Deliverable 4.5

Integrated Knowledge Sharing Environment

Author: Thomas Franz, University of Koblenz, franz@uni-koblenz.de

Renata Dividino, University of Koblenz, dividino@uni-koblenz.de

Christoph Ringelstein, University of Koblenz, cringel@uni-koblenz.de

Miriam Fernandez, Open University, M.Fernandez@open.ac.uk

Vanessa Lopez, Open University, V.Lopez@open.ac.uk

Andriy Nikolov, Open University, A.Nikolov@open.ac.uk

Johannes Busse, Ontoprise, busse@ontoprise.de

Aba-Sah Dadzie, University of Sheffield, a.dadzie@dcs.shef.ac.uk

Andre Busche, University of Hildesheim, busche@ismll.de

Massimo Ferraro, Quinary, m.ferraro@quinary.com

Work-package: WP4 -Knowledge Lenses and Process Support

Type: Report

Distribution: Public

Status: Final

Date: January 15, 2010

Deliverable Coordinator: Thomas Franz, University of Koblenz

Reviewers: Steve Fullerton and Ian Pallen, Solcara Ltd

Area Coordinator: Sergej Sizov, University of Koblenz

Project Coordinator: Fabio Ciravegna, University of Sheffield

EU Project Officer: Leonhard Maqua

ABSTRACT

This deliverable reports on the integration of user interfaces developed within workpackage 4 of X-Media. Specifically, this deliverable focuses on their integration with respect to the following aspects:

1. Integration among user interfaces developed in X-Media.
2. Integration of X-Media technologies developed outside of work package 4.
3. Integration with X-Media infrastructure.

Unlike prior developments in task 4.5 focusing on novel methods for knowledge sharing and process support, the focus is in this deliverable is on the integration of the different end user tools developed before and the integration of the methods developed within and outside of this workpackage.

This deliverable highlights a variety of tools and their integration with respect to above aspects. Among the different X-Media use cases, different combinations of the presented tools are applied to support diverse tasks handled by engineers at Rolls-Royce and Fiat. The range of tools and their tight integration, especially on the user interface side, documents the successful completion of the overall task of providing an integrated tool suite that supports complex knowledge work across different media.

Contents

1 Introduction

This deliverable reports on the integration of user interfaces developed within workpackage 4 of X-Media. Specifically, this deliverable focuses on their integration with respect to the following aspects:

- Integration among user interfaces developed in X-Media.
- Integration of X-Media technologies developed outside of workpackage 4.
- Integration with X-Media infrastructure.

These different aspects of integration are addressed along the lines of the working task 4.5 defined for the last phase of workpackage 4. As highlighted also in the description of work, in the last phase, the development of the end user tools that started in month 29 is continued, however, with a different development focus. Unlike prior developments in task 4.5 focusing on novel methods for knowledge sharing and process support, the focus is now on the integration of the different end user tools developed before and the integration of the methods developed within and outside of this workpackage. The subtasks of task 4.5 correspond closely to actual user tasks as supported by the user interfaces developed in workpackage 4, e.g. searching and browsing, document annotation, or knowledge analysis. This deliverable is organized along these tasks and presents for each task

- the user interfaces developed,
- their design,
- their integration (according to the above aspects),
- and the user benefits gained by the integration.

2 Browsing of Meta Knowledge

One of the key benefits of the X-Media technologies such as knowledge-based problem resolution, workflow optimization, lifecycle monitoring of complex products, or analysis of product evolution is the better support of decentralized, self-organizing knowledge exchange between users. In such applications we are faced with a highly varying quality of information. In order to investigate the value of information, humans usually base their analysis in the knowledge about the provenance of information, its certainty/reliability, recency and its contributors/agents (e.g. meta knowledge).

Accordingly, within task 4.5.1 of workpackage 4, the integration of meta knowledge capabilities as developed in workpackage 2 has been targeted. More precisely, the focus of the task has been to develop a user interface that enables users to evaluate knowledge with respect to available meta knowledge.

2.1 Meta Knowledge

Utilizing meta knowledge in complex, large scale semantic applications is still a rather difficult enterprise. On one hand, although many sources (such as provenance records) are available at almost zero cost, they were not considered as serious value for knowledge management frameworks in the past. On the other hand, even more important, the lack of flexible mechanisms for capturing and representing this kind of information makes aggregation and utilization of meta knowledge even harder.

In the X-Media project the *metak* framework has been developed which is a generic approach to the treatment of meta knowledge that is able to adapt to many dimensions of meta knowledge and that is open to accommodate to new dimensions when the need arises. It is implemented as a principled, original framework for RDF repositories with SPARQL query processors[1].

2.2 Knowledge Browser LENA

Furthermore, in the X-Media project a toolkit for search/browsing tools has been developed and tested for at least one use case. In order to allow X-Media users to easily

[1] For a more detailed description of the project produced meta knowledge framework, we refer the reader to the deliverables D1.3 “Prototype for management of process knowledge”, D1.4 “Prototype for management of provenance / change knowledge”, D1.5 “Report on reasoning with integrated provenance and change statement support” and D2.4 “Report on query answering under uncertainty”.

access meta knowledge information making using of X-Media UI tools, in this deliverable, we describe the integration of the LENA semantic browser [Franz et al., 2010, Koch et al., 2008] that can browse over semantic information and the *metak* framework.

LENA stands for LEns based NAvigator. 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 i.e. the provision of multiple renderings for the same data to conform to the specific information needs of different users. 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. Additionally, LENA enables the user to combine locally available RDF data with data available on the Semantic Web by implementing a linked data browser.

2.3 Integration of the Meta Knowledge Framework

LENA was presented as part of the X-Media vision demonstrator for the FIAT competitor scenario forecast use case. It is intended to support the user role of the knowledge engineer. Considering this, in this deliverable we present the integration of LENA and the *metak* framework for the FIAT scenario highlighting the benefits of this integration for the knowledge engineers.

As shown in Figure 2.1, LENA lists available lenses in a box (named Lenses) on the left. The box below (named Local 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. The biggest portion of the screen (the right side) displays instance data. Figure 2.1 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 2.1-a is an example of the standard view of LENA and Figure 2.1-b is an example of how resources are presented with associated meta knowledge. At the right side of the screen a check box is presented. When the check box is activated, LENA presents the resources and the meta knowledge information associated to them. When the checkbox is not activated, LENA ignores the meta knowledge information and presents the resources as defined in its lenses.

LENA with *metak* assists users in browsing resources and gives them a means to explore the meta knowledge information associated with the available resources. LENA with *metak* enables users easily to identify the value of information by presenting meta knowledge information whenever it is requested and thus reducing, for knowledge engineers, the negative effect of dealing with many different tools for browsing and analysing semantic information.

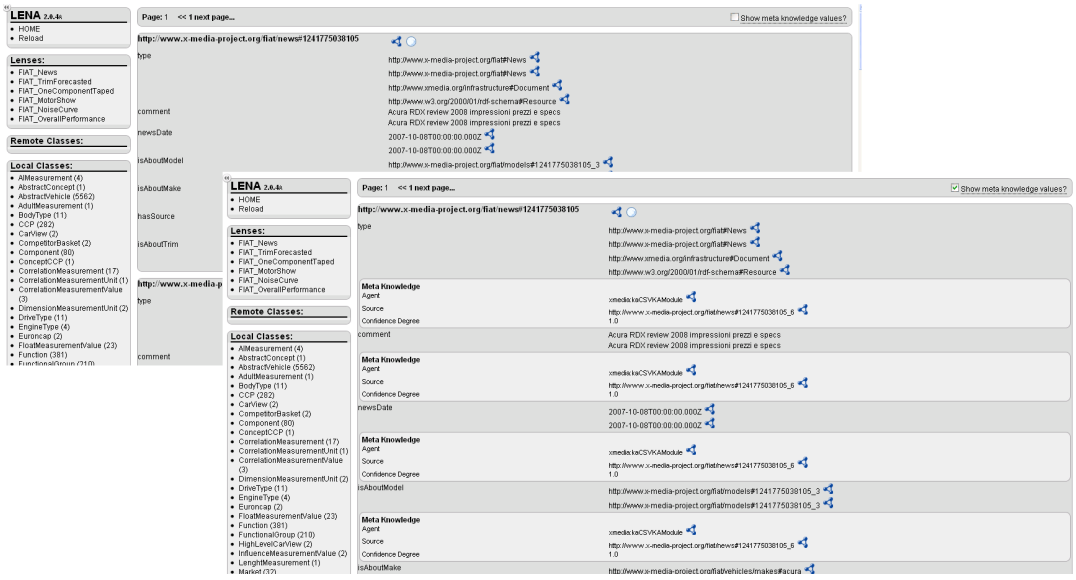


Figure 2.1: Screenshot of LENA for the FIAT resources (a) and the presentation of resources with associated meta knowledge information(b).

2.4 Integration with X-Media Infrastructure

LENA runs as a Java servlet application accessible from common web browsers. It utilizes an implementation of the *metak* framework, in order to support meta knowledge browsing. In the following, we describe the work-process of LENA, review its architecture (see Figure 2.2) and shortly describe the core *metak* components.

Upon incoming HTTP requests from web browsers, the requested RDF resource is rendered either, if requested, according to a specific Fresnel lens definition or according to a standard Fresnel lens, which selects all triples in the available graph where property-values are object and subject of the requested resource. In parallel, the *metak* components are invoked and requested by means of an AJAX call. Thus, the user can immediately inspect the requested resource and the associated meta knowledge information. The *metak* components are processed consecutively, whereas the resulting meta knowledge information is injected into the user interface by means of AJAX.

LENA implements the Model-View-Controller (MVC) pattern. The controller manages incoming requests from the browser, queries data from the model, and serves the view with data for the final rendering output. The model allows for requesting semantic information, e.g. by means of SPARQL queries. It provides a triple cache of local repositories to reduce network overhead. The view consists of XHTML templates used to generate the final rendering output with help of the provided controller information. Since LENA relies on the Fresnel display vocabulary for selection and rendering of requested datasets, the controller utilises the JFresnel Java library in order to apply Fresnel definitions. Meta knowledge is requested from the *metak* implementation.

The *metak* implementation is separated into two core components; namely, the query rewriter and query evaluator components. The query rewriter component is responsi-

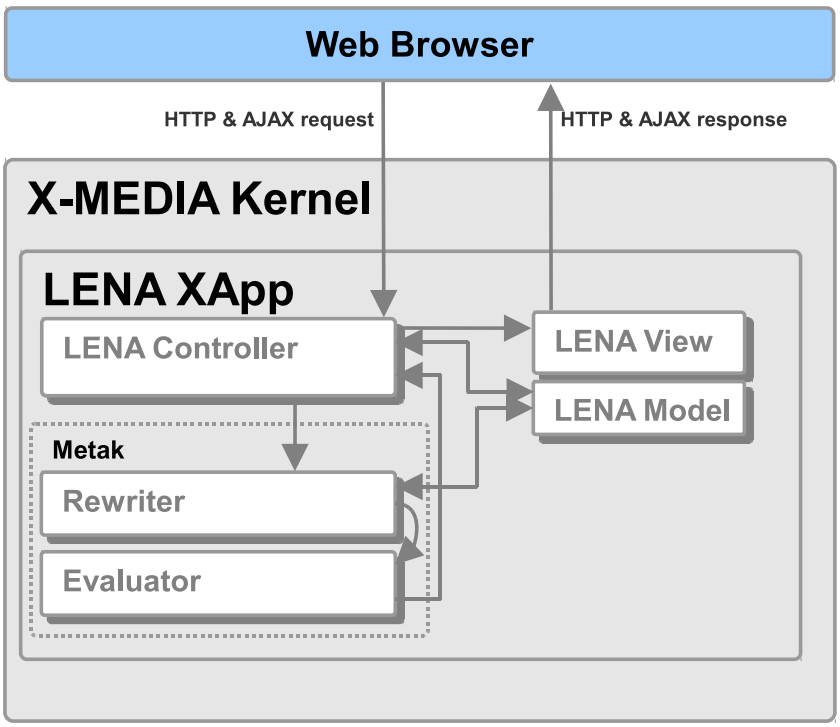


Figure 2.2: Architecture of LENA with *metak*.

ble for extending the SPARQL query by including individual variable bindings about meta knowledge. The query evaluator component extends SPARQL query processing to calculate conjunction, disjunction and negation for all meta knowledge properties.

LENA with *metak* Web application is integrated in the X-Media Kernel (for more details see `https://projects.isweb.uni-koblenz.de/svn/xmedia/trunk/lenaMetakXApp`) and has been used not just in the FIAT use case as described in this deliverable, but also in the Experimental Vibration and Issue Resolution use cases.

3 Integrated Searching and Browsing using LENA and SemSearch

Tools for searching and browsing the X-Media knowledge base have been developed within X-Media. As a final step of their development, we have worked on their integration. This section illustrates the results of the integration of the search tool SemSearch with further X-Media tools.

3.1 Integration of SemSearch and LENA

SemSearch, as presented in [Uren et al., 2007, Uren et al., 2008] is a semantic search engine that automatically maps a keyword-based user query to a specific ontology and its corresponding Knowledge Bases (KBs) in order to retrieve a ranked set of semantic entities as a final result to the user. SemSearch's best feature is its simplicity, providing support for the ordinary casual user to query the semantic data. As we can see in Figure 3.1 it provides a Google-like interface, so the user can type queries like *news: toyota* to obtain all the ranked set of answers, in this case all the instances of the class news that are in relation with the Toyota individual.

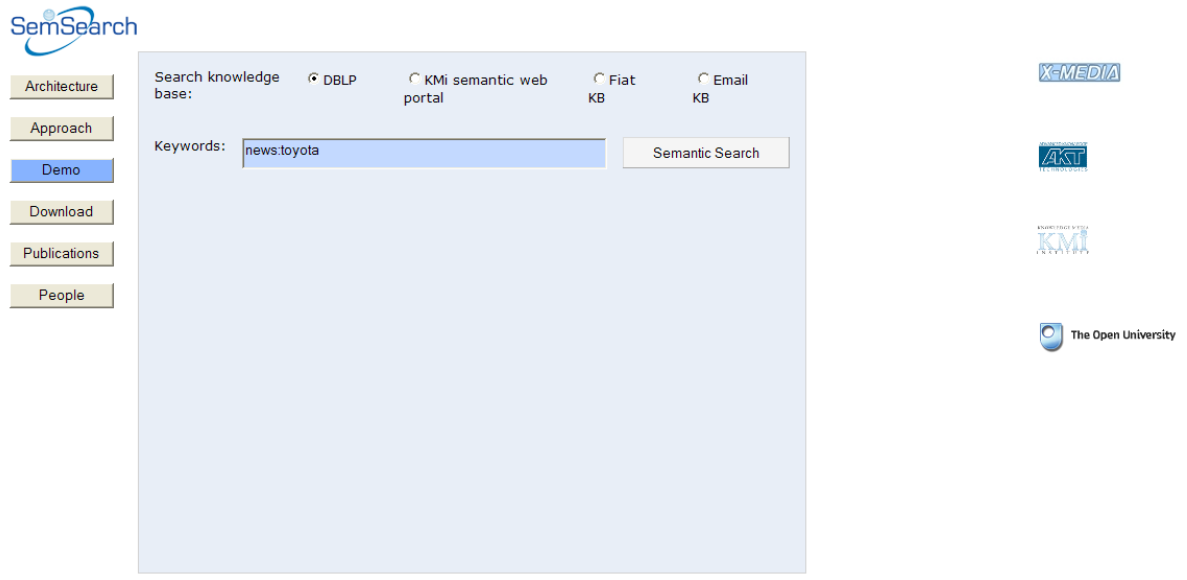


Figure 3.1: SemSearch Query interface

On the other hand, LENA (cf. Section 2) is a browser for semantic meta data stored in the X-Media knowledge base. The aim of combining both interfaces is to enhance, in a cyclic way, the visualization and browsing of results provided by LENA with the query capabilities of SemSearch. The following figures provide a detail description of the process of querying and browsing with the union of both technologies. As we can

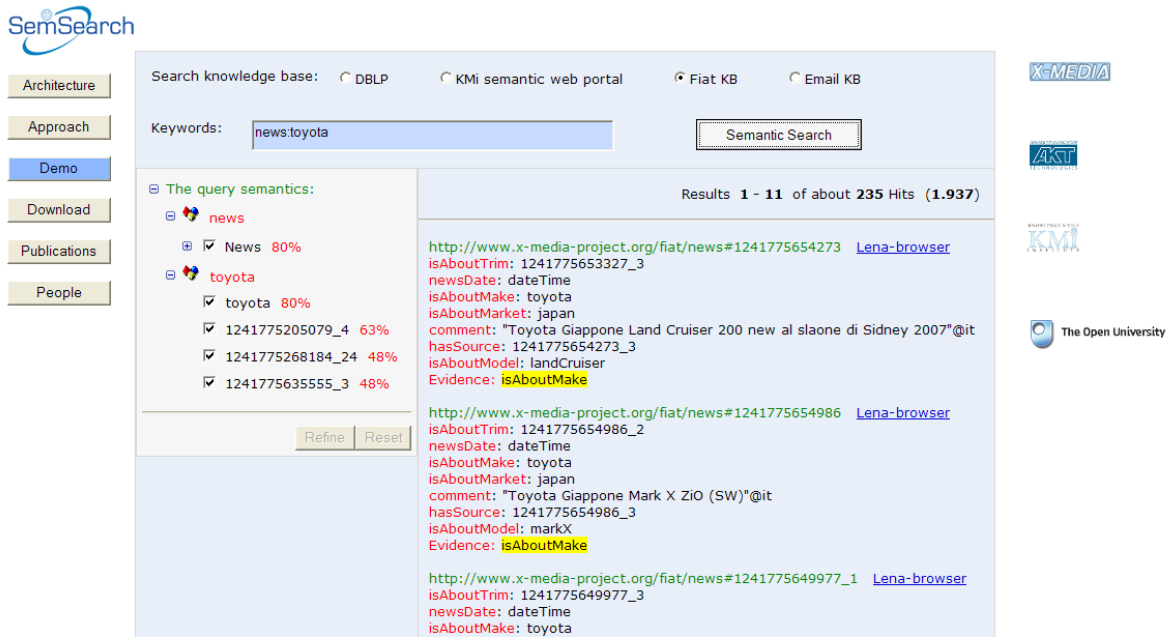


Figure 3.2: SemSearch results interface

see in Figure 3.2, SemSearch interface is divided in 4 main sections:

- The KB Selector panel (located at the top part of the interface)
- The query input panel (located just behind the KB selector panel)
- The user feedback panel (in the left hand side of the interface)
- The results panel (in the right hand side of the interface)

During the query process, the user selects the Knowledge Base over which he wants to perform the query. In this example, SemSearch handles four different KBs as we can see in the top part of the figure: DBLP, the KMi semantic portal, the Fiat KB and the Email KB. Once the KB is the selected and the query is introduced, in this example news:toyota, SemSearch displays the user feedback and the results panel.

The user feedback panel, located in the left hand side of the interface, allows the user to refine the query, in a relevant feedback way, resolving the ambiguities detected during the query process. Following our example, SemSearch detects that news is a class in the ontology, without any ambiguity. However, for Toyota, SemSearch suggests to the user the actual instance of Toyota (with 80% of accuracy) but also suggests several entities

that are related to Toyota in the ontology with different levels of accuracy (63% and 48%). The user can then select which semantic entity he is interested in and press the bottom refine to introduce his feedback into the query process.

Once the user is satisfied with the query, the final list of results is displayed in the right hand side panel of the interface. For each retrieved individual, SemSearch displays its URI and a list of its main properties and corresponding values. In our example, for each Toyota new individual, SemSearch is able to display the URL and its subset of properties (isAboutTrim, newsDate, isAboutMake, isAboutMarket, etc,). Also, as basic performance metrics for the user, SemSearch displays on the top part of this panel the amount of results that have been retrieved with this query (in this case 235) and its time (in this case 1.937 seconds)

3.1.1 LENA

Although the SemSearch interface is able to display basic properties for each of the retrieved individuals, it would be desirable to provide a tool that allows the browsing and exploration of these results in more detail. Here is where we have introduced LENA as a key visualization tool to improve the exploration of results retrieved by SemSearch. As we can see in the interface of SemSearch, close to the URI of each result is a linked LENA-browser. When the user click this link, the LENA interface is displayed (see Figure 3.3).

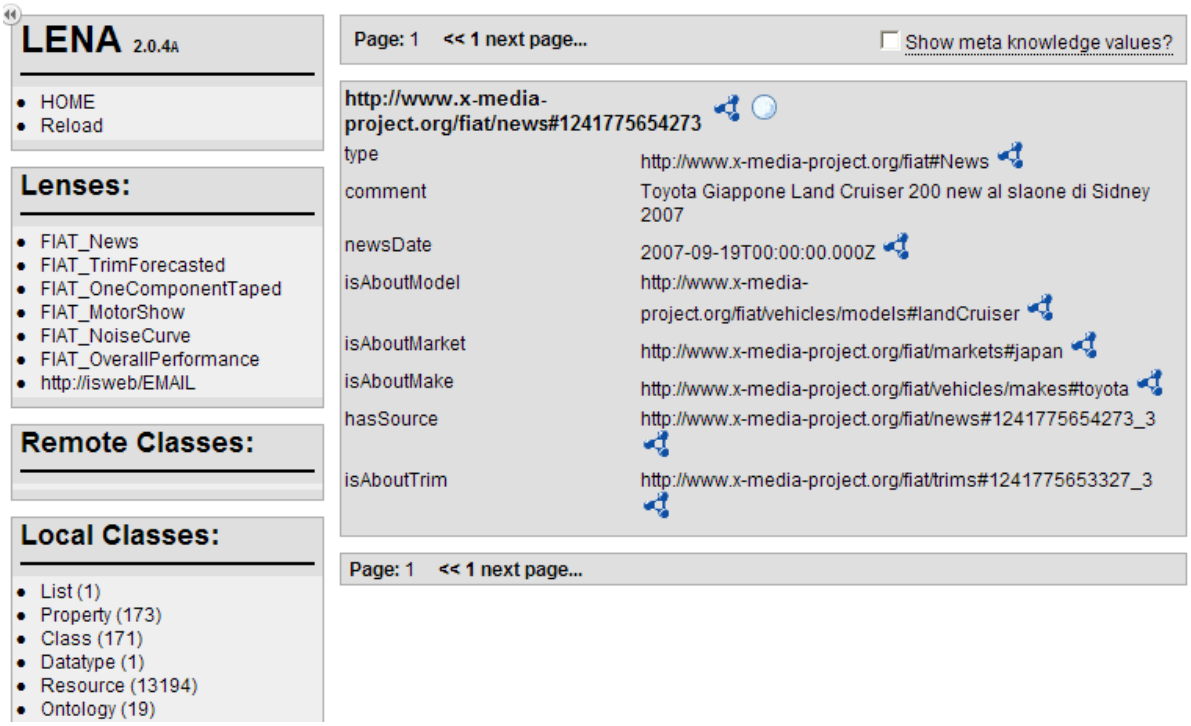


Figure 3.3: LENA: semantic entity interface

As we can see in Figure 3.3, the LENA interface is divided in two main panels:

- The visualization options panel (located in the left hand side)
- The visualization results panel (located in the right hand side)

We are focused on the visualization results panel, which, at the same time is divided in three main sections:

- The MetaKnowledge option panel (in the top part)
- The result panel (in the middle part)
- The information panel (in the bottom part)

The result panel is the most relevant one, because it displays the information about the entity, and allows the navigation across the different related entities in the KB. For example, for the individual selected, LENA is able to display the URI and to display all its properties with the completed value. The users can then click in these new URLs and navigate across the ontology and the KBs to gather more information without the necessity of performing new queries. The result panel also displays close to the URI, the icon to select a Lens. If the user clicks in this icon, he can refine the visualization of the same result. In the case of our example, the user might only be interested in the properties trim, model and date of the new object. LENA displays the specific visualization, reducing noise and providing the user more with an efficient way to interact with the information (see Figure 3.4).

Figure 3.4: LENA: semantic entity Lens

Finally, if the user also wants to extract information about meta-knowledge values of the properties, he does not need to perform any other query, because LENA also considers the display of this information within its interface. As we can see in Figure 3.5, for each property related to a semantic entity of type news, LENA is able to display its agent, its source and the confidence degree value of this information.

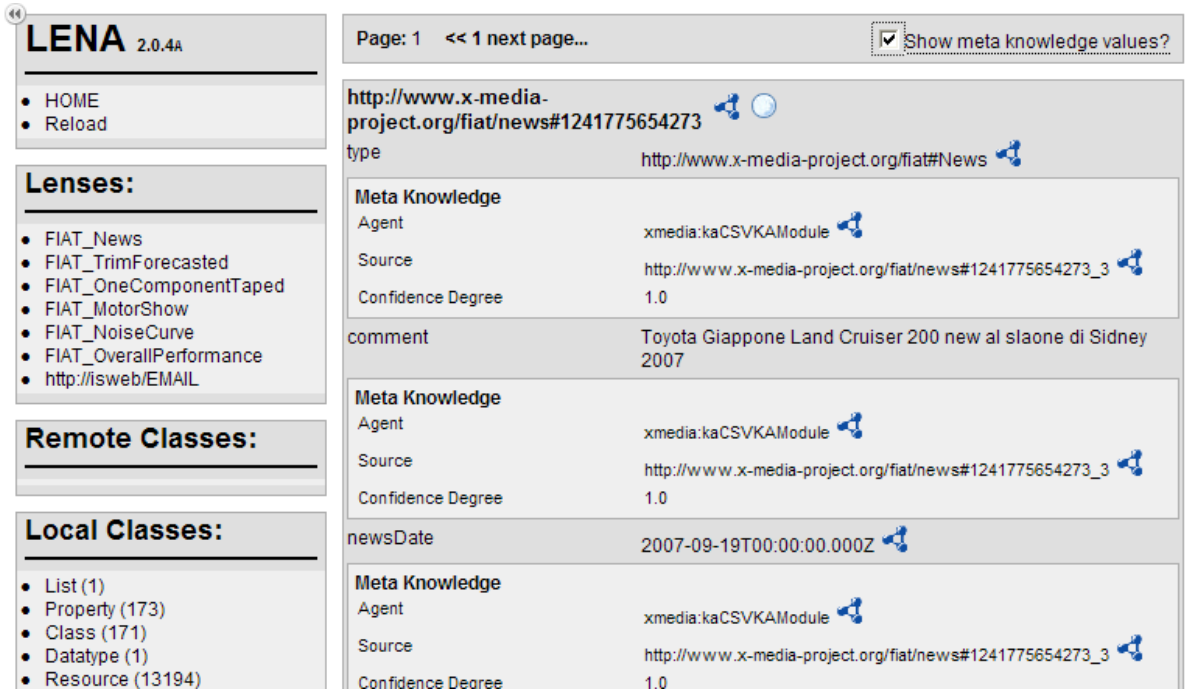


Figure 3.5: LENA: semantic entity meta-knowledge

3.1.2 User Benefits

The improvements obtained by the integration of SemSearch and LENA are threefold.

- The efficiency of the user increased in terms of spent time as he does not have to perform several queries to obtain the information he is looking for, but he can combine the query capabilities of SemSearch with the browsing navigational capabilities of LENA to find faster the information he is looking for.
- The effectiveness of the user in terms of querying the KB also increases. The visualization capabilities of LENA, performed using its lenses, reduce and enhance the amount and quality of information that the user has to manage. This fact, joined with the capability of displaying meta-knowledge, significantly improves the process of searching and selecting information.
- There is a clear gain with respect to user satisfaction. Since the user obtains in one tool three functionalities: querying, browsing and visualizing information. This clearly reduces her time introducing queries and exploring results, which is translated in higher levels of satisfaction.

3.2 Enhancement of SemSearch with SemSearchXplorer

Aiming to enhance the user experience in the process of exploring semantic data, another visualization tool has been implemented on top of SemSearch, SemSearchXplorer. This

tool supports the user in three respects: (1) it supports querying of the semantic data with a keyword based approach using SemSearch, so the users do not need to learn a semantic query language, (2) it helps users find relevant results both by using semantic enriched information about the results and semantic filter options to narrow down the set of results, and (3) it provides information exploration capabilities through semantic visualizations recommended by the system. Filtering of semantic search results helps to narrow down the result set to a more manageable amount of information. Besides searching for relevant information, facilities for the exploration of the results help users to gain insight in the context of results. With several semantic visualizations, we try to help users making sense of the raw data. Based on the assumption that there is no single visualization that fits all exploration needs, SemSearchXplorer recommends visualizations based on the information selected by users.

3.2.1 Searching

The query interface of SemSearchXplorer (see Figure 3.6) allows either to search for a keyword or to specify the expected type (the subject) of search results. The latter is supported through the syntax subject:keyword provided by SemSearch. With Boolean operators like and/or we can compose complex queries (for details see [Lei et al., 2006]). For example a query about news of projects has the following form: news:project.

Figure 3.6: Initial query interface of SemSearchXplorer

3.2.2 Refining the Result Set

SemSearchXplorer visualizes the query results as an ordered list (see Figure 3.7). A list representation has the advantage that users can read very quickly through the text and scan for information. The order of the set of results is sorted according to the ranking mechanism of SemSearch. Each hit contains the instance using the fragment part of an URI, or if available, the label information (rdfs:label). In addition to the instance we provide context information consisting of the class names and the relations to other instances, which belong to the query. This makes it easier for users to search for relevant hits. Each instance also provides a tooltip with statistical information about the result.

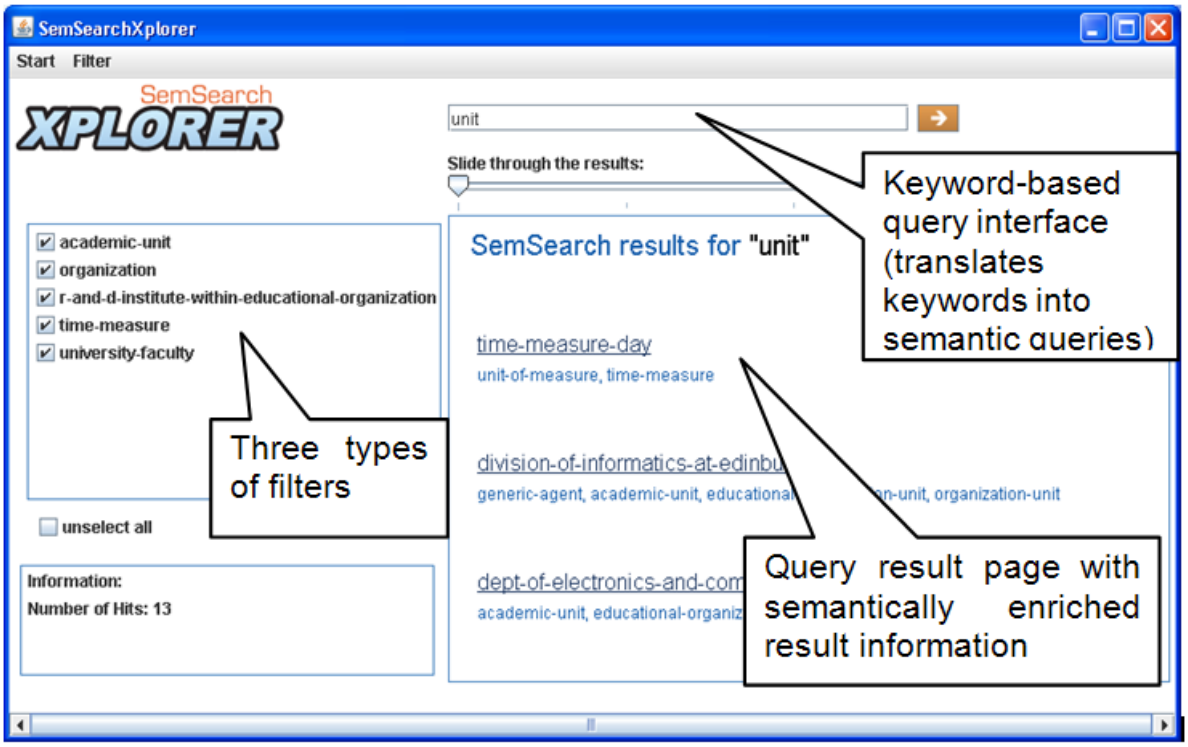


Figure 3.7: SemSearchXplorer query result page

For users is it important to be able to narrow down the result set to a set of interesting results. There are tradeoffs between filters with a small number of general class items, which allow a quick reduction of the result set, and filters with a large number of filter items, which allow a fine-grained narrowing, but have the drawback that the user has to select from a large set of filter possibilities. To allow filtering of the result set, we implement three types of filters. The first filter uses the direct super-classes of an instance but not all the other classes higher in the class hierarchy. Compared to the second filter, the all class filter, which uses the whole class information of the class hierarchy of the result set, the direct class filter generates less filter items, while the all class filter produces more filter items allowing precise filtering. The third filter uses the

key concepts (KCE Key Concept Extraction [PMA08]) of an ontology. With KCE it is possible to extract the best descriptors of an ontology. Therefore, the algorithm tries to find the most important concepts of an ontology as a human expert would do. Currently, the KCE algorithm considers three concepts. Natural categories (that are concepts that are information-rich in psycholinguistic sense), density (concepts which are information-rich in an ontological sense), and popularity (frequency of terms returned from a query to the Yahoo search engine). Although this algorithm's goal is to generate automatic ontology summaries, we consider KCE as a method applicable for filtering the result set according to the most meaningful items of an ontology while ignoring classes that are less important. Although the algorithm tries to maximize the coverage of the ontology, some concepts are not contained in the filter. We add hits of classes not belonging to the key concepts or subclasses to a new filter entry called *other*.

With these three types of filters, the user has effective mechanism to filter quickly to relevant information. Once found, the user can start the exploration of the hit.

3.2.3 Exploration

After the user selected a result, a new view of SemSearchXplorer shows the exploration page (see Figure 3.8). By now, we support users with three types of visualizations to explore the context of the selected hit: a graph, cluster and chart visualization.

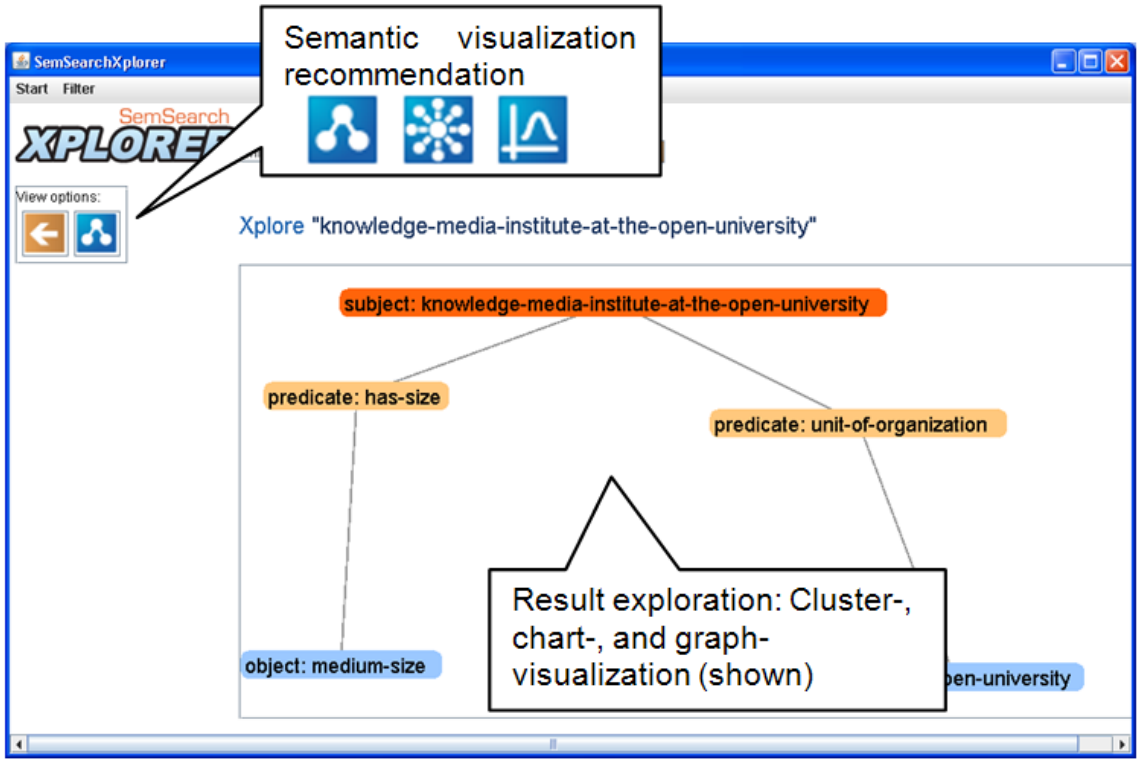


Figure 3.8: SemSearchXplorer exploration page with graph visualization

Graph visualizations are suitable for representing the context of a result. In contrast to a list-based visualization, a graph visualization is able to represent the connections between classes and instances. The list representation does not reflect the relational structure of the semantic result. Within our graph visualization instances are connected with edges representing the properties of the semantic result. While graph visualization helps users to understand the relations between resources, large graph visualizations become cluttered or need a large space on the screen. Our cluster visualization (see Figure 3.9) is used for showing larger amounts of information. Information that is not necessary for the first exploration is hidden in a cluster. If users need this information, they can expand the cluster. With this technique, we can reduce the problem of cluttered visualizations to a certain degree. As the graph visualization the cluster visualization tries to support users making sense of the relational structure of the backing data. Features of the cluster visualization are:

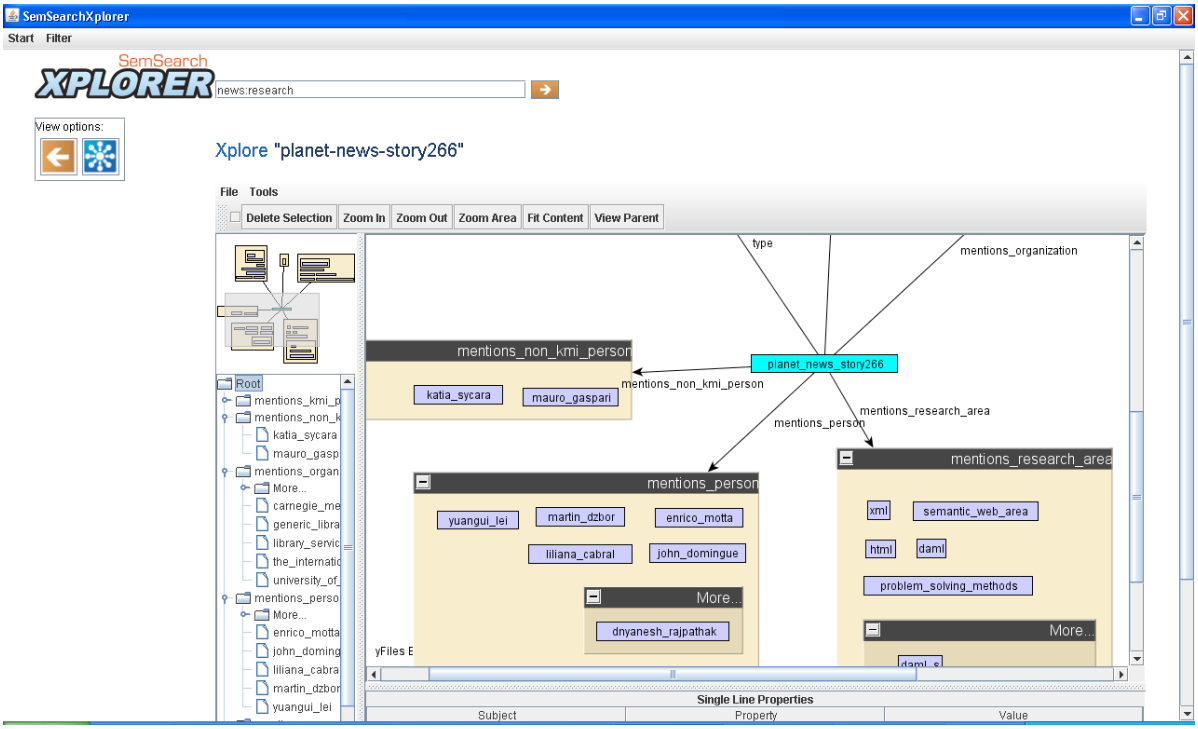


Figure 3.9: SemSearchXplorer exploration page with cluster visualization

- The instance to explore is highlighted and central in the graph representation.
- The object instances of the triple are ordered in a cluster with the name of the property. A cluster can contain a sub-cluster. We order higher ranked instances into the first level group and lower ranked instances into the more cluster.
- On the top left position of the cluster visualization we provide an overview and below the overview a tree representation of the whole visualization.

However, not only could the information about the context be interesting for users but also statistical information about the instance. This allows users to compare instance according to their relevance. For an example of our chart visualization (see Figure 3.10). The outer right blue chart of the chart visualization has the meaning that 12 instances have as line manger Enrico Motta. For a detailed description of the cluster and chart

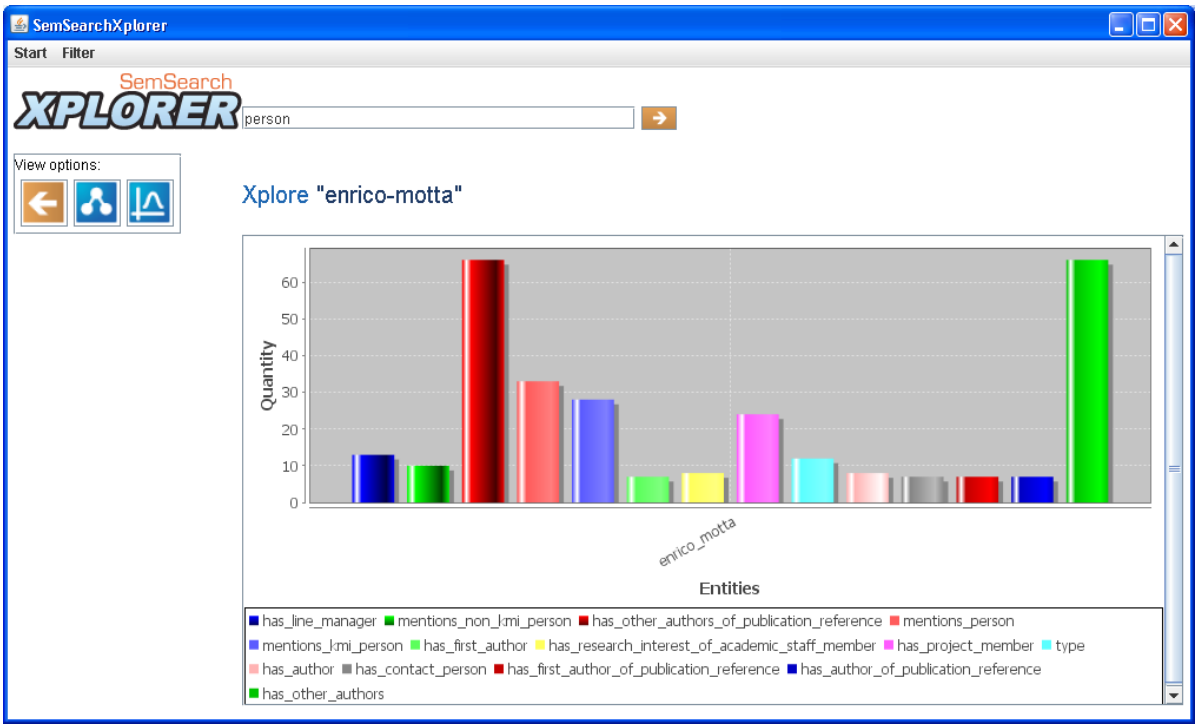


Figure 3.10: SemSearchXplorer exploration page with chart visualization

visualization and the underlying algorithms, see [Nazir et al., 2009].

3.2.4 Recommendation

The most notable feature of the exploration page is the recommendation component consisting of a variable number of recommended visualizations. Until now, we have focused on content-based recommendations of visualizations. The goal would be to provide users with the most suitable view for a particular set of data (consisting of instances, classes and/or relations). This is quite an ambitious vision, but for the first implementation of SemSearchXplorer we consider heuristics which serve as a starting point for future recommendations. The heuristics follow these simple rules:

- If the number of properties and related instances is above a threshold, then offer a cluster visualization.
- If this number is below a threshold, then offer a graph visualization.

- If filtering influences this number significantly, then provide cluster or graph visualization according to the threshold.
- For chart visualizations, we need three clusters of data (x and y-axis, and the values to plot). Therefore, if we can generate three clusters of data, then offer a chart visualization.

Based on the chosen entities of the user SemSearchXplorer recommends the user visualizations, which are able to visualize the selected information. As seen in Figure 3.10 the recommendation of visualization is represented in iconic form below the SemSearchXplorer logo. From left to right the icons are ordered according to which of the possible visualizations the system predicts are more suitable.

The improvement obtained with the integration of SemSearchXplorer is to help people find relevant information more quickly and to help them to explore the context of this information. For the case that users do not exactly know what they are looking for, semantic search (SemSearch) supports the user in finding a suitable starting point for their exploration and with the visualization layer of SemSearchXplorer semantic visualizations help to explore the context of the result.

SemSearchXplorer recommends visualizations according to the information the user provides and the capability of the visualizations to make use of the provided information. Thus, the decision is more a technical one, than based on user experience.

After the user entered the query, a list-based representation of the visualization is generated. The advantage of lists is that users are accustomed to read quickly through linear text. This representation works well, if the user is only interested in the highest ranked query results. However, if the user wants to get an overview of the whole result set another visualization, which provides an overview of all hits at once, would be more suitable.

3.3 Integration with X-Media Technologies

This section covers the benefits in terms of user, system and performance achieved from the integration of SemSearch and X-Media technologies.

In the simplicity of SemSearch lies its main limitation, the query expressivity. The query keyword interface limits the query expressivity, especially at the level of relations. (e.g. the user cannot differentiate in the query between persons who born in a country or persons who live in a country). Also, although SemSearch is domain independent and the user can choose with KB he wants to query, SemSearch does not allow to simultaneously query different KBs and to combine these information across different sources. To tackle these two limitations, we have integrated PowerAqua [Lopez et al., 2006], an ontology-based Question Answering (QA) system, with the X-Media technologies in order to extend SemSearch capabilities to a multi-ontology scenario, and enhance its query expressivity through the use of Natural Language (NL) queries. Technical details of how PowerAqua resolves the queries have been detailed in [Lopez et al., 2009b] and [Lopez et al., 2009a]. In the X-Media scenario, PowerAqua has been used to allow users

to input more expressive NL queries over the FIAT semantic repositories. Figures 3.11, 3.12 and 3.13 describe the User Interfaces exposed by PowerAqua to resolve the query: show me all vehicle models of the seat automobile. As we can see in Figure 3.11 the

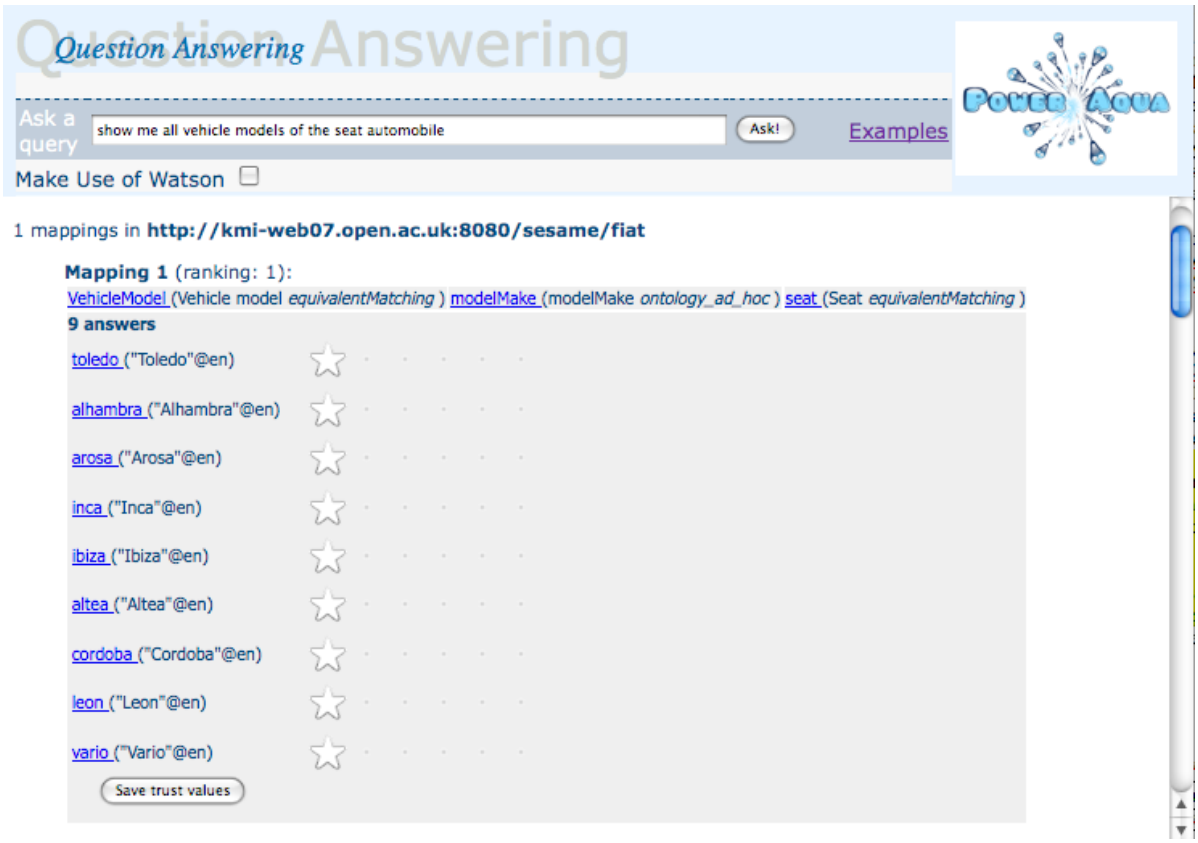


Figure 3.11: ontology mappings found in the FIAT ontology

interface of PowerAqua is divided in two main parts:

- The query panel (located in the top part)
- The results panel (located just below the KB selector panel)

The user introduce a NL query in the query panel *show me all the vehicle models of the SEAT Automobile* and get a set of mappings in different ontologies that are displayed in the results panel. Figure 3.11 shows the set of answers found in the FIAT ontology, while Figure 3.12 shows the set of answers found in the DBPedia ontology. The results panel displays, in a first step, the ontology that has mapped the query and the type of mappings. In the case of the Figure 3.11 the ontology is http://kmi-web07.open.ac.uk:0808/sesame/fiat and the mappings are: VehicleModel and seat (as equivalent matchings) and modelMake (as an ad-hoc matching). Figure 3.12 provides the mapping with the ontology http://kmi-web07.open.ac.uk:0808/openrdf-sesame/dbpedia3.3 and the mapping are: automobile and seat (as an equivalent matching) and manufacturer (as an ad-hoc matching).

Behind the ontology and the corresponding mappings, the results panel display the list of results returned for that query. A little star is added close each result allowing the system to collect feedback from users. A button is added at the end of the results to save trust values to store this feedback. As we explain before, PowerAqua extends

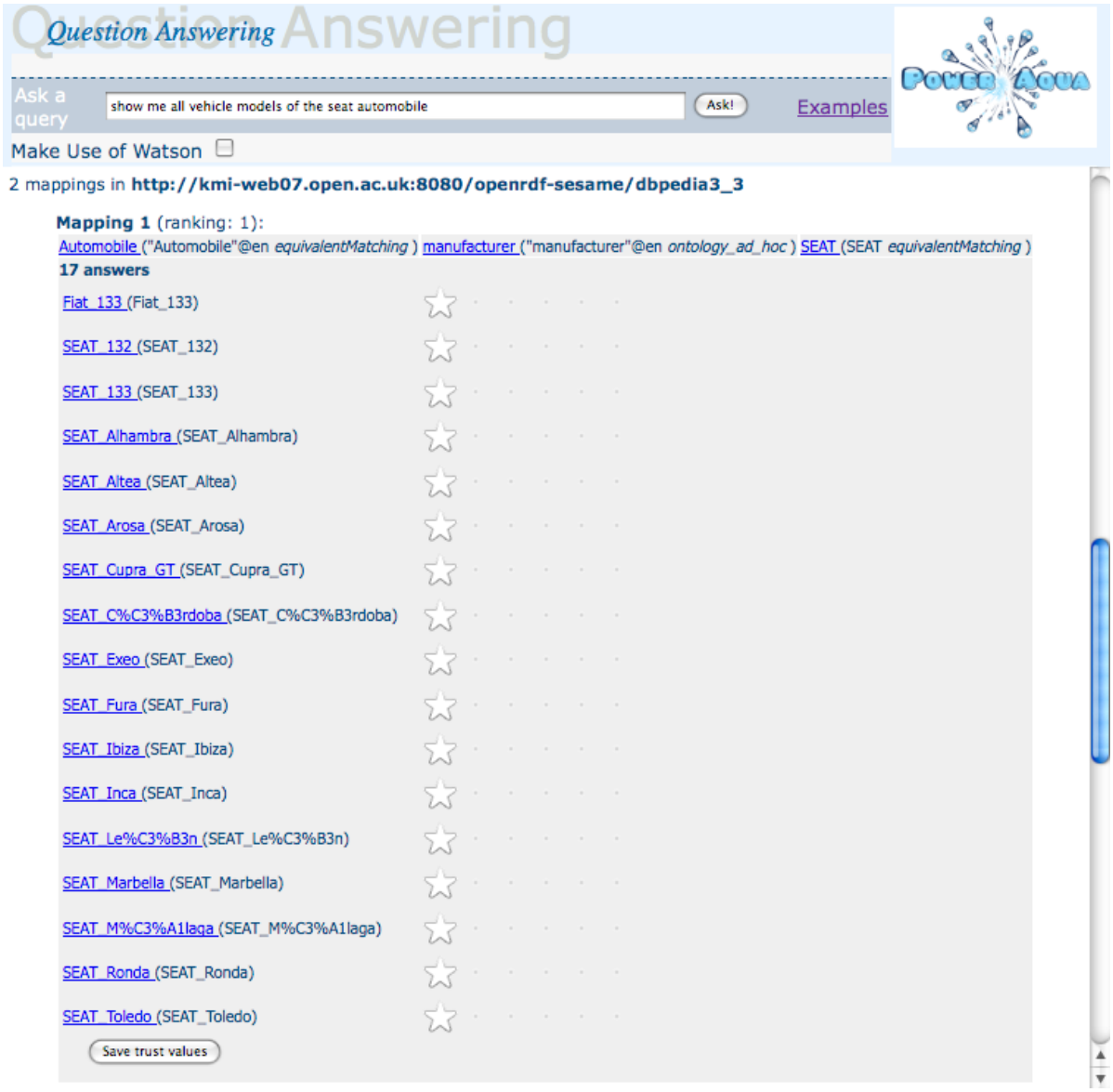


Figure 3.12: ontology mappings found in the DBPedia ontology

the ontology search capabilities in a multi-ontology scenario. For instance, a query like show me all type of vehicles is answered not only by FIAT but by 18 ontologies in total (ATO, KIM, aktivesa, mid level ontology, fao agrovoc, tapfull, chevrolet, coastal ontology, KIMO, SWETO, SUMO, nciontology, etc). In order to integrate the answers coming from all those different ontologies, PowerAqua makes use of light-weight fusion techniques developed within the X-Media project. An example of how the fusion is per-

formed to obtain a final set of answers is shown in Figure 3.13. This screenshot shows the different vehicles obtained in ontologies like ATO, KIM, activesa or the FIAT ontology among others for the query show me all types of vehicles. A link 'Group identical answers' is shown in the top part of the results panel in order to merge equivalent results provided by all these different ontologies and give a final answer to the user. As we can see two main enhancements can be reported from the integration of PowerAqua with the X-Media technologies. On the one hand, the integration with the X-Media repositories allows users to write more expressive queries in the form of NL, where different relationships between entities can be explicitly express as part of the user needs. This helps to generate more accurate answers, increasing the effectiveness and satisfaction of users. It also keeps the simplicity of the interface, since the user does not need to learn any query language or interact with complex query interfaces in order to express his needs.

On the other hand, the integration of PowerAqua with the fusion algorithms of X-Media allows the integration of partial answers from several ontologies and retrieved results that, without the fusion process, would be impossible to retrieve. It also helps to identify the same answer coming from different data sources, which significantly reduces the noise of the final set of results retrieved to the user. Examples and details of how fusion has been integrated and evaluated within PowerAqua can be found in [Lopez et al., 2009a] where the introduction of merging capabilities within the system has proved to enhance the quality of results, therefore increasing the effectiveness of users.

3.4 Integration with X-Media Infrastructure

The following section covers the benefits in terms of user, system and performance achieved from the integration of different X-Media user interfaces.

SemSearch is also integrated as part of the X-Media infrastructure within the Fiat noise use case [Giordanino et al., 2008]. In this case, only the information retrieval capabilities of SemSearch are exploited and integrated within the Fiat portal. SemSearch performs seven specific functionalities or queries for this use case:

- Querying competitors cars: returns a list of cars with the associated spectrum information.
- Querying predicted spectrum: returns a list of spectra with detailed values.
- Querying similar spectrum: returns a list of spectra with detailed values.
- Querying recommender components: returns a list of components.
- Querying better solutions: returns a list of competitor cars that have better solutions on the given components.
- Comparing two spectra: returns a list of spectra that are associated with the given vehicle

The user interfaces also varies according to the functionality. The Fiat portal user interfaces have been reported in [Giordanino et al., 2008] but as a main example we include here the interface of querying competitors cars shown in Figure 3.14. As we can see the interface is divided in two main parts:

- The summary panel (located in the top part)
- The work area panel (located in the bottom part)

While the summary panel provides basic information about the study, the work area show the results obtained by SemSearch in the different steps of the study. Therefore, as we can see in the figure, the integration of SemSearch search capabilities within the Fiat portal interfaces constitute a main achievement in terms of the system functionalities. At the same time, the indexing process of SemSearch results in a significant decrease in the time the user needs to retrieve an answer which is translated into a significant increase of user efficiency and satisfaction.

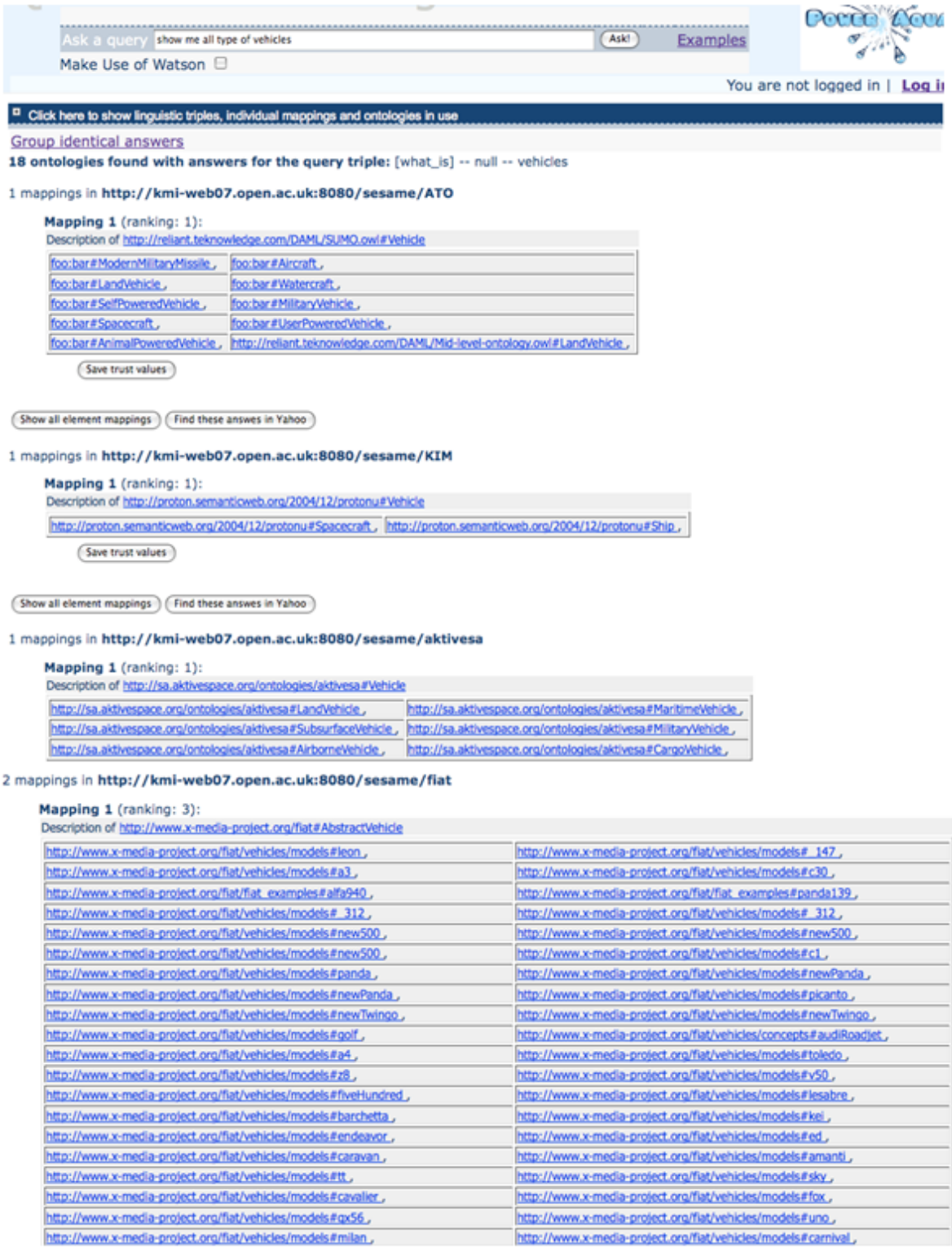


Figure 3.13: ontological answers to the query: give me all type of vehicles in the FIAT ontology

Figure 3.14: Querying competitors car

4 Integrated Searching and Browsing within the XMediaBox

4.1 Legacy Corpora Search

Legacy data plays an important role in the retrieval of the knowledge that informs decision-making in enterprise organisations and other similar environments. Using the Issue Resolution (IR) use case for the Rolls-Royce aerospace engineering domain this chapter illustrates information and knowledge retrieval via ontology- and keyword-based search for text and cross-media documents, and content- and feature-based image similarity search for image and cross-media documents.

4.1.1 K-Search & K-Forms

K-Search, developed as part of the IPAS project[1] [Bhagdev et al., 2008a] provides hybrid search – user-specified combinations of ontology and keyword search, in order to increase the ability to retrieve knowledge from large, legacy data stores. K-Search uses form fields mapped to an ontology in a web interface (see Figure 4.1) to support the construction of simple boolean (OR and AND) queries using different combinations of keyword, keyword-in-context and ontology only searches. This removes the need for users to learn formal query syntax, while supporting a simple, flexible method for information retrieval that increases the capability for information and knowledge retrieval for the typical knowledge worker.

A comparative evaluation of the hybrid search afforded by K-Search recorded precision of 0.85, an increase of 51% over keyword only search, and recall of 0.83, an increase of 46% and 109% over keyword only and ontology only search respectively, for the ranking of the first 20 and 50 documents in a result set [Bhagdev et al., 2008b]. K-Search has further been evaluated in a long term trial at Rolls-Royce and shows the potential to results in significant savings in the resources required to extract information from the large legacy data stores in regular use.

K-Forms uses web forms to support ontology-based knowledge acquisition [Bhagdev et al., 2008a]. K-Forms is used with K-Search to support the capture and retrieval of information and

[1] IPAS - Integrated Products and Services - a project funded by the UK Technology Strategy Board's Collaborative Research and Development Programme and Rolls-Royce plc, to study knowledge transfer between the three worlds of new service design, new product design and the continued operation of existing services and products. (See also `http://www.3worlds.org`)

knowledge in the modern enterprise organisation, by providing a more structured method for capturing and reporting the procedures followed during normal working activities. Backed by a semantic repository, the creation of a new form results in the transparent construction of a backing ontology, with concepts and properties mapping to form fields and relations between fields as appropriate. Re-using regions of existing forms results in the automatic creation of links across their corresponding ontologies, supporting the retrieval of related information across different corpora. The flexible support for data capture supports the work of the multiple, inter-connected communities of practice (CoPs) typically formed in such domains to carry out knowledge-intensive activities. Each CoP is able to generate custom forms directly from relevant parts of the formal organisational ontology, or by extending this ontology to provide a community-specific perspective on the data and knowledge required for their work. The links automatically created to the formal organisational data stores ensures that other CoPs are able to make use of the knowledge generated by other (related) communities, from the perspective that best suits each CoP's requirements. A client-server architecture allows multiple clients to connect to a single server, so that the results of the interaction of each CoPs with the knowledge stores are immediately available to all others connected to the central server and repository. Offline clients use a delayed update mechanism to synchronise with the server.

Figure 4.1 shows the results obtained from a simple keyword-in-context query: Product Family = 'Trent 800'. A single result is displayed, with annotated sections of the document colour coded to highlight matches for specified concepts. This version of K-Search, integrated with K-Forms in a single user interface (UI), allows users to add new annotations to search results. The annotations are immediately available for future searches; the new annotations are however flagged till they are validated, in the Rolls-Royce environment, as is typical for other similar domains, usually by a human gatekeeper.

K-Search and K-Forms are being used to support information and knowledge retrieval in the X-Media project. The integrated knowledge acquisition and retrieval tool is in use for the Rolls-Royce IR and Experimental Vibration use cases. It is also being ported to the Bicycle Brakes public use case.

4.2 Visual Browsing - XMPlots

The *XMPlots* module was developed to support the visual exploration of legacy corpora, using multiple views (or lenses) customised to reveal patterns within the data by focusing on common, significant properties found across multiple document types. Document properties are captured as metadata based on a pre-specified domain ontology. This implementation focuses predominantly on time and geographical location. For the Rolls-Royce domain, for e.g., plotting the properties of different documents against a timeline allows a history of various components, systems and entire product families to be visualised against other attributes of interest. Figure 4.2 shows the module design, and [Petrelli et al., 2009] provides further information on the design rationale and

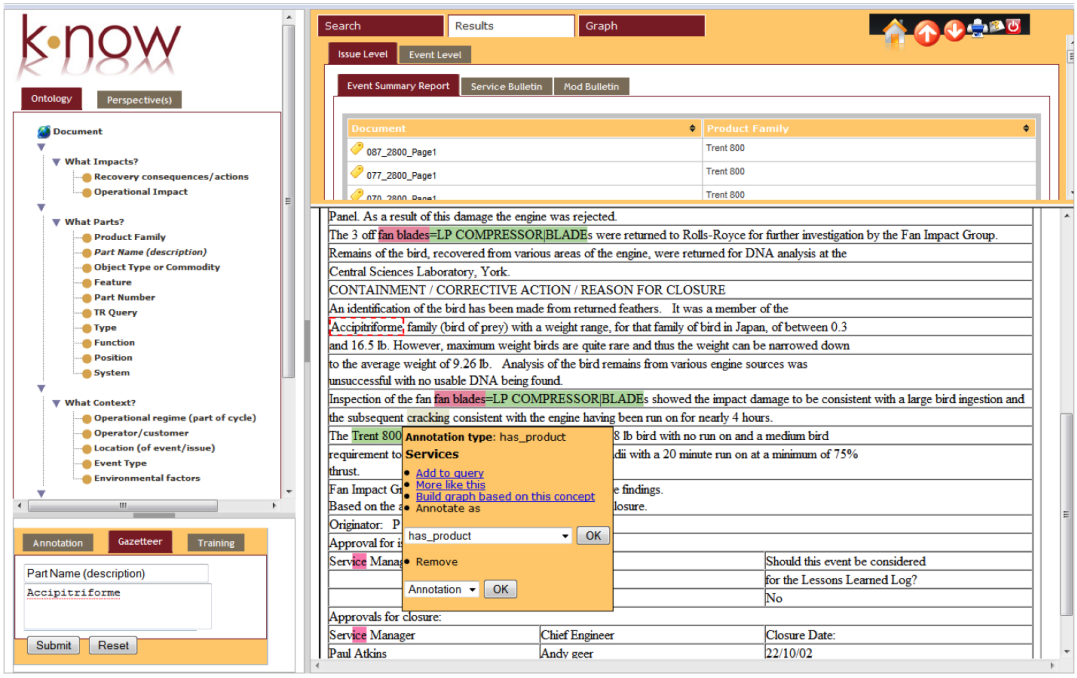


Figure 4.1: The K-Search hybrid search tool, integrated with K-Forms in a web-based UI, to provide seamless, ontology-based knowledge capture and retrieval.

a preliminary evaluation of the user interface.

The *Timeline* plot may also be used to obtain a direct comparison between any two defined properties in one or more corpora, by using a domain (or other relevant) ontology to map across corresponding properties in related corpora. For the RR IR scenario, extracting information on the geographical location of incidents, where this is available, provides additional information that contributes to the resolution of issues that occur. The *GeoPlot* reveals patterns in the data where topology and climactic conditions influence the occurrence of specific issues. Applying a filter to, say, component or Product Family, or at an even higher level of refinement, serial number, allows the user to focus on a specific component or engine type. Plotting the result set against time reveals patterns in the history of the component or engine of interest, overlaying this on the geographical plot reveals yet another perspective on the data set.

The (visual) filter component, which is used across all views, is built based on the ontology used by K-Search to index the data visualised. This focuses the visual browsing obtained on the metadata contained within each document. Each object property defined in the ontology maps to a filter, based on the data type for the property. A numerical data type, for instance, will be used to build a query slider, with the range spanning the minimum and maximum values for the complete data set visualised, e.g., that for flight cycles illustrated in Figure 4.3. Similarly, date data types map to the use of range filters. Nominal data may be used to build list and/or checkbox filters, where one or more distinct items may be selected, e.g., airport codes or location identifiers. Free text fields are provided where an object property represents an open set, e.g., report author.

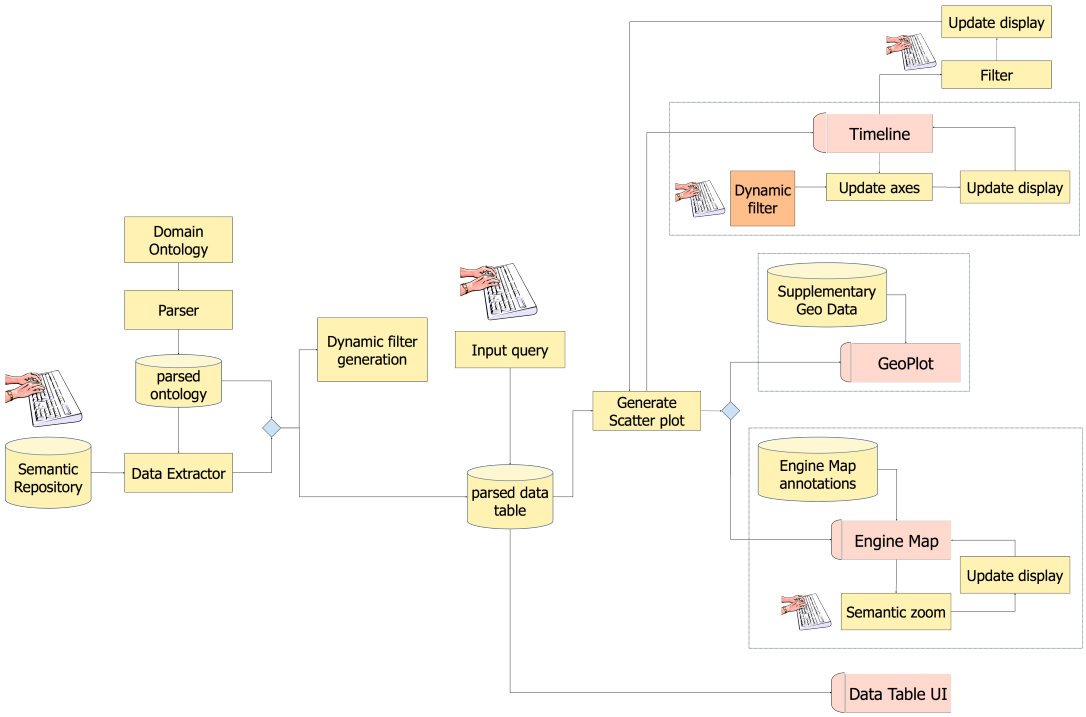


Figure 4.2: The architecture diagram for the design of the XMPlots module, showing the interaction between the alternative views available for browsing the Rolls-Royce legacy data repository.

Using the different visualisations in concert, especially across multiple, related corpora, supports the identification of the causes and/or drivers of an issue by highlighting the links across data and properties stored in distributed locations and repositories. One example is the recognition that while the failure of a specified component may occur at a specified location the root cause of the issue may be due to a flight path over or through another location, probably due to geographic and/or climactic conditions (such as increased ingestion of dust in desert areas leading to greater friction and wear). A second example is the differences in maintenance and inspection procedures in different locations leading to variation in the reporting of issues across locations. The visual exploration enables greater insight, allowing users to focus quickly on regions of interest, within the context of the overall data set. The visual filtering method further allows users to quickly expand or restrict queries to extract closely related information. Figure 4.4 provides a snapshot in the use of XMPlots.

A quick overview of the Timeline view (top, left, in Figure 4.4) indicates, for the plot of (flight) cycles since new (CSN) against event date, a gradual increase in the number of events reported along both axes, which would be the expected trend. There are however three major gaps in the plot for three date ranges - indicating either a potential anomaly in the data store or perhaps a significant event such as a change in design, maintenance and overhaul procedures, or the introduction of a new product. Further examination of the reports on either side of each gap should provide more information on the pattern that

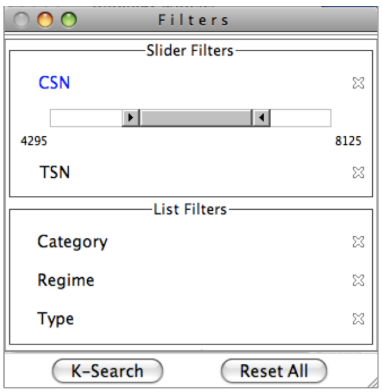


Figure 4.3: The filter applied to the visual displays in Figure 4.4. A single filter – a range query slider for flight cycles – is used to highlight a region of interest in the display.

emerges. A single document is found in a region with immediate sparcity of documents, albeit lying in the overall most dense region of the display. This indicates what may be a significant event or anomaly; this document is extracted (bottom, right, in Figure 4.4) in order to allow its contents to be examined.

Moving to the geographical layout (bottom, left, in Figure 4.4) there appears to be a large concentration of events in southeast and far eastern Asia, with a second, smaller concentration in central Europe. The user may decide to examine the influence of geography and climate and location-specific procedures on the events recorded, in order to determine if any of these contribute to the patterns observed.

4.3 Integration with the XMediaBox User Interface

The XMediaBox, developed to provide a common UI for the different modules that support the investigative analysis process in complex scenarios, provides a single front end from which the search and browse modules described may be accessed. The current prototype provides access to the independent search modules; however each requires independent querying that returns exclusive result sets. The ultimate aim is however to provide more fully integrated search that also learns from the actions of users within the integrated XMediaBox user interface and propagates these to a cycle of learning and enrichment of the underlying semantic repositories, in order to obtain increasingly higher precision and recall for information retrieval for all search modules.

Work is also in progress to propagate all queries formulated within the XMediaBox to all search methods, to prevent the need to repeat similar searches in the different tools. This is already available for the *retrieval of similar* documents from a selected knowledge object stored in an XMediaBox session. A keyword-in-context query is formulated from the metadata attached to the knowledge object. This is then passed to the KSearchAPI tool, which enables transparent querying of the K-Search semantic repository, in order to retrieve matching documents from the K-Search repository. For an image document

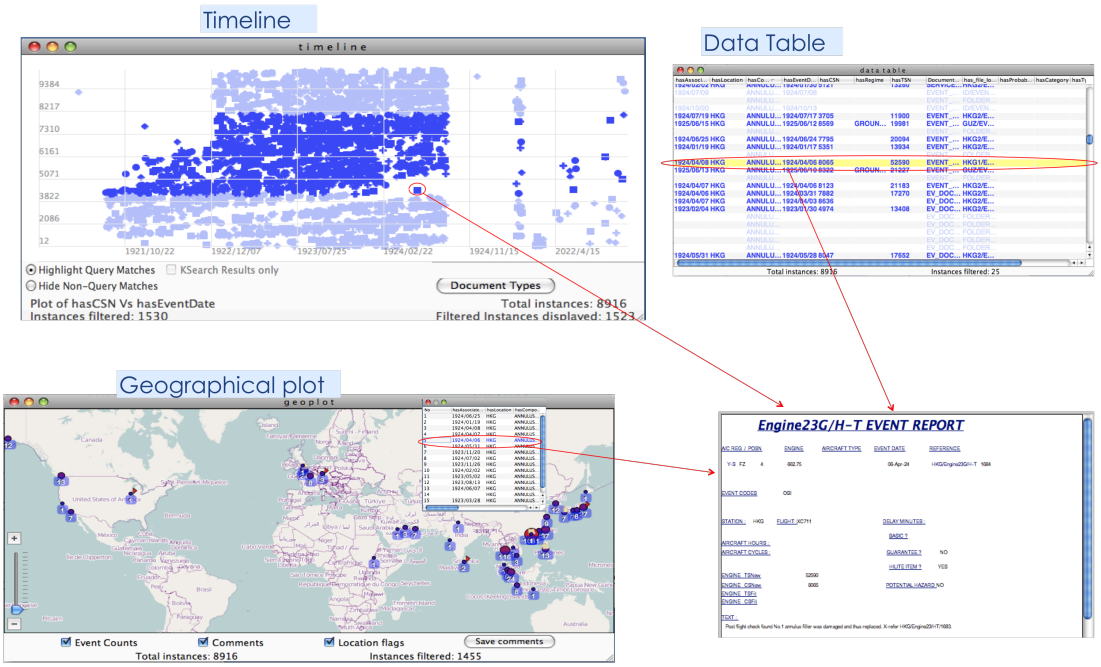


Figure 4.4: A snapshot of the Timeline, GeoPlot and Table views, for a filtered data set, to highlight the different patterns revealed by each perspective on the data set. A single document lying in a region with immediate sparcity of documents in the Timeline view is displayed on the bottom, right.

an image similarity search query is also formulated from the input and passed to the corresponding search module. It should be noted that the results returned from this integrated search are not ranked; the user is left to select from the result set those items that are most relevant to their current task.

4.3.1 K-Search

Starting from the second phase of the X-Media project the K-Search UI is no longer integrated within the XMediaBox. A link is provided to the web interface from the search tab of the XMediaBox, and selected search results are imported into the XMediaBox UI by dragging the link to each result URL into the XMediaBox clipboard. This action generates and stores a reference to the HTML representation of the source document and triggers the X-Media indexing modules for the document in question, by placing a call to the X-Media Kernel (§4.4 provides further detail on the integration with the underlying X-Media architecture).

The KSearchAPI tool is also used in the initial legacy corpora search for the IR use case with a restricted ontology focusing on the requirements of the IR use case, based on the ontology created for the Rolls-Royce domain as part of the X-Media project. The XMediaBox system formulates a query (transparently) by parsing the (free text) description of each new issue using a terminology recognition and disambiguation module

to extract key terms and match these to pre-specified concepts. The KSearchAPI tool then runs this query against all document repositories indexed using the ontology, and any others for which mappings are found to the ontology in use. Search results are presented in a table, collating the results for each concept for which information is retrieved for each document in the result set. While the result set is not ranked, the table columns may be sorted (using natural ordering, according to data type) in order to cluster the results on a concept of interest.

4.3.2 Image Similarity Search

Content and feature-based image similarity retrieval and clustering modules have been developed in WP6 as part of the X-Media project. The XMediaBox provides a UI for the search module, allowing end users to submit an input image (by dragging from the clipboard into the input field in each form) as the subject of a query to the X-Media semantic repository. Figures 4.5 and 4.6 show the use of the UI to display snapshots of the input image and the result set, which includes both images and the cross-media documents containing images matching the input.

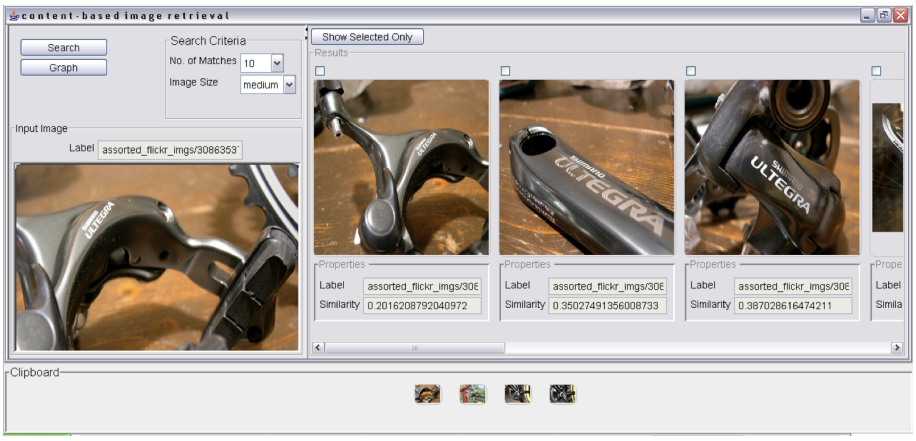


Figure 4.5: Content-based image retrieval - the UI ranks, from left to right, the search results returned, and displays the similarity value (between 0 and 1) that indicates the level of similarity to the input image.

Users may import individual results into their knowledge workspace by dragging selected results into the clipboard. This allows (further) annotation to be performed, explicitly by tagging images with relevant concepts from the ontology browser in the XMediaBox, or by attaching comments to the images. Implicit, transparent annotation may be obtained as a result of users' interaction with an image, e.g., by attaching as evidence to the root cause analysis structures during issue investigation (see §6.1.1).

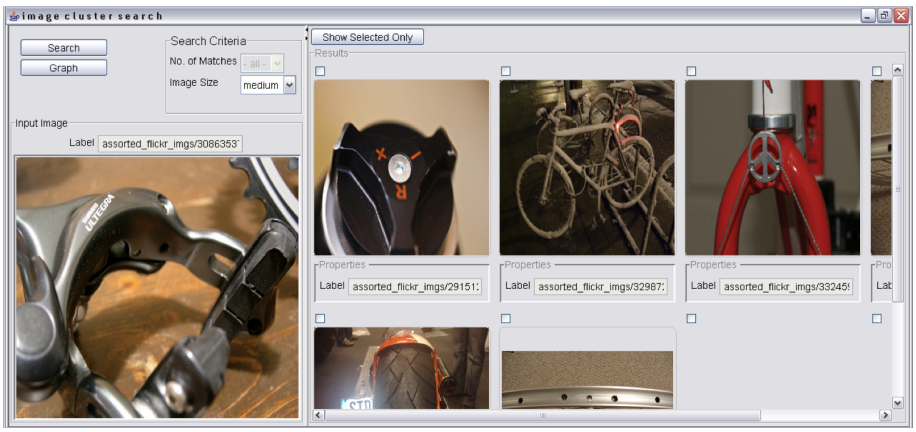


Figure 4.6: Content-based image clustering - the non-ranked result set is presented in a grid, retrieving not just visually similar images but also those that fall in the same category as the input image, e.g., all components of a specified type.

4.3.3 XMPlots

XMPlots is integrated into the XMediaBox as a plug-in. The current version of the tool is domain-dependent, customised to visualise legacy Rolls-Royce data indexed and stored in the K-Search repository using the restricted ontology described, such that selected metadata for the indexed documents is highlighted as users browse the document content. As is available for the K-Search UI, documents displayed in the XMPlots modules may be imported into the user's workspace by dragging the visual references in the plots and the links in the table view into the XMediaBox clipboard.

4.4 Integration with X-Media Infrastructure

The X-Media Kernel indexers are triggered when a new document (reference) is imported into the XMediaBox – from the K-Search repository or the file system – as part of the user's information retrieval and analytical activity. This generates a new XMUri, the handle subsequently used to retrieve the actual document and its metadata. Document metadata is updated as users interact with the knowledge contained within the XMediaBox, e.g., by tagging a document using the ontology browser contained within the XMediaBox, or using key terms (transparently) extracted from comments attached to a document, using a domain-specific terminology recognition module. The X-Media image index is also updated when an image or a cross-media document is imported into the XMediaBox. The new knowledge thus obtained becomes available for re-use beyond the immediate point of generation.

The current implementation maintains a separate repository for K-Search, due to conflicts in the server models for JBoss (used for the X-Media Kernel) and Apache Tomcat (used for K-Search) and implementation-specific requirements for the X-Media Kernel and K-Search. Full integration of the search technology may involve the implementation

of pipes to allow the X-Media Kernel indexers to run on the K-Search repository, in order to allow querying of the complete K-Search repository using Kernel-specific tools without the replication of the K-Search repository. This would be especially useful for cross-media documents, as K-Search does not include feature- and content-based image indexing and retrieval.

5 Email and Process Support

The semantic email tool COSIMail (cf. Figure 5.1) adds additional linkage among transferred messages and attachted files. A detailed description of the utilization and evaluation of COSIMail has already been reported in [Franz et al., 2009, Dadzie et al., 2008]. There, it has been shown that semantic email can significantly improve the efficiency of knowledge workers. StickyMail is a further semantic email extension that has been

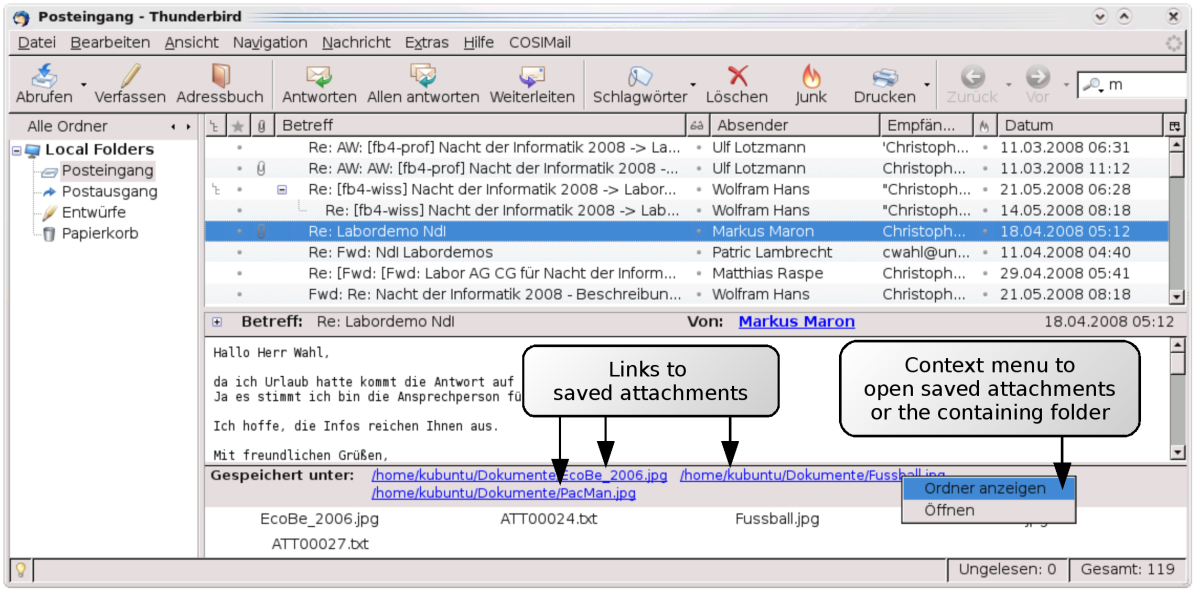


Figure 5.1: The Semantic Email Tool COSIMail

developed in X-Media. StickyMail builds upon the exploitation of X-Media research conducted in workpackage 1, where the tracking and representation of process knowledge has been targeted. The outcome of these works are DiALog, a formal model for process knowledge [Ringelstein and Staab, 2009, Ringelstein and Sizov, 2009a], and a logging mechanism for process provenance (PPL), the implementation of the DiALog approach [Ringelstein and Sizov, 2009b]. The PPL is implemented as JBoss handler, which is integrated into the X-Media kernel.

5.1 StickyMail

Like COSIMail, StickyMail has been implemented as an extension for the Thunderbird email client. Accordingly, it can be easily installed by its add-on manager (cf. Figure 5.2).

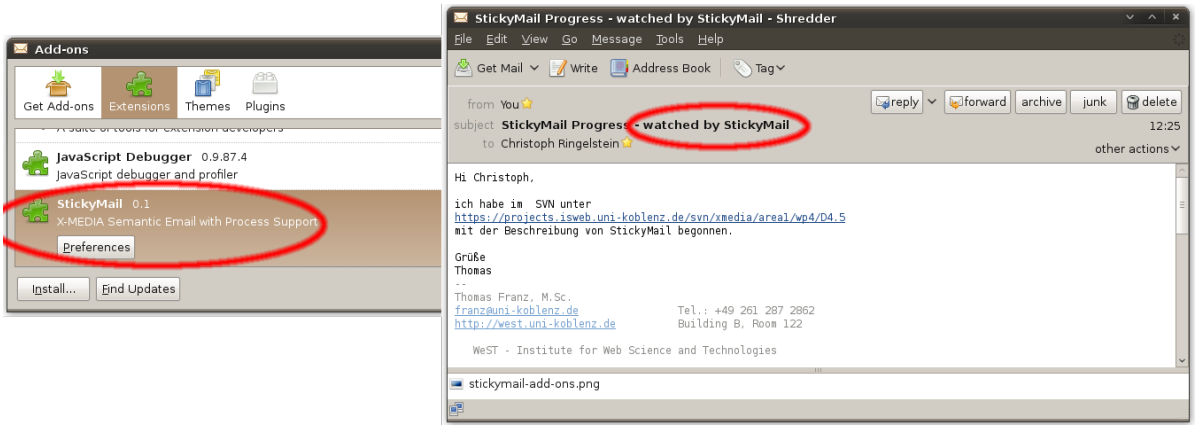


Figure 5.2: StickyMail Interface

Whenever an email is sent that has an attachment, StickyMail logs information on the email and the attachment, e.g. the name of the attachment and its path on the local filesystem of the sender. These logs are automatically stored in the X-Media Knowledge Base. Upon the sending of an email, StickyMail indicates its availability by adding the text - *watched by StickyMail* to the subject header of outgoing messages (cf. Figure 5.2).

5.2 Integration with X-Media-Kernel

To use PPL with any kind of application, including Web services, we built a Web service providing operations for logging of provenance information (PPL-WS). The Web

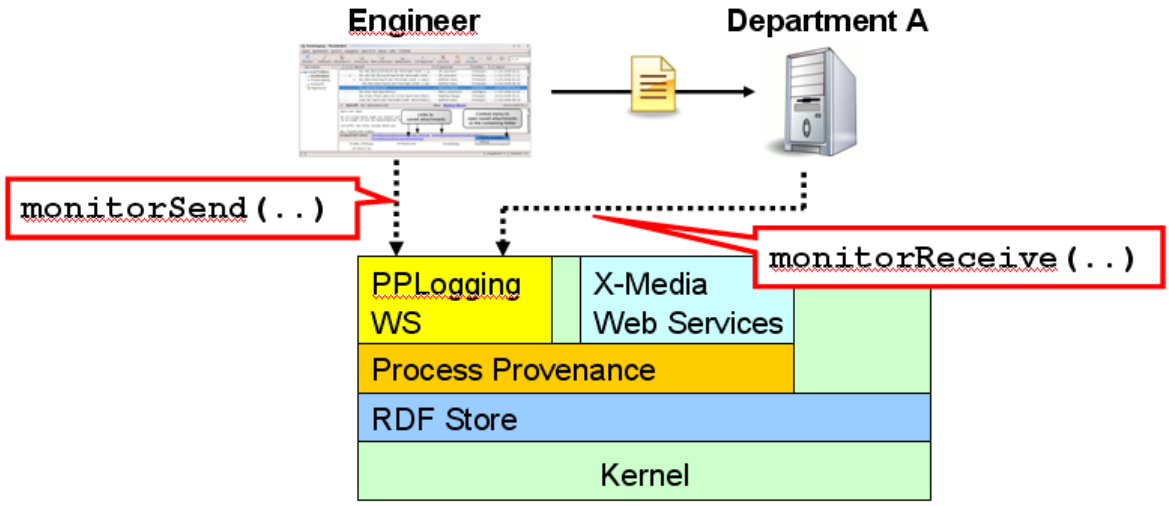


Figure 5.3: StickyMail Architecture

service is deployed on the application server provided by the kernel. This allows for fully

integrating the Web service with the rest of the PPLogging mechanism (cf. Figure 5.3).

The Web service provides operations that can be used by applications to collect process provenance. The Web service provides these operations also to applications that are not part of the kernel. The StickyMail client, which is such an application, makes use of the operations of the PPL-WS whenever an action or event occurs that should be logged (e.g. the sending of an email). The PPL-WS operation makes use of the PPL mechanism and stores the information into the central knowledge base provided by the kernel.

5.3 Integration with LENA

By running StickyMail, metadata on transferred files logged through PPL is available in the X-Media knowledge base. Thus, this data can be searched and visualized by X-Media tools, e.g. as presented in Section 3 and Section 2. For instance, using the browser LENA, attachments of incoming mails can then be further inspected to find out who has dealt with the attachment, to whom it has been sent, and when. Figure 5.4 shows

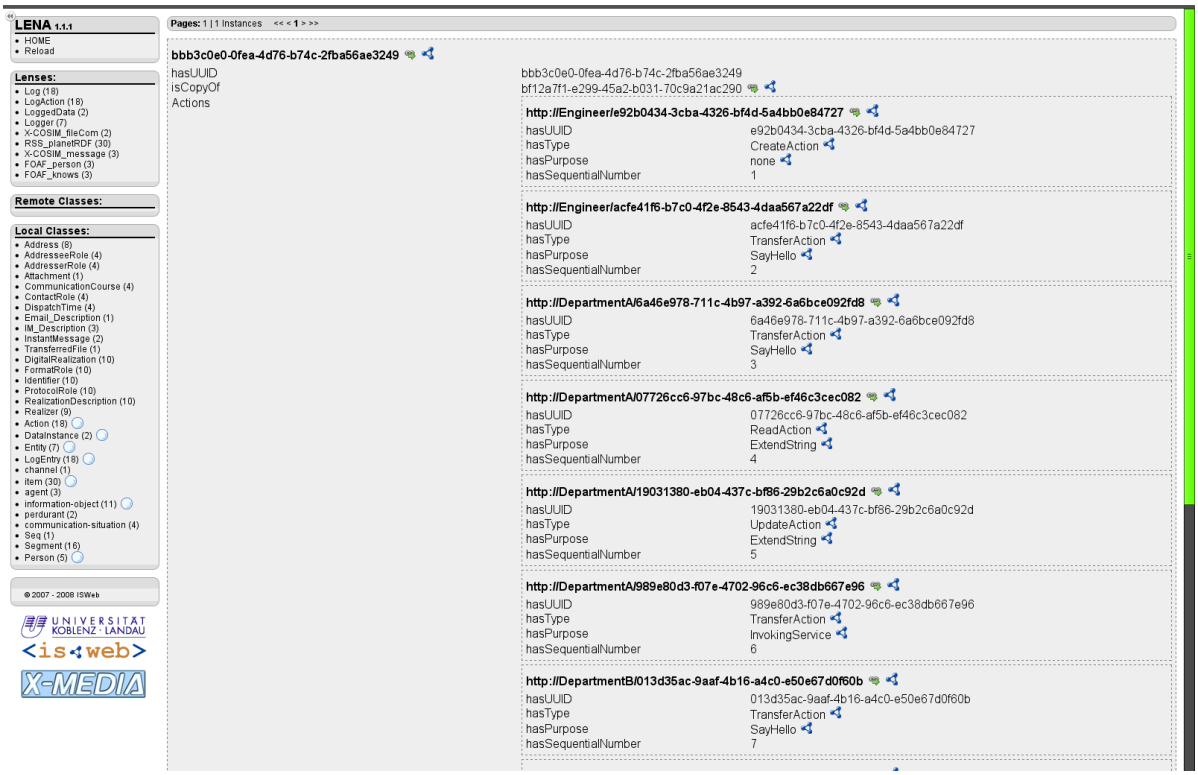


Figure 5.4: LENA Lens to View Actions Performed on a Specific Data Instance

how such information can be browsed using a lens in the LENA browser that shows for a data instance, its UUID, if it is a copy of another data instance, and a list of action taken on the data instance. Each of the actions can be further inspected by clicking on it.

Figure 5.5 shows how the details of a logged action can be browsed with LENA. The screenshot shows the specific action type, *TransferAction*, when it occurred, its purpose label and further links, e.g. to browse back to the data instance (by following the link *performedOnDataInstance*) or to the entity that logged the action. The latter can also

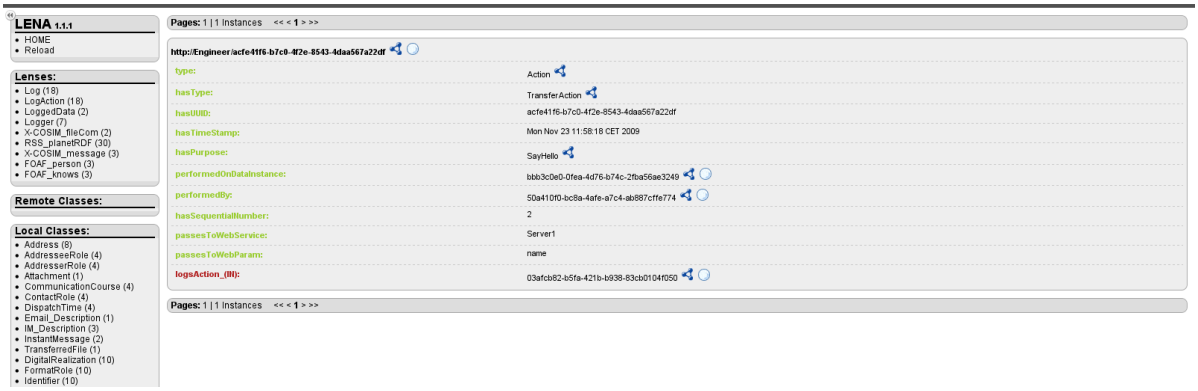


Figure 5.5: Detailed View of a Logged Action

be inspected as shown by Figure 5.6. In this case, the logging entity is an engineer that has the address *Universitaetsstrasse 1; 56072 Koblenz*. Using such and similar

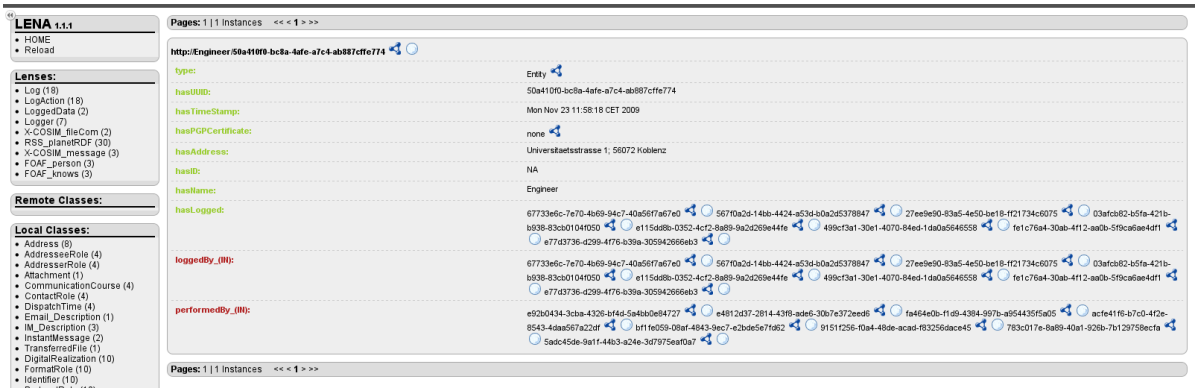


Figure 5.6: Detailed View of a Logging Entity

views, X-Media users can track in a very finely grained manner what happens to the documents they created and what the provenance of a document is. Such information is extremely important in engineering scenarios at X-Media use case partners where the determination of the trustworthiness and validity of information is crucial.

6 Visual Hypothesis Analysis

6.1 K-Views

K-Views, *Knowledge Views*, was originally designed to support the visualisation of the results of search and analysis in WP4 by providing alternative lenses (or perspectives) on the knowledge contained. It now includes support for hierarchical graph analysis, as is required for instance, in the hypothesis investigation during root cause analysis in Issue Resolution (IR) at Rolls-Royce. K-Views has also been applied to the Bike Brakes public use case, which is used here to illustrate the use of the tool.

Three modules have been built - the:

Analysis Tree which uses interactive tree graph construction to support ontology-guided, visual hypothesis analysis.

Knowledge Graph used to visualise the results of search. The Knowledge Graph also initially provided an alternative perspective for visualising the results of the hypothesis analysis obtained using the analysis tree.

Knowledge Cloud which provides an alternative, co-ordinated (visual) perspective, using a network graph to visualise the results of the analysis obtained using the analysis tree. The knowledge cloud may also be used to browse the structure of ontologies and semantic repositories.

6.1.1 Analysis Tree

Dadzie *et al.* in [Dadzie and Petrelli, 2009, Dadzie and Ciravegna, 2009] describe the use of the *K-Views Analysis Tree* to support visual root cause analysis. The design of this module was guided by the requirements of aerospace engineers at Rolls-Royce for the identification of the root cause of issues under investigation. The module makes use, for this use case, of the Issue Resolution sub-module that forms part of the ontology developed during X-Media for the Rolls-Royce aerospace engineering domain. This sub-ontology described the symptoms and drivers of commonly recurring issues and their resultant effects. The flowchart in Figure 6.1 summarises the customisation of the visual analysis module to support the root cause analysis process.

The aim is to support human reasoning during hypothesis exploration, by (re-)using the formal, agreed knowledge captured in domain ontologies to suggest paths for users to follow during their analysis. The inherently hierarchical nature of the hypothesis investigation lends itself to hierarchical graph visualisation - hence the use of an interactive tree graph to capture users' analysis structure and translate this to a visual

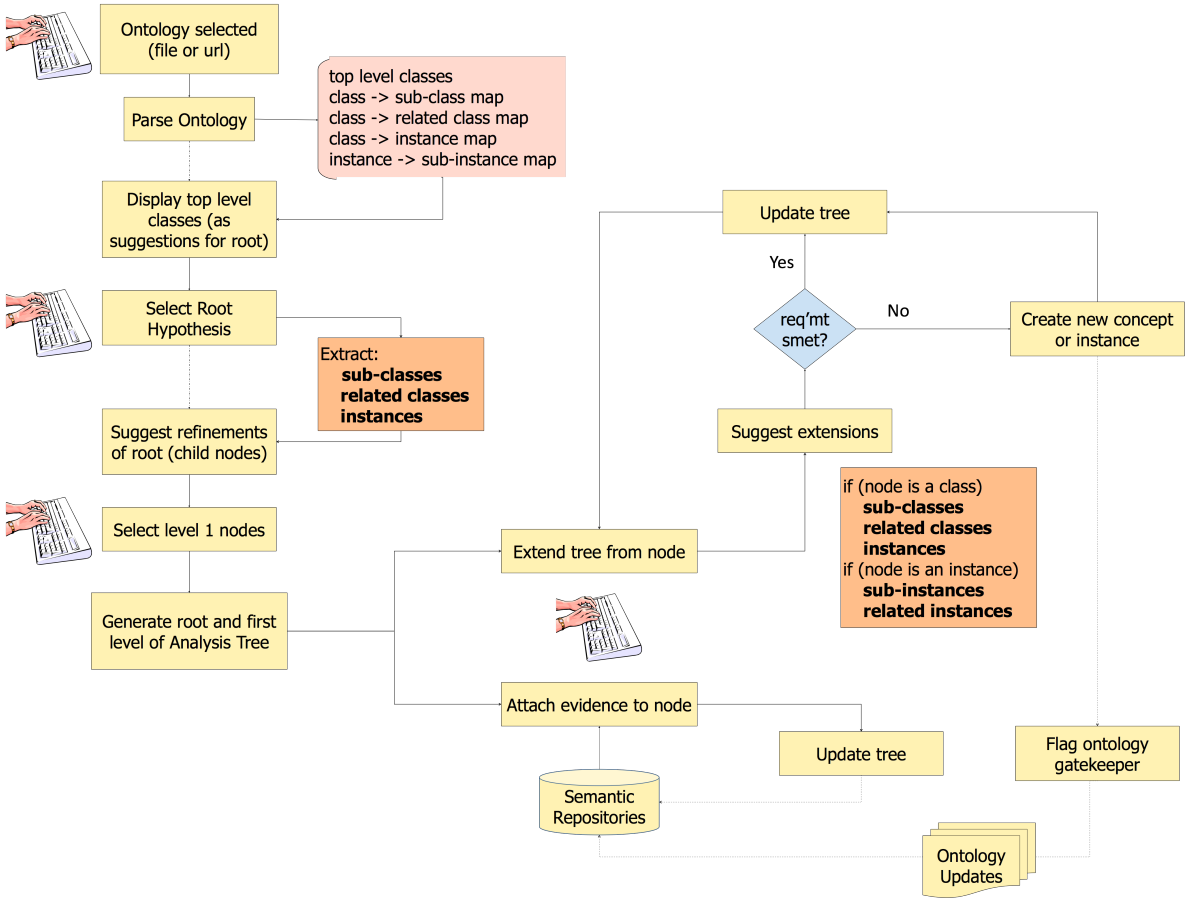


Figure 6.1: Flow chart illustrating the parsing of the backing ontology and the generation of a root cause analysis tree.

representation. The visual overview obtained lowers the cognitive load associated with complex data analysis by harnessing the more intuitive perceptual ability of humans; greater insight is obtained and end users are able to more easily identify regions of interest (ROIs) and previously undiscovered patterns within the knowledge structure that results from the analysis. These requirements are easily translated to other complex, hierarchical investigative analysis for similar scenarios; for each case the analysis is guided by a domain or other relevant ontology. An example is the root cause analysis described in the Bicycle Brakes use case (see [Lavelli et al., 2009]), using a domain ontology built using the ontology engineering tool described in §8 and [Lavelli et al., 2009].

The visual analysis begins with the selection of an appropriate ontology. The user defines the hypothesis from which they wish to start their investigation, based on the information collected as part of the overall investigation. The system then attempts to match the user's input to defined concepts and instances in the ontology. If a match is found suggestions for related hypotheses to explore are presented – these may be:

- a refinement – a sub-class or instance – of the current hypothesis, e.g.,

  - the class Deterioration has an instance Wear, and
  - the instance Wear *has_sub_type* Corrosive_Wear.

- a class for which a defined relation exists from the concept that corresponds to the current hypothesis, e.g.,
  - Wear *has_driver* Material_Properties.

Extensions to the default string matching, in order to capture a wider set of options, include the following:

- a synonym, an abbreviation or a variant of the current hypothesis, e.g.,
  - worn is a variant of Wear.
  - IP is an abbreviation of Intermediate_Pressure.

- a term(s) contained within the current hypothesis, where the hypothesis is encapsulated within a phrase rather than a single word, e.g., worn brake pad may be broken down into:
  - Wear
  - on the *sub_part* pad
  - of a brake
  - which is an *instance* of the *class* component.

  While the phrase as a whole will not match any concepts or instances in the ontology each of its component parts results in a match, allowing the system to suggest an alternative sub-structure for the analysis tree and a set of suggestions for paths to continue the exploration.

The user continues their analysis by extending the tree graph to define sub-hypotheses and alternative hypotheses along with their corresponding sub-graphs. The analysis tree is populated with comments describing users' analysis and (visual references for) evidence for each hypothesis, extracted from documents and other knowledge objects such as conversation threads, meeting minutes and memoranda. The analysis is terminated when all relevant paths have been explored, and for root cause analysis, a conclusion is reached for the root cause of the issue being investigated. Figure 6.2 shows an advanced stage in the investigative, exploratory analysis to determine the root cause of the failure of the brake system on a pedal bicycle (see [Lavelli et al., 2009] for a complete description of the use case and scenario).

6.1.2 Knowledge Graph

The *K-Views Knowledge Graph* was designed to serve two purposes: an alternative perspective on the analysis tree, and to visualise the results of search and analysis. Figure 6.3 illustrates the generation of a knowledge graph from an analysis tree. Relative size of the hypothesis nodes is mapped to the number of evidence items attached to each

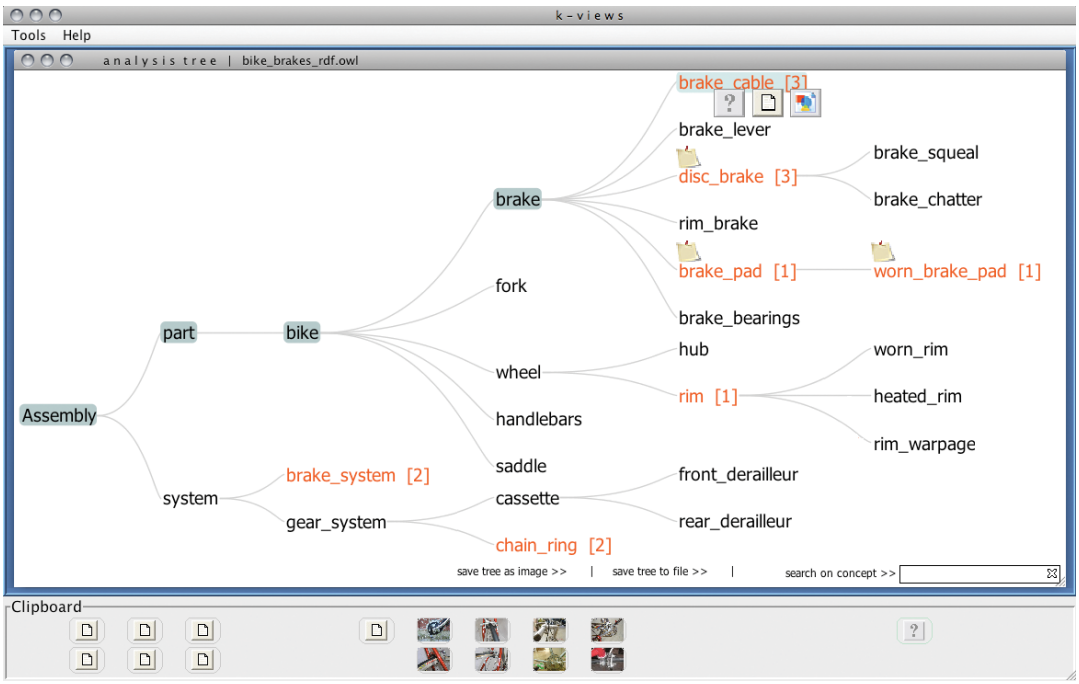


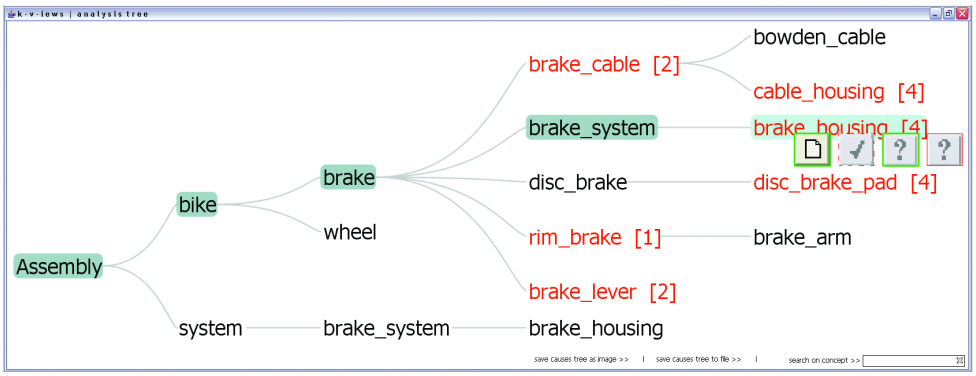
Figure 6.2: Ontology-guided, hierarchical graph visualisation using the K-Views Analysis Tree. Evidence collected (some of which is visible in the clipboard of the XMediaBox) is attached to corresponding hypotheses in the analysis tree.

hypothesis. Additionally, for root cause analysis the colour of the hypothesis nodes maps to the probability that each hypothesis is the root cause of the issue - based on the overall confidence for the evidence attached to a hypothesis.

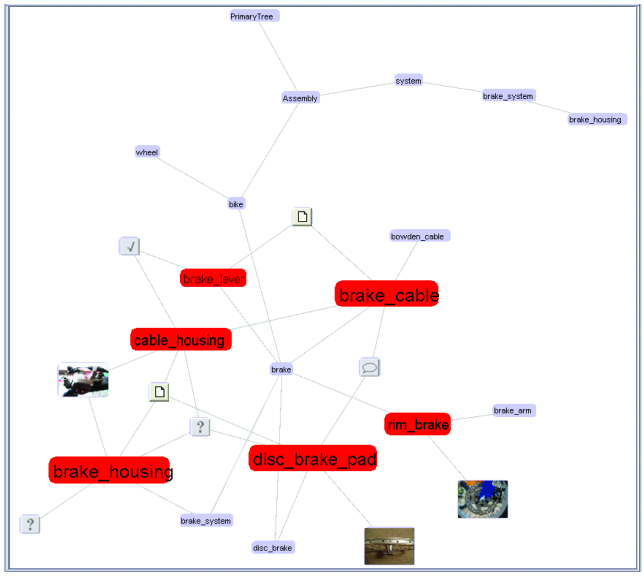
A significant advantage over the tree graph structure is the visualisation of the implicit relationships between node pairs (for which explicit relations are not defined in the backing ontology. For instance, hypotheses repeated in different parts of especially large graphs may not be easily identified; in the graph the need to clone nodes is removed - the relationships such nodes have to all others to which they are connected are easily discerned. A second example is distinct nodes that have evidence in common; this indicates interdependence between the hypotheses represented, information which may not be recognised otherwise.

Graphing Search Results

In a simplified version the knowledge graph is used to visualise the results of similarity search. Relative node size for each search result is mapped to its similarity to the input. Figure 6.4 illustrates this for mode shape similarity search in the Rolls-Royce Experimental Vibration use case.



(a)



(b)

Figure 6.3: Figure 6.3b shows the knowledge graph derived from the root cause analysis in Figure 6.3a. The graph structure allows cross-linking in the knowledge structure to be displayed, highlighting relationships between nodes in different parts of the tree graph due predominantly to evidence held in common.

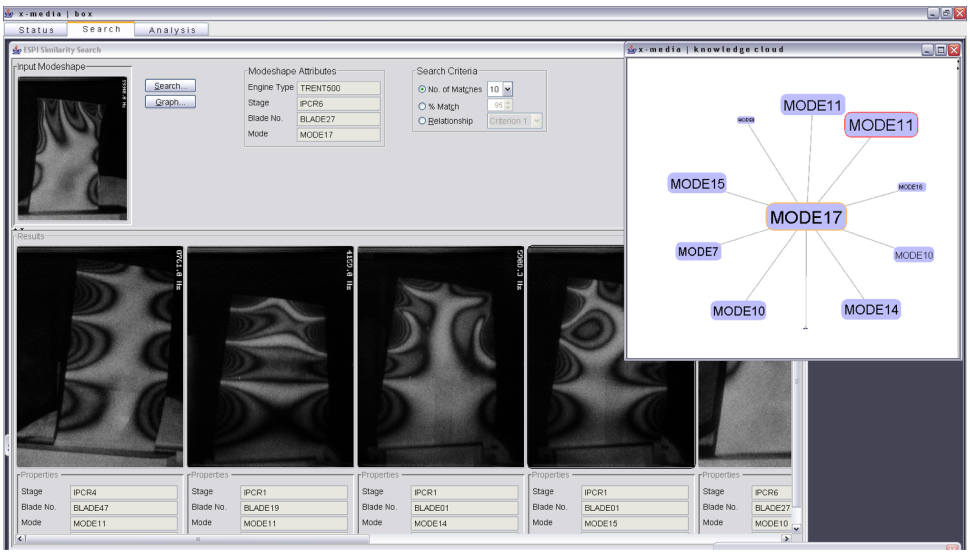


Figure 6.4: The graph in the top, right corner illustrates the use of the network graph to present the results of image similarity search. The node size for each result maps to its similarity to the input in the centre.

6.1.3 Knowledge Cloud

The *K-Views Knowledge Cloud* was developed after the usability evaluation at the end of the first phase of the X-Media project, based on user feedback in the use of the knowledge graph. Rather than simply providing an alternative visual representation of the same data set the cloud provides extra information not directly accessible in the analysis tree. This new perspective places the user's analysis within the overview of the entire semantic repository, allowing a direct comparison between the user's knowledge structure and that obtained by following the structure of the domain ontology. Additional features, illustrated in Figure 6.5, include:

- colour-coded edges to highlight the (primary) relationship between node pairs,
- edge thickness mapped to the number of relationships stored between node pairs,
- variation in node borders to differentiate between:
  - hypotheses in the user's tree that match defined concepts and instances in the backing ontology (thick border),
  - hypotheses in the user's tree not defined in the backing ontology (broken border), and
  - concepts and instance in the ontology with a defined relation with a hypothesis in the user's analysis tree, but which are not present in the user's tree (no border).

Advantages over the knowledge graph (cf. Figure 6.3b) include a greater overview of the complete knowledge repository and more intuitive, visual support for the identification

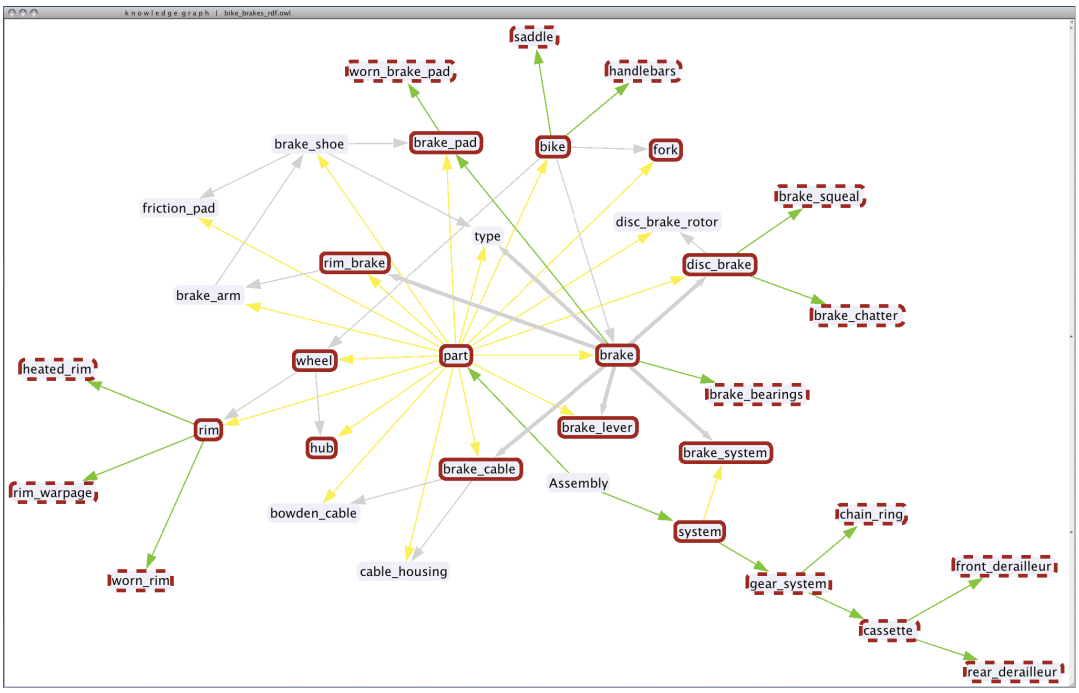


Figure 6.5: The corresponding knowledge cloud for the root cause analysis in Figure 6.2, showing the user's analysis within the overview of the backing ontology. (Note that the evidence items are hidden in this view.)

and exploration of new paths and ROIs. The use of colour and size encoding for the edges also provides more information to the user about the relationships between node pairs without the need to explicitly query the underlying data structure.

6.2 Integration with the XMediaBox User Interface

The XMediaBox was developed to provide a common user interface for the different modules used to support the IR process and analysis for other similar, complex scenarios. The XMediaBox has also been used with the Experimental Vibration and Bicycle Brakes use cases (see [Dadzie et al., 2008, Lavelli et al., 2009]). Independent modules communicate with each other via the X-Media Kernel (by listening to and handling events triggered by other modules), and the semantic clipboard in the XMediaBox (which stores references to the knowledge objects, e.g., documents and conversation threads used within each session).

The K-Views module is used as a plug-in within the XMediaBox. The end user is provided with a visual representation of their analysis, to support the greater intuition obtained through advanced human perception. The analysis tree provides a structured method for investigating ontology-suggested and user-defined hypotheses for the root cause of an issue (or other similar analysis). The alternative perspective, the knowledge cloud, uses a more relaxed visual representation to reveal relationships hidden by the

tree structure, and further, provide an overview of the user's analysis within the overall knowledge structure provided by the backing ontology. This supports more readily the identification of related hypotheses not already discovered by the user via visual, exploratory browsing.

Evidence for hypotheses in a user's analysis tree, contained in documents and other knowledge objects, are attached to the corresponding hypotheses by dragging the visual representations (and references to each knowledge object) from the XMediaBox clipboard onto the analysis tree. This action results in (transparent) tagging of the documents and other knowledge objects referenced, and a visual reference to the context in which the analysis is performed. §6.3 discusses the integration with the X-Media Kernel, and details the impact this has on the semantic repository and subsequent information and knowledge retrieval.

6.3 Integration with X-Media Infrastructure

The user is able to support their analysis with the knowledge stored in the domain ontology, which represents the formalised knowledge held by the organisation, the experience obtained over time by knowledge workers. As the analysis progresses new hypotheses formulated in each user's analysis tree provide new knowledge that should, once validated, be fed back into evolution of the backing domain ontology, to allow sharing and re-use beyond its immediate use. Figure 6.6 illustrates the contribution the visual analysis makes to the knowledge lifecycle. The new knowledge generated may fall under one of the following categories:

1. a new class or a sub-class of an existing class,
2. a new instance of an existing class, e.g., two new *sub_type*s of Wear are Fretting_Wear and Erosive_Wear, (all of which are instances of the class Deterioration),
3. a synonym, abbreviation or variant of a class or an instance,
4. a new relation (object property in the ontology) between concept (class) pairs, e.g., the recognition that a specified deterioration mechanism has a driver not previously recorded.

For the Rolls-Royce aerospace engineering domain, as is likely for other similar domains, the requirement for a human validator means that the evolution of the ontology is not immediate. However this new knowledge is captured within the individual XMediaBox sessions and is available for searches across sessions. Within each session the knowledge cloud (see §6.1.3 and Figure 6.5) provides an immediate visual update that highlights new concepts and instances encapsulated by user-defined hypotheses.

Updates to the domain ontology should be fed to all other users, allowing the new knowledge to be re-used in other relevant analysis. This is reflected in the update of the metadata in the X-Media semantic repository; documents attached to nodes in the analysis tree that define new concepts or instances should have their metadata updated to

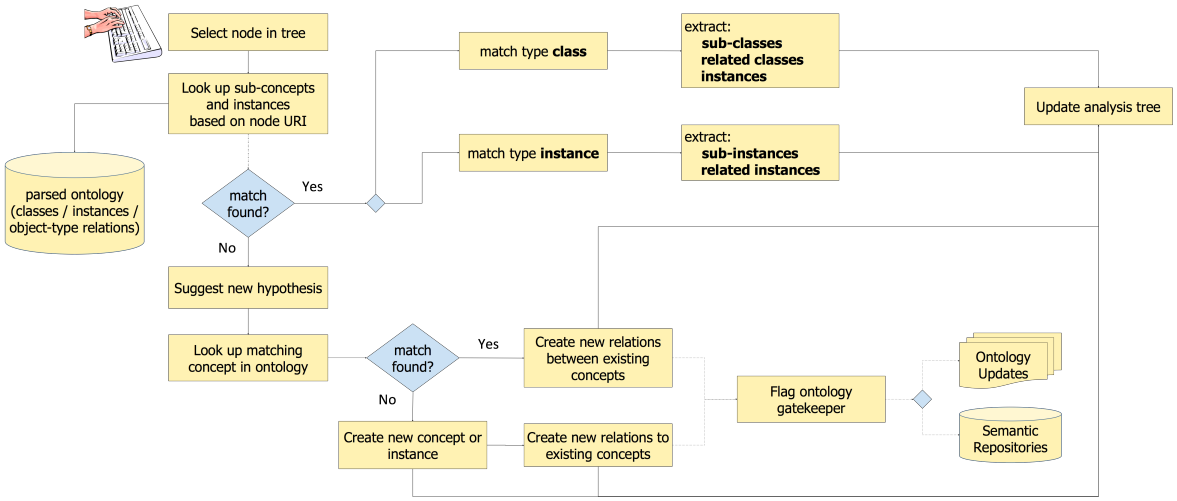


Figure 6.6: Flow chart illustrating the process followed in extending and populating the analysis tree and the resultant update and evolution of the semantic repository and the corresponding ontology(ies).

capture the more specific annotation obtained. A resultant effect is improved information retrieval; the existence of new instances and classes allows further refinement of queries and more accurate retrieval of similar items, based on the richer metadata describing data referenced in the visual analysis.

7 User Feedback Integration

In the process of working the user may notice errors in the main repository produced by extraction or fusion tools. While these errors can be corrected directly in the main repository, the extraction and fusion tools can also benefit from this user feedback and change their behaviour to avoid similar errors in future: for example, a machine learning algorithm can insert an erroneous case into its training set and re-train itself. In order to make use of these user corrections, the system has to provide means to propagate them from the end-user tools to the relevant modules. As described in [Buza et al., 2009], apart from the end-user tools, this requires collaboration of several server-side components of the X-Media architecture, in particular:

- Area 3 (kernel) to pass information about user corrections to relevant Area 2/WP3 modules
- Area 2/WP3 tools to capture and utilize the corrections.

The relevant module, which is responsible for the error, can be determined using the type of error corrected by the user and the provenance of data. The errors in which distinct individuals have been incorrectly merged or there were several individuals denoting the same real-world entity have to be propagated to the fusion module. The errors in which an incorrect statement was introduced have to be propagated to the corresponding information extraction module. The actual module can be determined by the URI of the agent in the provenance descriptor. The information is passed using the kernel event system. The event corresponding to the user feedback propagation carries the following information: actual statements to be deleted/inserted into the main repository Information about the relevant module which caused an error

Events of this type always have to be processed by the fusion module, which performs the actual update of the main repository, and, in case the error was related to fusion, decides about the update of its decision model.

The extraction errors then can be processed by the Area 2 extraction tools. They can subscribe to the user correction events, filter out those relevant to a particular tool, and make decisions about whether the particular error correction can be utilized.

8 Document Annotation

Objective of task 4.2.1 was to develop a tool (i.e. a functional prototype) for the integrated maintenance of a schema and a fact base by means annotating text documents.

In the first two years of X-Media several prototypes of ODF (Open Document Format) based text annotation tools were crafted and demonstrated. It turned out that ODF in principle is well suited to serve as a basic container format for storing and maintaining multimedia (text, images, tables etc.) annotations. However, some problems arose with ODF: while we technically can work directly on ODF there is no mature and well established user front end beside OOo (OpenOfficeOrg). Even as of 2008 (i.e. two years after project start) the OOo API was not well enough developed; worse: transforming input documents like MS Word, ppt, and pdf to ODF proved to be too slow for large scale context. There were also signs that it would be very difficult to establish OOo as an alternative to Microsoft products within the area 4 partners. Evidence suggested finally that a traditional office tool doesn't supporting editing of the type of highly structured text which is needed for a fine grained semantic annotation of small portions of text, i.e. on phrase or even word level. As a consequence of thorough technical evaluations X-Media decided in 2008 to abandon ODF as the overall multimedia interchange format.

Complementary to these technical findings X-Media got a deeper insight into the end user's (industrial) knowledge management contexts of X-Media. It soon became clear that there are several distinct tasks like annotating legacy documents, working with (incl. annotating) new (i.e. "living") documents w.r.t. a given X-Media ontology and extending and amending the X-Media ontologies It also became clear that interactive text annotation driven by users is very expensive in terms of work time. Relying mainly on end users to perform the tasks of annotating documents semantically is not an option to deal with large text corpora. Consequently X-Media decided to make use of supervised and semi supervised text annotation techniques. The seeding text annotations needed were provided by specific expert annotation tools. These are not intended to be used by the end users.

As a result of these findings the objectives of the text annotation tool of WP4 to be developed in phase two read as: allow for (a) maintaining the various X-Media ontologies (R-R, CRF, bike brake use case) by (b) annotating text (instead of using a heavy weight ontology engineering workbench) such that (c) knowledge management processes are supported adequately, (d) end users may read (and optionally extend) given conceptualizations, (e) various different ontology design patterns are supported and (f) one single user interface and interaction paradigm for conceptual modelling can be used (g) to generate RDFS, OWL and F-logic ontologies from a single source.

8.1 The new Interaction Paradigm of the Tool mm2flo

Analyzing the X-Media requirements w.r.t. ontology engineering and ontology population lead to the idea that engineering an ontology (with a focus on the schema part) and populating an ontology (with a focus on the A-box part) doesn't necessarily call for distinct tools. Ideally there should be no tool change required between T-Box definition and A-box population. The requirements towards a text annotation tool suggest that there is no clear borderline between annotation as ontology population (i.e. adding facts to the A-box) and annotation as terminology refinement (i.e. adding new concepts to the T-Box).

The tool m2flo realizes these objectives by reviving Donald Knuth's concept of literal programming (c.f. http://www.literateprogramming.com/index.html): instead of authoring code and documenting it afterwards the idea of semantic authoring toggles these two steps: with the new annotation tool a user generates a properly annotated documentation of a domain in order to gather automatically the respective ontology from it. "Literate ontology engineering" is a new paradigm in ontology engineering, characterized by the identification of text annotation, ontology authoring and text production.

As opposed to other approaches of literate authoring we do not rely on a plain text editor or traditional WYSIWYG office tools. While these tools are fine for plain (and only very weakly structured) text, they are only sub-optimally suited for very highly structured conceptual modelling.

Analyzing the major ontology engineering workbenches and other basic text structure visualization techniques soon led to the decision to take the cognitive technology "mindmapping" as a basis for the new tool: the structure "tree" forms one of the most basic and widely used structuring paradigms we know. Working with mindmaps is one of the most established brainstorming and information structuring methodologies we have in knowledge management.

The simple approach of maintaining content within a tree of elements (technically with a mindmap tool like freemind) serves as the fundamental overall interaction paradigm. It is the structure of the tree itself (making use of edge annotations) which mirrors the structure of the text, while it is an icon-tag which defines the semantics of a certain text region formally. This double-layered text annotation principle results in an isomorphism between text structure and semantic.

Having decided to use a mindmap as the overall representation paradigm of text and semantics led to the decision to fathom how far we could go purely with that approach. Would it be possible to integrate visualization, structuring and authoring of text and it's semantics such that authoring schema and facts of an ontology follow the same interaction paradigm?

In fact we developed a mindmap tagging system (cf. Figure 8.1) which allows for representing unstructured text content, text structures, schema, fact base and rules completely within one notation! To our best knowledge we believe that this is a genuinely new and innovative modelling technique, which is definitely not covered by current major ontology engineering workbenches or by well known office tools. In fact it is this mindmap annotation methodology which defines a new, sound, powerful and simple

interaction paradigm. End users (who often already use mindmaps for their personal knowledge management) now can author semantic models directly with a common and lightweight technology.

The tool mm2flo was used to prepare the demonstration ontology; a human readable documentation of the ontology is included in D19.2 in section "The bicycle brake universe". A more detailed documentation of the tool (of course generated by the tool itself) is included in the tool's distribution (get it from the X-Media Software download page; the distribution also contains some modelling examples).

8.2 Integration with Knowledge Management Challenges

In X-Media the use case ontologies are already developed by joint work between ontology engineering expert from research institutes, technology providers and domain experts of the end user partners. In order to communicate the ontology it has to be visualized and documented in several channels - ideally both in an ordinary text document and a bird's eye visualization. The tool mm2flo interprets as the native format (a very simple XML) of the lightweight and very common open source tool freemind.sourceforge.net. Conjoint editing and visualization of ontologies and their documentation can be performed even between organizations with very heterogeneous technology stacks and software usage policies. The freemind map is editable without the need of having the script mm2flo installed as even a minimally configured computer while offline can be used to edit an ontology.

Ordinary users are empowered to suggest ontology refinements without having to change the ontology. However, this has to be done in a way that allows the users to suggest extensions (and communicate the refinements to the ontology engineers) without having to commit them. While in traditional ontology engineering workbenches this only can be done by changing directly the ontology (which easily may result in an incoherent or otherwise ill formed ontology), the new annotation approach allows for communicating refinement suggestions in a much less viral way: users suggest how to extend the ontology simply by adding text content to the mindmap without tagging it semantically, i.e. annotating the ontology with tree structured text.

Terminological knowledge may be outlined, fully formulated and refactored within the same tool. In order to make use of workbenches like Protégé or OntoStudio in ontology engineering projects an ontology engineer has to bring a full understanding of a conceptualization with her; and using these workbenches for iterative re-structuring of a conceptualization requires to make use of complex (and error prone) refactoring functions. Our new annotating approach exploits the fact that a mindmap is able to represent text and text structures independent from edge tags or semantic icon tags. Changing the structure of a taxonomy or reverting a class tree (which is built on class level) into a tree of skos:broader-related instances of skos:Concept can be done very easily, fast.

The ease of playing around with semi formal and more formal aspects of a conceptualization suggests to make use of mindmaps both in early (brainstorming) phases and late (formalization) phases of terminology projects - completely within one single tool, without having to change work and tool contexts while proceeding in a project. While earlier the need for changing tools resulted often in an inflexible and expensive waterfall approach of ontology engineering, our new approach is suited for cyclic or agile ontology engineering methods. In effect our approach results in an a simple and powerful solution to tie early and late, informal and formal, explorative and productive phases in ontology engineering seamlessly together.

8.3 The SkosLiteFLO Example

In the past we have shown in several X-Media deliverables how we have crafted the public dissemination show case ontology (cf. D19.2). In order to show the universality of the tool also w.r.t. differences between OWL and F-logic we here give a short overview over an F-logic version of SKOS.

SKOS (simple knowledge organization system, c.f. http://www.w3.org/2004/02/skos/) is a very lightweight ontology schema which represents on a semantic level knowledge which is organized as a concept thesaurus. SKOS plays a role in X-Media because in the CRF ontology it was decided to use the skos labelling properties (like prefLabel, altLabel etc.) instead of less expressive RDF labels.

The normative SKOS schema is modelled in OWL full. It can be loaded and maintained e.g. with Protégé 4. However, OWL full ontologies can't be handled appropriately with standard inferencing engines (and thus not be translated to F-logic). In order to make way for large scale inferencing within lightweight (formally: of OWL DLP complexity) terminologies we have decided to hand craft an F-logic pruning the SKOS model with mm2flo.

Figure 8.1 shows a mm2flo representation of the original and normative OWL full representation of the SKOS Ontology.

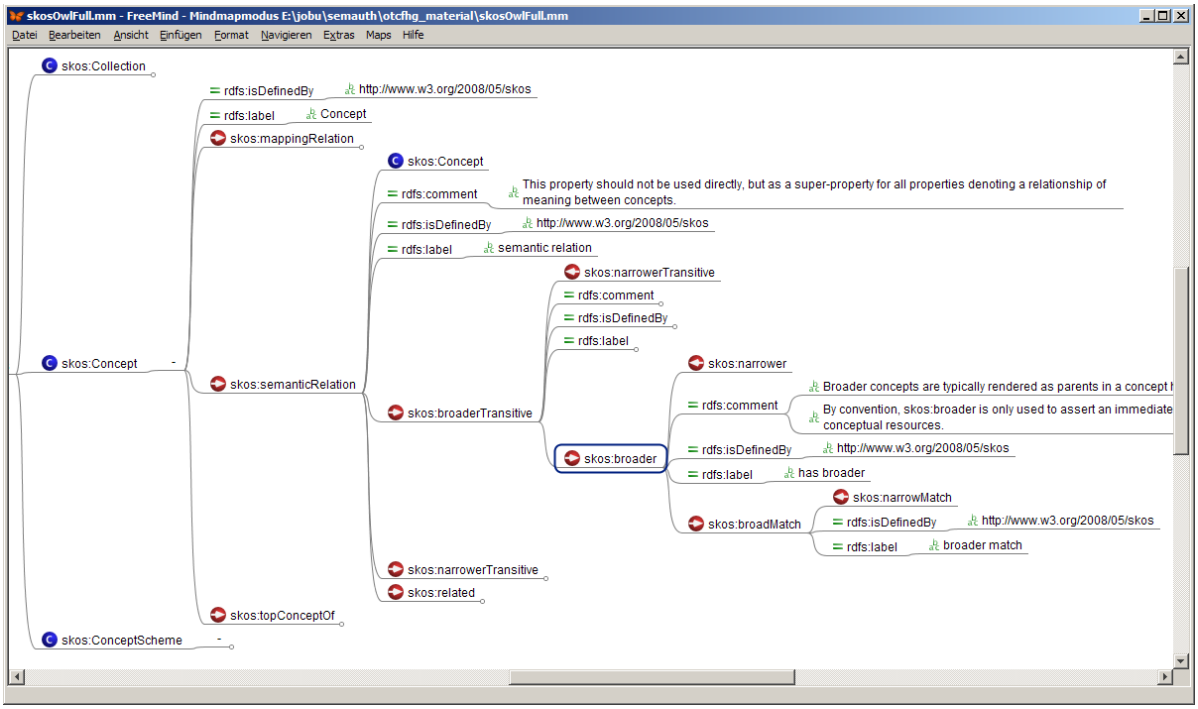


Figure 8.1: Screenshot of mm2flo.

9 Global Tools-Integration using the Semantic Scratch Pad

This task was concerned with the integration of different tools developed in the Experimental Vibration Use Case. To overcome issues related to lack of integration of individually developed tools as raised in phase 1, a global integration mechanism was integrated into the X-Media Kernel Architecture in D11.6 to combine the tools for the use case. A well-balanced solution had to be found which on the one hand should not affect the individual tools that much (e. g., deep changes in source code), while on the other hand giving the end-user the feeling of having a fully-integrated, yet deeply linked final software system. Challenges increased also because of different programming languages used in the tools:

- The Scratchpad (D11.6) used *Struts 2*.
- The Zmod-based tools (D7.7) developed by UHil were using Java Server Faces (JSF).
- The Mode Shape similarity tool (D6.5) provided by Labri uses plain *JSP*.
- The Mode Shape classification tool (D6.5) in implemented using *PHP*.
- The Mode Annotation Tool by UHil (D7.7) is available as an *applet* running in a browser.

Besides of the need of integrating completely different programming languages, further the X-Media Kernel Architecture with its distinction between document and ontology URIs introduced some interlinking problems. To this end, automated transformation strategies were used. Here, `meta:source` and `owl:sameAs` properties were used, depending on the context in which a mapping was required. From the architectural level, the integration challenge can be best explained by using Figure 9.1.

All (deployed) XApps work on the same ontology structure (here, Lena and RRExpVibXApp are given as a running example). The main idea to integrate the tools is based on the fact that all of them work on the same individuals. As a consequence it should be possible to seamlessly switch between the different tools in a resource-centric way: the user should be able to continue his work while switching between tools, while remaining on the same, fixed individual without the need to re-locate it using some proprietary and/or redundant search and/or browsing mechanism. Furthermore, the overall mechanism was required to automatically recognize which XApps are deployed in the

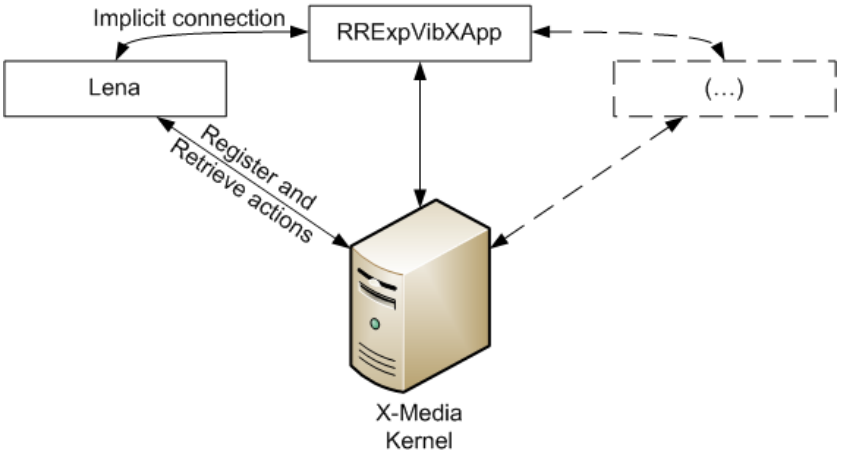


Figure 9.1: Structure.

kernel so that no broken or dead links are shown. For this reason, static links between the tools were impossible. A solution was found by using the following abstraction layer:

1. Each XApp registers multiple *actions* into an *action-repository*. These actions represent possible *entry-points* to an XApp, e. g. representing tasks such as 'Browse Ontology', 'Show Details', 'Classify Mode Shape', or 'Search Similar Zmods'.

2. By using this action-repository, once the user wants to perform an action on the user interface (e. g. further working on a Zmod), the currently used XApp queries the repository for available actions on the currently used individual, and presents them to the user.

3. If the user chooses one action, the XApp acts accordingly: for all the tools being available through an URL, open that one by passing all parameters as returned by the action-repository using POST.

By decoupling the XApps using an intermediate action-repository, it is possible to only show those actions which are exposed by XApps which are actually deployed in the kernel. Figure 9.2 shows the integrated version of LENA (the part on the bottom left 'Applications' contains integration links).

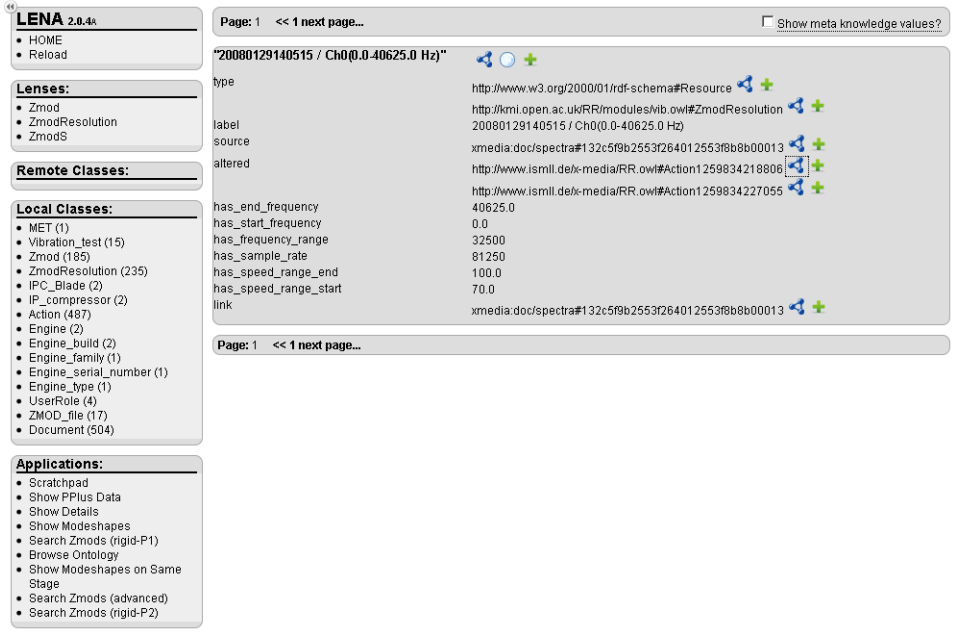


Figure 9.2: Integration-enabled version of LENA (bottom left).

For the tools available through the RRExpVibXApp, JSF technology was used. More specifically, the Ajax-framework IceFaces (`http://www.icefaces.org/`) had been used. By using this technology, easy technical solutions for context-menus and partial submission of HTML forms could be found. Figure 9.3 shows a context menu for the same entity as displayed in LENA in Figure 9.3, using the Data Browser provided by UHil.

In Figure 9.3, by invoking the 'Browse Ontology' action, LENA is opened. Implemented as a utility function/library, it was possible to add the integration functionality all over the web frontend as depicted in Figure 9.4 for their generic Data Browser.

While the action-repository provides a general mechanism to get possible actions for individuals stored in the shared knowledge base, the bookmarking facility could not only be used per XApp to store and re-use bookmarks, but also to provide a generic 'Open-As' functionality as shown in Figure 9.5.

Using the Semantic Scratchpad we suceeded in integrating all tools for the Experimental Vibration Use Case in a couple of weeks. As one result of the final evaluation conducted in November 2009, the users highly appreciated the fully integrated tools delivered by the partners.

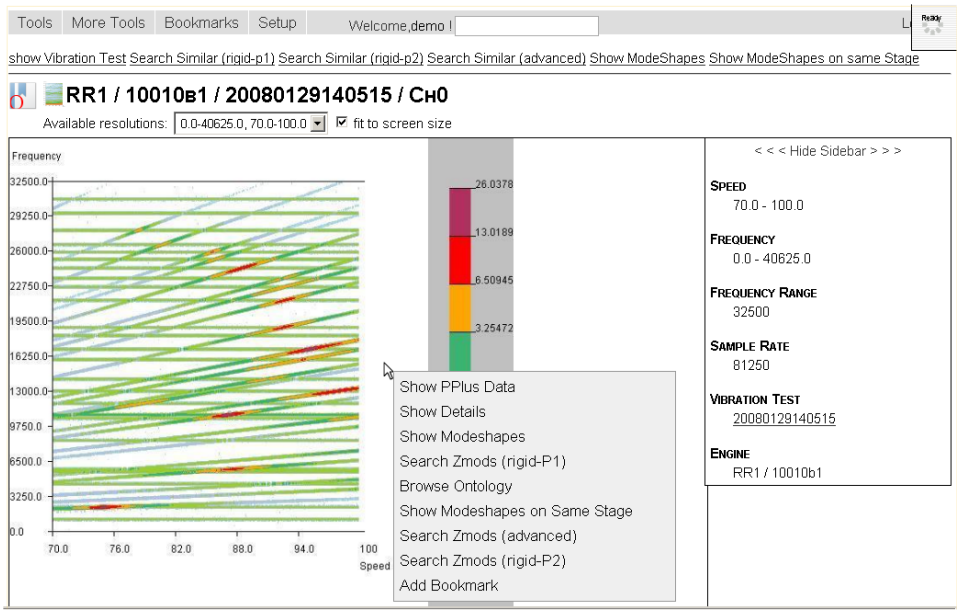


Figure 9.3: Global integration available for resource-centric pages.

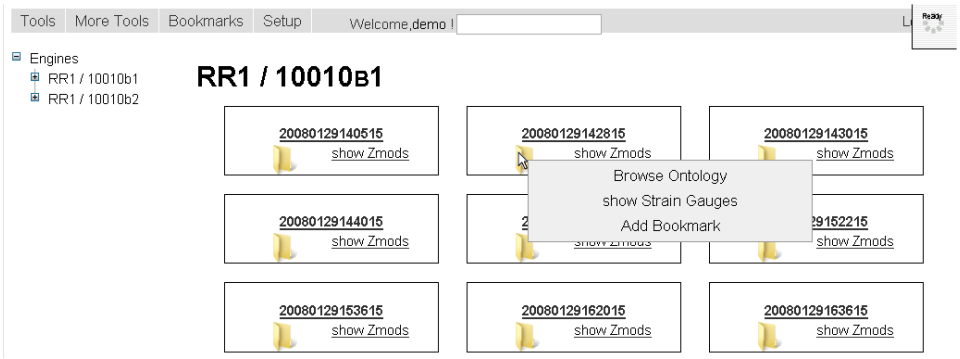


Figure 9.4: Global integration on list-like overview pages.

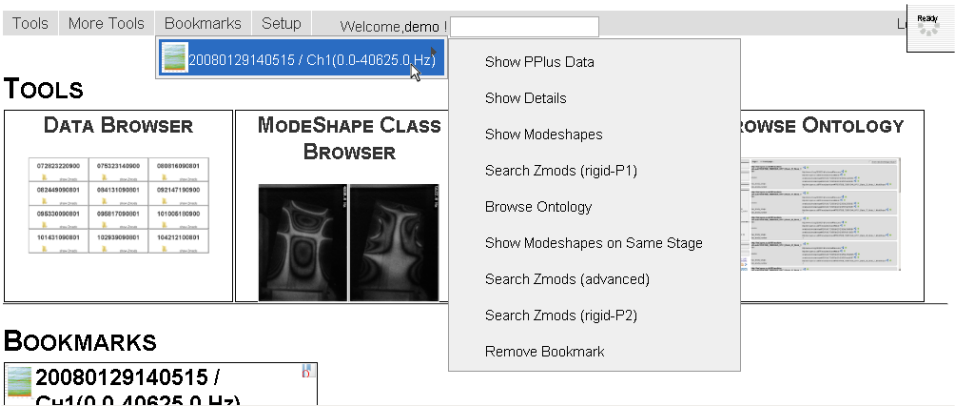


Figure 9.5: Fully integrated bookmarking facility.

10 Conclusion

This deliverable has shown a variety of tools and their integration with respect to user interface integration, technology integration, and integration with X-Media infrastructure. Among the different X-Media use cases, different combinations of these tools are applied to support diverse tasks handled by engineers at Rolls-Royce and Fiat. The range of tools and their tight integration, especially on the user interface side, documents the successful completion of the overall task of providing an integrated tool suite that supports complex knowledge work across different media.

Bibliography

[Bhagdev et al., 2008a] Bhagdev, R., Ajay Chakravarthy, Sam Chapman, F. C., and Lanfranchi, V. (2008a). Creating and using organisational semantic webs in large networked organisations. In *ISWC '08: Proceedings of the 7th International Conference on The Semantic Web*, pages 723–736. Springer-Verlag.

[Bhagdev et al., 2008b] Bhagdev, R., Chapman, S., Ciravegna, F., Lanfranchi, V., and Petrelli, D. (2008b). Hybrid search: Effectively combining keywords and semantic searches. In Bechhofer, S., Hauswirth, M., Hoffmann, J., and Koubarakis, M., editors, *The Semantic Web: Research and Applications, Proceedings, 5th European Semantic Web Conference (ESWC 2008)*, pages 554–568.

[Buza et al., 2009] Buza, K., Nikolov, A., and Rendle, S. (2009). D3.8: Report on methods for enhanced knowledge fusion. Technical report, X-Media Consortium.

[Dadzie and Ciravegna, 2009] Dadzie, A.-S. and Ciravegna, F. (2009). Flexible knowledge generation and re-use in aerospace engineering using the XMediaBox. In *European Semantic Technology Conference (ESTC 2009)*.

[Dadzie et al., 2008] Dadzie, A.-S., Franz, T., Lei, Y., Petrelli, D., and Preisach, C. (2008). D 4.3: Evaluation of knowledge sharing tools. Technical report, X-MEDIA Consortium.

[Dadzie and Petrelli, 2009] Dadzie, A.-S. and Petrelli, D. (2009). Visual knowledge exploration and discovery from different points of view. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST 2009) - Poster Session*, pages 227–228.

[Franz et al., 2010] Franz, T., Koch, J., Dividino, R., and Staab, S. (2010). Lena-tr : Browsing linked open data along knowledge-aspects. In *LinkedAI 2010*.

[Franz et al., 2009] Franz, T., Scherp, A., and Staab, S. (2009). Are semantic desktops better? In *International Conference on Knowledge Capture, K-CAP*.

[Giordanino et al., 2008] Giordanino, M., Cresta, M., Tola, R., Vieux, R., Romano, L., Nikolopoulos, S. N., A.Nikolov, Kuznar, D., Biasuzzi, C., and Ferraro, M. (2008). Deliverable 13.4:. Technical report, X-Media Consortium.

[Koch et al., 2008] Koch, J., Franz, T., and Staab, S. (2008). Lena - browsing rdf data more complex than foaf. In *International Semantic Web Conference (ISWC), Demo Session*.

[Lavelli et al., 2009] Lavelli, A., Busse, J., Dadzie, A.-S., Xia, L., Nikolopoulos, S., Franz, T., and Ringelstein, C. (2009). D19.2: Application of x-media technology to the public use case. Technical report, X-MEDIA Consortium.

[Lei et al., 2006] Lei, Y., Uren, V. S., and Motta, E. (2006). Semsearch: A search engine for the semantic web. In *EKAW*, pages 238–245.

[Lopez et al., 2006] Lopez, V., Motta, E., and Uren, V. S. (2006). Poweraqua: Fishing the semantic web. In *ESWC*, pages 393–410.

[Lopez et al., 2009a] Lopez, V., Nikolov, A., Fernandez, M., Sabou, M., Uren, V., and Motta, E. (2009a). Merging and ranking answers in the semantic web. In *The 4th Asian Semantic Web Conference*.

[Lopez et al., 2009b] Lopez, V., Sabou, M., Uren, V., and Motta, E. (2009b). Cross-ontology question answering on the semantic web an initial evaluation. In *Knowledge Capture Conference*.

[Nazir et al., 2009] Nazir, F., Uren, V., and Nikolov, A. (2009). Algorithms for generating ontology based visualizations from semantic search results. In *Workshop MoViX*.

[Petrelli et al., 2009] Petrelli, D., Mazumdar, S., Dadzie, A.-S., and Ciravegna, F. (2009). Multivisualization and dynamic query for effective exploration of semantic data. In *Proceedings, International Semantic Web Conference (ISWC 2009)*, pages 505–520.

[Pietriga et al., 2006] Pietriga, E., Bizer, C., Karger, D., and Lee, R. (2006). Fresnel: A browser-independent presentation vocabulary for rdf. In *International Semantic Web Conference*, pages 158–171.

[Ringelstein and Sizov, 2009a] Ringelstein, C. and Sizov, S. (2009a). Formal Definition of Data Centric Process Knowledge. Deliverable, X-Media Consortium.

[Ringelstein and Sizov, 2009b] Ringelstein, C. and Sizov, S. (2009b). Internal Deliverable INT1.6. First Prototype with API for management of Centric Process Knowledge . Deliverable, X-Media Consortium.

[Ringelstein and Staab, 2009] Ringelstein, C. and Staab, S. (2009). Dialog: Distributed auditing logs. In *Web Services, IEEE International Conference on*, pages 429–436, Los Angeles, CA, USA. IEEE Computer Society.

[Uren et al., 2007] Uren, V., Chapman, S., Dadzie, A.-S., Slavazza, P., Franz, T., Lei, Y., Busse, J., Rabus, D., Korf, R., duc Thanh, T., and Lech, T. C. (2007). Deliverable 4.2: Knowledge lenses and process support tools. Technical report, X-Media Consortium.

[Uren et al., 2008] Uren, V. S., Lei, Y., and Motta, E. (2008). Semsearch: Refining semantic search. In *ESWC*, pages 874–878.