PRODUCT WHITE PAPER

THE ONTOPIA KNOWLEDGE SUITE

Abstract

This white paper gives a quick introduction to version 1.3 of the Ontopia Knowledge Suite™ (OKS), describing the architecture, functionality, and composition of the suite. There is also some discussion of the possible usage areas of the suite.

A basic understanding of the main concepts of topic maps is assumed; for explanations of these, please see the various topic map introductions available on our web site at http://www.ontopia.net/topicmaps.

1. An Overview of the OKS

The Ontopia Knowledge Suite is a well-designed suite of products that together enable the development of diverse types of topic map-enabled applications. The suite is designed to be integrated into any application to provide it with *topic map capabilities*. It is stable, performant, and highly advanced; it handles all the difficult aspects of developing with topic maps for you.

In version 1.3 the Suite consists of three main products, with several smaller add-on components. These can be purchased in three different configurations, which are described below. The main products are:

The Ontopia Topic Map Engine

The Ontopia Topic Map Engine is the heart of the suite: a Java SDK which has all the functionality needed for writing software that works with topic maps. The engine loads, stores and keeps track of the topic maps, and provides interfaces through which applications can access and manage the topic maps. The other products in the OKS are built on the engine and use its functionality.

Three add-on components to the Engine provide additional power for advanced applications. These are the query engine, which supports a topic map query language, the schema tools, which support a topic map schema language, and the full-text integration, which adds support for full-text search of topic map data.

The Ontopia Topic Map Navigator

The Ontopia Topic Map Navigator is a web application framework that greatly simplifies the development of topic map web applications. It is based on Java Servlets and Java Server Pages (JSP), and provides a set of tag libraries and Java components that make developing scalable topic map web applications with advanced functionality fast and easy. Using it can be compared with using a kind of XSLT specially designed for topic maps.

The Ontopia RDBMS Backend Connector

The Ontopia RDBMS backend connector is an add-on to the engine that enables it to store topic maps in relational databases, and to access and modify topic maps stored in such databases. Most RDBMS servers are supported.

In addition to these main products, the OKS contains some smaller components which add important functionality to the suite. These are:

The Ontopia Query Engine

Using the Query Engine you can easily specify complex criteria for retrieving information from the topic map. This greatly simplifies application development, since much can be accomplished with very little code, and significantly lowers the learning curve for developers.

Since queries specify retrieval actions at a very high level they also provide great scope for optimization, allowing the query engine to perform retrieval very efficiently.

The Ontopia Full-text Search Integration

The Ontopia full-text search integration allows users to look up topics by name (and the text appearing in occurrences) easily and quickly. Users can jump to a specific point in a topic map immediately, without knowing the topic map structure at all, which makes it enormously much easier to find information.

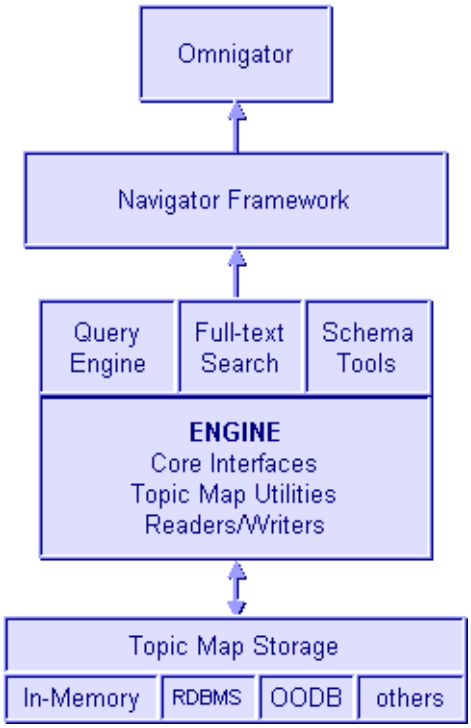
The Ontopia Schema Tools

Certain tasks can only be performed by humans, but humans make mistakes. If your topic maps are written by human authors, in whole or in part, you will want some way of ensuring that the authors use the correct topic map structure. Using the Ontopia Schema Language you can easily and concisely define the correct topic map structure and use the schema the schema tools to enforce that structure.

The Omnigator

The Omnigator is a demonstration tool built using the Navigator Framework. It is a topic map browser that can e used to browse any topic map. Ontopia has made the Omnigator freely available in order to promote topic map awareness, but it is also distributed as part of the OKS, as many customers have found it useful for looking at their own topic maps. (One customer called it a topic map debugger.)

Below is a diagram that shows the structure of the OKS, and the relationship between the different parts.



The OKS at version 1.3

Future versions will add more products and components.

1.1. Editions of the Suite

Version 1.3 of the OKS comes in three different editions:

Personal Edition

This edition contains the Ontopia Topic Map Engine, which means that it is mostly useful for integration and visualization projects, and for adding topic map functionality to existing products.

Professional Edition

This edition contains the Ontopia Topic Map Engine with the three add-on components, as well as the Navigator Framework. It is useful for the prototype web projects, smaller web projects, and for adding topic map capabilities to existing products which need a web interface.

Enterprise Edition

This edition adds the RDBMS backend to the professional edition, and is thus the edition of choice for deployment of full-scale projects.

The table below shows more precisely which components are included in what editions of the OKS.

The prices of the different editions as well as an online purchasing system can be found on our web site http://www.ontopia.net/solutions/products.html

Components	Personal	Professional	Enterprise
Engine	Yes	Yes	Yes
Omnigator	Yes	Yes	Yes
Schema Tools		Yes	Yes
Query Engine		Yes	Yes
Full-text integration		Yes	Yes
Navigator Framework		Yes	Yes
RDBMS Backend Connector			Yes

1.2. Application areas for the Suite

The Ontopia Knowledge Suite is not an end-user product, but a set of tools useful for building such products. In general, the OKS can be used to advantage in nearly any application that manages information. The problem solved by the Ontopia Knowledge Suite is the perennial problem of information management: the problem of navigating the information ocean to find the knowledge you seek.

We all know how difficult it can be to find the information you are looking for. The reasons for this are many and deep-rooted, and not the least of them is that creating appropriate information structures is in itself difficult. Topic maps, the product of decades of work and experience in information management, provide a model for the organization of information that can make creating appropriate structures much easier. If your information repository is organized with a topic map, structuring it becomes much easier, and so locating the document or piece of information you want also becomes much easier.

An important aspect of topic maps is that they do not require you to replace your information repository with one based on topic maps. Instead, you can maintain a topic map *separately* from your repository and have the topic map refer into the repository. The topic map can then be navigated to locate the information you need from the repository.

With the Ontopia Knowledge Suite you have what you need to create and manage topic maps for your repositories, as well as to build the interfaces used to navigate the resulting topic maps. Using the Ontopia Knowledge Suite all the difficult problems of topic map management, persistence, concurrency, transactions, and scalability are taken care of for you.

Some of the most important and interesting areas where the Knowledge Suite can simplify development and improve the result are:

Web portals

Portals, being essentially large information repositories, are perfect use cases for the OKS. Using topic maps for the portal content you will find yourself guided by the topic map model into creating a well-organized portal. Using the OKS, developing the actual portal is greatly simplified, thanks to the Ontopia Topic Map Navigator SDK.

Knowledge-base intranets

Using topic maps you can capture and manage the knowledge that constitutes corporate memory in a kind of *semantic network* that represents the people, products, roles, procedures and other artefacts of your organization; the relationships between these entities; and the links to the documentation that describe them in detail. With the OKS you can enrich your intranet with topic maps and turn it into a real collaborative knowledge base.

Content management systems

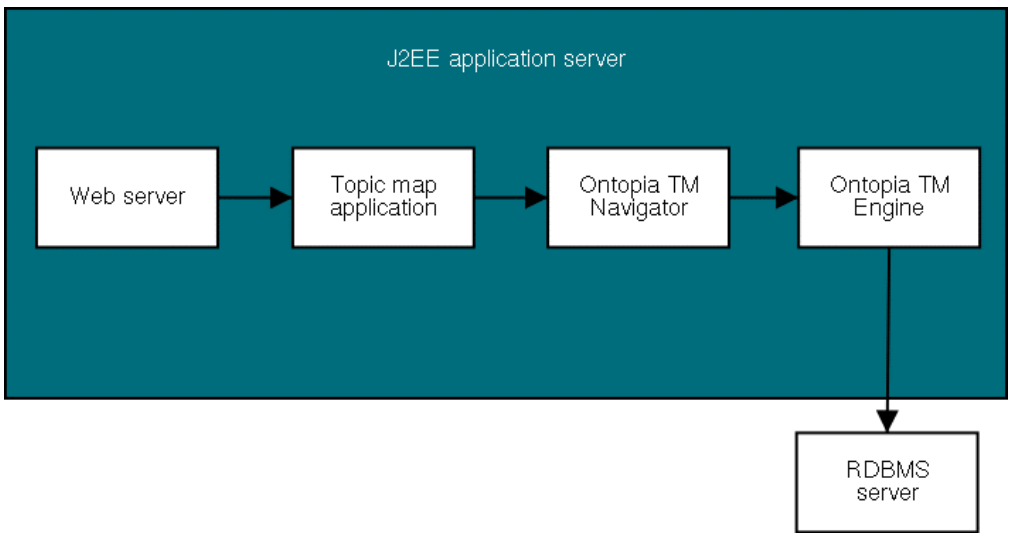
Any content management system can benefit from a better way of organizing information, which is what topic maps provide. Using the Ontopia Topic Map Engine you can integrate topic mapping capabilities into your content management system. The integration could take the form of making the entire system topic map-based, or it could create a virtual topic map of the system. And with the Ontopia Topic Map Navigator the web interface is there almost out of the box.

Enterprise Application Integration (EAI)

The topic map data model is designed to allow automated merging of information from diverse sources. Using the Ontopia Topic Map Engine's support for merging, information from many different sources can be integrated into a coherent whole. One application of this facility is the integration of products in a product suite; another is the provision of a single, unified access layer to information stored in otherwise disparate and incompatible formats and repositories. A web interface to this unified information can be developed using the Navigator framework.

The list of possible application areas for this technology is endless, and as topic map technology is still new and fresh, exciting new uses for it will undoubtedly continue to be found.

Below is a figure that shows how the Ontopia Knowledge Suite might be used to build a web site based on topic maps, storing the topic map data in a relational database.



Web application built on the OKS

In this example the web application is deployed in a Java 2 Platform, Enterprise Edition (J2EE) web server, together with the Ontopia Engine and Navigator. The topic map is stored in a relational database server, managed by the engine, and the web application is built using the Navigator Framework.

1.3. A velvet revolution

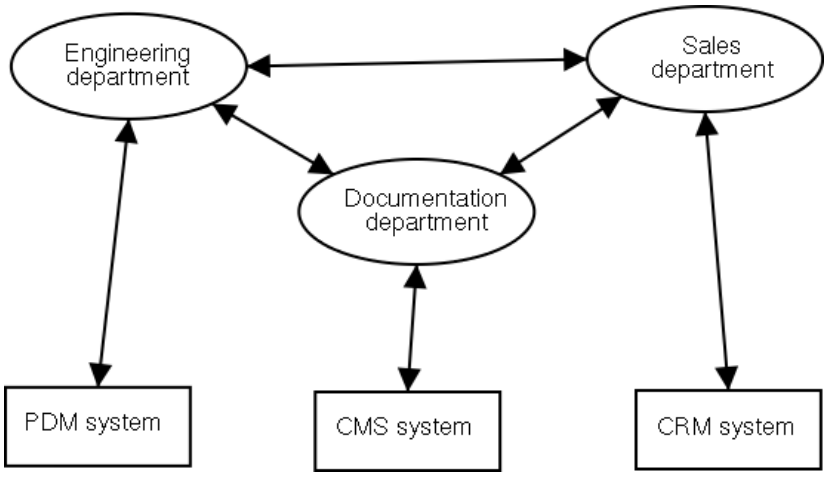
To help you understand more clearly how these products can be used, and how topic maps can be made use of in practice, this section presents an example topic map application.

Dingbat Inc., invented for the purposes of this example, is a manufacturing company that produces construction equipment like drilling rigs, trucks, and so on. It has a large product catalogue, consisting not only of the machinery, but also chemicals used by the machinery, spare parts, and so on. Keeping track of what variations of a particular drilling rig exist, which pieces of documentation go with which variant, which spare parts with which variant and so on has proven to be a daunting task.

Dingbat already uses a product data management (PDM) system, and so can manage the CAD data, technical properties, and inter-product relationships using this system. Much of the related information, however, is not managed in the PDM system. For example, the product documentation is in a separate content management system (CMS); the information about which customers have bought what products are in the customer relationship management (CRM) system; the information about which branches of the company that can sell and provide training for what products is in people's heads, and so on.

Information mismanagement

In short, Dingbat Inc. is in the same position as countless other companies around the world. The structure of their IT systems mirrors the organization of the company, but while the departments communicate, their IT systems do not. And, as usual, some information that ought to be stored electronically is not. This is the situation illustrated in the figure below:

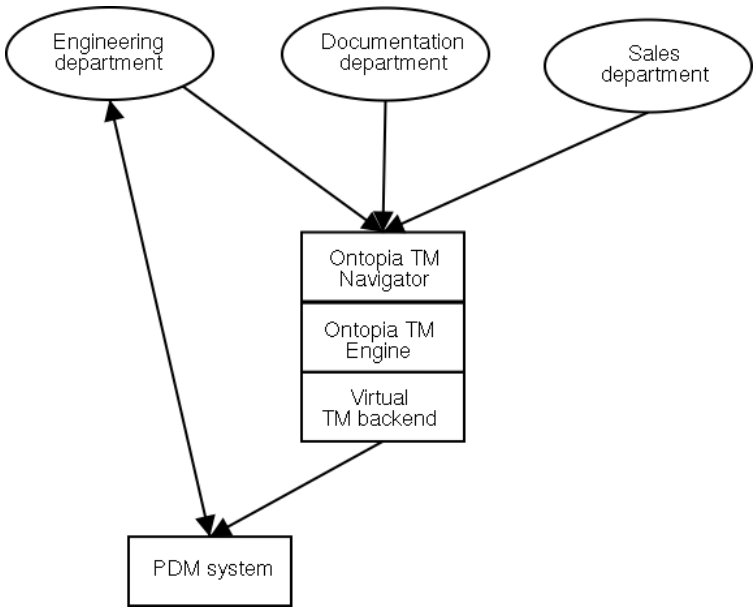


Dingbat Inc., before

Enter topic maps…

Dingbat decides to tackle the problem of managing their information assets head-on, and decides to use topic map technology to do so. The engineering department is happy with its PDM system, and does not wish to replace it, so the IT department decides to use the Ontopia Topic Map Engine to create a virtual topic map view of the PDM system. This results in a topic map containing the product catalog of the company, mirrored live from the PDM system, with information such as the products, their relationships, their most important technical properties and so on.

The Ontopia Topic Map Navigator is then installed and made to display the virtual topic map created from the PDM system. This provides instant gratification to everyone in the company, who can now browse their product catalog on the web. At the same time, no revolution has been necessary, and the engineering department continues to use its already-established PDM system. The architecture of this solution is shown on the next page.



Topic mapping the PDM system

The next step is to apply the same formula to the CMS system of the documentation department. Another virtual topic map backend is created, this time for the CMS system. The CMS system really only contains the documentation of the company products, and product codes are used to identify which document documents which product. The resulting topic map for the CMS system consists of a topic for each product variant, with the documents attached to the products as occurrences of various types, and some metadata about each document.

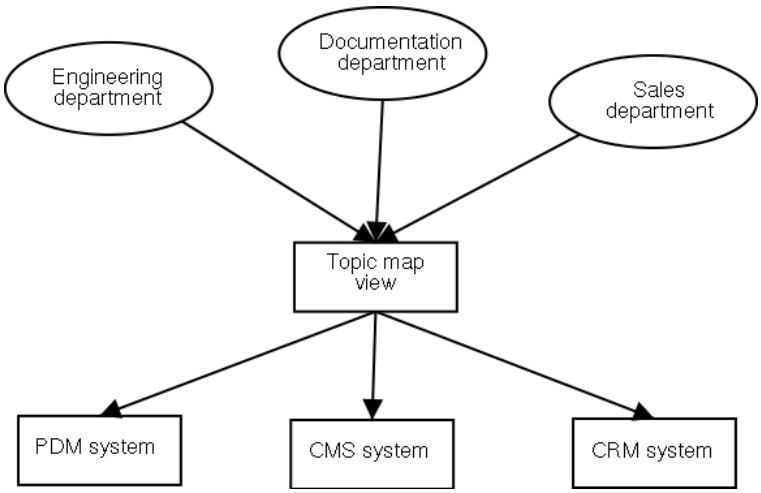
This may not sound very interesting, since the topics in the CMS topic map are unrelated to each other and do not even have names. What the products do have, though, are subject indicators (topic map-speak for globally unique identifiers) created from the product codes. The PDM topic map also has subject indicators for its product topics. This means that these two topic maps can be merged automatically, on the basis of the product codes.

Dingbat Inc. still has a single topic map describing its product catalogue, but this has now been enriched with connections from the products to their documentation. Suddenly, figuring out what documentation exists in what state for what variants of the same product can be done in seconds using a simple web interface. It is about this time that the employees of Dingbat Inc. start to get really excited about topic maps.

On the insistence of the sales department, the procedure is now repeated with their CRM system, yielding a topic map that has information about which customers have bought which variants of what products, the internal relationships of the customer companies and branches, as well as the responsible customer managers for each customer company. On the basis of the product codes, this topic map is merged with the PDM+CMS topic map.

Suddenly, the sales representatives of Dingbats Inc. can quickly discover information such as the fact that no training material or documentation available for certain products. Could there be a connection between this fact and the low sales figures for auxiliary items to those products?

The overall IT system at Dingbats Inc. now looks as shown in the figure below.



Dingbats Inc., after

At this point Dingbats Inc. was very pleased indeed, and trying to decide whether to first integrate more systems into their topic map, or whether to start building applications on top of the topic map. Wouldn't it be nice, for example, if the system discovered when a product variant had no maintenance manual, and automatically started a workflow task in the documentation workflow system with the purpose of creating one? And so on.

By now you would probably like to know more about the products that enabled this system, and so we will return to the details of the Ontopia Knowledge Suite.

2. The Ontopia Topic Map Engine

The Ontopia Topic Map Engine is what topic map applications and products use to work with topic maps. This SDK lets applications load topic maps from XML documents, store topic maps in databases, modify and access topic maps, and generally do all an application may need to do with a topic map. The engine has a core topic map API which all applications use to access topic map data, regardless of where those data are stored. Thus, whether the topic map is in-memory, in a database, or a virtual view is all the same to the application.

The Ontopia Topic Map Engine is the product of three years of topic map software development and sports a number of impressive features, as well as some powerful add-on components. The features are described immediately below, while the components are described in separate sections below.

A well-designed API

The core API of the engine has been carefully designed to be consistent, intuitive, and scalable. A number of well-considered policies describing the rules topic map data must abide by have been designed and documented. The result is an API that is quickly learned, and that protects the integrity of the underlying topic map data. The core API consists only of data objects, with all action objects separated out. This separation means that the core API is essentially only a thin layer over the underlying data.

The core APIs provide an object model that fully supports both the ISO 13250 standard and the XTM 1.0 specification. All aspects of the models of both standards are supported, as are extensions for other addressing syntaxes than URIs.

Extensive utilities

A rich set of utilities built on the core API is provided. These utilities provide functions like topic map-wide deletion of topics, association walking, the filtering of characteristics by scope, name selection, and topic map merging. The utilities are general and will work for any implementation of the core API.

The index API

The engine also has an index API which can be used for quickly and efficiently looking up information about the topic map, such as all instances of a given type, all topics that have been used as topic types, all topics used as scopes for occurrences, and so on. This API enables applications to quickly and efficiently locate the information they need, and also provides a mechanism for inferring information about the design of the topic map.

XML import and export

The engine has APIs for importing and exporting topic maps to and from XML documents. There is full support for the XTM 1.0 format and all its features, as well as for an XML version of the ISO 13250 format. Topic maps can easily be imported from one format and exported into the other. There is also support for validating XTM topic maps against a DTD and verifying that there are no dead references.

There is also support for importing topic maps from the compact and easy-to-author Linear Topic Map Notation (LTM) format developed by Ontopia. This can be used to author and demo simple topic maps very quickly. An introduction and a syntax description can be found at our website at http://www.ontopia.net/download/ltm.html.

Cross-platform support

The Engine is written in pure Java (though not certified as such), and has been tested with the Java 2 Software Development Kit (SDK) 1.3 on Windows 98, Windows 2000, RedHat Linux 7.0, SuSE Linux 7.1, Debian Linux 2.2, and Solaris SunOS 5.7. No known platform dependencies are in the code, and so it should run on any platform that has a Java 1.3 environment.

Internationalization support

The Engine correctly imports and exports topic maps to and from any legacy encodings supported by Java, which means that most character encodings are supported. Use is also made of the internationalization features of Java to provide internationalized sorting and similar operations.

Comprehensive test suite

To support the development and maintenance of the engine we have developed an extensive automated test suite for the engine. This test suite consists of nearly 500 test cases and verifies that all aspects of the engine are working correctly. All aspects of the core APIs, the utilities, the XML import/export, and the index API are tested. This ensures that the quality of the engine is high and that it remains so over time.

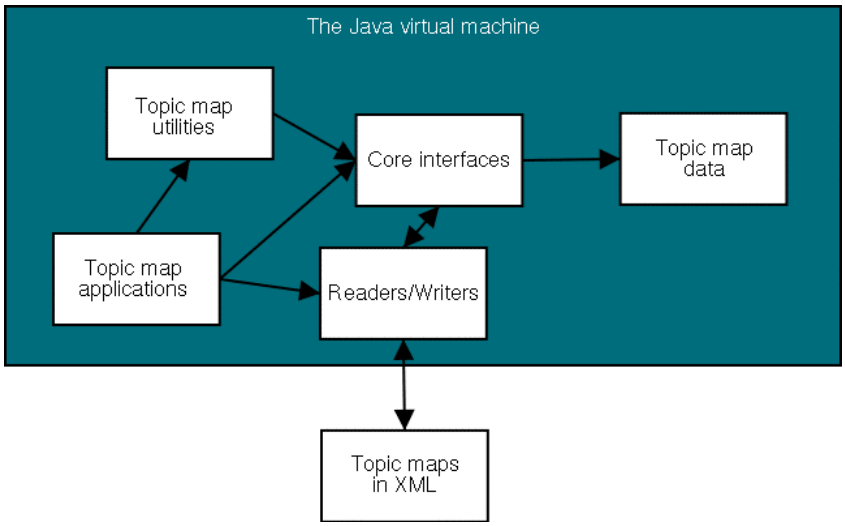
The test suite is shipped with the product, which enables customers to quickly verify that the engine works correctly in their particular environment. The test suite is also used to verify that the various implementations of the core APIs behave identically.

Enterprise-level robustness

The engine has been tested extensively to verify that it is thread-safe, scalable, and performant. A collection of automated testing tools has been developed to verify that the engine maintains these properties as new features continue to be added.

2.1. The architecture of the engine

As is shown in the diagram below topic map applications access the topic map data through the core APIs and the topic map utilities. The utilities also use these APIs, and so all interaction with the topic map data passes through the core interfaces. Import and export of topic maps serialized as XML, whether in XTM 1.0 or ISO 13250 format, is also done through the core interfaces.



The engine architecture

This architecture allows applications and utilities to be implemented independently of how the topic map data is represented. The data can be an in-memory object structure (which is what this diagram shows), a special implementation that stores its data in a relational database, or even a virtual view onto another data source. All applications and utilities will still work with the topic map in the same way.

2.2. The Query Engine

The Query Engine is an implementation of the tolog query language for topic maps. This language can query topic maps for topics of specific types, which participate in

certain combinations of associations, and supports inference rules. You can sort query results, and there is also support for counting query matches, and sorting by counts.

Using tolog allows complex retrieval operations to be expressed compactly and easily, making it easier to develop and maintain applications. tolog is *not* a standardized query language, but is provided while ISO completes its standardized TMQL query language. Once completed, Ontopia will provide full support for TMQL.

The query engine analyzes queries in order to find the most efficient way to perform the query, which means that using queries generally yields much more efficient code than implementing the query yourself on top of the APIs. The effort of writing a query is also of course at least an order of magnitude smaller.

More information on the tolog query language is available in an early paper presented at XML Europe 2001 (http://www.ontopia.net/topicmaps/materials/tolog.html) , as well as the tolog language tutorial that can be found at the Ontopia's web site http://www.ontopia.net/omnigator by following the link within the documentation.

2.3. The Schema Tools

The Schema Tools are an implementation of the Ontopia Schema Language (OSL), which is a schema language for topic maps that allows the expression of constraints on a topic map. For example, it can be used to say that composers must have only a single name, they must have a date of death and of birth, and they must have composed at least one opera.

Using the Schema Tools applications can easily validate whether or not the topic maps they work with follow the prescribed topic map structure or not. Schemas are also useful as compact and precise documentation of the structure of a topic map.

The main use for the Schema Tools is to ensure that human authors follow the editorial guidelines laid down in a particular topic map project, but they can also be used to guard against unpleasant surprises in automatically generated topic maps.

OSL is described in the OSL tutorial and can be found at http://www.ontopia.net/omnigator by following the link within the documentation.

2.4. The Full-text Search Integration

The Full-Text Integration allows topic maps to be indexed and then searched for topics by their names and the contents of their occurrences. This can be very helpful for users new to topic maps who need to find something quickly in a topic map. Using the full-text search they can type the name of what they are looking for and immediately be presented with a list of alternative topics. From the topics they can then easily find the information they are looking for.

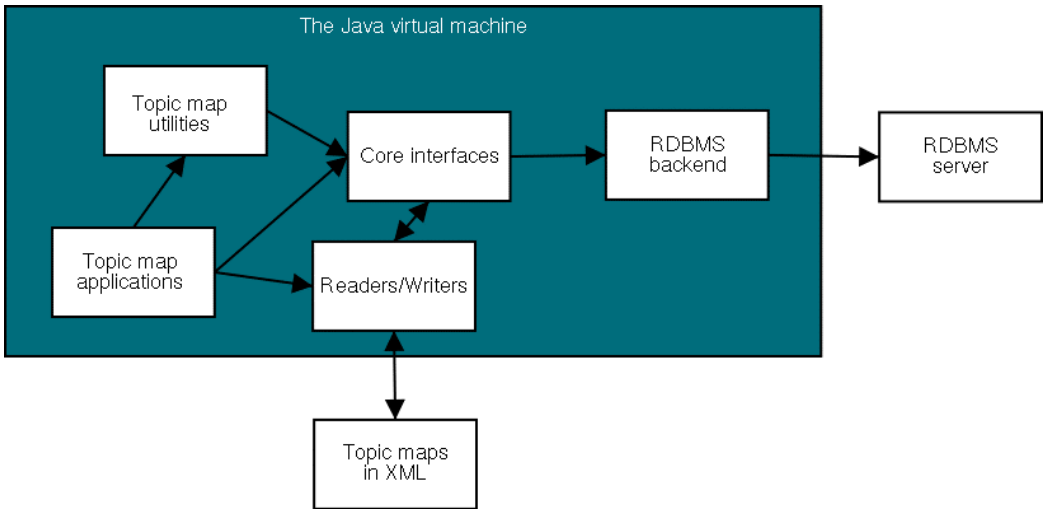
Using full-text searching in this way gives users an entirel different user experience, since although they are doing an unstructured search the users get a result set that is anything but unstructured. Since users are shown their search results in terms of topic map constructs it is much easier to judge *what* they have found, and one can more easily search for obscure or vague concepts by searching for related topics and then navigating around.

The Java-based search engine Lucene (http://jakarta.apache.org/lucene/docs/Lucene) comes bundled with the integration. Lucene is open source, powerful, robust, scalable, and lightning fast. A plug-in for the Omnigator providing an easy-to-use user interface is also part of the distribution. Other full-text search engines can also be plugged in. You can try this out in the online demo at http://www.ontopia.net/omnigator/.

2.5. The RDBMS backend

The RDBMS backend is an add-on to the Ontopia Topic Map Engine that uses the architecture described in the previous section to store topic maps persistently in relational databases. The applications access the topic map data through a different implementation of the core interfaces, which means that applications (as well as the engine utilities) need not care how the data is stored. The test suite is used to verify that the two API implementations behave in precisely the same way.

The diagram below shows the architecture of the engine with the RDBMS backend.



The engine architecture (with RDBMS)

Using this backend topic map application can scale to handle enormous topic maps, and also get benefits such as transactions and concurrency management between different processes. The architecture of the engine allows prototypes to be developed quickly using topic maps generated by scripts and stored in XML files. The prototype can then be upgraded to production quality by switching to the RDBMS backend and upgrading the generation scripts. The applications themselves do not need to change, since they are independent of where the topic maps are stored.

The RDBMS backend has been tested with Oracle 8i, PostgreSQL 7.1, and hsql 1.61, and is known to work with these databases. It can also be used with MySQL 3.23 (with the InnoDB transaction support). Whether the backend works on a given server or not can easily be verified using the test suite.

3. The Ontopia Topic Map Navigator

The most natural way to access the information in a topic map today is through the web, and the Ontopia Topic Map Navigator greatly simplifies development of topic map web applications. The Navigator is a general development framework based on the engine that lets you create your own web interfaces to your topic maps.

The framework consists of a set of Java utility APIs designed for web development, and a set of JSP tag libraries. The tag libraries allow much of the logic in each web page to be written using HTML-like tags, thus dramatically simplifying the development process. The tag libraries are very high-level and allow developers to do scope filtering and other advanced operations simply by tweaking the attributes of tags.

Another important benefit of the tag libraries is that since they operate at a high level of abstraction, they allow for many advanced optimizations, like caching of the most frequently retrieved topic names, object pooling, and so on. Thread-safety, session management, and so on is also handled automatically by the framework. Applications using the tag libraries get these benefits with no extra effort. The tag library is also integrated with the tolog query language, allowing you to use tolog queries to build your web applications.

The framework is fully internationalized and can correctly render topic maps written in any script. We have successfully viewed both Chinese and Japanese topic maps in our navigator applications. Applications built using the navigator inherit these features with no extra effort required.

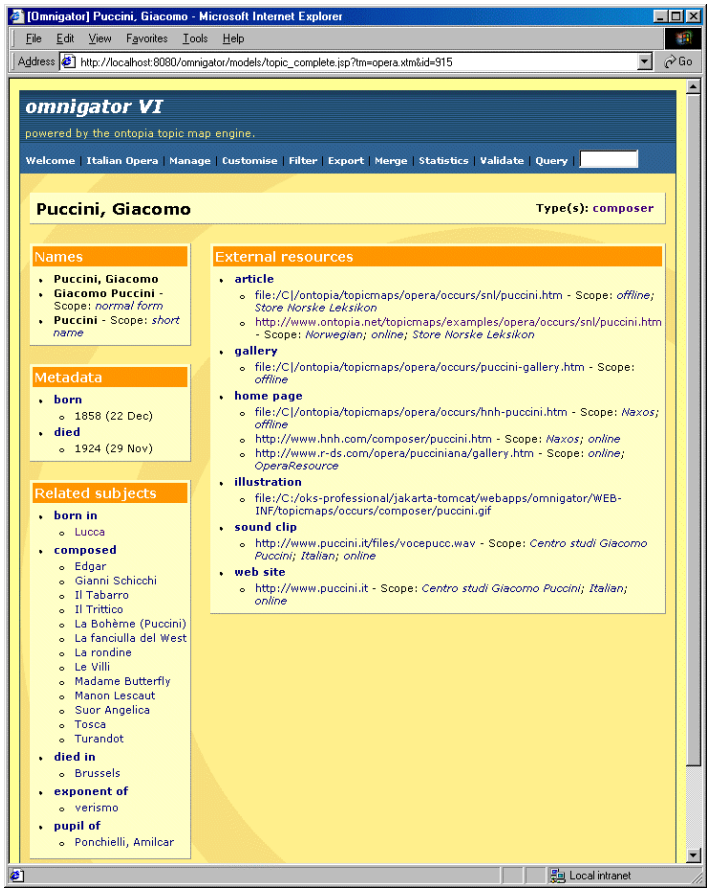
The navigator framework is based on the Java 2 Platform, Enterprise Edition (J2EE), and applications developed with it can be deployed into any J2EE container. The Tomcat web server was used during development, but using an automated test suite it is easy to verify that the navigator works in other containers as well. We have verified

that it works in the Resin (http://www.caucho.com/products/resin/) application server as well.

3.1. The Omnigator

As a demonstration of the power of the navigator framework we have developed the Omnigator, which is a short for “the omnivorous topic map navigator”. This is a topic map web application that can be used to navigate any topic map. You can access an online demo of the Omnigator at the Ontopia web site.

Below is shown a screenshot from the Omnigator, showing the automatically generated page for a topic in Steve Pepper's Italian Opera topic map. The topic shown is Puccini, a well-known composer.



The Omnigator

Although written for demonstration purposes, the Generic Navigator is a useful tool in its own right, since it turns any topic map into a web site with no need for programming at all. This makes it very useful as a tool for quickly creating topic map prototype applications that can be used to test out the applicability of topic maps to specific problems (and also to convince management that topic maps are the correct approach). All that is needed is to generate your topic maps as XML documents, load them into the Navigator, and you have a simple but convincing topic map demo.

The Omnigator is an advanced topic map application, supporting all aspects of both the XTM 1.0 and ISO 13250 standards. Some of the more advanced concepts supported are listed below.

Reification

Reification allows topics to represent identifiable resources within a computer, such as a web resource or a construct in a topic map. This can be used to make statements about associations, topic names, resources, and even occurrences (that is, the relationship between a topic and a resource). The Navigator recognizes reification and

uses it to optimize its presentation of the topic map. (See “The Omnigator User’s Guide for examples of reification.)

Scope

Scope can be attached to occurrences, names and associations to specify the contexts in which they are valid. This can be used to enable multiple views on single topic map, for example by language or in other ways. The Omnigator includes a User Context Filter plug-in (described below) that automatically builds user preference forms based on the use of scope in the topic map.

These concepts are also supported by the underlying framework, which makes it easy to develop topic map web applications using these constructs.

3.2. Using the Navigator Framework

The screenshot from the Opera topic map above shows the same view that is presented by the Omnigator for *all* topics. This is so because the Omnigator is designed to display any topic map, and so does not know the difference between a composer, a country, or an engine part. Using the Navigator Framework you can develop web applications that are specific to your topic maps, and this allows you to give the applications a user interface designed for your topic map. Below is shown the page for Puccini in the Operamap web application, which is written with the Navigator Framework, and designed for the Opera topic map.

The Operamap Application

The application is simple, and yet it shows that a topic map-driven web site can look just like any other web site, except that it is easier to find your way around. Using the navigator framework, writing this page is very simple. Most of the page is produced by a template, which is used to ensure that the site has a common look and feel. The rest of the page is produced by the `composer.jsp` file, which produces the pages for composers. It fills the area with the main content by pulling information from the topic map and writing it out as HTML..

Below is shown the JSP code used to produce the list of operas. Like XSLT the code is a mix of HTML elements (which are written out as output) and logic elements, which are executed to produce the output.

The code first finds all topics associated with the composer via the `composed-by` association, which gives us the list of operas (automatically sorted by name). We then loop over the operas (using `logic:foreach`), and write an HTML list item for each opera, with a link to that opera. We also find the `premiere-date` occurrence of the opera, and, if there is one, we write out its contents. This is the white text you see after the name of each opera.

List of Operas

```
<logic:set name="composed">
<tm:associated from="composer" type="composed-by"/>
</logic:set>
<p><b>Operas:</b><br/>
<logic:foreach name="composed">
<li><a href="opera.jsp?id<output:objectid/>"><output:name/></a>
<logic:set name="date">
<tm:filter instanceOf="premieredate">
<tm:occurrences/></tm:filter>
</logic:set>
<logic:if name="date">
<logic:then>
<output:content of="date"/></logic:then>
</logic:if></li>
</logic:foreach></p>
```

In this way, creating a web application for a topic map becomes a matter of no more than writing some HTML and topic map tags. No Java development is necessary, although if desired it is possible to integrate Java code into the framework in various ways. The full power of the topic map engine, the tolog query language, and the full-text search is available to application developers.

3.3. User context filter

The navigator framework as well as the Omnigator support a user defined context filter. The user can set to control what parts of the topic map the web application is to show them. This filter uses the scopes defined in the topic map, and lets users decide which names they prefer to see, and what occurrences and associations they do not wish to see.

This feature can be used to let users decide which language they want to see a topic map in, if the topic map has names in several languages for each topic. They can also decide whether they want to see the full names of topics, for example, or just the abbreviated forms.

In very information-rich topic maps, with many occurrences and associations for each topic, the filter may be used to remove occurrences and associations that are less interesting to the user, in order to keep the amount of information presented to a manageable level.

Users can control the settings of the user context filter by following the 'Filter' link shown in the Omnigator screenshot above.

3.4. Rendering with Model-View-Skin

The Navigator Framework uses a Model-View-Skin approach to rendering topic maps, cleanly separating graphical layout from information extraction. This simplifies application development, and allows application developers and graphic designers to collaborate more easily.

View

A view is essentially a web page template, written as a JSP page, with a set of named slots where content can be filled in. This makes it easy for graphic designers to create a single template for all pages in an application, thus simplifying the development, and at the same time separating application development from graphic design.

Model

The model is a JSP page that generates blocks of content and attaches them to the named slots defined by the view. The model pages are generally written by application developers, and do all the hard work of extracting information from the topic map and other sources, and turning it into web page content.

Skin

The last piece of the puzzle, the skin, is simply a CSS stylesheet that is applied to the generated page to produce the page style. This will generally define aspects such as the colors, background, fonts and text size.

The end-user using the web application sees the combination of a model, a view and a skin, and each user can freely select whichever combination is found the most appealing. For example, the Omnigator has both a frame-view and a no-frames-view. Each user can freely select which view to use. Similarly, the Omnigator has a detail model, displaying as much information as possible, and a simple model, showing only essential information.

A set of skins is provided with the Omnigator, ranging from the tasteful to the less so. New skins can be added simply by copying them into a specific directory. Each user can freely combine views and models as desired.

3.5. Extending the Navigator with plug-ins

The Navigator has a concept of *plug-ins*, which allows extensions to topic map web applications to be developed separately from the applications and easily plugged in. The Plug-ins can be deployed by simply dropping them into a directory, and the navigator will automatically detect their presence and include them in the interface.

Essentially, a plug-in is a web page that is linked to from every page of a topic map web application. When users follow this link, the web page is informed about the current topic map and topic, and can thus present relevant information. The topic map statistics plug-in, for example, will use these parameters to display statistics about the current topic map.

This mechanism allows plug-ins to be developed separately from specific topic map web applications and yet to be easily integrated into them. The result is greatly simplified development, installation, and administration.

3.6. The Navigator plug-ins

The Omnigator comes with a set of plug-ins which can be helpful when creating topic maps and developing web applications. These plug-ins are:

Statistics reporter

The statistics reporter is a simple plug-in that provides statistics about a topic map, such as the number of topics, the number of topics of each type, the association structure of the topic map, and so on.

Merge plug-in

The merge plug-in allows you to merge any topic map in the navigator's topic map registry with the current topic map, and enters the merged topic map into the navigator as a new topic map. You can then browse the resulting topic map in the navigator. This allows you to easily and quickly test the effects of automatically merging topic maps.

Export

The export plug-in allows topic maps to be exported in either ISO 13250 or XTM 1.0 format, and to be viewed or downloaded. Two topic maps can be merged using the merge plug-in and then exported in merged form using the export plug-in. Topic maps can also be loaded in one format (XTM, ISO, or LTM) and exported in another.

Full-text search

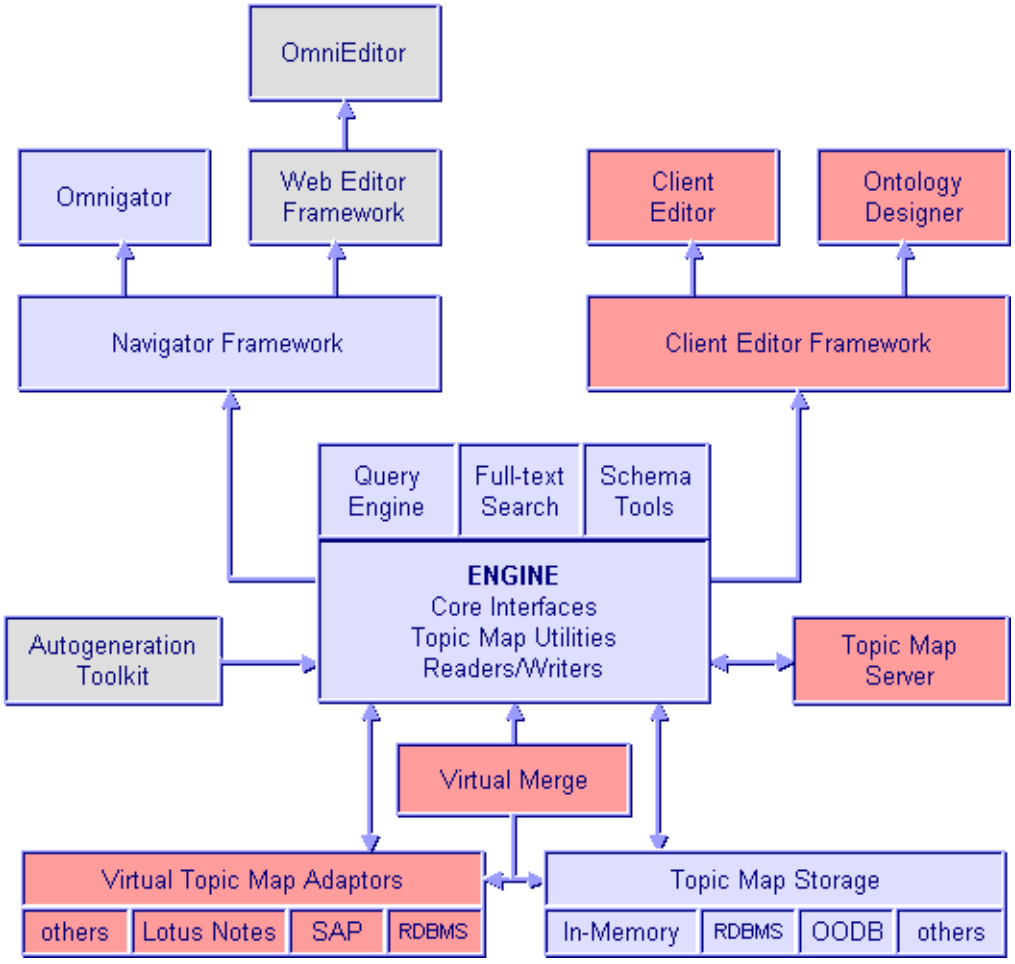
This plug-in comes with the full-text integration product and makes it very easy to do full-text searches on the topic map. Results are displayed in an intuitive manner, with links both into the topic maps and to any external resources in which matches were found.

More plug-ins are likely to be developed in the near future, since the simplicity of the plug-in concept makes it very easy to develop new plug-ins. A number of trivial examples in addition to those mentioned here are distributed with the Omnigator.

4. Future development plans

Ontopia will continue to improve the Ontopia Knowledge Suite, intending to make it a complete solution for all aspects of topic map application development. Upcoming releases will enhance the Engine and Navigator products, and also extend the Suite with new products that further simplify the creation of topic maps.

The diagram below shows the structure of the OKS as it is intended to be once completed. The parts in green are available now, those in blue in development (but available under some conditions), and those in red to follow at a later date.



Web Editor Framework

The Web Editor Framework is a toolkit that makes it easy to develop web-based topic map editing applications. The extensions will be general, so that organizations can develop their own web-based editing interfaces. This will help organizations solve the problem of how to create the topic maps used by topic map applications. There will also be a demonstration product (omniEditor), which can be used to edit any topic map.

Alpha versions of the editor are available for evaluation on request.

The Autogeneration Toolkit

This is a framework for automatically processing various kinds of data sources into topic maps. It can read XML documents, CSV files, RDF data, emails, various kinds of text formats, and much more, process, wash, and improve the extracted information, and convert it into a topic map. The framework is only automatic in the sense that

once configured with an XML document that sets up the processing chain it can produce the output automatically. It can not automatically configure itself.

The toolkit is currently used in consulting projects and can already be purchased in connection with our consulting services.

Virtual merge backend

This engine backend sits as an umbrella above a set of topic maps and presents a unified view where the topic maps have been merged into a single topic map. The unified view will be a dynamic view of the data in the underlying topic maps, and will change as the underlying topic maps change. This can be very powerful when integrating information from diverse information sources.

Virtual Topic Map Adaptors

These engine backends provide live views of the data stored in structured repositories of various kinds as topic map data. They are configured by declaring how to express the information in the repository in topic map term and then allow access to RDBMS servers, LDAP servers, and other repositories as if they contained topic map information.

Together with the Virtual Merge backend these Adaptors can be used to integrate disparate applications together with the other information resources of an enterprise into a unified whole.

Topic map server

The topic map server will allow client applications to connect to a topic map server to retrieve topic map information, and also to write topic map information to a server. This will be a great help in developing scalable distributed topic map applications for cases where distribution via a persistent backend is not appropriate.

Products and features beyond what is mentioned here are also planned. Ontopia is dedicated to providing a complete suite of topic map products ready to revolutionize application development, and what is described in this white paper is only the first stage.

Waldemar Thranes gt. 98
N-0175 Oslo, Norway
http://www.ontopia.net