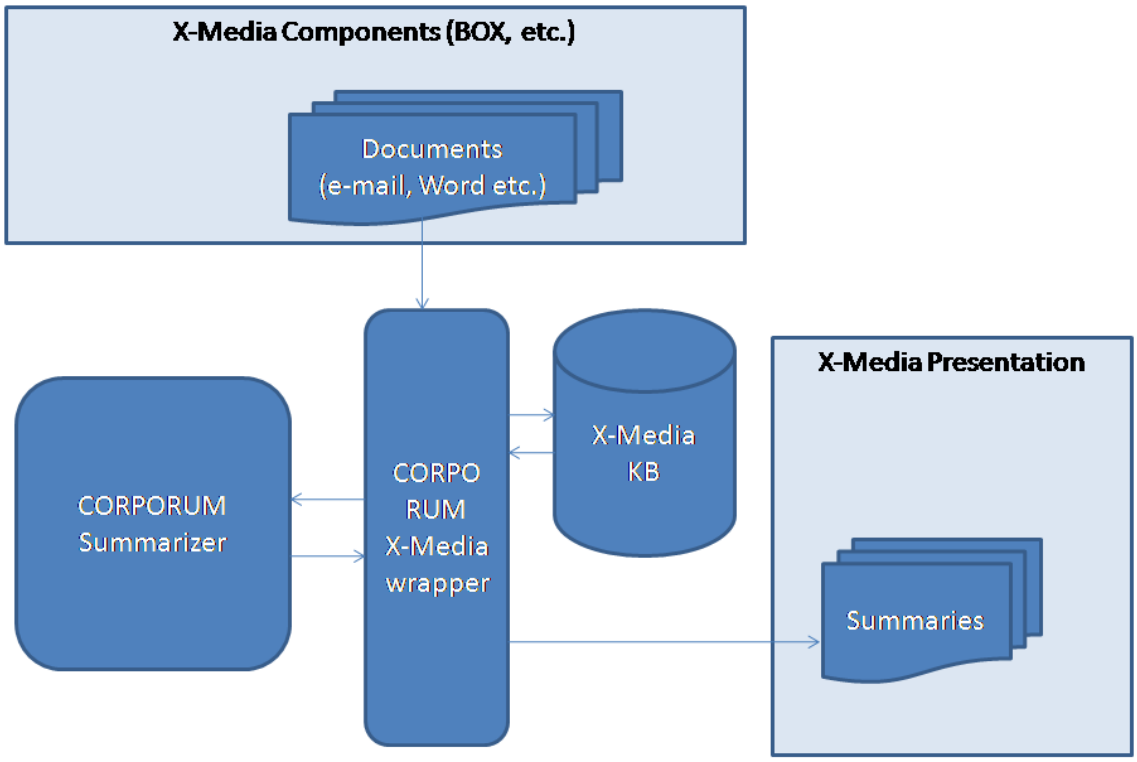


Figure 14 The CORPORUM summariser in the X-Media architecture

4.2.2 Meeting the X-Media requirements

Requirements from the X-Media Use Cases

A typical use case for the X-Media summarisation tool will be constituted within TIP 2 and 3 (D4.1) of the issue resolution use-case, where the harvesting, gathering and browsing of large amounts of information is required.

Interoperablility

The X-Media CORPORUM wrapper will provide a JAVA API that ensures the interoperability with other X-Media System Components. RDF/OWL interoperability will ensure the interaction with the X-Media Knowledge base.

5 Sub task 4.3.1 Big Organizer for X-Media

The Semantic Scratchpad (or BOX) is a special personalized, task-based and virtual semantic repository storing task-specific facts. A single BOX object provides *read-only* access to a given subset of the knowledge stored within an X-Media system;

means are provided to access such knowledge in tabular and graph format (however, it is not guaranteed that *both* formats are supported by a given BOX instance).

5.1 Semantic Scratchpad

The Semantic Scratchpad is instantiated by means of a Java component named BOXHandler (available through the X-Media Kernel), which manages BOX objects. In the first Scratchpad release, it is provided as a BOX implementation which selects knowledge by means of a SPARQL "SELECT" query dispatched to a given AccessKnow repository; such implementation provides access to the selected knowledge only in tabular format. At later stages of the project, other implementations may be provided which will enable access to the knowledge in graph format also, and to select knowledge by accessing the Kernel component XMSearchAndIndexHandler (see [D11.1]); moreover, more powerful ways of selecting knowledge will be also evaluated, such as Networked Graphs (see [Schenk 2007]).

5.1.1 Technical Description

A BOX object is referenced by means of a URI; moreover, it has associated an X-Media user which is the BOX "owner"; possibly, a BOX is also described by an extensible set of metadata, such as title, description…

The BOXHandler manages an "index" of BOXes, allowing for insertion, deletion, and retrieval of BOXes. Entries of such index hold associations between a BOX and a pair user, task (or user group, task). The BOXHandler allows to add (or remove) an association between a BOX and a pair user, task (or user group, task), and to retrieve the BOXes associated to a pair user, task. Users, tasks and BOXes are referred by means of their URIs. Each action is invoked on behalf of some authenticated X-Media user; a BOX security layer exists which checks whether a given user has the proper rights to perform the requested action; such security layer also manages allocations of access rights: the first Scratchpad release may not feature an implementation of such security layer, and later versions will provide a Unix-like implementation (i.e. the BOX owner will grant read/write permissions to its groups and to the other users).

According to deliverable [D4.1], Section 5.4.1, the Scratchpad is also supposed to feature means to set up access control on the knowledge it manages. However, given that the Scratchpad basically provides filtered access to other X-Media knowledge repositories, responsibility of access control checking is also delegated to the target knowledge repositories.

BOX Browser

As a sample usage of the BOX APIs, a (generic) BOX Browser web application will be provided, which will allow an X-Media user to login, manage his/her BOXes, and browse their content (possibly by exploiting the presentation facility of the X-Media knowledge lenses).

The BOX Browser code will provide a baseline that may be further customized by developers in order to build domain specific applications.

Interoperability

The Scratchpad largely depends on other X-Media Kernel modules, such as AccessKnow, the XMSearchAndIndexHandler, and the Process Support components (see [D11.1]). It is integrated (and deployed) in the X-Media Kernel. Moreover, it uses standards such as SPARQL.

5.1.2 Meeting the X-Media requirements

Performance

The Scratchpad performance relies on the performance of the targeted X-Media knowledge repositories.

Handling Heterogeneous Knowledge

The Scratchpad handles heterogeneous knowledge by providing support for targeting multiple knowledge repositories, and by also addressing (at later stages) the Kernel XMSearchAndIndexHandler component – which in turn is supposed to handle both semantic and full-text search (see [D11.1]).

Fundamental Functions

As the Scratchpad doesn't envisage a direct interaction with end user, the requirements listed in the "Fundamentals Functions" Section of deliverable [ID4.1] don't apply.

Innovative X-Media Functions

The Scratchpad doesn't directly fit in any of the categories outlined in Section "Innovative X-Media Functions" of deliverable [ID4.1]; nonetheless, it is rather to be considered a fundamental means to achieve all of the innovative functions therein illustrated: indeed, it can be used as task-oriented selection and filtering tool by Knowledge Lenses components, it can used in order to contextualize user activity by Process Support components, and it is related with Perspective Building in that it

gives valuable information about relation between user activity (in terms of tasks being accomplished) and consumed knowledge.

Use cases

The Semantic Scratchpad is explicitly mentioned as Technology Insertion Point in all the use cases described in deliverable [D13.2], and in the "Issue Resolution Test Bed" and "Non Destructive Evaluation Test bed" use cases in deliverable [D12.2]. It may be the case that, in some of the cited use cases, also the BOX Browser will be employed in order to fully match the requirements.
Within project Phase 1, the Semantic Scratchpad will be deployed in the "Competitors Scenario Forecast" FIAT use case (see [D13.2]).

6 Subtask 4.3.2 Basic Process Support

Tools for process-support are primarily dedicated to assist individual users in keeping track of and staying aware of tasks, in particularly with respect to different information that may be associated to emails, i.e. reports, emails, spreadsheets, images.

6.1 The Koblenz Email tool

Email is a vital tool for debate, knowledge sharing and distribution (e.g. by sending around attachments). One goal of the Koblenz Email tool is to provide emails and associated metadata about persons, tasks, and attachments to the X-Media knowledge base. Additionally, easy-to-use task management features will be supported focussing on the provision of task-awareness to users and leveraging retrieval based on metadata not only about documents, but also people and tasks. Moreover, services such as automatic notification based on different criteria will be supported, e.g. notification based on a newly defined issue resolution process.

6.1.1 Technical Description

The tool is implemented as a plugin for the Thunderbird email client. It makes use of the X-COSIM (Franz et al. 2007) framework that is partly used in X-Media knowledge representation for storing email, person, and task metadata compliant to the X-Media KB.

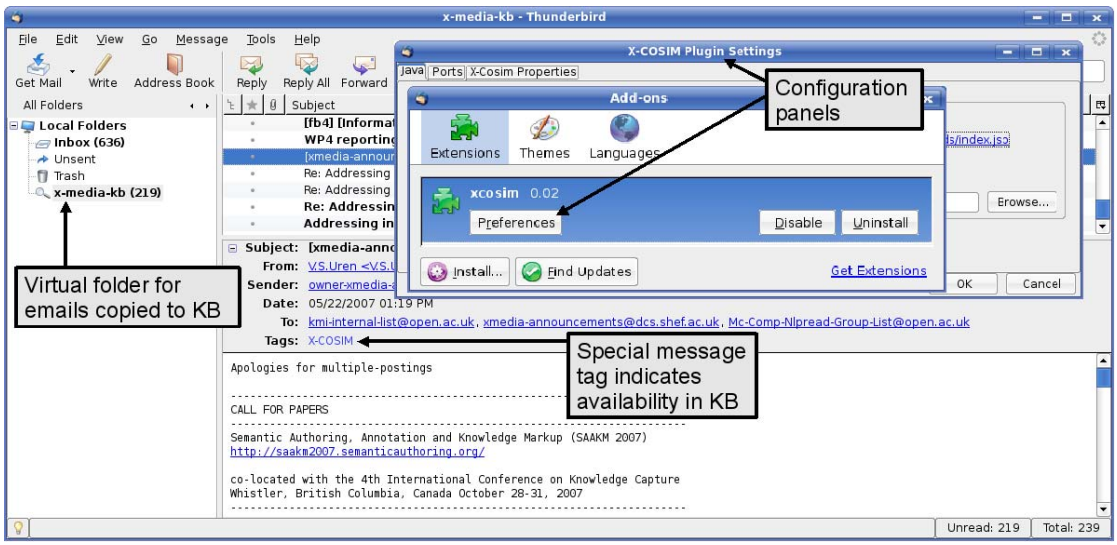


Fig 15 The semantic email client

6.1.2 Meeting the X-Media requirements

Being a common knowledge management tool, email is part of nearly every use case provided by the industrial partners of X-Media. We will focus, however, on the scenarios identified by TIP 9 from Fiat and TIP 1 from Rolls-Royce.

The Thunderbird mail client is available for multiple operating sytems and supports different mail standards like IMAP and POP. The plugin developed by Koblenz is platform-independent as well, being based on the Thunderbird runtime environment and the X-COSIM Java API. To integrate email-related support within X-Media scenarios where no client-side installation is possible, Koblenz and Quinary are also developing a server-side solution that reads emails sent to a particular account, e.g. x-media@example.com, and transfers them to the X-Media KB.

7 Conclusions

This deliverable has reported our progress in developing the knowledge sharing tools that will be used to build the first phase of X-Media test beds. In summary our achievements have been:

- Building annotation tools for open document format and images which can annotate regions.
- Four different search and browse tools each exploring different research lines on user interaction and search refinement. Search is a central knowledge sharing task and the comparison of multiple search modes will allow us to evaluate the advantages and disadvantages of different approaches.

- Innovative work has been undertaken on the presentation of search results, particularly of summary views of semantic data.
- A ground breaking semantic scratchpad prototype has been built. The scratchpad idea came directly from requirements work with users and is central to the envisioned X-Media knowledge sharing and reuse paradigm.
- A first prototype of a semantic email tool has been built to demonstrate key process support ideas.

Already in phase 1, these tools address many of the X-Media requirements. These are itemised for each tool but here we note particularly that the tools all address the issue of interoperability. In this phase, interoperability is gained through compliance with the X-Media kernel and through the wide use of the Eclipse RCP which will facilitate the packaging of task driven tools into use case driven platforms. In addition, user-centred design is important in many of the tools, whether through better search support or the presentation of knowledge, using technologies like Fresnel lenses or summarisation of text or semantic metadata.

The X-Media consortium has responded to the requirements for knowledge sharing and process support with a range of task centric tools that make are individually innovative while working within the constraints of the X-Media architecture to facilitate interoperability. The next tasks are to assemble the individual tools into the use case test beds and to evaluate the tools in context.

8 References

[ID4.1] V. Uren, D. Petrelli, P. Cimiano, T. Franz, V. Lanfranchi, Y. Lei, M. Weiten, P. Slavazza, T. Christopher, F. Ciravegna, L. Schmidt-Thieme, L. Gilardoni, *Specification of Methods for Knowledge Sharing*, X-Media Internal Deliverable 4.1, 2006.

[D4.1] V. Uren et al., *Specification of Knowledge Sharing Systems*, X-Media Deliverable D4.1, February 2007.

[D11.1] C.Biasuzzi, P.Slavazza, L.Gilardoni, M.Ferraro "*Integration Framework Specification*", X-Media Deliverable 11.1, 2007.

[D12.2] Dadzie, A et al., *Specification of System Functionalities for the First Release of the Rolls-Royce Test Bed*, X-Media Deliverable 12.2, 2007.

[D13.2] Giordianino, M. et al., *Specification of System Functionalities for the First Release of FIAT Test Beds*, X-Media Deliverable 13.2, 2007.

[Ahlberg 1994] C. Ahlberg and B. Shneiderman, *AlphaSlider: A compact and rapid selector*. Proc. CHI '94 (1994) 365-371.

[Arndt et al. 2007] Richard Arndt, Raphael Troncy, Steffen Staab, Lynda Hardman, Miroslav Vacura, *COMM: Designing a Well-Founded Multimedia Ontology for the Web*, proceedings of the 6th International Semantic Web Conference, 2007.

[Bremdal 2000] Bremdal, B.: The CORPORUM Summarizer. CognIT white paper. http://www.cognit.no Halden, Norway: 2000.

[Chakravarthy et al. 2006] Chakravarthy, A., Ciravegna, F., Lanfranchi, V.: Cross-media document annotation and enrichment. In: Proceedings of the 1st Semantic Authoring and Annotation Workshop (SAAW2006).

[deSmedt et al. 2005] de Smedt, K., A. Liseth, M. Hassel, H. Dalianis 2005. How short is good? An evaluation of automatic summarization. In Holmboe, H. (ed.) Nordisk Sprogteknologi 2004. Årbog for Nordisk Språkteknologisk Forskningsprogram 2000-2004, pp 267-287, Museum Tusculanums Forlag.

[Engels and Lech 2002] Robert H.P. Engels and Till C. Lech: Generating Ontologies for the Semantic Web. In: J. Davies, D. Fensel, F. van Harmelen (eds): Towards the Semantic Web: Ontology-driven Knowledge Management. Wiley, 2002.

[Franz et al. 2007] Thomas Franz, Steffen Staab, Richard Arndt, *The X-COSIM Integration Framework for a Seamless Semantic Desktop*, proceedings of the 4th International ACM Conference on Knowledge Capture, 2007

[Hassel and Dalianis 2005]. Hassel, M and H. Dalianis. Generation of Reference Summaries. In the proceedings of the 2nd Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, April 21-23 2005, Poznan, Poland.

[Heer 2005] J. Heer, S. Card, and J. Landay, *Prefuse: A toolkit for interactive information visualization*. Proc. CHI '05 (2005) 421-430.

[Hovy and Lin 1997] E. Hovy and C-Y Lin. 1997. Automated Text Summarization in SUMMARIST. in Proceedings of the Workshop of Intelligent Scalable Text Summarization, July.

[Lei 2006] Yuangui Lei, Victoria Uren, and Enrico Motta, *SemSearch: A Search Engine for the Semantic Web,* in proceedings EKAW 2006, Managing Knowledge in a World of Networks, Podebrady, Czech Republic, 2006.

[Lei et al.] Yuangui Lei, Vanessa Lopez, Enrico Motta, and Victoria Uren, *An Infrastructure for Semantic Web Portals*, to be published in Journal of Web Engineering.

[Pietriga et al. 2006] Emmanuel Pietriga, Christian Bizer, David Karger, Ryan Lee, *Fresnel: A Browser-Independent Presentation Vocabulary for RDF,* proceedings of the 5th International Semantic Web Conference, 2006

[Schenk 2007] S. Schenk, S. Staab. *Networked RDF Graphs*. Technical Report 3/2007, Institute for Computer Science, University of Koblenz, 2007.

[Spärck Jones 1999] Spärck Jones, K. 1999. Automatic Summarizing: factors and dimensions. In Mani and Maybury (eds.): Advances in Automatic Text Summarisation. Cambridge, Mass.: MIT Press

[Tran06adap] Duc Thanh Tran, Philipp Cimiano, Anupriya Ankolekar. *Rules for an Ontology-based Approach to Adaptation*. In Proceedings of the 1st International Workshop on Semantic Media Adaptation and Personalization. Athen, Greece. December 2006.

[Tran07int] Duc Thanh Tran, Philipp Cimiano, Sebastian Rudolph, Rudi Studer. *Ontology-based Interpretation of Keywords for Semantic Search*. In Proceedings of the 6th International Semantic Web Conference, pp. 523-536. Busan, Korea, November 2007.

[Tran07fil] Duc Thanh Tran, Peter Haase, Bernado Grau, Ian Horrocks, Boris Motik: *Representation and Querying of Metalogical Information*. Planned for submission to AAAI. 2008.

[Tran07res] Duc Thanh Tran, Stephan Bloehdorn, Philipp Cimiano, Peter Haase *Expressive Resource Descriptions for Ontology-Based Information Retrieval*. In Proceedings of the 1st International Conference on the Theory of Information Retrieval (ICTIR'07), 18th - 20th October 2007, Budapest, Hungary, pp. 55-68. October 2007.

[Uren 2006] Victoria Uren and Enrico Motta, *Semantic Search Components: a blueprint for effective query language interfaces*, in proceedings EKAW 2006, Managing Knowledge in a World of Networks, Podebrady, Czech Republic, 2006.

[Uren et al.] Victoria Uren, Yuangui Lei, Vanessa Lopez, Haiming Liu and Enrico Motta, Marina Giordanino, *The usability of semantic search tools: a review*, to be published in Knowledge Engineering Review.

[Zhou 2007] Qi Zhou, Chong Wang, Miao Xiong, Haofen Wang and Yong Yu, *SPARK: Adapting Keyword Query to Semantic Search*, proceedings of the International Semantic Web Conference (ISWC/ASWC), Busan, Korea, 2007.