Project Number: 032518

QVIZ

Query and context based visualization of time-spatial cultural dynamics

Specific Targeted Research Project

Information Society Technologies

Toolkit architecture report version 1
D 5.1

Due date of deliverable: 31/04/2007

Actual submission date: 10/05/2006

Start date of project: 01/05/2006 Duration: 24 month

Umeå University Revision 1

Abstract

Number and name	Project Number: 032518 QVIZ, Query and context based visualization of time-spatial cultural dynamics		
Workpackage	WP 5		
Task	D 5.1		
Date of delivery	**Contractual: 31/04/2007**	**Actual:** 10/05/2007	
Code name	032518	**Version**	draft ☐ final ☑
Nature	Toolkit architecture report version 1		
Distribution Type	Final		

Authors (Partner)

Fredrik Palm fredrik.palm@humlab.umu.se

Vojtech Kupca vojtech.kupca@humlab.umu.se

Bob Mulrenin, bob.mulrenin@salzburgresearch.at

Mona Bonta Bergman, mona.bontabergman@humlab.umu.se

Kalev Koppel, Kalev.Koppel@regio.ee

Johan Forssell, johan.forssell@ladok.umu.se

Contact Person Mona Bonta Bergman mona.bontabergman@humlab.umu.se

Abstract Toolkit architecture report version 1 for the QVIZ project, co-funded by ICT Research Framework Programme of the European Union. Handling components structure and general architecture.

Keywords List Architecture, Components, Functionalities, Interface, Protocols, Interactions, Technologies, Tools, Framework, Platforms, User Requirement, User scenario

Table of contents

1. Executive Summary

This report describes the architecture of components used in the prototype implementation of the QVIZ portal. Specifically, the Query Visualization Environment and Collaborative Environment parts will be discussed. We will present current results as well as future plans.

The Query Visualization environment helps the user search, retrieve and organize large amounts of information. At this stage the information consists of an administrative unit ontology and its associated archival resources. Two tools are provided for this task, the Time-Spatial client and the Faceted Query component. There is at the moment a small amount of interaction between these two, with a larger amount planned for future release. The Time-spatial client uses a map and time bar to show administrative units and associated archival resources. The Faceted query component displays facets; lists of categories, which might be hierarchically ordered. Choosing different facets and categories will show different search results, in this case archival resources.

Time-Spatial and Faceted Query coherent:
http://polaris.regio.ee/qviz/index_qviz3.php

Time-Spatial
http://polaris.regio.ee/qviz/

The Collaborative Environment can currently be accessed through a Salzburg Research test webpage where users can utilize basic functions which implement collaborative knowledge building. The main features include basic resource search, classification, annotation and resource structure visualization. An integration with Query Visualization environment is expected in the second year of the project.

Collaborative Environment
http://qviz.salzburgresearch.at/tc/qviz/

2. Introduction

Use Cases developed in WP2 provided us with the approximate picture of how the main QVIZ components should interact among one another and with users. These Use Cases, presented in D4.1.2 System Specification and Requirement Report (1st Phase), laid foundations to our understanding of the implementation side of the system. Further development of the Use Development was carried out through the wiki, skype discussions s well as through physical meetings hosted by the partners. Decisions were made on how to split up responsibilities among partners during the implementation phase. The part of Query Visualization Environment was assigned to UMU (Faceted Query Environment) and REGIO (Time-Spatial Component) wheras the part of Collaborative Environment was done by Salzburg Research.

2.1 General architecture

As was stated above, the QVIZ portal is composed of two main parts; the Query Visualization Environment and the Collaborative Environment. For the purposes of the first prototype these two parts will function separately and will be used mainly as a demonstration of the concept and as indispensable learning tools for the next software cycle and future implementation efforts during the second year of the project.

3. Query Visualization Environment

Visualizing queries is one of the more important tasks in this project. With a faceted query system one can narrow down search results by selecting different facets. The result of this search can be visualized on a map. If we are choosing facets describing different levels of administrative units, the map might center on each chosen units. If we have chosen a facet with time periods as well, the map might animate changes in administrative units over that time period. The basic interface will include a faceted browser, a map interface and a time bar interface. This environment is purely a search tool.

3.1 Description of Components

The architecture, data structure and interface communication specifications of the different components are described below as well as interrelations.

3.1.1 Core Portal Components QVIZ User Management

Core portal components QVIZ user management is not being developed in this phase of the project.

3.1.2 Time Spatial Client

The Time Spatial Client consists of two core components – map and time-bar. The purpose of TSC includes visualizing and filtering query results made in the system and provide other Internet GIS capabilities. The map component and time bar are interlinked with other system components through the Javascript controller as a client (Figure 1).

The core components of the time spatial client are:

1. Faceted browser
2. Result list
3. Contextual data area
4. Time bar
5. Map
6. Javascript client – controller which links together components in client

The core components of the time spatial back-end are:

1. Faceted browser back-end
2. Map server

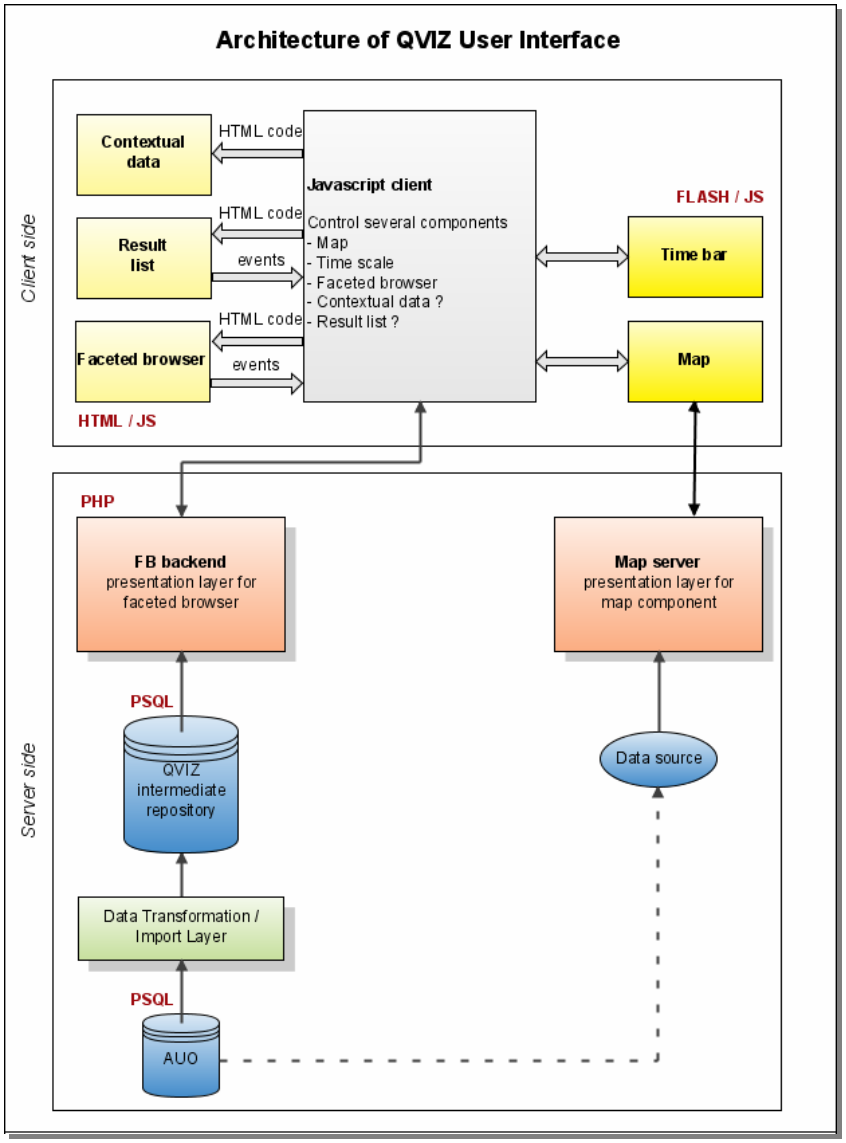

Figure 1. Architecture of QVIZ user interface.

3.1.2.1 Map component architecture

The Map component is based on the *Macromedia Flash 7.0* software what is a wide-spread platform for creating web applications. The Map component can handle both raster and vector graphics. Raster images in the background are ordinarily used as a base map while vector graphics in the foreground stand for the thematic information such as points of interests etc. The Flash map client loads raster images from the map server and vector data in XML format from some database server. Vector and raster data are combined in the end user's browser.

Raster data handling is based on raster-tiling technology. The tiling consists of cutting a large raster file in many rectangles which could re-assemble on demand. The fact that tiles are loaded asynchronously when navigating on the map makes the application fast and improves usability.

The following figure gives an overview of the communication between the components within the map component architecture.

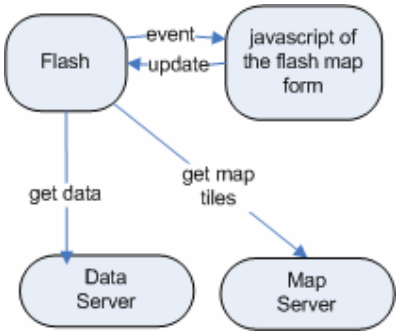

Figure 2. Communication Diagram

The map component is integrated into HTML page. JavaScript is used to load flash component, to update its state and to handle its events. The Flash component uses a data server to get vector data (XML). The Flash component loads map images from map server.

The following system requirements must be fulfilled for using the map component:

Web server with PHP support	Apache or IIS web server with PHP (version 4 or higher)
XML data source for layers and objects	Any database with ability to convert database data into XML files according to the required schema (PostgreSQL, Oracle, DB2, MySQL, MSSQL, etc)
Tiling server	A file server containing the pre-generated tiles (i.e. map images to be shown in the map component).

The end user, i.e. the visitor of the web page, must have Adobe (Macromedia) Flash installed and JavaScript enabled in web browser.

The flash map client is highly configurable through a set of configuration settings and its functionality is controlled by Javascript API.

3.1.2.2 Features in the first prototype

1. Zoom in and out using zoom-bar. Each AU level (type) has predefined zoom-ranges and when zooming in and out, the AU-s displayed on thematic layer would be replaced automatically.
2. Pan map window. Change the location of the map without changing the zoom factor

3. Send the extent of map window to the JavaScript client. The Map client sends the coordinates of the bounding box, the coordinates of centre point of the map window and the zooming factor values to the client.
4. Thematic mapping. The Map client is able to display coloured choropleth map using frequency numbers about the spatial distribution of resources or community activities.
5. Zoom to object. Map zooms to selected object and highlights its borders using administrative unit id code.
6. Select object from map. When clicked on, the map client sends the administrative unit id code to the JavaScript client.
7. Background map layer. Display background map image using raster streaming technology.
8. Select time moment or period. The time bar component has 3 slider arrows for selecting either time period or discrete time moment. The time period is a so called time window in which the timebar has its slider arrows (start_date, end_date)
9. Refresh time-slider component using values sent by the JavaScript client.

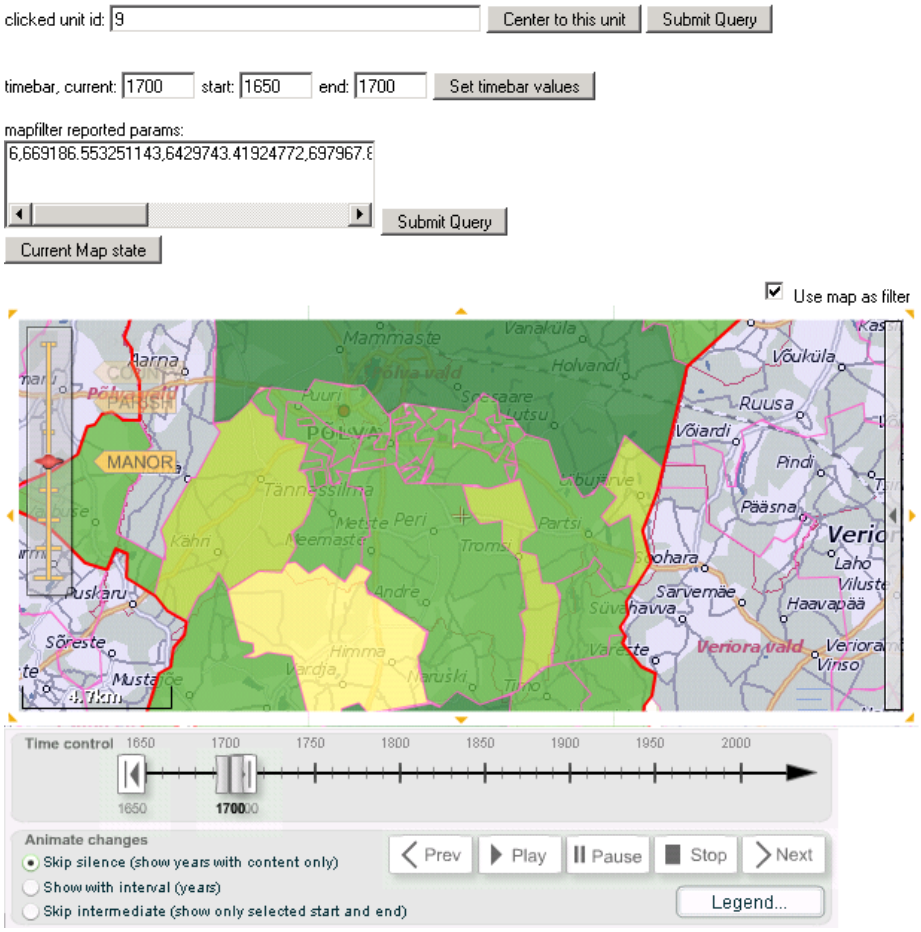

Figure 3. Test page of TSC (http://polaris.regio.ee/qviz/).

3.1.2.3 Challenges and solutions

In the next phase of the project we focus on the following features:

- The Thematic map frame-by-frame like animation support on client- and server-side
- User editable map-layer. User can draw point, line and area objects in the system and add comments to them.
- The Dynamic time-bar. The Time-bar should be dynamic according to the time period and time ranges presented in it. The Time-bar should indicate historical periods.

Minor issues:

- Multi-polygon support. The Map component does not support multiobjects on client side. A large proportion of administrative units consists of separate parcels and might have holes in them. Multiobjects include multipoints, multilines and multipolygons, but also collections which contain different geometry.
- Highlighting objects on mouse roll over event.
- Fine tuning of map visualization.

3.1.3 Faceted Query Component

This part will deal with the Faceted Query Component (FQC) and look at its main features from both the user and technical points of view.

FQC is an integral part of the QVIZ environment. It is meant to be a convenient starting point in a data search process done by users. It consolidates different types of data into logical groups or topics. It comprises a set of so called *facets*, each containing a list of categories related to a particular topic. For example, a facet called "Countries" would have individual countries listed as its categories.

Faceted classification (and browsing) allows users to access a general conceptual hierarchy through a user-defined path. This is done by changing the *order* in which individual facets (or topics) are displayed. In this way, users can give different priorities to how the data are explored and presented.

A hierarchical structure of data was considered; where categories in the same facet have their direct predecessors in a *common* higher-level facet creating a classical tree hierarchy (e.g. regions "Västerbotten" and "Norrbotten" have a common predecessor "Sweden"). In QVIZ, however, and specifically in browsing of Administrative Unit Ontology (AUO) data, categories in the same facet can have a direct predecessor in *different* higher-level facets creating an oriented graph.

3.1.3.1 FQC architecture

From the user point of view, the FQC is a part of the front-end user interface and is closely linked with the Time-Spatial Component (TSC) (i.e. a Map with a timebar). Internally FQC has two basic parts:

- a client-side which is essentially a dynamic web user interface whose purpose is to display facets and their content, register user events, fetch data from the backend server and call TSC function if necessary;
- a server-side which acts as a backend to the client side. Its purpose is to provide the client with the data optimized for the needs of the user interface.

The FQC is implemented using dynamic web development techniques broadly called AJAX. These techniques include HTML, CSS and JavaScript on the client-side as well as PHP programming on the server-side.

We use a simple exchange format between client and server where individual client-side webpage elements are populated upon obtaining *"ElementCode1 | ElementData1 | ElementCode2 | ElementData2 | …"* pairs from the server. The *ElementCode* can be made of three different kinds: a facet number, character 'R' and/or character 'C'. These codes are transformed into corresponding HTML elements in the webpage: either into a particular facet element, or a *Result List* element or a *Contextual Data* element.

3.1.3.2 The first prototype features

The functionality of the first prototype is limited to basic browsing of hierarchical AUO data. Users can perform the following actions:

- Add an arbitrary facet from the list of *Inactive Facets* by clicking on a facet name label (see Figure 4). A selected facet is moved to the first free position in a list of *Active Facets*. If the facet is the only one in the list of *Active Facets*, it is automatically populated with all available data related to that facet.
- Remove an arbitrary facet from the list of *Active Facets* by clicking on a facet name label. A selected facet is moved back to the list of *Inactive Facets*. All lower-order facets (to the right of the removed one) are erased. If the removed facet is the first one, the list of facets shifts to the left and the new first facet is automatically populated with data.
- Swap two adjacent facets by clicking on swap symbols in the upper corners of active facets. This operation allows users to alter the order in which facets are displayed. The data in all lower-order facets (to the right of the swapped ones) are erased. If the swap includes the first facet, the data are automatically loaded into the new first facet.
- Browse hierarchical AUO data by clicking on individual facet categories. A click initiates a series of actions: a) the content of the next facet is loaded; b) an information related to the currently selected category is displayed in the *Contextual Data* frame; c) a list of resources related to the currently selected category is displayed in the *Result List* frame (see Figure 5).

It is necessary to note that data in all lists are sorted in either alphabetical order (for AUO categories) or descending numerical order (for years). Also notice that for the time being, resources shown in the *Result List* contain only a set of blind links.

The data used for the first prototype cover several Estonian administrative units (AU's) structured in a simple tree hierarchy. Included in our example are Estonian

AU levels of *maakond* (county), *kihelkond* (parish) and *mõis* (manor). The following table gives an overview of the size of the data used (numbers in parentheses indicate the number of resources for each AU).

Table 1 Estonian Administrative Units

maakond	kihelkond	mõis
Võru (67)	Põlva (67)	Jaanimõisa (1), Kähri (6), Tilsi (9), Timo (8), Tõdu (9), Uibujärve (6), Varbuse (9), Vastse-Koiola (9), Virumaa (1), Võru (9)
Pärnu (5)	Karksi (5)	Pöögle (5)

Interaction with the TSC is currently limited to a) the highlighting of the selected category in the map and b) the updating of Timebar based on the selection in the year facet.

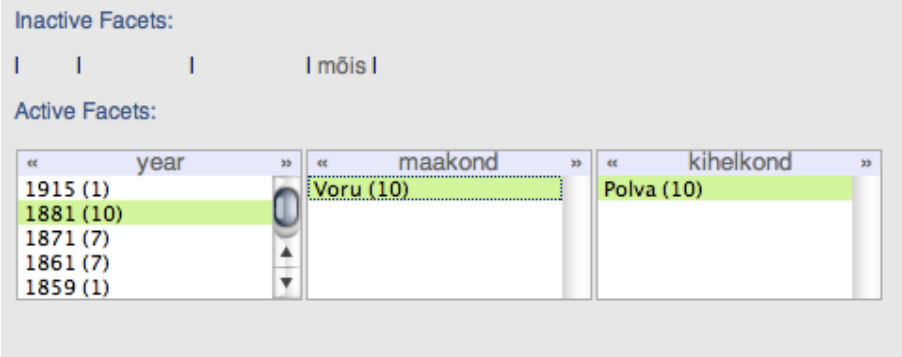

Figure 4 Faceted Query (http://polaris.regio.ee/qviz/index_qviz3.php)

200250813862: (1881-1890) Personaalraamat XXVIII (Kioma, Varbuse, Mooste, Kauksi, Jaanimõisa)
200250813861: (1881-1890) Personaalraamat XXVIII (Peri, Kähri, Tännassilma, Karilatsi, Tõdu, Kioma)
200250813861: (1881-1890) Personaalraamat XXVIII (Peri, Kähri, Tännassilma, Karilatsi, Tõdu, Kioma)
200250813853: (1881-1890) Personaalraamat XXVII (Väimela, Tilsi, Joosu)
200250813862: (1881-1890) Personaalraamat XXVIII (Kioma, Varbuse, Mooste, Kauksi, Jaanimõisa)
200250813866: (1881-1891) Personaalraamat XXIX (Vana-Koiola, Partsi, Viira, Uibujärve, Saarjärve, Põlva pastoraat, Vastse-Koiola)
200250813874: (1881-1891) Personaalraamat XXIX (Vastse-Koiola, Timo, Laheda)
200250813874: (1881-1891) Personaalraamat XXIX (Vastse-Koiola, Timo, Laheda)
200250813866: (1881-1891) Personaalraamat XXIX (Vana-Koiola, Partsi, Viira, Uibujärve, Saarjärve, Põlva pastoraat, Vastse-Koiola)
200250813851: (1881-1895) Personaalraamat XXVII (Võru, Väimela)

Figure 5 Result List (http://polaris.regio.ee/qviz/index_qviz3.php)

3.1.4 Time Spatial Middleware

The Time Spatial Middleware component provides interconnection with spatial databases like PostGIS.

We have implemented a PostGIS driver which converts data requested from Postgres into XML format suitable for the map client.

The Server component has proprietary algorithms for processing geospatial data to be effectively transferred over the Internet. It means that we are using vector data generalization algorithm to reduce data traffic.

The Time Spatial middleware does not render any map images; they are requested from the separate map server by the map client.

3.1.4.1 Challenges and solutions

1. The Generalization algorithm can handle Cartesian coordinates only, but the QVIZ data model requires geographic coordinates in the WGS84 system. The problem could be solved by implementing separate coordinate converters or make conversation on the fly in Postgres.
2. Thematic mapping uses static ranges for frequency numbers. We implement dynamic thematic mapping generation algorithms on server.
3. This map component uses its own XML format, and currently it is not structured according to OGC standard such as WFS for vector Data.

3.1.4.2 System requirements

- Apache Tomcat --- servlet engine used for running Java servlets
http://apache.datanet.ee/jakarta/tomcat-5/v5.0.28/

- Log4j --- library used for logging

http://apache.datanet.ee/jakarta/log4j/binaries/jakarta-log4j-1.2.8.zip

Apache DB --- database pooling

http://db.apache.org/

Licensing terms of these libraries are given at:
http://www.apache.org/licenses/LICENSE-2.0.html.

3.1.5 Faceted Query Component Middleware

This component is referred to as “Data Transformation/Import Layer” in Figure 1. The fully developed FQC will need quick access to a number of data sources in order to populate facets with contents. The two primary sources of data are the Collaborative environment and the Administrative Unit ontology. For the first prototype of the FQC only the AUO is considered. As the FQC only needs parts of the AUO structure there are some advantages to extract and restructure necessary data into an intermediate database for fast client access. See QVIZ intermediate database in Figure 1 above. At this stage the only purpose of the faceted query component middleware is to periodically import data from different data sources into the intermediate database. Right now it is only the AUO and Resource data that is being imported, but in the future all data which is to be presented in the faceted browser will have to be imported into this database (e.g. data from the collaborative environment).

3.1.6 Portal Data Storage

Portal data storage is not being developed in this phase of the project.

3.1.7 Administrative Unit Ontology Incorporation, GIS Storage

The Administrative Unit Ontology is described by a complex database schema, created to model administrative unit hierarchies for any country. Included is also GIS data for administrative units. The FQC only uses parts of the ontology model, which the middleware extracts for it, see above.

4. Collaborative Environment

A Collaborative Environment will add to the context provided in the QVIZ portal. The goal is for users in communities to create new knowledge and content, and associate knowledge from Communities of Practice (CoP) with archival resources referenced in archive portals, as well as new content in the collaborative environment. By use of semantic annotation to describe or add value to resources and interrelate them, users can enhance the comprehension and access of both new content and archival content. Furthermore, social objects help to interrelate threads and comments relating to content.

Overall, the focus is to enhance access to archival resources by community based annotation of resources and their relationships to other resources. Users, however, must be motivated to add value to resources, and therefore, content creation and relevant domain ontologiess are considered, especially to support publishing needs, social software features, resource and result collection management, simple knowledge organisation and building (thesauri, taxonomies, etc), and visualisation.

Additionally some users might enhance or extend the domain ontologies as well within the environment.
In the future, there will be additional features to support CoPs, administrative unit map social activities, and archive portal social bookmarking.

4.1 Description of Components

The relevant high-level components or interfaces relating to WP4 are described below.

4.1.1 Collaborative Environment Tools

Comprehensively this is one "portal" that will later be integrated with the Query Visualisation portal. This portal is responsible for the collaborative environment for knowledge building. It is deployed as a java web application; however, it is heavily supported by AJAX technologies using Dojo AJAX libraries.

The portal can be accessed at http://qviz.salzburgresearch.at/tc/qviz/

Users can register and login, perform a basic search, create content, assign ontology classes to gain access to annotation properties, perform semantic annotations, and visualise connections between resources based on the ontology properties.

The knowledge content query manager will provide services for external components in the future.

The domain ontology is the first version, however. It is expected that users will need to extend to include the property relations and classes relevant to their needs. We describe the domain ontology in another section.

The content management and ontology or knowledge base provide a means to associate ontology instances to content 1:1. Each instance in the ontology has a

corresponding content page. Likewise, any ontology class or property also has a content page.

Each content page can be typed by one or more ontology classes and therefore infer the class properties for annotating the content page. The goal is to build connections between resources using named relations from the ontology.

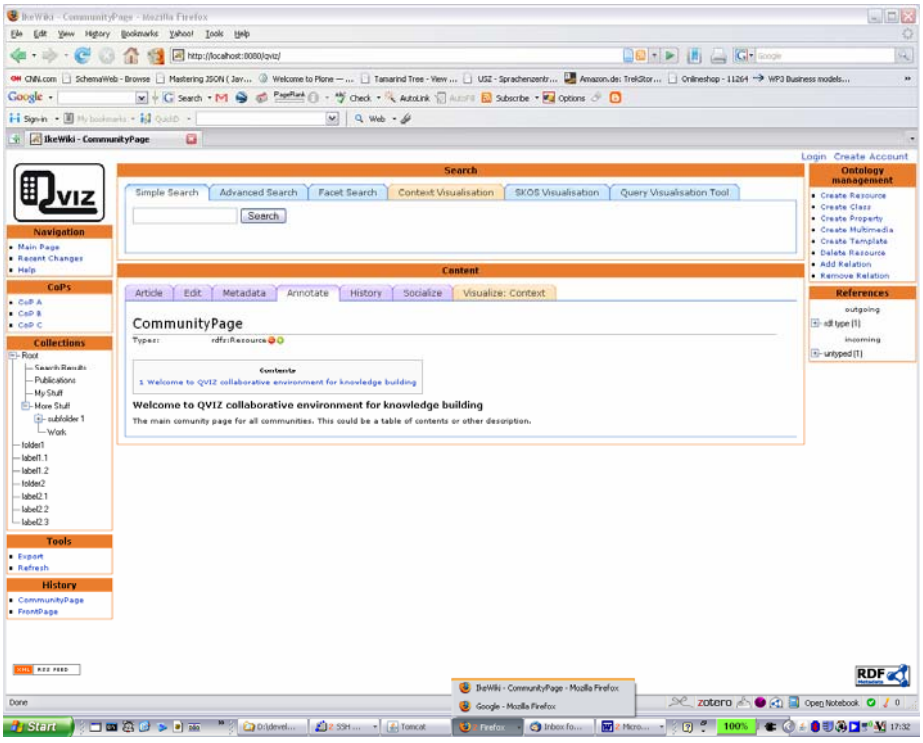

Figure 6 - Basic layout

In Figure 6, the basic layout of the portal is shown. It is composed of portlets and some portlets can have additional divisions such as a tabbed layout.

In the main centre area, a variety of portlets can be stacked, currently there is a search portlet and content portlet where the user can perform various activities to create content, link and provide semantic relationships between resources.

The collections portlet is a semi-functional test portlet which will provide the user with features to manage references to resources and share with the CoP as well.

Basic visualisations include a graph visualisation displaying connected resources and a list of relations and types associated with this resource and others.

Ontology management is also available, but like the semantic annotation functionality, the usability needs to be addressed in WP6 which focuses on User interfaces and User interactions.

Basic user management and access control are available, but the CoP based user roles and access control need to be further elaborated and tied to the user management solution in the next phase. Each CoP will have a unique namespace, and that will be one basis of the controlling access to resources created by CoPs.

4.1.2 Knowledge Content Query Manager

The knowledge content query manager will provide the main access for external components. This will provide access to the internal search components. It is possible to access this by clients using JavaScript to perform a remote call. As part of the dojo plug-ins for the query visualisation portal, we will provide it with the means to access the knowledge content query manager API. The API development is dependent on the facetted query development and when integrating the portals, the facetted query component and the Map GUI interface need to provide requirements to develop the content query manager API.

4.1.3 Plug-ins

The (current) plug-ins;

- Access from the client browser environment to the server environment for selected objects such as the collection manager.

Relevant plug-ins in the near future;

- Integration related plug-ins are not yet available, but will enable communications between the collaborative environment portal and the query visualisation portal.
- Communication between the Archive portals and the collaborative environment at the level of the internet browser client.
- General Dojo based plug-in for handling messaging between browser applications for query visualization or archive portals; each with client libraries to communicate with the collaborative environment.

4.1.4 Social Knowledge Content Tool

Provides the ability to;

- create content using a basic WYSIWYG editor and wiki-style open linking.
- perform semantic annotation on resources to type and describe the resource, and to interrelate resources using the domain ontologies.
    - however, we need to address usability and user interactions in WP6 to address additional Use Cases
- perform basic visualisation of related resources and
- perform a simple search
- perform basic ontology management tasks

- requires expertise, usability must be addressed

The basic WYSIWYG editor will be replaced with a more powerful editor for content building and we will add addition features such as:

- Wiki style open editing link with helpers for typing the content (ontology)
  - Usability will be addressed later to replace this wiki-like linking syntax with one or more editor buttons and dialogs
- Link building combined with search & result selection

4.1.5 Collection Manager

We provided the initial infrastructure for the collection manager and the means to perform (JSON-RPC) remote procedure calls to the collection manager. Currently, it supports testing of the client side (Internet browser) for collection handling actions on the client using AJAX technologies using the dojo library. The Dojo based User interface features are currently being developed and consequently the User Interface/AJAX development drives the features and API provided by the collection manager.

4.1.6 Communities and Member Management

- Ability to create a CoP, however, WP6 will add more user actions to address Use Cases.
- User management, but currently no specific CoP user management until the user management is integrated with the SIOC based ontology model for users in forums - a CoP is a subclass of a SIOC Forum.
- Permission management - current permission management will be extended to include namespace permission handling on resources from CoP (depending on the kind of relation of a resource to a CoP).

4.1.7 Publication Manager

When the KCO related tools are available, the publication manager will be addressed. It has not been implemented for the first prototype; however, users can create publications based on the domain ontology within the collaborative environment.

4.1.8 RDF based Repositories

4.1.8.1 Overview RDF based Repositories

We worked with both Jena and Sesame version 2 based repositories. Currently the 1st prototype uses Jena. The Jena model created is based on both RDFS and OWL.

Currently the Sesame V2 was alpha, now beta level. Our interest was to work with a future OWLIM version based on Sesame V2. If this works out, then commercial users would have the means to buy a commercial solution (BIGOWLIM) later.

Additionally, we have an experimental Sesame V2 testing environment, using our experimental SKOS API that is not part of the first prototype, but keeps us familiar with the new features and issues regarding the collaborative environment.

4.1.8.2 Domain Ontology

The domain ontology includes a number of existing ontology's including base ontology's to support portal features, community, and mapping between certain ontology's. There is an emphasis on the social activities that benefit from Argumentation, SIOC, Trust. Using FOAF we describe and relate persons, and tie in to SIOC. The FOAF document and FOAF Agent are used as either subclass or super class and help unite different ontology's.

The following lists the ontologies, excluding the base or mapping.

Table 2. Ontology's

DILIGENT Argumentation: Ontology		DILIGENT V.3 Includes classes such as: Issue, argument, argumentation, person, agree, disagree Any resource can be annotated with the argumentation ontology, although we see more usefulness combined with the social objects provided by SIOC (social "posts"). A user can use the argumentation ontology to augment the social context such as a web blog, comments on articles, the articles themselves might be part of the argumentation. In the future we will also see how to annotate segments with an article with argumentation ontology.
creative commons (cc)		Creative Commons Provides basic rights description for resources
Dublin Core (dc) Dublin Core Terms (dct)		Dublin core and Dublin core terms.
FOAF FOAF relationships		Friend of a Friend FOAF provides the basis for most of the

		domain ontology. The experimental FOAF relationships is provided only as a guide to users for creating more relationships between persons
SKOS Simple Knowledge Organisation		Simple Knowledge Organisation We expect more visualisation/navigation features and user features for creating SKOs vocabularies. Currently it is possible to create SKOS, but WP6 will provide more usability and user interactions. SKOS (requires more extensions to facilitate organisation of vocabularies, additional vocabularies to help classify vocabularies themselves). We must also look at content sensitive repository (named graphs) to enable isolation each vocabulary
SIOC		SIOC – Semantically linked online communities / This ontology supports Social interaction on resources (support blogging, forum, comments). Initially, we consider social interaction about the resource, but potentially, user could reference parts or segments or the resource and associate with a social object. Classes include: User, role, group, forum as superclass of "CoP“ Post, Site, community (not CoP, but more global); We include FOAF mapping and document mappings to other ontologies.
SIOC mappings		SIOC + mappings for FOAF, SKOS, internal system, user, roles. A CoP is a subclass of Forum, a group of users interested in a topic, and we are interested in the User model for relating groups, users, roles, forums and sites. A user can have one or more roles in a forum; groups of users can belong to a forum etc.
swportal		swportal Semantic web portal The focus is on community and publishing related activities, projects, proposals, work

		packages, organisations, publications, publication container, groups, events, persons, organisations, groups, etc)
Trust (mindswap)		Trust Ontology Trust of person or agent for a particular subject or resource. This is a means of ranking as well. We can use it to indicate trust of a resource, a quality assessment. Further extensions are expected, we expect that a visualisation for Trust would be useful. Users might consider combinations of trust, SKOS and Argumentation
VANN		"a vocabulary for annotating descriptions of vocabularies with examples and usage notes" http://purl.org/vocab/vann/
Archive Social book marking		Not included at this time until it is stable A Social bookmark, like any collaborative resource can have associated social objects. It was one issue whether to make the social bookmark a SIOC Post or not.
Not included in 1st prototype		
LOM		Learning object metadata
Geonames		geospatial test ontology The relevance of this ontology in the collaborative space to admin units from the admin ontology must be discussed
ISADG owl		ISAD(G) owl ontology Not included, but we must evaluate and understand how a tool can be created to handle this and whether it is useful. Currently, we use the social bookmark archive metadata to describe basic archive collection descriptions.
DOAP - experimental		Swportal does already consider "project", however, DOAP might influence us in another iteration of the domain ontology.

Experimental Not included		- Another rights related ontology – ipronto - conference_ESWC2006 - ebiquity ontologies - bibtex and SWRC were once considered however, swportal is better suited for our needs especially because of its relationships to FOAF and other popular ontologies.

4.1.9 General Storage

The Collaborative environment for knowledge building utilises a PostgreSQL Database server (version 7.4 or greater). It currently handles content storage for text, XML, and images.

4.1.10 Specialised Content Repository

The specialized Content Repository is a Java content repository (JCR) implementation. Its usefulness currently stands to its support of the collection manager which facilitates the organization and activities of the user and CoP workspaces and helps to also interrelate resource references from the collaborative environment, but also archive portal social bookmarks, query visualization and map activities.

The content storage needs are currently addressed using General Storage, until we need to switch. There are a few different ways to deploy this repository, currently it is an embedded repository, and however, it might be more useful deployed as a server function in the future.