REGNET

Cultural Heritage in REGional NETworks

IST-2000-26336

REGNET - Demonstration

Trial Service

Deliverable D10

Version 1

March 2003

REGNET

Cultural Heritage in REGional NETworks

Deliverable Report D10

REGNET - Demonstration (Trial Service)

Project acronym	REGNET	**Contract nr.**	IST-2000-26336
Type and Number	D10 – REGNET – REGNET - Demonstration (Trial Service)		
Work package	WP4 - Demonstration, Assessment and Evaluation		
Task	T 4.2 Refinement of system and services		
Date of delivery	2003-03-31		
Code name	RN_D10v01	**Version** 01 draft ☐ final ☑	
Objective	Report		
Distribution Type	Restricted		
Authors (Partner)	AIT, SR, TARX, MOT, SPAC, ZEUS, SI, CERT, VALT		
Abstract	This deliverable includes the final version of the REGNET prototype. It includes the changes and extensions based on the result of the validation activity (deliverable D7) as well as the modifications needed to improve the trial service. It refers to task 4.2 of work package 4.		
Keywords List	REGNET software, user and installation manual		

Table of Contents

Executive Summary

This document refers to deliverable D10 which includes the final version of the REGNET prototype. It includes the changes and extensions based on the result of the validation activity (deliverable D7) as well as the modifications needed to improve the trial service. It refers to task 4.2 of work package 4.

The deliverable D10 contains the code developed during the project and necessary to run a REGNET CSC (Cultural Services Center). It includes all documentation needed to install and run the system.

The first paragraph of this document contains internationalisation improvements done during the work package. Next paragraphs are dedicated to software ameliorations. Main improvements deal with integration of software components both in front and back-end point of view. Broker and connector components are described in detail, the first allows communication between software modules, and the second allows collaboration between REGNET stakeholders. General consideration about the need for integration is given then you find analysis and design artefacts for the modules.

Annexes are:

- A: Installation manuals. This part contain installation manual for REGNET components. It describes necessary software and installation procedures. The first paragraph called “Regnet Cultural Service Centre installation Guide” is dedicated to the overall installation of the software material from the REGNET CSC CD-ROM. Next paragraphs give details for each module installation in a standalone point of view.
- B: User manuals. These user manuals are dedicated to the REGNET components part of a CSC softwares. Users who want to learn how to use the software can use them.

Situation

	Demonstration, Assessment and Evaluation (WP 4)							
	Task	**Leader**	**Document**	**MM**	**Task**	**Leader**	**Document**	**MM**
Execution of the demonstration phase (trial service)	4.1	TARX	IR 4.1 → **D9**	55				
Refinement of the system and services were appropriate and necessary	4.2	VALT	IR 4.2 → **D10**	20				
Analysis of trial service, assessment and evaluation of the system	4.3	IAT	IR 4.3 → **D11**	6				

The demonstration efforts and all operational aspects of the trial service through the Cultural Service Centres approach are handled in Deliverable 9. D10 covers the technical refinements and additions carried out following the feedback of the tests while D11 treats the final assessment, evaluation and recommendations of the whole REGNET system.

1 Introduction

1.1 Purpose

This document contains artefacts from the task 4.2 "Refinement of system and services".

Main objectives of this task are to integrate into the REGNET system, feedback coming from the validation activity. According o this feedback, we identified the following activities:

- Integration: closely integrate REGNET application in order to provide implementation of main business processes identified during the previous Work Package. This requirement deals with the following software components: data entry, PCM/Eshop, Eprocurement and Z39.50 server.
- Improvement of software modules according to user feedback and internalisation needs.
- Development of a software component in order to integrate REGNET Cultural Heritage catalogs into the OAI (Open Archive Initiative) network.

As far as process used for REGNET project is based on Unified Process, this document contains iterations and artifacts description.

1.2 Overview and document structure

This document is structured in the following way:

- Introduction: this part. Contains a general description of the document.
- Internationalisation: describes new development done in order to internationalise the REGNET system.
- Back end integration: this part began by a general description of the need for back-end integration and describes design patterns used.
- Topic Map: extension of the topic map tools.
- Portal: extension of the portal component.
- Distributed REGNET: this part describes software modules (B2B connector and Broker) developed in order to provide integration.
- E-Publishing: extension of the component.
- Eshop: extension of the component.
- Product Catalogue Management: extension of the component.
- Auction: extension of the component.
- Index+ gateway: extension of the component.
- Cooperative thesauri: extension of the component.

2 Internationalisation

Internationalisation is based on forms defined by each partner in charge of a REGNET's module. These forms contain word to be translated in the different languages.

Each module is free to manage these forms according to the more suitable algorithm.

This work began during previous work-package. This part contains enhancements per module.

2.1 Portal

REGNET solution for internationalisation takes advantage of the localisation resources of Jetspeed tool to easily support multiple languages without hard-coding messages in the Java source code.

The main part of the internationalisation consisted in gathering all of strings that will be displayed to the end user into so called resource bundles. These are a set of files with a list of couples key-values like the following:

```
TOP_TITLE=Welcome to REGNET
CONTACT_US=Contact Us
```

The first string, TOP_TITLE in the example above, replaces a resource string that can be adopted in the presentation pages. There must be one file for each language required for the internationalisation.

Referencing internationalised strings in Velocity templates is accomplished with the localisation global tool, with a rather odd name of $l10n. It stands for the word "localisation", where the 10 middle letters are cleverly represented by the number 10. Any string in the resource bundle can be referenced within the presentation language statements as shown below (in the example there is an HTML piece of code):

```
<td align="center">
<h2>$l10n.TOP_TITLE</h2>
</td>
```

Turbine (the software layer on which Jetspeed resides) has a concept called modules, which is a special class loader path. Multiple module class paths can be configured by simply adding a path to the root directory of your modules. Modules are made up of different kinds of modules: actions, layouts, screens, navigations, pages, and localisations. Jetspeed uses the module path to find the resource bundles. Resource bundles and the default language are also specified in the Turbine.properties file:

```
locale.default.bundle=org.apache.jetspeed.modules.localization.JetspeedLocalization
locale.default.language=en
locale.language.supported=bg-ca-de-el-en-es-fr-it-ru-nl-sv
```

In particular, the language-codes are specified with an ISO-639 standard two-character language abbreviation, and stand respectively for the following languages: Bulgarian, Catalan, German, Greek, English, Spanish, French, Italian, Russian, Dutch, and Swedish.

Particular attention must be made in the encoding of the properties files, since Cyrillic and Greek charsets differ from the Western European charset supported by the web browsers. It was adopted the UTF-8 charset that supports all the required languages but this gave problems with Western European special characters like "àèìòù". This was solved by converting these special characters with an HTML escape code. Escape codes are used in HTML to convert the higher part of ISO-8859-1 characters (codes from 160-255). Below it is an example:

à (small a, grave accent) --> "&agrave;" otherwise "&#224;"
á (small a, acute accent) --> "&aacute;" otherwise "&#225;"
â (small a, circumflex accent) --> "&acirc;" otherwise "&#226;"
ã (small a, tilde) --> "&atilde;" otherwise "&#227;"

ä (small a, umlaut mark) --> "&auml;" otherwise "&#228;"
å (small a, ring) --> "&aring;" otherwise "&#229;"

To make simple the upgrading of the set of languages of the set of terms for the internationalisation, the properties files were isolated in an external library (Localization.jar) mounted in the library folder of Tomcat web server:

<tomcat home>\webapps\jetspeed\WEB-INF\lib

PSML resources (Portlet Structure Mark-up Language, that is the descriptors of portal resources for each user and role) may also be optionally localised. PSML Resources are localized by placing them in sub-directories based on language and country code abbreviations. The language-code sub-directory contains one or more country-code subdirectories. The language-code directory name is specified with an ISO-639 standard two-character language abbreviation. The country-code subdirectory is optional, and is specified with an IS0-3166 standard two-character country code abbreviation. An example:

```
user
  |-- david
      |-- html
          |-- fr                    // french language
              |-- FR                  // France country-code
              |-- BE                  // Belgium country-code
```

For a given locale of fr_FR, the search order for the default accounting resource would be:

groups/accounting/html/fr/FR/default.psml
groups/accounting/html/fr/default.psml
groups/accounting/html/default.psml
groups/accounting/default.psml

Nevertheless, this approach is not practicable because of the number of languages and the duplication of files to maintain aligned in case of upgrade. Therefore, another solution was adopted to internationalise elements that do not come from the templates but from the PSML resources, as for example, the portlet titles. Other templates (for example jetspeed.vm) process the PSML information as part of the Jetspeed logic. These templates were customized to intercept the *$portlet.title* property in order to substitute it with a resource bundle key matching into the *JetspeedLocalization* property files. This solution requires a new customisation of the tool whenever you want to add a new portlet, and should be revised in phase of engineering of the product.

As regards very verbose and static pages, it would be not practicable to manage them with the approach of the resource bundles, because this would lead to difficult to handle property files. A key could have a value as big as a paragraph, or there should be several keys for each paragraph line. Therefore, it was decided to embed the language selection directly into the templates of this kind. In other words, into the template, that is processed server side before sending it to the user, there is a test based on the current language, and based on the result, the correct translation is sent. This requires that all the translations must be into the template. As a result, these templates are usually very large. To intercept the current language, many methods would be adopted. It was decided to read the bundle key *LANG* that is translated differently in each language (e.g. "en" for English, "it" for Italian, and so on). An example of this approach can be found into the on-line help pages.

2.2 Data Entry

General improvements of the system:

1. Interface and preparation of internationalization update:

The Data Entry tool primarily focuses on the target group "expert user". Nevertheless the user interface was not user friendly enough. Previous efforts concerned mainly the internal functionalities of the system and did not take into account very much usability requirements.

- a navigation bar was implemented on the left upper corner containing the buttons for "restart", "new seach", "new record", "administration".

- all buttons for confiration (confirm search, confirm changes, save,...) are now on the bottom right corner.

- all buttons concerning the same action have same names.

- the term "document" was replaced by "record".

There are two ways to display the search result: the detailed view as now available in nearly all REGNET databases and the summary as now available in the SUL application (http://csc000.cscaustria.at/sul).

Now the work on the user interface is finished and we will have to send out tables to the partners in oder to update existing internationalization.

2. ONB application

Following feature have been developed for the ONB application (http://csc000.cscaustria.at/onb):

- a differenciated display of the search result. First of all so-called header cards are displayed and after selecting one or more header cards the user sees the index cards that refer to real objects/pictures.

- the user can select items, specify his needs for publication of the image and send an order to ONB.

- the design proposed by ONB is implemented

3. Synchronization of "doubled data"

Within the REGNET system object data is mapped into DublinCore fields that are contained in a seperate section of the object xml record. We have now implemented a feature called "synchronization" for automatic update e.g. of DC contents in case of changes in the object data fields. This feature is implemented using Xpath.

4. Display of special characters via the browser

Displaying special characters via browses caused a lot of problems in the last months, so the display of characters is now implemented on the basis of UTF-8 encoding.

2.3 Search and retrieval

No siginificant changes during this period.

2.4 Product Catalogue Management

In this section there are new words that were added in the Product Catalogue Management

Index Form

Return To Portal
To go back to the page of the portal

Insert item form

The word *price* has changed in *price per item.*

Price per item	Insert it to E-shop?	YES	NO

Update, delete or look form

The word *Let's go* has changed in *Go!*

Go!

B2B form

Return To Portal	Name

2.5 Eshop

Index page form

The new words are:

RETURN TO PORTAL	TERMS AND CONDITIONS	lower than	Higher than
RESULTS	Buy All	Currency	

Basket form

Buy All	Save Checked In Wish List	Clear Checked Boxes	Delete Checked	Update Quantities

Buying form

The words *Zip, Reser* do not exist anymore.

Clear	You have chosen to complete the next order	Days of Delivery	Choose the way of deliverance	Type	Sippment Cost
Choice	Normal Shippment	Express Shippment	Your order will be delivered in the following address	Do you want to continue?	Please select the payment
Payment with Credit Card	Deposit in a Bank Account	Please select the payment	An email with the comfirmation of your order will be delivered to you in a few seconds Please finish your order by submiting you credit card number in a Internet Secure Environment.	An email with the comfirmation of your order will be delivered to you in a few seconds The details of the payment are included in the e-mail.	Go To Credit Card Transaction

Wish list Form

Buy All	Clear Checked Boxes	Delete Checked	Update Quantities

New Search form

Lower than	Higher than

Confirmation form

REGNET E-shop, Confirmation Form	Order Comfirmation	You have ordered the following goods	Order ID	You want to pay by depositing in our bank account
You have to deposit	in the	The account Number is	After that please mail us a copy of the receipt of the bank in the address	These goods will be delivered in
If you have any problem please contact with e-shop administrator at kvotis@zeusnet.gr	Dear			

2.6 Ebusiness

Index form

Search for items	Search for services	Return to portal	Name	Category	UPCCode
Manufacturer	Store_Area	All	Price Type	Retail	Wholesale
Company ID	Company Name	Search items			

Search for items form

Currency	Results	Item_id	Category_id	Product Quantity	Order	Submit order
Contract #	Item	Supplier	Id_type	Street	City	State
Province	PostCode	Country	Phone	Fax	E-mail	URL
Warehouse address	The order was not submitted	The order was submitted	Sorry, there are no goods with your requirements			

2.7 Auction

Index form

Currency Converter	Return To Portal	User Agreement	Privacy Policy

Register new item form

There are additional messages in cases of not providing all the required information.

Enter product's name	Enter product's description	Enter product's producer	Enter product's start price
Enter product's reserve price	Wrong reserve price for item	Reserve price can't be less than start price	Enter year
Wrong value for year	Enter year in the end time	Wrong value for year in the end time	The start day is sooner than today
The end day is sooner than the start day	Please insert a image		

2.8 Topic Map generation

The internationalisation aspects of the knowledge layer on top of the information resources within REGNET pertain solely to the mechanisms used to express this knowledge. The retained mechanism here is the topic map paradigm. As a reminder, topic maps are in essence constructs in XML of topics (can be everything), associations between topics and occurrences information resources related to topics. One of the characteristics of a topic is its "base name". This base name can be "scoped", i.e. one can express its validity range. By this, the base name is the ideal means to realise internationalisation for the knowledge part. This is done by defining different base names, for one topic, everyone of them scoped by the corresponding language (which is also a topic referenced by an official subject indicator, e.g. the 2-letter ISO code for Dutch, nl, English, en, French, fr, German, de, etc.

So, the topics can be expressed in any language but what about associations and occurrences. Well, the topic map standard allows to express associations and occurrences as topics too. This means that we can use the same mechanism to express internationalisation aspects when the creator or end users find this necessary. By experience we know that this is done for associations together with an additional scope for the direction of the association between topics. In the SAINTS topic map example we have: "Saint Roche patronises the plague" and "the plague is patronised by Saint Roche". In this case we have two base names for the association between "Saint Roche" and "the plague", scoped by "Saint Roche" in the first case and by "the plague" in the second case. Again, we can apply the language scooping to every of these base names.

As far as the occurrences are concerned, the same method can be applied as for associations but a scope can also be put directly on an occurrence. This enables an immediate indication of the language the information resource is written in. But here the main effort lies in the content creation in different languages of the information resource.

Within REGNET there exist different ways to produce topic maps: on line structured TM data entry, off line structured TM data entry and on line topic per topic data entry. But all of them have in common that they generate the exchange format of topic maps, namely the XTM format.

Conclusion: by using standard features of the topic map standard, i.e. topic-base name-scope and occurrence-scope an excellent and easy means is at our disposal to express internationalisation aspects.

For an example of multilingual topic map generation, see next figure.

Example of a trilingual part of the Linnaeus topic map

<topic id>	<instanceOf>	<scope>	<baseName>	<scope>	<baseName>	<scope>	<baseName>	<end/>
organisation		en	organisation	se	organisation	nl	organisatie	
academy_of_science	organisation	en	scientific academy	se	vetenskapsakademi	nl	academie van wetenschappen	
geographical_area		en	geographical area	se	geografiskt område	nl	geografisch gebied	
city	geographical_area	en	city	se	stad	nl	stad	
country	geographical_area	en	country	se	land	nl	land	
region	geographical_area	en	region	se	region	nl	regio	
region_within_country	geographical_area	en	region within country	se	region inom land	nl	regio binnen land	
language		en	language	se	språk	nl	taal	
en	language		English	se	engelska	nl	Engels	
se	language	en	Swedish	se	svenska	nl	Zweeds	
nl	language	en	Dutch	se	holländska	nl	Nederlands	
name		en	name	se	namn	nl	naam	
old_name	name	en	old name	se	tidigare namn	nl	oude naam	
<end/>								

2.9 Topic Map Viewer

A new website has been set up in order to simplify topic generation. Currently each content provider has his/her own website to access the workspace for topic map creation and generation of visualization (see the manual).

Every topic map viewer uses as input the exchange format for topic maps: XTM. This means that, during the generation of an XTM file after data entry, one can choose to do this in only one specific language or in several languages. A program like the topic map viewer will reflect the chosen (first) language of the base names in the XTM file and stay with it all the time. If one wants to differentiate between different languages, two options are possible. The first is letting the end user a choice before the viewer is activated. The viewer will load then the appropriate language version. Changing languages means restarting the viewer. The second way is to add intelligence to the viewer in order to switch base names of a multilingual XTM file during the viewing process but this can be a rather complex operation.

2.10 Publisher

2.10.1 Web-based Java Prototype

The web based Java prototype is available in the languages we received translations in (Spanish, English, German and Bulgarian.) The excel sheet provided for the translations was transformed into Java resource bundles for internationalisation and therefore is compliant with the standard internationalisation mechanism of Java.

2.10.2 Macromedia Director Prototype

The Macromedia Director 8.5 is available in several languages, but the individual applications are not customizeable to different languages, i.e. there is no internationalisation for the Director Prototype available.

For the internationalisation of the e-Publishing modules, we are mainly bound to the means put at our disposal by the used third party software, Macromedia Director.

As a reminder, Macromedia Director uses the film paradigm for the generation of multimedia productions:

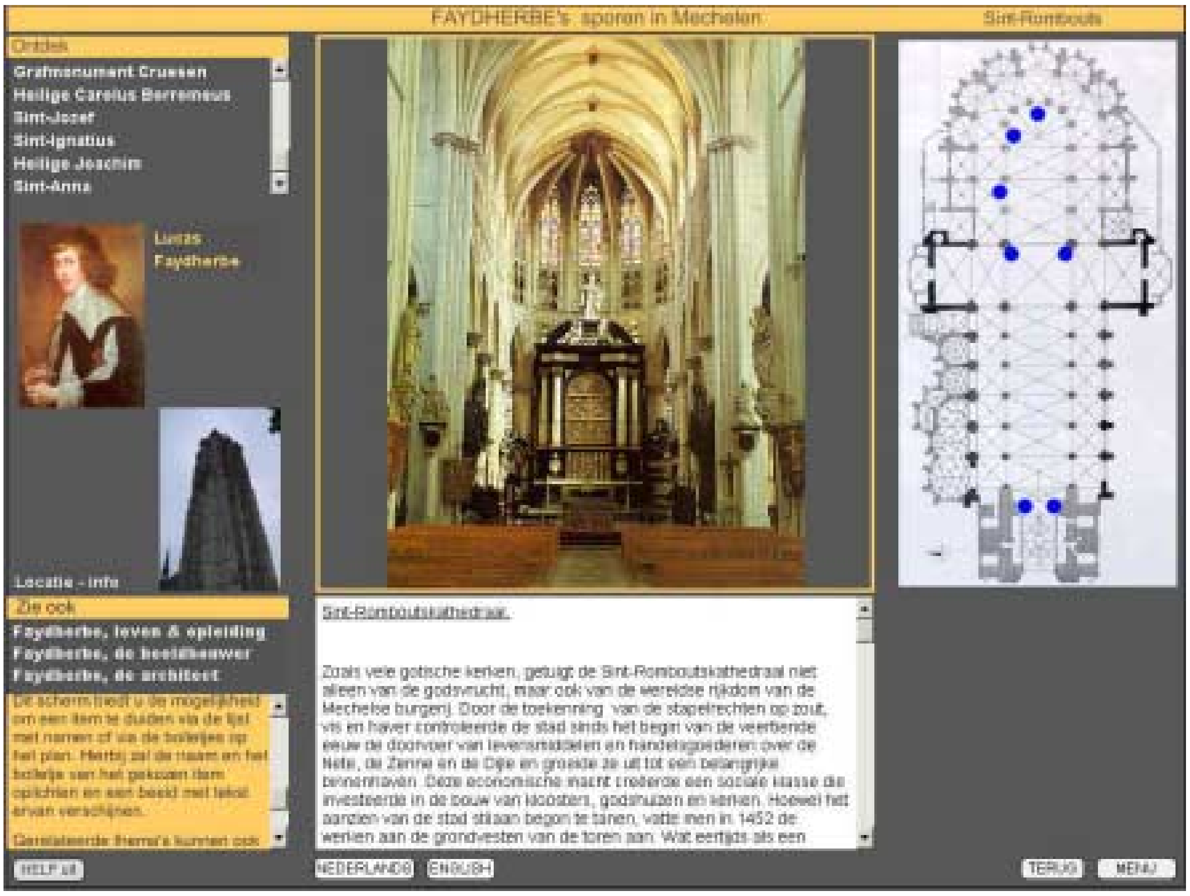
- stage: where the cast members will appear
- score: when and how long the cast members appear
- cast: the actors and stage accessories = cast members.

In the case of a multimedia cultural heritage production most cast members consist of images and texts. It was obvious to concentrate on the cast manipulation, namely cast switching, to realise internationalisation.

In order to obtain a flexible structure for real time cast switching and to guarantee a conveniently arranged cast generation and editing, the following structure was adopted:

- 1 internal cast containing all behaviours
- 2 push button casts (one in Dutch and one in English)
- 1 external images cast (mainly jpeg images)
- 1 internal images cast (mainly vector images)
- 2 external text casts (mainly thematic texts in rtf, one in Dutch and one in English)
- 2 internal text casts (mainly titles and captions, one in Dutch and one in English)

This method allows to switch, during the consultation or navigation of the production, languages at any moment and at any place. When doing this, the system changes the casts and the screen is immediately refreshed into the new language.

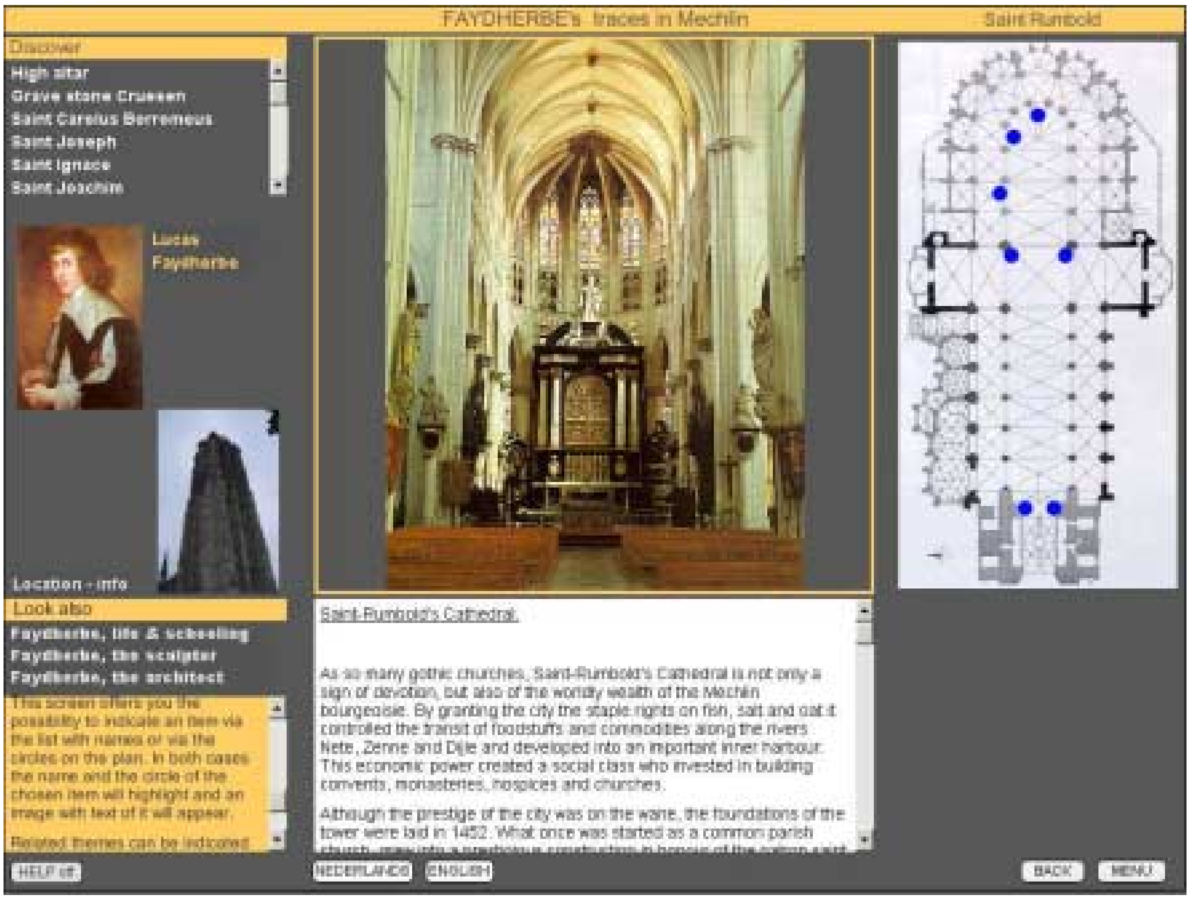


Dit scherm biedt u de mogelijkheid om een item te duiden via de lijst met namen of via de bolletjes op het plan. Hierbij zal de naam en het bolletje van het gekozen item oplichten en een beeld met tekst ervan verschijnen.

Gerelateerde thema's kunnen ook

Sint-Romboutskathedraal

Zoals vele gotische kerken, getuigt de Sint-Romboutskathedraal niet alleen van de godsvrucht, maar ook van de wereldse rijkdom van de Mechelse burgerij. Door de toekenning van de stapelrechten op zout, vis en haver controleerde de stad sinds het begin van de veertiende eeuw de doorvoer van levensmiddelen en handelsgoederen over de Nete, de Zenne en de Dijle en groeide ze uit tot een belangrijke binnenhaven. Deze economische macht creëerde een sociale klasse die investeerde in de bouw van kloosters, godshuizen en kerken. Hoewel het aanzien van de stad stilaan begon te tanen, vatte men in 1452 de werken aan de grondvesten van de toren aan. Wat eertijds als een



This screen offers you the possibility to indicate an item via the list with names or via the circles on the plan. In both cases the name and the circle of the chosen item will highlight and an image with text of it will appear.

Related themes can be indicated

Saint-Rumbold's Cathedral.

As so many gothic churches, Saint-Rumbold's Cathedral is not only a sign of devotion, but also of the worldly wealth of the Mechlin bourgeoisie. By granting the city the staple rights on fish, salt and oat it controlled the transit of foodstuffs and commodities along the rivers Nete, Zenne and Dijle and developed into an important inner harbour. This economic power created a social class who invested in building convents, monasteries, hospices and churches.

Although the prestige of the city was on the wane, the foundations of the tower were laid in 1452. What once was started as a common parish

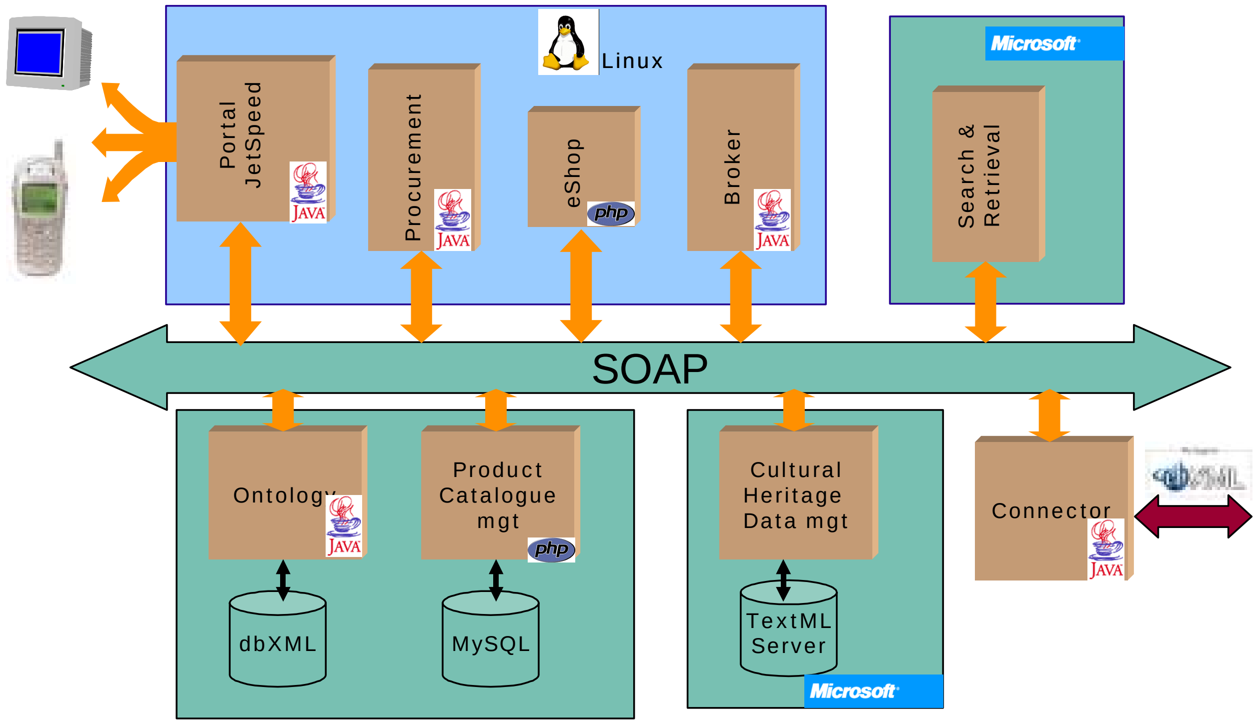
3 Back End Integration

The aim of these part is to give an overview of technologies used in order to take into account integration into the REGNET architecture.

3.1 The need for Integration

As described in previous deleverables from WP1 and WP2, two kinds of integration are necessary for the REGNET platform (see figure below):

- A2A (Application to Application) integration. Inside a CSC REGNET information system. Based on Web Services approach, which means that each component (Data Entry, PCM, Delivery, etc.) provides a WSDL interface and a set of services available through the SOAP protocol. The REGNET Broker describe later allow to transfer data from one component to another.
- B2B (Business to Business) integration. Between CSCs. Based on ebXML approach, it provides necessary framework in order to set up collaborations between partners from (not mandatory) many CSCs. The REGNET Connector describe later manage these collaborations. Lend process between partners has been implemented yet.



Next paragraphs will detail the approach. Its began by some remarks about the business position of the leading technologies which are Web Services, ebXML and EDI, then we give some pattern for B2B integration which are relevant to REGNET context. Broker and Connector are describing later (§ **Fehler! Verweisquelle konnte nicht gefunden werden.**).

3.2 Web Services, ebXML and EDI

Choice between the three technologies, which are Web Services, ebXML, and EDI in order to sup up B2B collaboration is not clear for many people. This topic is address in this part.

One can visualize EDI, ebXML, and Web services on a continuum rather than three distinct alternatives. EDI provides a fixed, predictable message format, which with high volumes and stable business processes make a lot of sense. With ebXML, one can have the messaging features of EDI, plus a larger framework of functions combining business process models, registries, company profiles, trading partner agreements, and semantic interoperability. While ebXML offers a complete framework, companies can implement parts of that framework, without having to swallow it all at once.

Web services offer some of the same business functions of ebXML - messaging, service descriptions, and registries - but in more component form. However, they both have their respective domains. For example, Web Services messages based on the SOAP protocol do not provide for reliable messaging per se. SOAP does not have the same support for reliable messaging as ebXML MS v 2.0. EBXML MS has a fire and forgets function that either reliably delivers the message or detects (and hopefully tells the sender) that the message failed. In general terms ebXML is about loosely coupled document-centric business collaboration while with Web services, the API-centric (or service interface) view dominates (rather than the document-centric view as with ebXML).

By providing a set of functions that users can assemble into sets of services, Web services also have the modular features of ebXML, but without the overall structure of ebXML. The business process determines the choice and configuration of those components.

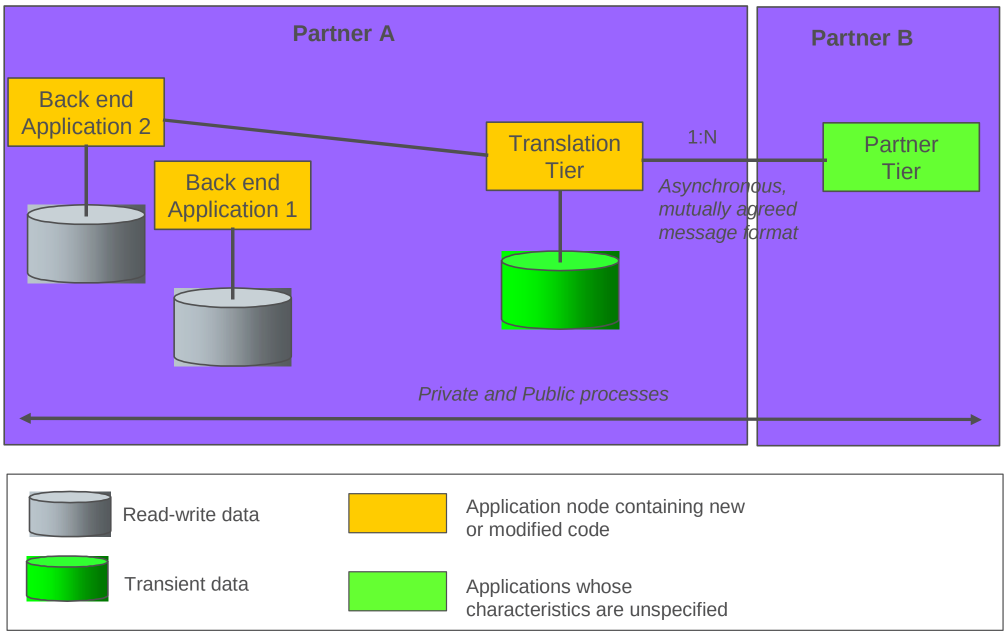
Basing technical solutions on business processes supports another feature of ebXML, namely the separation of business process from technology. This dichotomy goes back to the Open-edi Reference Model, ISO standard 14662 published in 1997, which recommends defining business processes independent of the technology. Open-edi recommends developing a business operational view (BOV) of a business relationship separate from the functional service view (FSV). The business operational view identifies the business processes, including the parties doing business, the interactions among those parties, the business messages exchanged as part of those interactions, and the semantics included in the messages. The functional service view discusses the information technology used in the interactions, in terms of functional capabilities, technical connections, and interchange protocols.

3.3 B2B design patterns

The specific business functionality supported by applications that automate the Extended Enterprise business pattern (also known as the Business to Business, or B2B business pattern) reveals certain common approaches that have been successful.

The five Extended Enterprise application patterns are presented here in order of increasing flexibility and sophistication. As the Application patterns build on each other, their capabilities and reliance on middleware increase, and they require less application development effort.

3.3.1 Document Exchange


Partner A
Partner B
Back end Application 2
Back end Application 1
Translation Tier
1:N
Partner Tier
Asynchronous, mutually agreed message format
Private and Public processes
Read-write data
Transient data
Application node containing new or modified code
Applications whose characteristics are unspecified

The Document Exchange application pattern helps to structure the batched electronic exchange of data using mutually agreed message formats.

Business and IT Drivers

- Improve organizational efficiency
- Reduce the latency of business events
- Support a structured exchange with business partner
- Leverage existing skills
- Leverage Legacy Investment
- Minimize Application Complexity

The primary business driver for choosing this Application pattern is to increase the efficiency of interactions between enterprises. Instead of exchanging paper documents, this Application pattern can be used to send and receive documents electronically. This eliminates the need for error prone manual re-entry of data.

This is the ideal Application pattern to choose if your current business needs would be satisfied by the batched exchange of electronic documents. In other words, at present your business requirements don't call for direct invocation of a business partner's systems in a real-time fashion.

Large organizations often require their business partners to exchange messages electronically. For example, they may mandate the use of Electronic Data Interchange (EDI) transaction sets over a particular Value Added Network (VAN) for certain interactions such as placing an order.

Solution

This Application pattern, as shown in the figure above, is divided into at least three different logical tiers: Partner, Translator and Backend application:

- The Partner tier represents a set of trading partner applications whose characteristics are unspecified. In other words, the technological implementation details of these systems are not disclosed. However, trading partners mutually agree upon the message format and the means of exchanging these messages.
- The Translator tier retrieves such mutually agreed upon messages from a persistent buffer and decodes them into messages that can be used by the internal business processes of the receiving organization. Decoded messages are then stored in a Work in Progress data store.
- The Backend application tier is responsible for processing the decoded messages. In doing so, it typically reads decoded messages directly from the Work in Progress data store. Because of this, there is a direct communication link between the backend application tier and the Work in Progress data store. After processing decoded messages, backend applications may in turn generate responses that have to be delivered back to the partner who originated the initial message. In such cases a reverse flow of messages may be observed.

If a business partner was requested to take part in a Request for Tender, a series of messages would need to be exchanged as part of this business process. In this case Partner A will have to build custom business logic which links together the applications processing these various types of messages in order to automate the overall business process. In such a case the private processes clearly get intermingled with the public process, and if the public process should ever change the whole business logic will need to be redone. Because of this, the figure above shows the Private and Public processes as a single entity rather than two separate entities.

Batched exchange of electronic documents implies asynchronous communication between partners. Because of this, there is a need for a persistent buffer that stores all the messages that are received from multiple partners to be translated and processed at a later time.

This Application pattern is best suited for implementing EDI based integration over a VAN between organizations. Alternatively one can use, EDI over web technologies that have emerged recently. In both cases the translator tier is implemented using EDI translation packages.

Guidelines for use

It is recommended that you choose an EDI translation package that is rules based that allows you to map EDI messages to internal messages using rules. This allows the quick definition and redefinition of messages. You may also want to consider more sophisticated translation packages that convert EDI messages into an automatic request to a transaction monitor such as an order processing system.

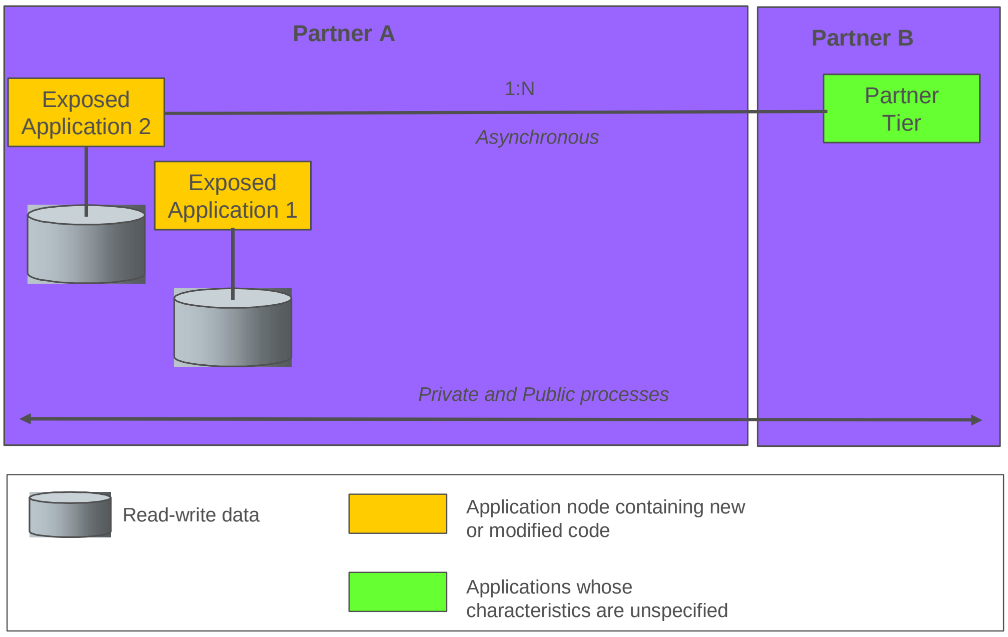
Benefits

The electronic exchange of documents increases the efficiency and reduces the latency of business interactions between organizations.

Limitations

As mentioned earlier, this Application pattern is best suited for implementing EDI based integration between organizations. EDI is a well-established standard but it is deployed only by a small number of companies. Participation in an EDI based network requires a subscription to a particular VAN that is typically expensive. Furthermore, it mandates the use of certain IT infrastructure by all partners. As a result you lose flexibility in connecting to business partners that might have different IT infrastructure capabilities. For these reasons, this Application pattern can be both expensive and inflexible.

3.3.2 Exposed Application


Partner A
Partner B
Exposed Application 2
Exposed Application 1
1:N
Asynchronous
Partner Tier
Private and Public processes
Read-write data
Application node containing new or modified code
Applications whose characteristics are unspecified

The Exposed Application application pattern helps to structure a system design that allows specific applications to be directly accessed by partner systems across organizational boundaries.

Business and IT Drivers

- Improve organizational efficiency
- Reduce the latency of business events
- Support a structured exchange with business partner
- Support partner real-time access to/ from applications
- Leverage existing skills
- Leverage legacy investment
- Backend application integration
- Minimize application complexity

The primary business driver for choosing this Application pattern is to enable business partner systems to gain direct and real-time access to specific applications. The underlying motivation behind these requirements is to improve the efficiency of interactions between organizations beyond what can be achieved with simple electronic document exchange.

Solution
The figure above illustrations how this Application pattern design is divided into at least two logical tiers: Partner and Exposed Application.

- The Partner tier represents a set of partner applications that are interested in invoking specific business logic on the Exposed Application tier.
- The Exposed Application tier may represent a new application, a modified existing application, or an unmodified existing application. This tier is responsible for implementing the necessary business logic and access to business data. Since this tier is directly exposed across organizational boundaries it must implement or exploit the necessary security features such as authentication, authorization, confidentiality, integrity and logging for non-repudiation purposes.
- Please refer to the previous Application pattern for an explanation of the Private and Public processes arrow.

Typically an asynchronous communication mechanism is used for inter-enterprise integration. The primary reason for this choice as opposed to using a synchronous communication mechanism is to minimize the dependency of the service levels of one organization's applications on another organization's applications. Even though companies agree on certain service levels such as availability and response time, it is hard to ensure across organizational boundaries that these service levels are met all the time. The use of an asynchronous communication mechanism ensures that during such a failure, requests can still be sent to partner systems to be processed at a later time. Meanwhile the application under consideration can still continue its processing without having to wait for the response from the partner systems. It also provides a degree of isolation between the two businesses so that the requesting application can never synchronously demand excessive resources or locks on the partner systems.

Note that asynchronous communication does not necessarily mean a delayed response. If the business requirement warrants a quick response one can consider fast asynchronous communication where the response is typically received immediately if the partner system is able to process the request at the time of receipt. If not, the request is processed at a later time. Fast asynchronous communication is becoming increasingly popular for inter-enterprise integration since it provides the benefits of both asynchronous and synchronous communication styles under most circumstances.

Predominantly an asynchronous Message Oriented Middleware (MOM) is used for implementing this Application pattern. It is possible to use synchronous middleware based on standards such as CORBA and DCOM. Most of the time the dominant partner in a trading relationship enforces the use of a particular middleware on the rest of the partners. Typically Virtual Private Networks (VPN) are used for interconnection between partner applications and to implement many of the security features at the network level.

Guidelines for use

Direct integration between applications is typically inflexible, in that any changes to one application may have knock-on effects on other applications. This is especially dangerous while integrating across organizational boundaries. Any changes to the exposed application tier may require changes to many partner systems. Such changes can be both expensive and time consuming. One can minimize such knock-on effects using message-based adapters that wrapper the applications in the exposed application tier. Message-based adapters are small programs that convert the mutually agreed upon messages into API calls to existing or new backend applications. This layering technique isolates the backend applications from partner systems and increases flexibility. Any changes to these backend applications would only impact the adapter, provided there is no need to change the mutually agreed upon messages.

Message definition should be generalized to further promote flexibility. In other words, messages should not be tightly coupled with backend application APIs. Rather the message should capture all the necessary information required for that logical interaction across business boundaries. Such generalization will help cope with changes to the backend application API without having to change the agreed upon message format.

Benefits

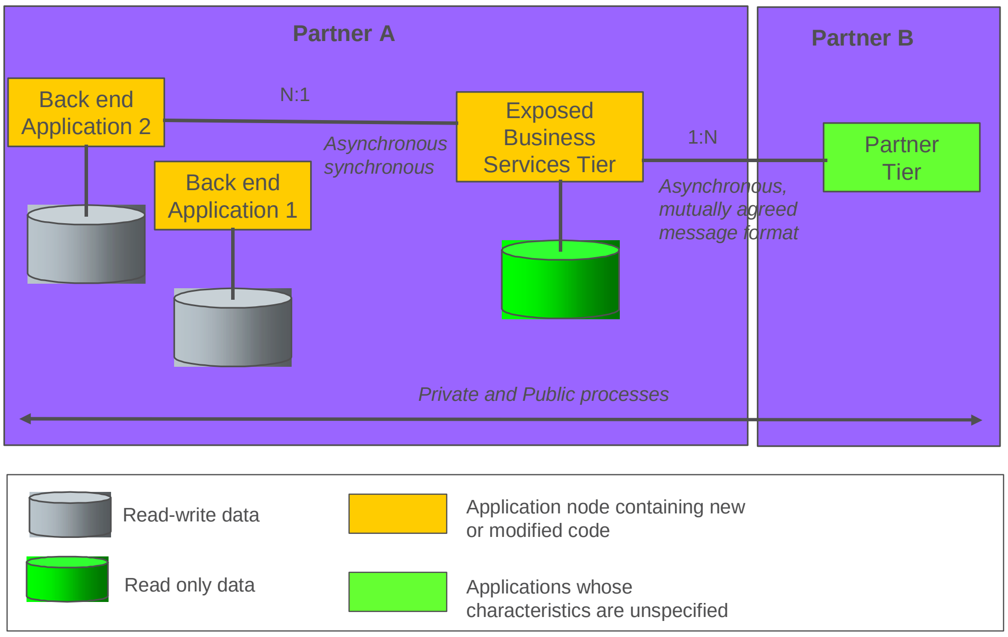
The common Message Oriented Middleware (MOM) enforced by dominant partners is a key enabler of this Application pattern today. The key features delivered by MOM are guaranteed delivery and once and once only delivery of messages. In future, open standards such as HTTP may play a more significant role in guaranteed delivery of messages between organizations.

Limitations

As mentioned earlier, for the most part this Application pattern implies the use of common middleware amongst all the participating partners. As a result you lose flexibility in connecting to business partners that might have different IT infrastructure capabilities. Because of this, this Application pattern can be inflexible.

This Application pattern implements a point-to-point interface between the partner systems and the exposed application tier. Hence it cannot be used for intelligent routing of requests, decomposition and re-composition of requests, and for invoking complex business process workflow as a result of a request that was received from a partner system. Under such circumstances one should consider a more advanced Application pattern.

3.3.3 Exposed Business Services


Partner A
Partner B
Back end Application 2
N:1
Exposed Business Services Tier
1:N
Partner Tier
Asynchronous synchronous
Back end Application 1
Asynchronous, mutually agreed message format
Private and Public processes
Read-write data
Read only data
Application node containing new or modified code
Applications whose characteristics are unspecified

The Exposed Business Services application pattern structures a system design that exposes specific services that can be directly invoked by partner systems across organizational boundaries.

Business and IT Drivers

- Improve the organizational efficiency
- Reduce the latency of business events
- Support a structured exchange with business partner
- Support partner real-time access to/ from applications
- Support partner real-time access to/ from business service
- Leverage existing skills
- Leverage legacy investment
- Backend application integration
- Minimize application complexity
- Minimize enterprise complexity
- Reduce partner dependency on specific application

The primary business driver for choosing this Application pattern is to enable business partner systems to gain direct access to specific business services. These services when invoked may in turn trigger multiple tasks on many backend applications. In other words, business requirements cannot be met by simple integration with a single backend application, as is the case with the Exposed Application application pattern.

A point-to-point interface between partner systems and many backend applications increases complexity and maintenance cost. The Exposed Business Services application pattern can be used to reduce the total cost of ownership by implementing a “hub and spoke” architecture instead of a point to point architecture between partner systems and backend applications.

Solution

As shown in the figure above, this Application pattern is designed using at least three logical tiers: Partner, Exposed Business Services and Backend Application.

- The Partner tier represents a set of partner applications that are interested in invoking specific services on the Exposed Business Services tier.
- The Exposed Business Services tier receives requests from multiple partner systems and intelligently routes them to the appropriate backend applications. It is also responsible for breaking down compound requests into several, simpler requests and routing them to multiple backend systems. Finally, it is responsible for message transformation and managing different levels of security. In doing so, it typically uses a local read-only database to store routing,

decomposition, re-composition, and transformation rules. These functions allow for one level of implementation of the private process rules employed in the later application patterns. The Exposed Business Services tier typically implements minimal business logic. The majority of the business logic is concentrated on the backend application tier.

- Please refer to the previous Application patterns for a description of the Backend Application tiers and the explanation for the Private and Public Processes description.

In the majority of cases this Application pattern is implemented using an asynchronous MOM prescribed by dominant players over a VPN. This Application pattern leverages many of the common services observed in Process-focused Application Integration patterns described in that Business pattern. One of the critical nodes that make up the Exposed Business Services tier **is a message broker node** that can perform intelligent routing, message transformation, decomposition and re-composition.

Guidelines for use

Mutually agreed upon messages should truly represent the electronic service that is being offered and should not be tied closely to the backend applications that are being invoked. This requires an even broader perspective in defining mutually agreed upon messages compared to the Exposed Application application pattern. Such generalization would further promote flexibility in changing or replacing backend applications without significantly impacting partner systems.

Benefits

This Application pattern can leverage investments already made in enterprise application integration to extend them beyond organizational boundaries.

The use of the Exposed Business Services tier isolates the internal business processes and backend implementation details from the partner systems and vice versa. This loosely coupled architecture makes it easy to change, replace, or add backend applications without heavily affecting partner systems. This increases maintainability and reduces the total cost of ownership.

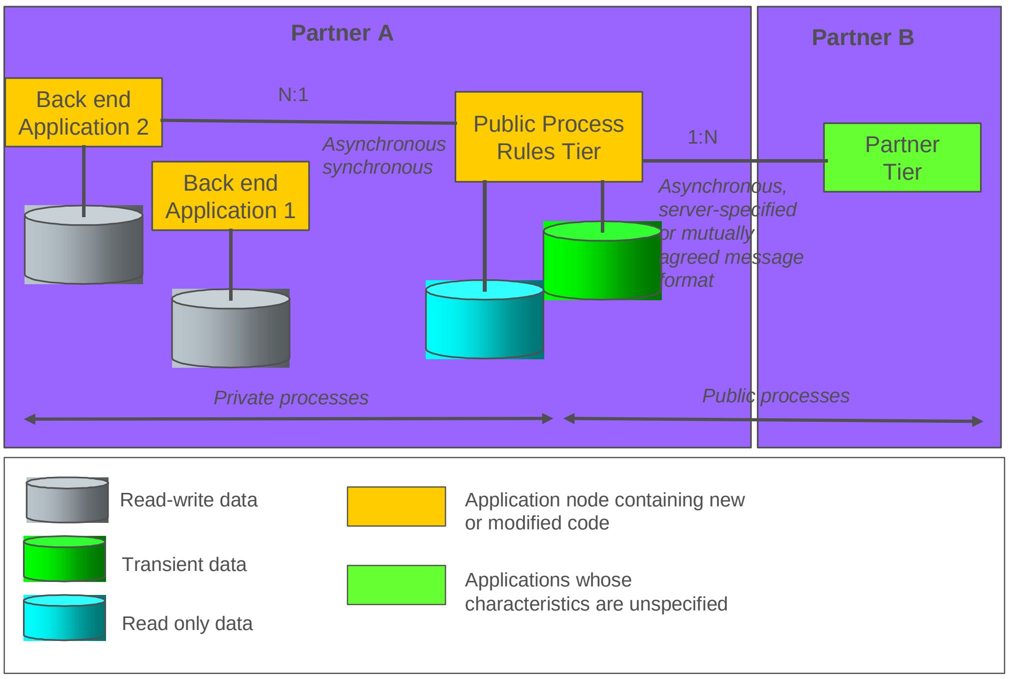
Limitations

This Application pattern currently implies the use of common Message Oriented Middleware (MOM) amongst all the participating partners. As a result you lose flexibility in connecting to business partners that might have different IT infrastructure capabilities. Because of this, this Application pattern can be somewhat rigid.

Corporations typically have special agreements with different business partners. We call such special agreements Business Protocols. When enterprise integration approaches are extended outside the company, as is the case with this Application pattern, one must allow for handling different business protocols with different business partners. This may require additional hand-crafted protocol management code per business partner to handle special agreements pertaining to that partner. Under such circumstances one should consider more advanced Application patterns such as the Managed Public Processes application pattern.

Much work has been done recently to make it easier to develop such service interface using the public Internet ant web infrastructure. Collectively, this technology is referenced as Web Services.

3.3.4 Managed public processes


Partner A
Partner B
Back end Application 2
N:1
Public Process Rules Tier
1:N
Partner Tier
Asynchronous synchronous
Back end Application 1
Asynchronous, server-specified or mutually agreed message format
Private processes
Public processes
Read-write data
Application node containing new or modified code
Transient data
Applications whose characteristics are unspecified
Read only data

The Managed Public Processes application pattern structures a system design that handles the management of shared business processes between business partners.

Business and IT Drivers

- Improve the organizational efficiency
- Reduce the latency of business events
- Support a structured exchange with business partner
- Support partner real-time access to/ from applications
- Support partner real-time access to/ from business service
- Leverage existing skills
- Leverage legacy investment
- Backend application integration
- Minimize application complexity
- Minimize enterprise complexity
- Avoid partner mandated infrastructure
- Reduce partner dependency on specific application
- Reduce partner dependency on specific business protocols

It is typical for corporations to enter into special agreements with different business partners. In business to business electronic commerce, there is a need to agree not only on the traditional terms and conditions but also on IT issues such as:

- Roles assumed by different parties
- Valid sequences of requests
- Electronic message formats
- Communication protocols to be used
- Security issues such as authentication, encryption, and non-repudiation
- Service level agreements such as response time and availability
- Error Handling Procedures

We refer to this combination of traditional and IT agreements between a set of partners as Business Protocols. The specific details of these business protocols vary from one set of partners to another. These details can also vary over a period of time between the same set of partners. The primary business driver for choosing this Application pattern is to support different business protocols with different business partners.

All the Application patterns considered so far require a common set of middleware and IT infrastructure amongst all the participants. For example, the Document Exchange application pattern enforces the use of a particular VAN amongst all the partners. Similarly dominant partners enforce the use of a common message oriented middleware in most of the implementations of Exposed Application and Exposed Business Services application patterns. Such requirements have been blamed by some for the slow acceptance of inter-business integration prior to the emergence of the Internet. Along with the Internet, several standards-based communication protocols such as HTTP and SMTP have emerged. By using such standards-based protocols companies can achieve complete independence from their business partners in implementing their backend applications and choosing their IT infrastructure.

In addition, all the business and IT drivers listed under Exposed Business Services application pattern apply here as well.

Solution

As shown in the figure above, this Application pattern is divided into at least three logical tiers: Partner, Public Process Rules and Backend Application.

- The Public Process Rules tier receives requests from multiple business partners. It is responsible for ensuring that the agreed upon business protocols with all these partners are satisfied. It maps the external communication protocols into internal communication protocols before forwarding the request to the Backend Application tier. For example an agreement with a particular business partner may require receiving requests using secure HTTPS protocol. Such a request may need to be converted into message before passing it over to the Backend Application tier. In addition, it is responsible for implementing the necessary security features such as authentication, authorization, encryption, message integrity, and non-repudiation. These security checks are much more important in this Application pattern compared to the earlier ones, because this Application pattern allows for receiving requests over the public Internet. Finally, this tier is also responsible for sending the required response back. In doing, so it typically relies on the response received from the Backend Application tier. A single Public Process Rules tier can be used to support multiple partners each employing different business protocols.
- Please refer to the previous Application patterns for the description of the Partner and Backend Application tiers.

The Public Process Rules tier is best implemented by using middleware products that can generate business protocol management code by interpreting electronically defined contracts.

Guidelines for use

Simple implementations involving a few business protocols with a limited number of business partners can be implemented using hand-crafted protocol management code. Typically such applications are written to run on a standard web application server. The emerging protocol in this arena to consider is SOAP.

For more extensive implementations, one should consider capturing the details of business protocols such as role, valid sequence of messages, message format, security issues, and so on using an electronic contract. These electronic contracts are called Trading Partner Agreements (TPA). They can be expressed using XML based languages such as tpaML or ebXML/CPA. A TPA describes all the valid visible and hence enforceable interactions between the parties and is independent of the internal business processes and backend applications of each party. Each partner builds its own internal business process to satisfy these external TPAs and interface them to the rest of its business processes and backend applications. The intent is to provide a high-level specification that can be easily understood by humans and yet is precise enough for enforcement by computers. TPAs can be compiled into business protocol management code that implements interaction rules in each party’s system. It should be understood that the information in the TPA is not a complete description of the application but only a description of the interactions between the parties. The application must be designed and programmed in the usual manner. Use of such an electronic TPA increases the flexibility, speeds the time to market and avoids error prone hand coding since changes can be made to the electronic contract and protocol management code can be generated from it.

To increase the ease of interoperability between multiple business partners, it is recommended that standardized business-to-business protocols such as ebXML be used.

Benefits

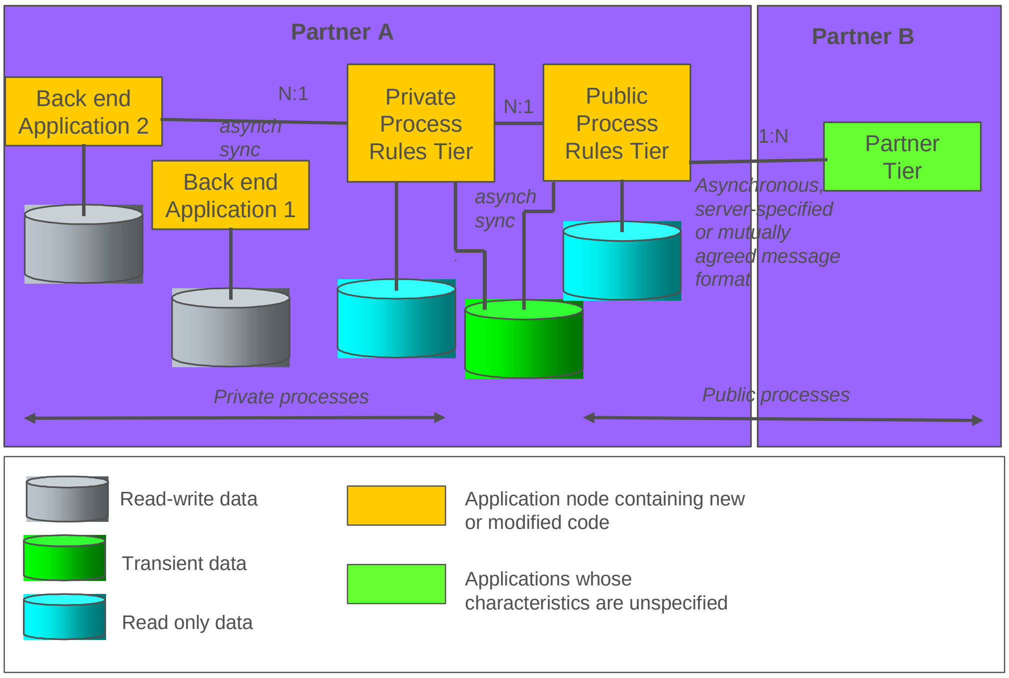
The Public Process Rules tier completely isolates the internal systems and business processes from the external parties. Hence it ensures that each party maintains complete independence from the other party both as to the IT infrastructure and the nature of the business processes resulting in a highly flexible architecture.

Limitations

This Application pattern can exploit universally accepted communication protocols such as HTTP and SMTP for integration between partners. However unlike MOM technologies these protocols do not guarantee reliable message delivery. Because of this, MOM techniques are still used in industries and business scenarios where reliability is of the utmost importance. It is important to note that one can expect improvements in the reliability provided by these open technologies over the next few years.

Even though this Application pattern can handle special agreements with different business partners and can ensure each party can maintain complete independence in terms of IT infrastructure, backend applications, and business process, it cannot map long running external transactions to internal business processes and workflow. Under such circumstances one should consider more advanced Application patterns such as the Managed Public and Private Processes application pattern.

3.3.5 Managed Public and Private Processes


Partner A
Partner B
Back end Application 2
N:1
asynch
sync
Private Process Rules Tier
N:1
Public Process Rules Tier
1:N
Partner Tier
Back end Application 1
asynch
sync
Asynchronous, server-specified or mutually agreed message format
Private processes
Public processes
Read-write data
Transient data
Read only data
Application node containing new or modified code
Applications whose characteristics are unspecified

The Managed Public and Private Processes application pattern structures a system design that handles different business protocols with different business partners and maps long running external transactions to internal business processes and workflow.

Business and IT Drivers

- Improve the organizational efficiency
- Reduce the latency of business events
- Support a structured exchange with business partner
- Support partner real-time access to/ from applications
- Support partner real-time access to/ from business service
- support for shared business process flows with multiple partners
- Integrate internal workflow manager with partner shared business process flows
- Leverage existing skills
- Leverage legacy investment

- Backend application integration
- Minimize application complexity
- Minimize enterprise complexity
- Avoid partner mandated infrastructure
- Reduce partner dependency on specific application
- Reduce partner dependency on specific business protocols

All the business and IT drivers listed under the Managed Public Processes application pattern apply here as well.

Furthermore, this Application pattern accommodates long running transactions across organizational boundaries. Dan and Parr observed that transactions spanning multiple independent organizations have different characteristics compared to traditional ACID transactions executed inside a single organizational boundary. They are typically composed of many related interactions dispersed in time resulting in long running conversations. Such long running conversations are particularly important since very little can be assumed about the target execution environment, response time, network availability, and the need for human intervention to complete the request. This results in the need to map such long running external transactions to internal business processes and workflows.

For example, agreed-upon inter-business workflows (Public Process Rules) can be defined to govern the long running transactions comprising entire business process cycles (such as submit RFQ - receive price quotation - issue purchase order - manage fulfillment). Private Process Rules can then capture lower level workflows including all human and machine interactions needed to complete each major step in the business process ("approve purchase order" for example). This "local" workflow will typically employ application integration techniques including message brokering and application adapters to execute its function.

Solution

As shown in the figure above, this Application pattern is built using at least four logical tiers: Partner, Public Process Rules, Private Process Rules and Backend Application tier.

- The Public Process Rules tier receives requests from multiple business partners. It is responsible for ensuring that the agreed upon business protocols with all these partners are satisfied. In doing so, it implements all the features described under the Managed Public Processes application pattern with one exception. Instead of passing external requests to the Backend Application tier directly, it passes them to the Private Process Rules tier.
- The Private Process Rules tier is responsible for mapping external long running transactions to internal business processes and workflows. To manage workflows efficiently, it combines the activities of a process, all the people in the organization, and the infrastructure such as computers and programs. In other words, it maintains the status of the long running transaction and moves work from one resource to the other based on the internal business process. Resources in this context refer to people in the organization and the business applications.
- Please refer to the previous Application patterns for the description of the Partner and Backend Application tiers.

The Private Process Rules tier leverages most of the Process-focused Application Integration services detailed in that Business pattern. It is often a manifestation of the Application Integration::Workflow application pattern.

Guidelines for use

The success of the Managed Public and Private Processes application pattern heavily relies on the ability to define generic inter-enterprise business processes and workflows. Such inter-enterprise workflow varies between different sets of business partners. Over time these business processes change between the same set of business partners. To accommodate this dynamic environment one should separate the flow dependency from the application. Such a separation allows for changing the workflow without impacting the applications. It is recommended that a message based workflow management system be used to define and implement these inter-enterprise workflows.

A holistic approach must be taken in defining inter-enterprise business processes and how they map to the internal workflow. It begins with a description of the roles and responsibilities, the task flows and processes, and the data definitions and documents underlying the joint business processes. The approach then employs a set of tools to transform these descriptions into a consistent and coherent set of message definitions, workflow definitions, directory entries, database schemas and so on, which can be deployed onto a runtime architecture representing a software implementation of these processes. This "ultimate" approach to business-to-business interactions can be modified in the future with increasing levels of depth and technical detail.

The Private Process Rules tier should be able to maintain a correlation ID, history, and status of a long running transaction so that an external party can effectively query the status of a transaction and also request one or a group of these interactions to be cancelled.

Please refer to the previous Application pattern for guidelines on expressing business protocols electronically using TPAs.

Benefits

This Application pattern, compared to the earlier ones, further enhances the complete independence gained with respect to the business processes. Internal business processes and workflows can be changed without impacting external business processes - hence not impacting external parties.

Using the principles of message-based workflow management, it separates the workflow information from applications. This results in a highly flexible architecture where business process flow can be changed without impacting applications. In addition, this enables the reuse of applications as software components in other processes. Furthermore, it allows applications to be replaced or changed without impacting the overall workflow. The combination of flexibility and reuse results in significant cost savings.

In addition, it provides all the benefits provided by the Managed Public Processes application pattern.

Limitations

Implementing a truly generic and complete business to business integration capability is a significant and complex undertaking and requires a considerable amount of effort, time, and resources.

Many legacy applications have built in workflow. Separating flow information from such backend systems is both complex and time consuming. In some cases it might not even be possible.

4 Topic Map

4.1 Topic Maps Generator

The Topic Maps Generator tool is developed to be used by the content providers for easy authoring of topic maps and direct storing to the REGNET knowledge Base. The tool is designed from scratch, according to content providers' comments and it is fully compliant with the XTM 1.0 specifications.

The implementation of the TMG is based on TM4J and Java, and the interconnection with the Knowledge Base is based on SOAP classes that are also implemented by CERT (these classes are also being used by every REGNET module which needs access to the Knowledge Base).

Taking into account the comments and proposals from the functional and demonstration tests performed by content providers, a number of improvements are made to the tool, namely:

- Addition of a data transformation utility: XALAN embodiment for real time transformation of the produced XTM to the "TGLink" internal format.
- On-line visualization of the edited XTM via integration of the "TGLink" graphical viewer.
- Real time visualization of topic maps.
- Support of "Instance Of" for occurrences (another block added in the interface).
- Parameterization of the tool for easy installation in any Jakarta server (this improvement concerns the technical partners).
- Replacing the "ResourceData" element with the "ResourceRef" element.

Apart from the major changes, there are also some minor ones that conduce to the final result. These are:

- Alphabetical order presentation of topics and associations.
- Removal of internal numbers after XTM importing.
- List of opened XTM files at any time.
- Extra links to XTM-specific sites, including an on-line tutorial for the use of the tool.

5 Portal

5.1 REGNET Portal Interoperability

In order to demonstrate the full potentiality of the Portal system, it was added the capability to manage web services not only in the context of Ontology interaction as in the previous releases, but to exploit services as an active REGNET component and not just an entry point. This will bring into the infrastructure the benefit of developing user applications with a flexible presentation layer already adopted for the Portal functions. The technology adopted to realise these services was the Web Services.

Web services are a new breed of web application. They are self-contained, self-describing, modular application that can be published, located, and invoked across the web. The goal was to demonstrate a sort of portals interaction in order to develop distributed services.

The approach was to store local information in each portal. When a user requires particular information at the portal he is connected (hereafter called the gateway), the gateway interoperates with others portals in order to retrieve and to integrate different pieces of distributed information. Actually, the gateway fits a generic concept of a broker specialized in searching and organizing information distributed throughout the REGNET infrastructure. Nevertheless, the fact that these information are housed in several different locations is transparent for the user.

Collaboration among portals is performed in term of Simple Object Access Protocol (SOAP) messaging. SOAP is a wire protocol similar to the IIOP for CORBA, ORPC for DCOM, or JRMP for RMI. While IIOP, ORPC, and JRMP are binary protocols, SOAP is text-based and uses XML for data encoding. SOAP is based on a vendor-agnostic technology, namely XML, HTTP, SMTP, in this way it appeals to all vendor.

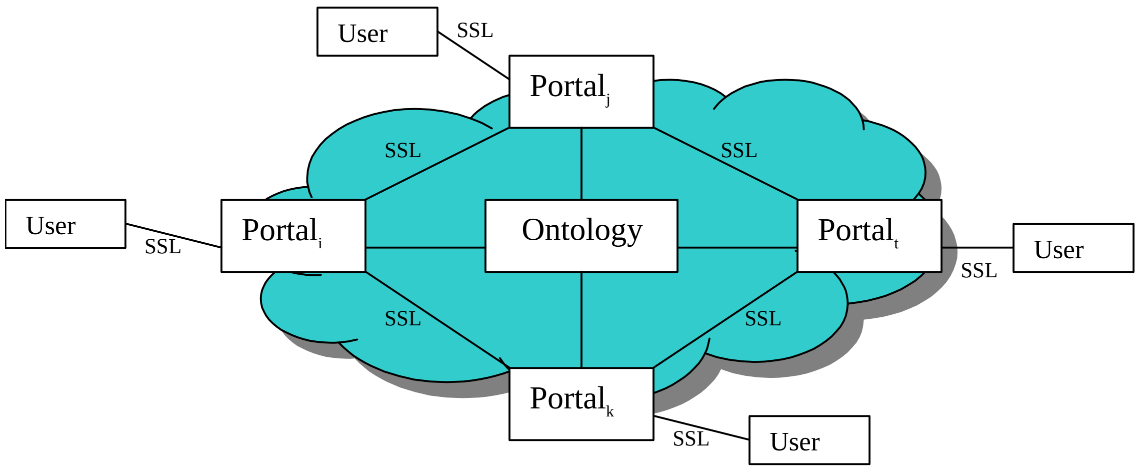
Each portal implements one or more web services, and the descriptions associated with these services are gathered into the Ontology. The Ontology is queried initially by the gateway to obtain a list of portals whose services match the given user query.

Obviously, more portals can implement the same web service, so increasing the availability of information for the final result. For example, in our demonstration we have realized a travel plane information service implemented in different portals, where each portal contains information regards a specific airplane company.

The Web Services Description Language (WSDL) provides an XML–based grammar for describing a web service interface. Portals wishing to provide services will publish on the Ontology their WSDLs, and portals seeking services will retrieve the WSDLs out.

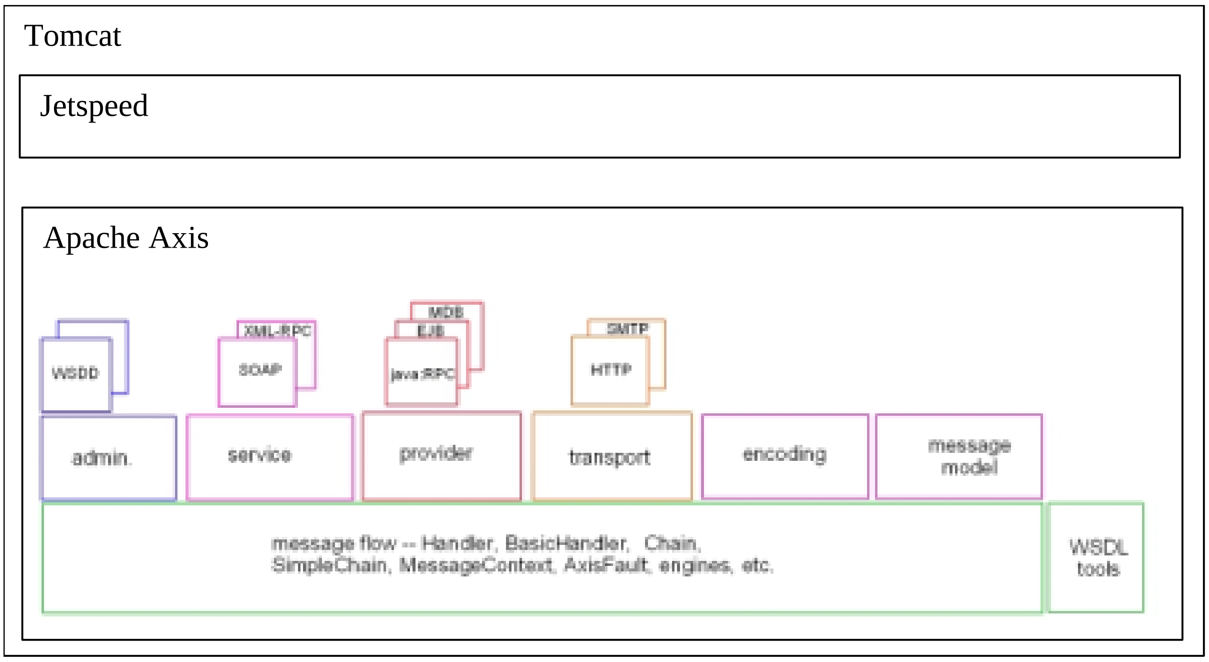
To maintain the confidentiality and integrity of data exchanged among portals, Secure Socket Layer (SSL) it is used. Portals Certificates guarantee non-repudiation and authentication. This would guarantee the support for future e-Business applications.

The figure below shows the REGNET network, in the context of the discussion above:



In our demonstration we have utilized Apache Axis provided by Apache as SOAP engine and Apache Jetspeed for building web services contents into a coherent front-end application for users.

The figure below shows the architecture utilized:



5.1.1 Apache Axis

Apache Axis is an implementation of the SOAP specification developed by the Apache Software Foundation. Axis is based on the Java API for XML Messaging and Java API for XML RPC specifications currently being drawn up by Sun. It has complete support for the WSDL 1.1 standard, for the SOAP 1.1 specification, and even offers partial compatibility with SOAP 1.2.

Features:

It is designed to be extensible and independent with regard to particular transport or exchange protocols. It is therefore to add on support for other transport protocols such SMTP, FTP and so on.

It is able to automatically generate WSDL descriptions mapping functionality of the JAX-RPC API.

It supports data types from the XML-Schema (2001) standard.

It supports user session and access authentication and authorization (via SOAP headers, independent from the transport protocol used).

It is able to process SOAP messages with untyped parameters. This brings greater interoperability with other SOAP implementations such Microsoft's MS-SOAP, which use untyped parameters in SOAP message exchange.

5.1.2 Service Description

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services).

A service type consists of the service name, the names of the operations (methods) it offers, as well as the names and types of those methods parameters and return types.

In addition to a service type, to invoke a web service, it is necessary to know technical details of locating and connecting to that service. The service provider must advertise its service access points, or endpoints, and the protocols supported by those access points.

The W3C recommends an XML meta-language, the WSDL, for describing a service's type and access information.

In the REGNET context, the basic idea is to store WSDL documents on the Ontology. Portals searching for a specific service can retrieve the WSDL document from the Ontology, convert the service-specific types to language-specific types, and then use the service's endpoint and protocol information to contact the service.

5.1.3 Using SSL with Axis

Secure Socket Layer (SSL) is a technology that allows two REGNET portals to communicate over a secured connection. In this secure connection, the data that is being sent is encrypted before being sent, then decrypted upon receipt and prior to processing. Both the portals encrypt all traffic before sending any data.

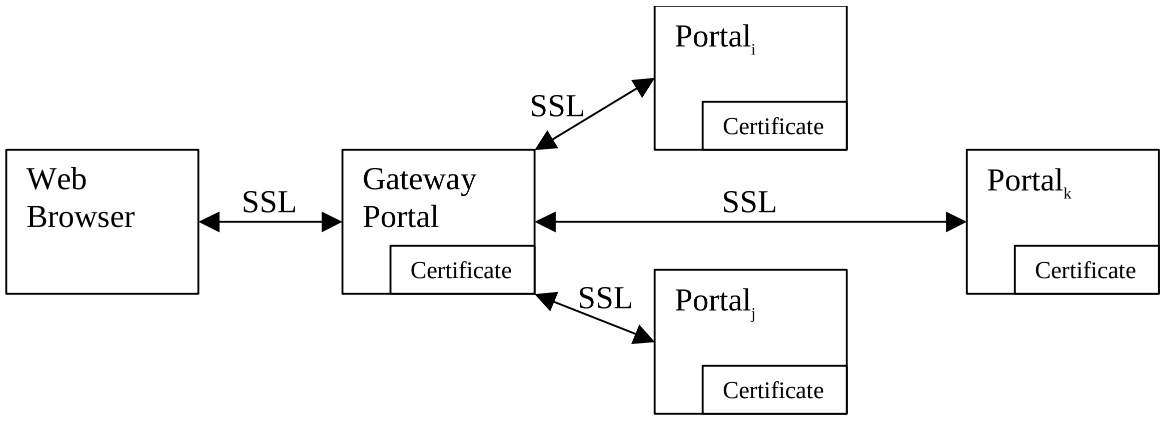
In order to establish a secure communication channel, the portals have to authenticate each other. The authentication requires that each portal have a digital certificate.

A user can choose to access at the gateway portal using HTPP or HTTPS protocols.

If he/she chooses to access with HTTPS, the gateway portal returns its digital certificate that must be accepted by the user before proceeding to the connection. In our demonstration prototype we have created a digital certificate for each portal not issued by a certificate authority (CA). When the user accepts the gateway certificate a secure communication channel is established between the web Browser and the gateway portal.

The connections between the portal gateway and other REGNET portals are secured although the user has selected HTTP protocol to connect to the gateway portal.

The figure below shows the scenario described.



5.1.4 Search Flight Prototype

The scenario implemented in our demonstration is a travel plane search that finds the best available prices for all planes departing and arriving in a specific date. Each portal, that implements the web service, manages a set of airplane companies. When a user makes a request to its local portal, this portal (the gateway) retrieves out of the Ontology the WSDL document describing the service, contacts each endpoint that implements the service and starts a timer. The WSDLs are stored in the root path on the Ontology.

The gateway portal manages multiple requests concurrently in a multi-user environment.

To contact each endpoint SOAP messages are used over SSL. The portal and the endpoints authenticate themselves using digital certificates. When the timer expires or all endpoints have answered, the portal collects the results in a single XML document and applies a specific XSLT to render the result to a web Browser or a WAP Browser.

6 REGNET B2B connector

6.1 Introduction

This part provides a high level and a detailed description of the conception of the REGNET ebXML Borrow/Lend application.

The REGNET ebXML Borrow/Lend application is design to allow a museum to borrow an item to another museum for an exhibition purpose for instance.

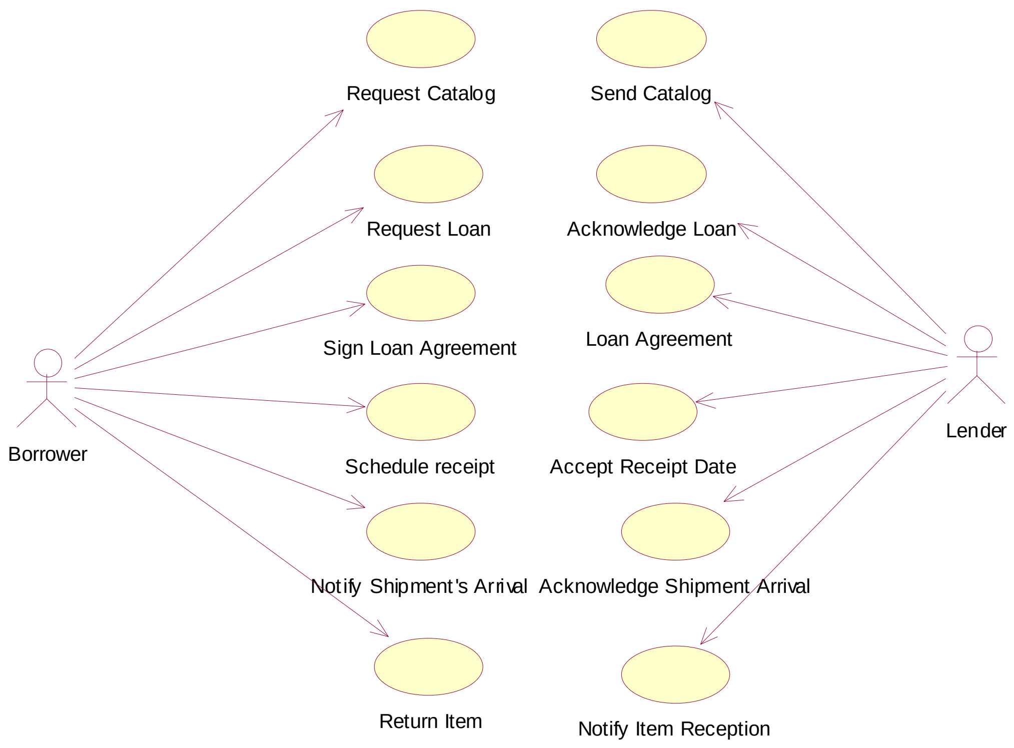
This part defines and explains the design of the REGNET ebXML Borrow/Lend application and is divided in the following paragraphs:

Section 2: Functional requirements. This section describes the different stakeholders belonging to the system, then it lists the available functionality of the application and finally lists and gives a description of the business objects.

Section 3: Detailed design. This section describes the content of the REGNET broker packages and for each one gives a detail description of the class that they contain.

6.2 Functional requirements

6.2.1 Use Case Overview



6.2.1.1 Actor : Borrower

The borrower actor initiates the borrowing business process. He means to borrow an item to its partner for an exhibition for instance.

6.2.1.2 Actor : Lender

The lender actor responds to the borrower in the business process.

6.2.1.3 Use Case : Request Catalog

The Borrower request an Items Catalog to the Lender.

6.2.1.4 Use Case : Request Loan

The Borrower request the loan of an item in the catalog.

6.2.1.5 Use Case : Sign Loan Agreement

The Borrower sign the loan agreement, send it via snail mail and inform the Lender he has done so.

6.2.1.6 Use Case : Schedule receipt

The Borrower chooses a date to receive the item

6.2.1.7 Use Case : Notify Shipment's Arrival

The borrower notify the lender he has received the borrowed item.

6.2.1.8 Use Case : Return Item

The borrower informs the Lender he has sent the item back

6.2.1.9 Use Case : Send Catalog

The lender sends its catalog of items.

6.2.1.10 Use Case : Acknowledge Loan

The lender acknowledge the loan requested by the borrower for a specified item of its catalog

6.2.1.11 Use Case : Loan Agreement

The lender informs the borrower that he has received and signed the loan agreement.

6.2.1.12 Use Case : Accept Receipt Date

The lender acknowledge the receipt date chosen by the borrower

6.2.1.13 Use Case : Acknowledge Shipment Arrival

The lender inform the borrower, he has taken notice that the item has been received.

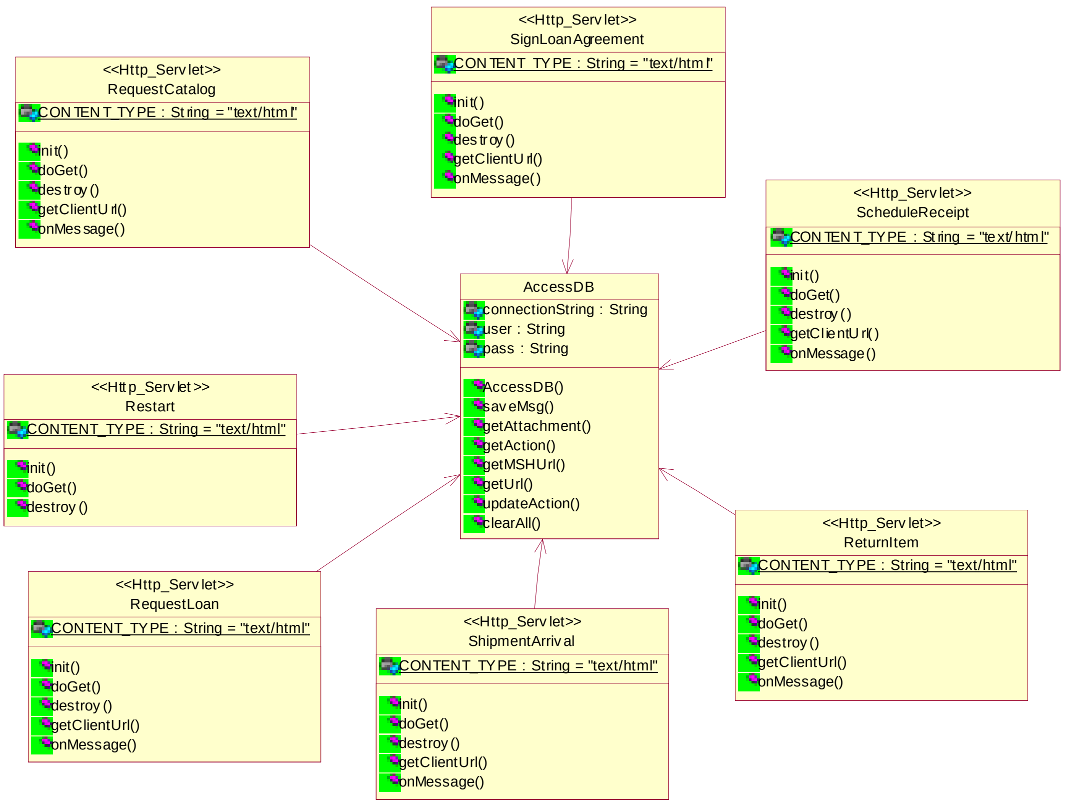
6.2.1.14 Use Case : Notify Item Reception

The lender informs the borrower that he has received the returned item.

6.2.2 Business model

6.2.2.1 Package Borrower : classes overview

Classes Diagram



Class AccessDB Description

This Class is used to access the Database, Insert or Modify data related to the application

Class RequestCatalog Description

This servlet is invoked in order to send an ebXML message to the parter and ask for an Items Catalog.

Class RequestLoan Description

This servlet is invoked when the borrower request a loan to the lender.

Class Restart Description

This servlet is invoked to restart the transaction. It removes all ebxml messages from the Database.

Class ReturnItem Description

This servlet is invoked to send en ebxml message to inform the parter, the borrowed Item has been shipped back.

Class ScheduleReceipt Description

This servlet is invoked to send en ebxml message to tell the parter about the date chosen to receive the borrowed item.

Class ShipmentArrival Description

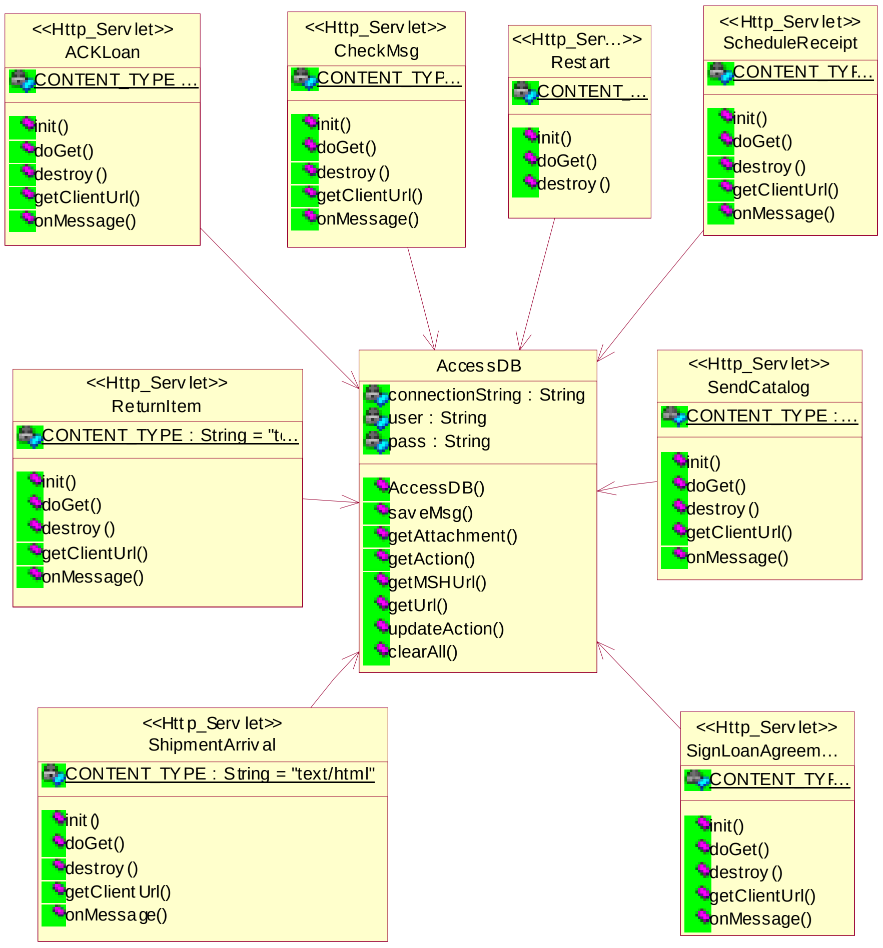
This servlet is invoked to send en ebxml message to inform the partner that the borrowed item has been received.

Class SignLoanAgreement Description

This servlet is invoked to send en ebxml message to inform the partner that the loan agreement as been received and sent back via "snail" (regular) mail.

6.2.2.2 Package Lender : classes overview

Classes Diagram



Class ACKLoan Description

This servlet is invoked to notify the partner that the loan has been accepted and that a loan agreement has been sent.

Class CheckMsg Description

This servlet is invoked by the application in order to check for incoming ebXML messages

Class Restart Description

This servlet is invoked to restart the transaction. It removes all ebxml messages from the Database.

Class ReturnItem Description

This servlet is invoked to informed the partner that the Item has successful been received and the transaction is over

Class ScheduleReceipt Description

This servlet is invoked to inform the partner if the Receipt Date has been acknowledge.

Class SendCatalog Description

This servlet is invoked to send a catalog (under a XML form) to the partner.

Class ShipmentArrival Description

This servlet is invoked to notify the partner that the Lender took notice that the Item has been shipped to the Borrower

Class SignLoanAgreement Description

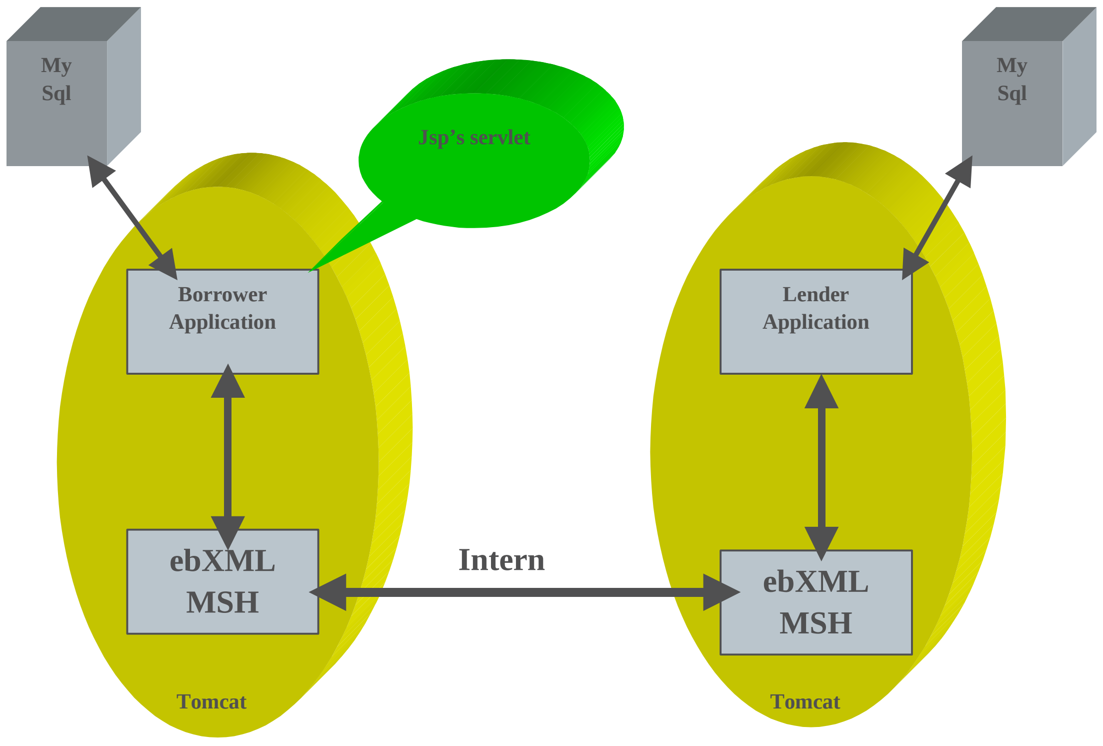
This servlet is invoked to inform the Borrower the Loan agreement has been received and signed.

Class AccessDB Description

This Class is used to access the Database, Insert or Modify data related to the application

6.3 Detailed design

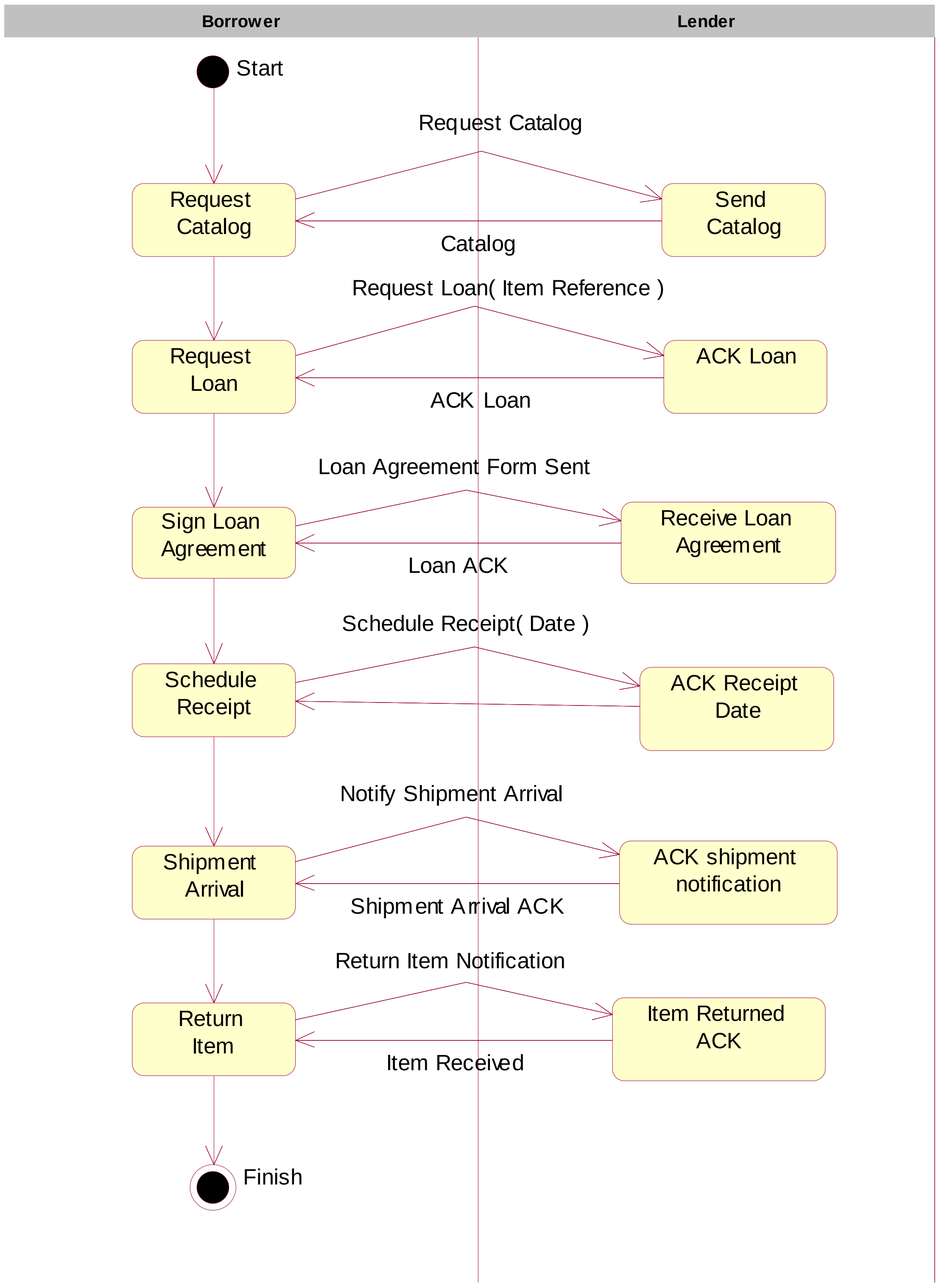
6.3.1 System Architecture



The application uses two tomcat 4 on which runs each application part: Borrower and Lender and one ebXML message service handler on each Tomcat.

A mysql database is used in order to store data related to the application and for the message service handler to work.

6.3.2 Business Process used by the application



6.3.2.1 Activivity : Start

The transaction starts.

6.3.2.2 Activivity : Request Catalog

The borrower requests a catalog

6.3.2.3 Activivity : Request Loan

The borrower request a loan for an item of the catalog

6.3.2.4 Activivity : Sign Loan Agreement

The Borrower signs the loan agreement

6.3.2.5 Activivity : Send Catalog

The lender send its catalog.

6.3.2.6 Activivity : ACK Loan

The lender acknowledges the loan

6.3.2.7 Activivity : Receive Loan Agreement

The Lender receives and sign the loan agreement

6.3.2.8 Activivity : Schedule Receipt

The borrower choose a receipt date

6.3.2.9 Activivity : ACK Receipt Date

The Lender acknowledge the receipt date

6.3.2.10 Activivity : Shipment Arrival

The borrower notify the lender he has received the shipment

6.3.2.11 Activivity : ACK shipment notification

The lender acknowledge the receipt information

6.3.2.12 Activivity : Return Item

The borrower informs the lender he has returned the item back

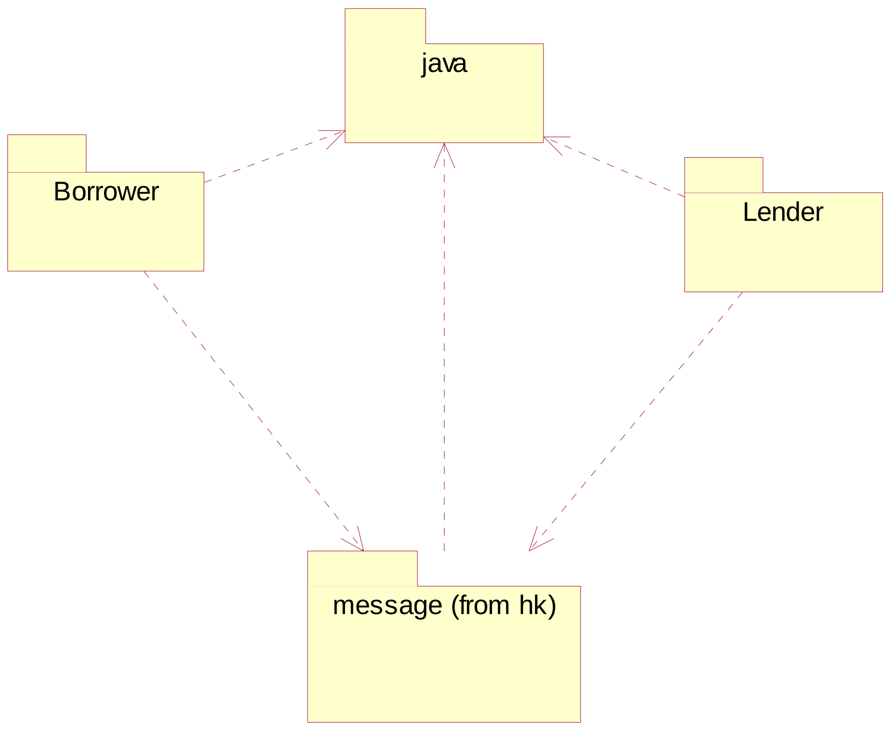
6.3.2.13 Activivity : Item Returned ACK

The lender inform the borrower he has received the item

6.3.2.14 Activivity : Finish

The Transaction is over

6.3.3 Package Overview



6.3.4 Package : Borrower

6.3.4.1 Class : AccessDB

Attributes	
connectionString	String used to connect to the database
user	Login used to access the database
pass	Password used to access the database

Operations	
AccessDB	Class Constructor. This methods gets the information contained in the Borrower properties file in order to set up the connectionString, user and pass variables
saveMsg	Save an ebXML message in the database,
getAttachment	Get the attachement of the message for a specific action
getAction	Get the Action to be performed.
getMSHUrl	get Url of the Msh of the trading partner
getUrl	get Url of the Borrower Application. Essentialy for portable purpose.
updateAction	Uptade the action field in the database.
clearAll	Clear all ebxml messages.

6.3.4.2 Class : RequestCatalog

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources
getClientUrl	This function returns null, because we don't want the Message Service Handler to use any Url to send the message back.
onMessage	This function is called when an ebXML message is received by the ebXML Message Service Handler.

6.3.4.3 Class : RequestLoan

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources
getClientUrl	This function returns null, because we don't want the Message Service Handler to use any Url to send the message back.
onMessage	This function is called when an ebXML message is received by the ebXML Message Service Handler

6.3.4.4 Class : Restart

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources

6.3.4.5 Class : ReturnItem

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources
getClientUrl	This function returns null, because we don't want the Message Service Handler to use any Url to send the message back.
onMessage	This function is called when an ebXML message is received by the ebXML Message Service Handler

6.3.4.6 Class : ScheduleReceipt

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources
getClientUrl	This function returns null, because we don't want the Message Service Handler to use any Url to send the message back.
onMessage	This function is called when an ebXML message is received by the ebXML Message Service Handler

6.3.4.7 Class : ShipmentArrival

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources
getClientUrl	This function returns null, because we don't want the Message Service Handler to use any Url to send the message back.
onMessage	This function is called when an ebXML message is received by the ebXML Message Service Handler

6.3.4.8 Class : SignLoanAgreement

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources
getClientUrl	This function returns null, because we don't want the Message Service Handler to use any Url to send the message back.
onMessage	This function is called when an ebXML message is received by the ebXML Message Service Handler

6.3.5 Package : Lender

6.3.5.1 Class : ACKLoan

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources
getClientUrl	This function returns null, because we don't want the Message Service Handler to use any Url to send the message back.
onMessage	This function is called when an ebXML message is received by the ebXML Message Service Handler.

6.3.5.2 Class : CheckMsg

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources
getClientUrl	This function returns null, because we don't want the Message Service Handler to use any Url to send the message back.
onMessage	This function is called when an ebXML

	message is received by the ebXML Message Service Handler.

6.3.5.3 Class : Restart

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources

6.3.5.4 Class : ReturnItem

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources
getClientUrl	This function returns null, because we don't want the Message Service Handler to use any Url to send the message back.
onMessage	This function is called when an ebXML message is received by the ebXML Message Service Handler.

6.3.5.5 Class : ScheduleReceipt

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources
getClientUrl	This function returns null, because we don't want the Message Service Handler to use any Url to send the message back.
onMessage	This function is called when an ebXML message is received by the ebXML Message Service Handler.

6.3.5.6 Class : SendCatalog

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources
getClientUrl	This function returns null, because we don't want the Message Service Handler to use any Url to send the message back.
onMessage	This function is called when an ebXML message is received by the ebXML Message Service Handler.

6.3.5.7 Class : ShipmentArrival

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources
getClientUrl	This function returns null, because we don't want the Message Service Handler to use any Url to send the message back.
onMessage	This function is called when an ebXML message is received by the ebXML Message Service Handler.

6.3.5.8 Class : SignLoanAgreement

Attributes	
CONTENT_TYPE	Type of data treated by the Servlet

Operations	
init	Initialize the global variables
doGet	Process the HTTP Get Request
destroy	Clean the ressources
getClientUrl	This function returns null, because we don't want the Message Service Handler to use any Url to send the message back.
onMessage	This function is called when an ebXML message is received by the ebXML Message Service Handler.

6.3.5.9 Class : AccessDB

Attributes	
connectionString	String used to connect to the database
user	Login used to access the database
pass	Password used to access the database

Operations	
AccessDB	Class Constructor. This methods gets the information contained in the Lender properties file in order to set up the connectionString, user and pass variables
saveMsg	This Class is used to access the Database, Insert or Modify data related to the application
getAttachment	Get the attachement of the message for a specific action
getAction	Get the Action to be performed.
getMSHUrl	get Url of the Msh of the trading partner
getUrl	get Url of the Lender Application. Essentialy for portable purpose.
updateAction	Uptade the action field in the database.
clearAll	Clear all ebxml messages.

7 REGNET Broker

7.1 Introduction

This part provides a high level and a detailed description of the design of the REGNET broker system.

The aim of the REGNET broker system is to provide data exchanges between the different datasources that appear in the REGNET project.

This part defines and explains the design of the REGNET broker system following the steps below :

Section 2: Functional requirements. This section describes the different stakeholders belonging to the system, then it lists the available functionalities of the broker and finally lists and gives a description of the business objects.

Section 3: High level design. This section presents the high level architectural organization of the REGNET broker system an the different components involved in it. It also gives a global overview of the packages architecture and describes the software component interactions.

Section 4: Detailed design. This section describes the content of the REGNET broker packages and for each one gives a detail description of the classes they contain.

7.2 Functional requirements

7.2.1 Actors description

This paragraph describes the different stakeholders who interact with the REGNET broker system.

7.2.1.1 Actors hierarchy

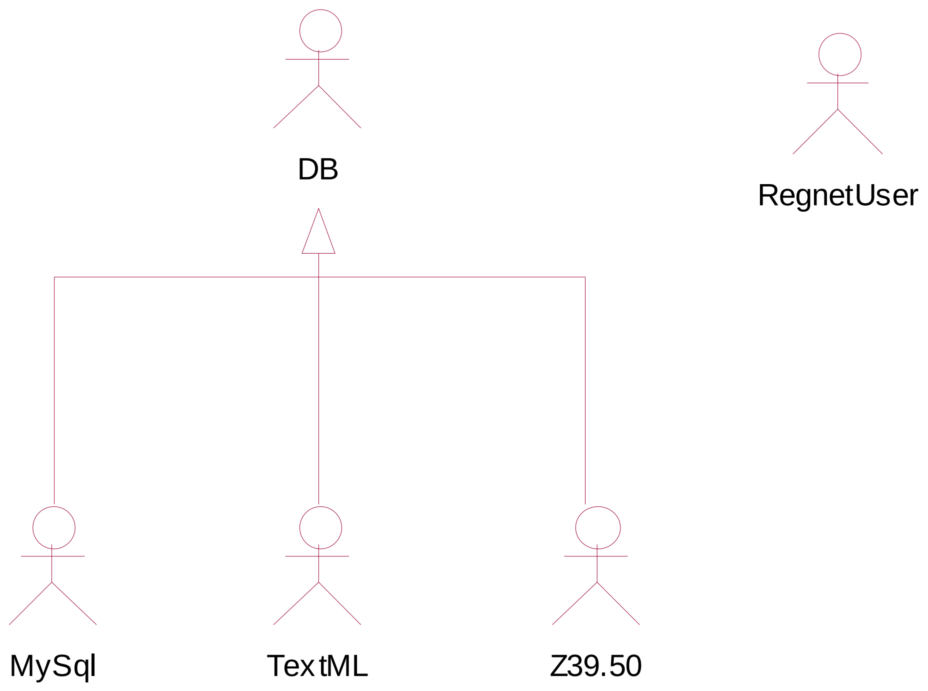


Figure 1 Actors hierarchy

7.2.1.2 Actor : DB

This actor stands for any database containing items that could be exchanged from a datasource to another.

7.2.1.3 Actor : MySql

Relational database belonging to the business applications (e-shop, auction, pcm, procurement, e-business). This database contains the items that belong to the PCM datasource.

7.2.1.4 Actor : REGNETUser

Any user who have permissions to connect to the REGNET portal.

7.2.1.5 Actor : TextML

Xml database of the data entry system.

This data contains xml description for items that are able to be exchanged.

7.2.1.6 Actor : Z39.50

Online database that provides records access via the Z39.50 communication standard.

7.2.2 Use case model

7.2.2.1 Use case overview

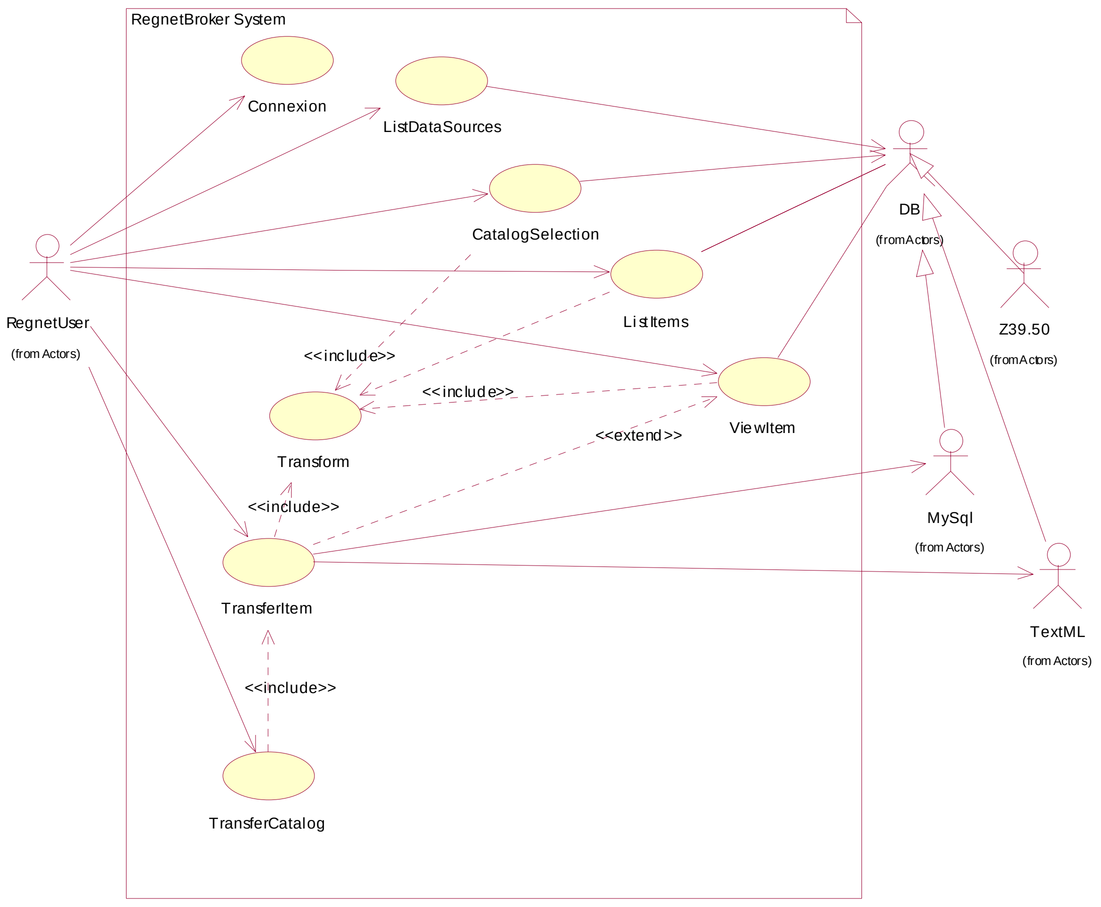


Figure 2 Use case overview

7.2.2.2 Use cases description

Use case: CatalogSelection

The user asks the list of available catalogues of the datasource that he has previously chosen.

Detailed description :

Title : CatalogSelection

Actors : REGNET user

Description : A REGNET user wants to select a particular catalogue of a specific datasource

Pre-conditions : The REGNET user has chosen the datasource for which he wants to get the catalogue

Post-conditions : The user has obtained the catalogue.

References : Transform use case.

Actor action	System response
1. This use case begins when the user wants to manipulate data (transfer or visualize items). He has selected the specific datasource for which he wants to get a catalogue.	2. The REGNET broker asks the appropriate datasource interface in order to retrieve the list of catalogues that contained.

3. The user select the catalogue desired.	4. The REGNET broker transform the xml list of catalogues and displays a page that offers many functionalities such as item search, item view, item transfer as regard as the catalogue that the user has chosen.
5. The user can now browse into the selected catalogue.	

Alternatives :

Any catalogue are available. The REGNET broker doesn't display catalogue list, it displays a message and no more functionality are available.**<u>Use case : Connexion</u>**

So as to use the REGNET broker system, the user must connect.

Detailed description:

Title : Connexion

Actors : REGNET user

Description : A REGNET user wants to connect to the REGNET broker system in order to handle data.

Pre-conditions : The REGNET broker system is initialised and ready to run. The user is logged on the REGNET portal (jetspeed)

Post-conditions : The user is connected to the REGNET broker system and can manipulate data.

Actor action	System response
1. This use case begins when a user wants to connect to the broker in order to do data exchange between datasources. He acces the broker system through the REGNET portal (jetspeed) by clicking on the appropriate link of the GUI.	2. The REGNET broker system try to authenticate the actor. If the actor is known, the system provides a page on which he can select a datasource.
3. The REGNET user is connected to the system and can manipulate data from different datasources.	

Alternatives :

The user is not known by the system, an error message is displayed.

<u>Use case : ListDataSources</u>

User asks for the list of available data sources. The REGNET broker system automatically presents a list of available datasources.

Detailed description :

Title : ListDataSources

Actors : REGNET user

Description : A REGNET user wants to list the different datasources so as to manipulate their data.

Pre-conditions : The user has logged on the REGNET broker system. The different datasources interfaces are available and have published their webservices.

Post-conditions : The user has obtained the list of the available datasources.

Actor action	System response
1. This use case begins when the user wants to list all available datasources. He clicks on the appropriate hyperlink that offers the functionality of listing datasources.	2. The REGNET broker try to bind all the datasources for which webservices implementaions are defined. Then a page listing the different datasources is displayed on the REGNET broker GUI.
3. Since the REGNET user has obtained the list of available datasources, he can select the datasource he wants to manipulate.	

Alternatives :

Any datasources are available. The REGNET broker doesn't display datasource list, it displays a message and no more functionality are available.

Use case : ListItems

After having selected a datasource, chosen a catalogue from it, the user can ask the list of items contained by the current datasource.

Detailed description :

Title : ListItems

Actors : REGNET user

Description : A REGNET user wants to list the items of a specific catalogue chosen before.

Pre-conditions : The user has chosen the catalogue for which I wants to list item.

Post-conditions : The broker displays the list of the items contained in the catalogue.

References : Transform use case.

Actor action	System response
1. This use case begins when a user wants to list the items contained in a catalogue. He has selected the desired catalogue.	2. The REGNET broker asks the appropriate data source for getting a list of items in the specific catalogue. Then it *transforms* the xml list of items and displays it on the GUI.
3. Since the user has obtained the list of items contained in the current catalogue, he can get information about a specific item	

Alternatives :

The catalogue doesn't contain items, in this case, the REGNET broker display a message and specifies that there isn't any item in the catalogue.

Use case : TransferCatalog

The user wants to transfert a whole catalog from a datasource to another.

To transfer the whole content of a catalogue, the broker will automatically use the transfer item functionnality for each item of the catalogue.

Detailed description :

Title : TransferCatalog

Actors : REGNET user

Description : A REGNET user wants to transfer a whole catalogue from a datasource to another.

Pre-conditions : Theuser has obtained the list of items of the catalogue which have to be transfered.

Post-conditions : The catalogue has been transferred to the desired target.

References : TransferItem use case.

Actor action	System response
1. This use case begins when a user wants to transfer a whole catalogue. He obtains the list of a catalogue for a specific datasource and clicks on the transfer button.	2. The system proposes an ihm that let to the user to select the target datasource and the catalogue in which he wants to transfer the catalogue.
3. The user selects the target datasource and the catalogue that will store the new Items. Then he clicks on the transfer button.	4. The REGNET broker realizes the transfer of each item of the catalogue. A message is displayed on the GUI to inform the user about the result of the operation.

Alternatives :

The transfer operation process doesn’t finish correctly, the broker displays a message on the GUI.

<u>Use case : TransferItem</u>

After having chosen a catalogue from a datasource, the user can transfer an item from a datasource to another. For this, he would select a target datasource.

Detailed description :

Title : TransferItem

Actors : REGNET user

Description : A REGNET user wants to transfer an item from a datasource to another.

Pre-conditions : The user asked the whole description of an item or he has obtained the list of items from a specific catalogue.

Post-conditions : The desired item has been transferred to the target datasource chosen by the user.

References : Transform use case.

Actor action	System response
1. This use case takes place when a user wants to transfer an item. He as selected the desired item into the list of item (see ListItems use case) or he has obtained the description of a particular item (see the ViewItem use case). Then he clicks on the transfer item button.	2. The system proposes an ihm that let to the user to select the target datasource and the catalogue in which he wants to transfer the item.
3. The user selects the target datasource and the catalogue that will store the new Item. Then he clicks on the transfer button.	4. The REGNET broker *transform* the item so as to make it compliant with the target datasource, and then gives the order to the target to store the new item. A message about the success of the operation is displayed.

Alternatives :

The transfer operation process doesn't finish correctly, the broker displays a message on the GUI.

Use case : Transform

All information that are retrieved by the REGNET broker are represented as xml files. The broker system uses xml transformations based on xml parsing or xslt transformations.

These transformations take place when information are to be represented on a ihm or during exchange process.

Detailed description :

Title : Transform

Actors : System

Pre-conditions : The REGNET broker system have retrieved any xml content.

Post-conditions : The xml content has been properly transformed.

Alternatives :

The transform process doesn't finish correctly, the caller is notified of the error.

Use case : ViewItem

For each item of a particular catalogue chosen in a specific datasource, the user can obtain information about it.

Detailed description :

Title : ViewItem

Actors : REGNET user

Description : A REGNET user wants to view the whole description of a particular item.

Pre-conditions : The user has chosen the catalogue in which he can find the desired item.

Post-conditions : The broker displays the whole description of the desired item.

References : Transform use case.

Actor action	System response
1. This use case begins when the user has selected a catalogue and he has obtained the list of the items belonging to it. Then he clicks on the hyperlink that let to view the whole description of a particular item.	2. The REGNET broker asks the appropriate datasource to obtain and retrieve the information belonging to the desired item. Then it *transforms* the xml description of the item and it displays the whole description of it.
3. The user can consult the description of the item. He can now decide to transfer this item to another datasource.	

7.2.3 Business model

7.2.3.1 Classes overview

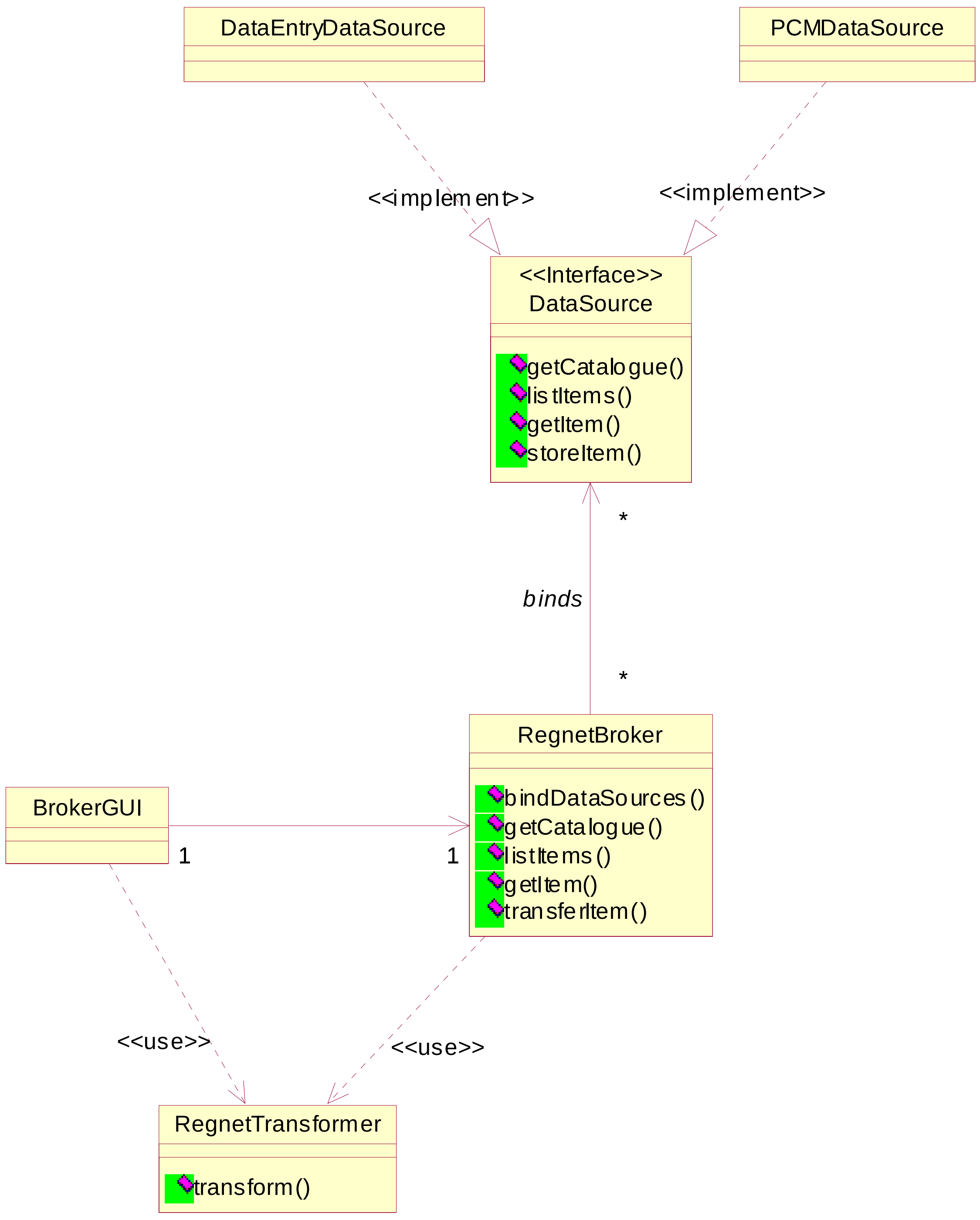


Figure 3 Business classes overview

7.2.3.2 Business classes description

DataSource

This interface provides services that offer a support for getting catalog, listing item, transfering item...

PCMDataSource

PCMDataSource implements the services described by the DataSource interface. The implementation of these services belongs to the interactions with PCM application.

DataEntryDataSource

DataEntryDataSource implements the services described by the DataSource interface. The implementation of these services belongs to the interactions with DataEntry application.

REGNETBroker

This class constitutes a mediator between the user and the datasources interfaces of the system.

REGNETTransformer

This class provides support to manage xml transformation in order to present result on the broker GUI.

BrokerGUI

This class represents the broker GUI. In reality the GUI is implemented as jsp pages.

7.2.3.3 Basic interactions

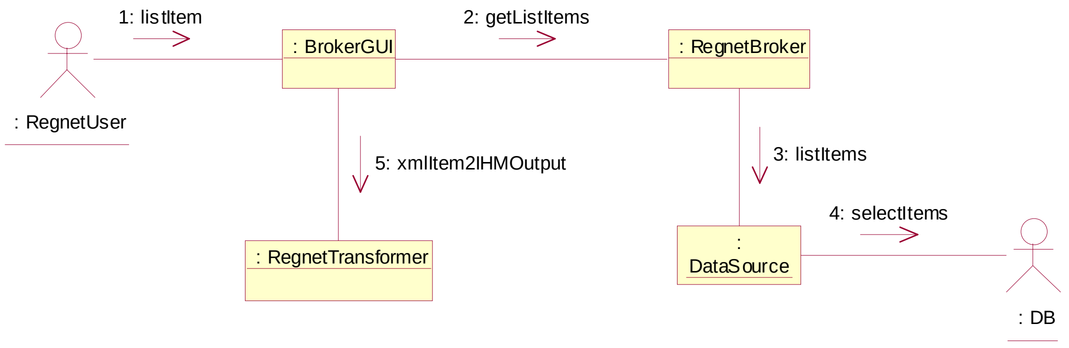


Figure 4 Basic interactions

This diagram gives an example of interaction between the different components of the REGNET broker for basic interactions such as retrieving information.

This example shows a scenario in which the user wants to get the list of items for a specific catalogue.

7.3 High level design

7.3.1 System architecture

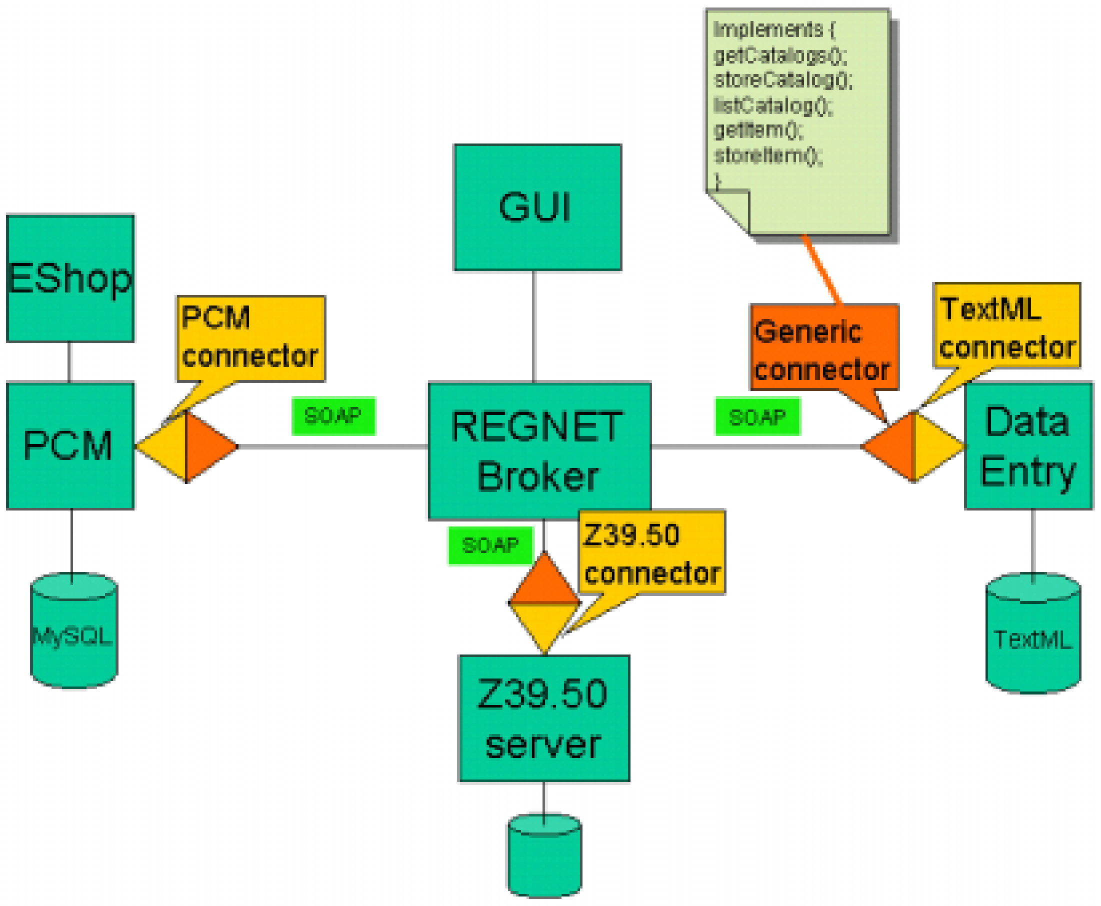


Figure 5 Broker system architecture

This diagram depicts the architecture of the REGNET broker system and the component with which it interacts (datasources especially).

The different datasources provide services via web services published thanks to the generic connector and give their own implementation (PCMConnector, TextMLConnector, Z39.50Connector).

To handle data the user can communicate with the broker GUI which gives orders to the broker. The broker then communicate with the datasources using the soap protocol.

7.3.2 High level description

7.3.2.1 Packages Overview

This section describes the packages involved in the REGNET broker system and illustrates the dependency that can exist between them.

Global overview

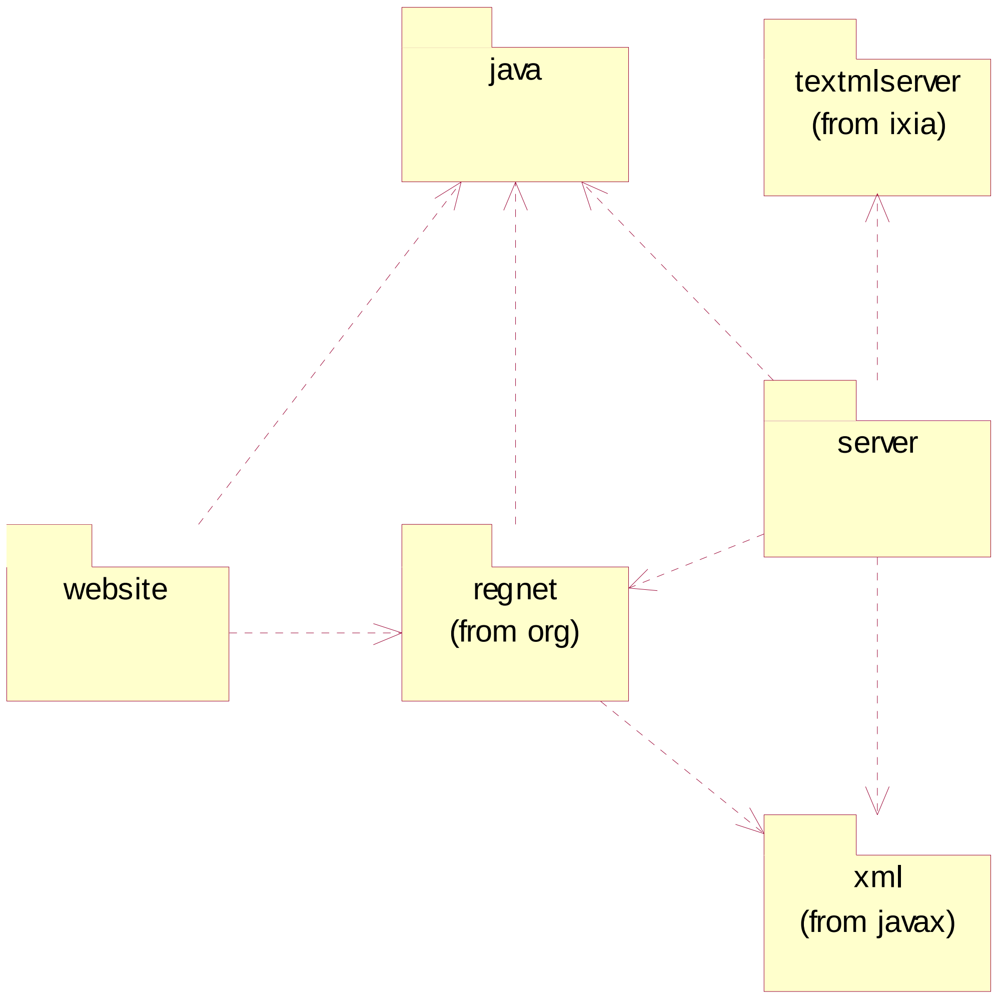


Figure 6 General packages overview

<u>Package java</u>

Java package containing all classes that come from the jdk.

<u>Package server</u>

This package contains classes that belong to the server part of the REGNET broker system.

<u>Package website</u>

This package contains all files belonging to the web ihm i.e jsp pages. All these pages constitute the Broker GUI.

<u>Package textmlserver</u>

This package provides support to manipulate xml database managed by the textmlserver software. The classes of this package let to use soap services to access this kind of database.

<u>Package REGNET</u>

This package provide the whole implementations classes of the REGNET broker system.

<u>Package xml</u>

This package contains java classes that provide support for processing xml files with SAX.

REGNET package overview

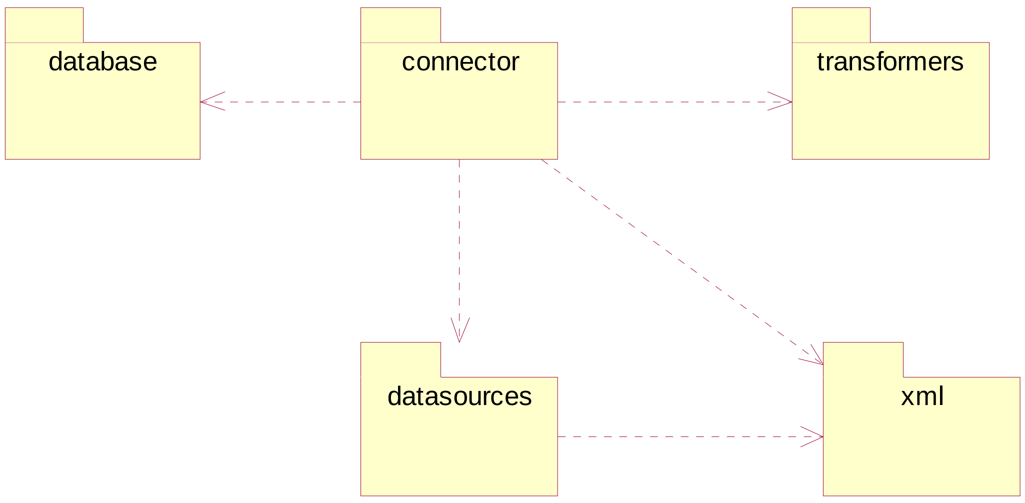


Figure 7 REGNET package overview

connector

This package contains the class belonging the the broker i.e the real mediator between the different datasources.database

This package contains all classes that provide support to connect to the different databases corresponding to the specific storage of datasources. datasources

This package contains the implementations classes of the web services provided by the different datasources.xml

This package contains classes that let to handle xml files belonging to the data exchange between datasources. transformers

This package contains classes that provide support for processing transformation of the xml files belonging to the data exchange between datasouces.Software components

Server side

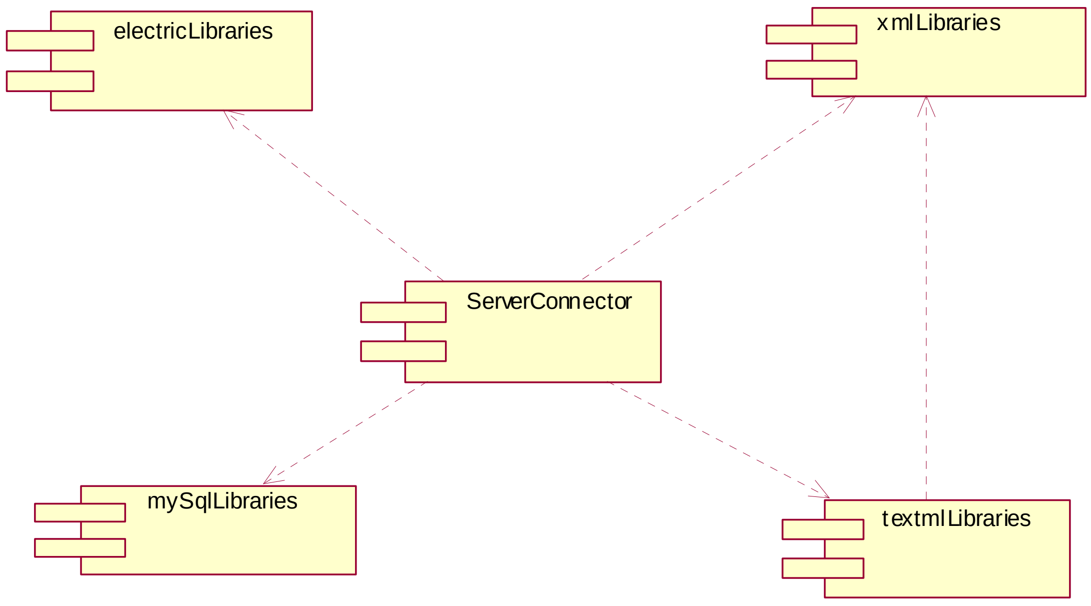


Figure 8 Server side components

ServerConnector component.

The ServerConnector component is the main of the server part. Thanks to it, the broker could access to the different datasources. For this, the ServerConnector component gives references on the different web services belonging to the datasources.

xmlLibraries component

The xmlLibraries is a set of component that contains all java archives giving full support to manipulate xml structures and to transform xml content according to different way : xsl transformation, xml parsing, … Note that there are two ways to manipulate xml content in the REGNET broker system : xsl transformation and xml parsing with sax.

The xml libraries used are the following :

jaxp-api.jar

sax.jar

xalan.jar

xerces.jar

xml-apis.jar

xmlParsersApi.jar

electricLibraries component

electricLibraries is a set of component for using the GLUE tool. GLUE is a simple, fast, comprehensive Java platform for creating and deploying applications with web services.

The java archives belonging to the use of this tool are the following :

collections.jar

GLUE-EXAMPLE.jar

GLUE-STD.jar

mySqlLibraries component

This component is a java archive containing classes that implements the interfaces described in the java.sql package of sun. The mm.mysql-2.0.4-bin.jar java archive has been used for this purpose.

textmlLibraries component

This component is a set of libraries that offer support for accessing a textml database. Accesses are possible thanks to the soap protocol.

Libraries used for this are the following :

textmlserver.jar

textmlserversoap.jar

activation.jar

all libraries described to manipulate xml contents

Client side

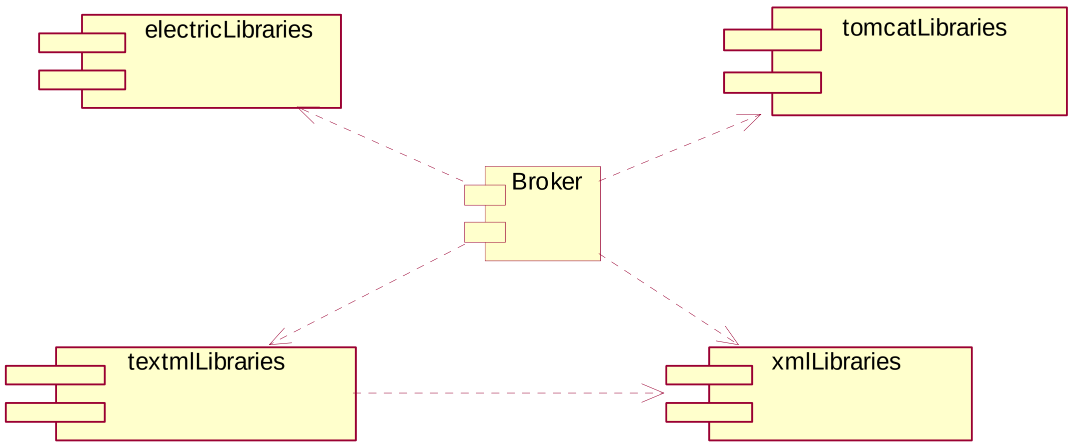


Figure 9 Client side components

Broker component

The Broker is the main component of the client side of the REGNETBroker system. Thanks to it, a REGNET user can access to the different datasources and then manipulate the content of each one.

The libraries used by this component are about the same that are used by the ServerConnector component. The main difference is that the broker is run in a tomcat context. For more information about the tomcat libraries please have a look at the tomcat-4.0.1 distribution.

7.4 Detailed design

7.4.1 Package server

7.4.1.1 Classes description

ServerConnector

The ServerConnector class publishes the web services provided by the datasources. It uses ServerProperties class so as to obtain information about publication properties (url, port, ...).

Operations :	
main	Publishes the web services of the different datasources.

ServerProperties

This class implements the Singleton Pattern.

His role is to give information such as url, port, implementation class usefull for the ServerConnector to publish the services provided by the different datasources.

Attributes :	
ELECTRIC_HOME	String

Operations :	
ServerProperties	Class constructor
getInstance	Returns an instance of the class. If there is no instance of it this operation create a new instance
getPort	Returns the port number of a given datasource
getServer	Returns the server url part of a given datasource
getImplementation	Returns the implementation class of a specific datasource
getInterface	Returns an interface for a given datasource
getDataSources	Returns a list of available datasources
getElectricHome	Returns the electric.home value
main	Implemented for test purpose

7.4.1.2 Server initialization

Sequence diagram

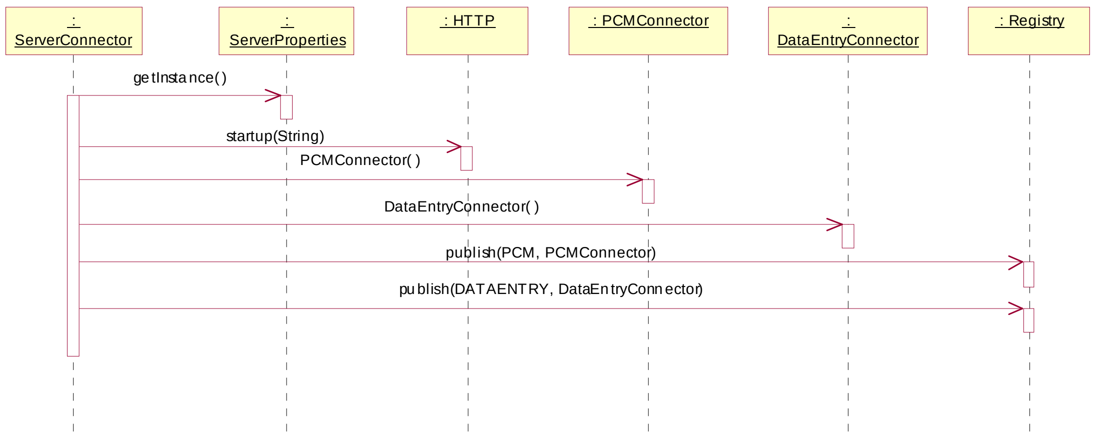


Figure 10 Server initialisation

This diagram shows how the initialization of the REGNET borker server part is done. The ServerConnector publish the web services provided by the different datasources. At the end of this scenario, two datasources will be accessible (PCM and DataEntry datasources).

Involved classes

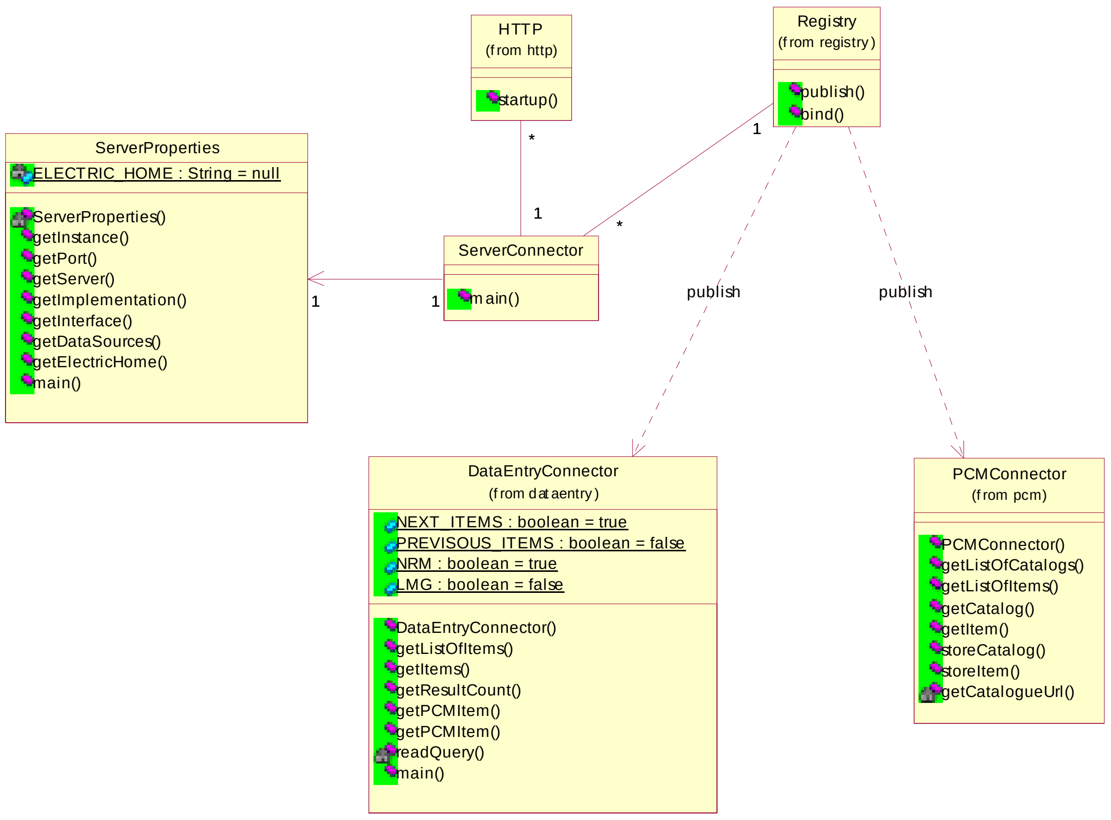


Figure 11 Server initialization. Involved classes

7.4.2 Package REGNET

7.4.2.1 Package connector

Classes description

BrokerProperties

This class implements the Singleton Pattern.

His role is to give information such as properties (pathes, url, ...) usefull for the broker.

Attributes :	
TMP_XML_FILEPATH	String
PCM_SERVER	String
PCM_PORT	int
XSL_PATHFILE	String

CLEANING_DELAY	long
STORE_AREA_ID	String
ADDRESS_ID	String
WARE_HOUSE_ID	String
USER_ID	String
TEXTML_URL	String
DATA_ENTRY_USER	String
DATA_ENTRY_PASSWORD	String
DATA_ENTRY_SERVER	String
DATA_ENTRY_DOCBASE	String
DATA_ENTRY_DOMAIN	String
TEXTML_QUERIES_PATH	String
TEXTML_NB_RECORDS	int
LOG_SERVER_PATH	String
LOG_BROKER_PATH	String

Operations :	
BrokerProperties	Class constructor
getInstance	Returns an instance of the class. If there is no instance of it this operation create a new instance
getTmpXMLPathFile	Returns the path to the temporary directory of xml files
getPCMServer	Returns the server url part of the PCM server datasource
getXSLPathFile	Returns the path of the directory containing the xsl files that manage xml transformations
getPCMPort	Returns the PCM port application
getCleanDelay	Returns the delay of the temporary directory cleaning process
getTextMLUrl	Returns the url of the textmlserver of the DataEntry datasource
getDataEntryUser	Returns the login connection to the textml database
getDataEntryPassword	Returns the password connection to the textml database
getDataEntryDomain	Returns the NT domain of the machin hosting the textml database
getDataEntryServer	Returns the name of the machine hosting the textml database
getDataEntryDocBase	Returns the name of the textml database
getTextMLPathQueries	Returns the path to the directory containin xml

	queries for the textml database
getTextMLNbRecords	Returns the number of records to be display on a single web page
getLogServerPath	Returns the path to the server log file
getLogBrokerPath	Returns the path to the broker log file

Broker

This class constitutes a mediator between the REGNET users and the distributed datasources. Thanks to it a user can obtain information about the content of a specific datasource and then exchange data from one to another.

Operations :	
doBinds	Binds the broker to the datasources webservices
getItem	Returns an item from a given catalogue
getListOfCatalog	Returns a list of catalogue for a given datasource
listDatasources	Returns a list of available datasources
listItems	Returns an item list for a given datasource and a given catalogue
transferItem	Transfers an item from a datasource to another

7.4.2.2 Package database

Classes description

DataBaseConnectionManager

This class implements the Singleton Pattern.

His role is to give connection instance to the datasources interfaces that provide services for datasources based on relational databases.

Attributes :	
DRIVER	String
URL	String
LOGIN	String
PASSWORD	String

Operations :	
DataBaseConnectionManager	Class constructor
getInstance	Returns an instance of the class. If there is no instance of it this operation create a new instance
getConnection	Returns a connection to a mySql database
main	Implemented for test purpose

TextMLConnectionManager

This class implements the Singleton Pattern.

His role is to give connection instance to the datasources interfaces that provide services for datasources based on textml databases.

Operations :	
TextMLConnectionManager	Class constructor
getInstance	Returns an instance of the class. If there is no instance of it this operation create a new instance
getSearchServices	Returns a serch service object that make possible queries
main	Implemented for test purpose

7.4.2.3 Package datasources

Classes description

IDataEntryConnector

This interface describes the web services that are available for the DataEntry datasource.

Operations :	
getListOfItems	Returns a list of items from the dataentry datasource
getResultCount	Returns the number of records retrieved by a querie
getPCMItem	Returns the object representation of a PCM item

IPCMConnector

This interface describes the web services that are available for the PCM datasource.

Operations :	
getListOfCatalogs	Returns a list of the catalogues of the PCM datasource
getListOfItems	Returns a list of item for a specific catalogue
getCatalog	Returns the whole content of a specific
getItem	Returns an item from a given catalogue
storeItem	Stores an item coming from another datasource

Item

This class is the object representation of items that come from datasources. This object is transfered thanks to the soap protocol through the network.

Attributes :	
categoryId	String

name	String
description	String
UPCCode	String
price	String
priceType	String
currencyId	String
quantity	String
manufacturer	String
weight	String
wMeasure	String
volume	String
vMeasure	String
height	String
length	String
width	String
dMeasure	String
export	String
image	String

Operations :	
Item	Class constructor
Item	Class constructor
marshall	Translates the representation of a item into the soap formalism in order to transfer it on the network
unMarshall	Operation that make it possible to build an object received in a soap envelope.

7.4.2.4 Package dataentry

Classes description

DataEntryConnector

This class gives an implementation to the web services provided by the DataEntry datasource.

Attributes :	
NEXT_ITEMS	boolean
PREVISOUS_ITEMS	boolean
NRM	boolean
LMG	boolean

Operations :	
DataEntryConnector	Class constructor
getListOfItems	Returns a list of item for a given catalogue
getResultCount	Returns the number of records retrieved by a querie
getPCMItem	Returns the object representation of a PCM item
readQuery	Reads an xml query that comes from an xml file
main	Implemented for test purpose

7.4.2.5 Package pcm

Classes description

PCMConnector

This class gives an implementation to the web services provided by the PCM datasource.

Operations :	
PCMConnector	Class constructor
getListOfCatalogs	Returns a list of the catalogues of the PCM datasource
getListOfItems	Returns a list of item for a specific catalogue
getCatalog	Returns the whole content of a specific
getItem	Returns an item from a given catalogue
storeItem	Stores an item coming from another datasource
getCatalogueUrl	Returns an url that let to connect to the PCM server

PCMRequest

This class is usefull for the PCMConnector to construct SOAP request according to the formalism used in the business applications.

Attributes :	
soapEnvBegin	String
soapEnvEnd	String

Operations :	
PCMRequest	Class constructor
getListItemsHTTPQuery	Constructs a soap request in order to retrieve a list of item from the PCM datasource
getItemHTTPQuery	Constructs a soap request in order to retrieve an item from the PCM datasources
getSoreItemHttpQuery	Constructs a soap request for storing an item in the PCM datasource

PCMRequestSender

This class is a tool which let to the PCMConnector to send soap request, established by the PCMRequest object, to the server of business applications.

Attributes :	
request	String

Operations :	
PCMRequestSender	Class constructor
send	Sends a PCMRequest to the soap server of the PCM datasource

7.4.2.6 Package xml

Classes description

XMLFile

This class is an utility for handling xml content belonging to the REGNET borker context.

Operations :	
XMLFile	Constructs an XMLFile from its string representation
getBytes	Return a xml content in a bytes array form

XMLFileCleaner

This class is a tool wich goal is to clean a temporary directory containing files that could be created by the REGNET borker to perform transformation of xml content. Periodically this thread empties the appropriate directory.

Operations :	
XMLFileCleaner	Class constructor
run	Activates the thread cleaner
clean	Cleans the temporary directory
main	Implemented for test purpose

XMLFileFactory

This factory provides support to created xml files belonging to the REGNET context. These files can be stored in a temporary directory so as to be transformed by xsl stylesheets.

It is aslo possible to load xsl files, stored in a directory, in order to perform transformations.

Operations :	
getXMLFile	Return a XMLFile object
getXSLFile	Return a XSLFile object

7.4.2.7 Package dataentry

Classes description

GetItemsSaxParser

This class is usefull for the DataEntryConnector to retrieve items of a textml database. It aims at parsing the whole files retrieved after a request to the textml database and producing a single xml file describing them.

Attributes :	

xmlFile	String
write	boolean
isItem	boolean

Operations :	
GetItemsSaxParser	Class constructor
getResult	Returns a query result in a xml form
startElement	Processes begining tags
endElement	Processes ending tags
characters	Processes tags content

7.4.2.8 Package pcm

Classes description

StoreItemSaxParser

This class lets to analyse the result of the storage operation in PCM datasource. Indeed, the result of such an operation is retrieved as an xml file which is parsed by this object to give a report of the operation process.

Operations :	
itemIsStored	Returns a boolean that gives a report of the storage process of an item
startElement	Processes begining tags
endElement	Processes ending tags
characters	Processes tags content

7.4.2.9 Package transformers

Classes description

BrokerTransformer

This class is a xml transformer dedicated to the REGNET broker context. It aims at manipulate XMLFile objects that are to be transformed by xsl.

Operations :	
BrokerTransformer	Class constructor
Transform	Transforms a given XMLFile in order to present result on the broker GUI

7.4.2.10 Operations samples

Sequence diagrams

List items from DataEntry

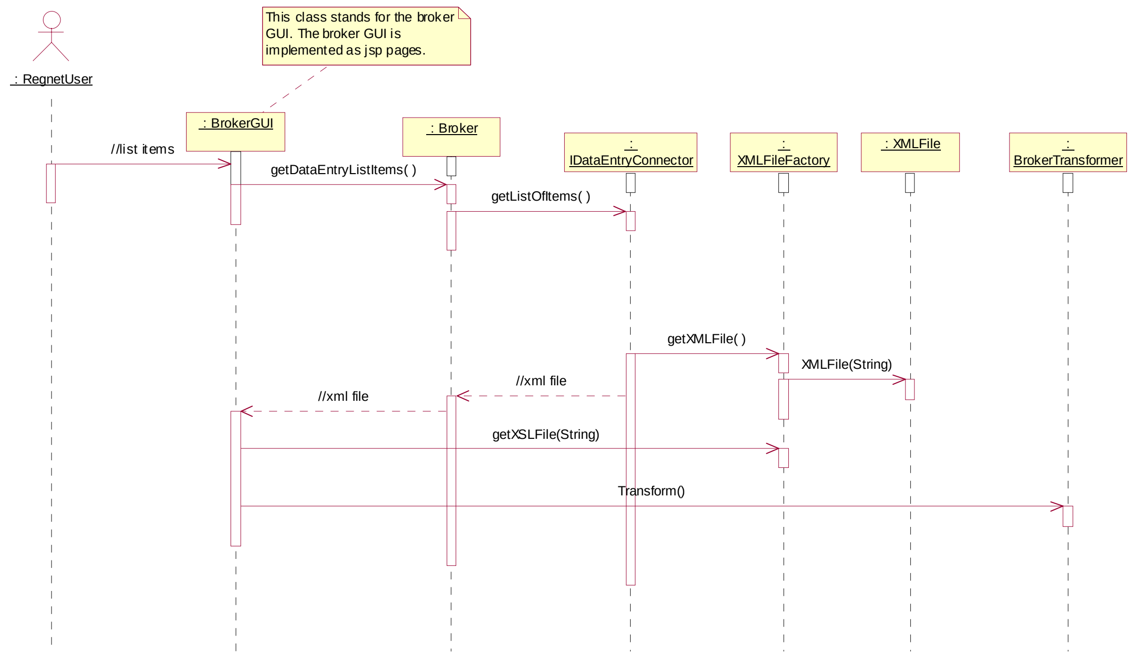


Figure 12 List items from DataEntry

This diagram illustrates a scenario in which a user wants to list items from the DataEntry datasource.

The user asks the list of items thanks to the broker GUI. Then the broker GUI requests the broker to get a list of items.

The dataentry datasource returns a list of items in a xml file obtain via the xml file factory.

Once the Broker GUI has retrieved this xml file, it uses the broker transformer to present the result list to the user.

<u>Transfer item from DataEntry to PCM datasource</u>

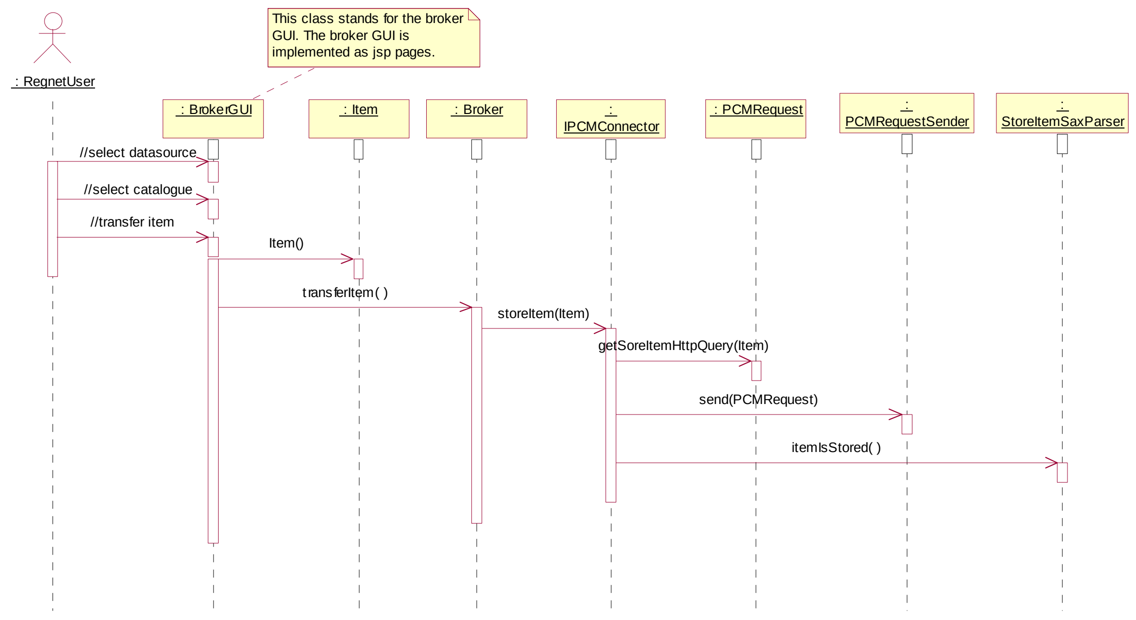


Figure 13 Transfer item from DataEntry to PCM datasource

This diagram illustrates a scenario in which a user decides to transfer an item from the DataEntry datasource to the PCM one.

The user starts by selecting the datasource (PCM in this case), the catalogue where the item to transfer is stored. Then he gives the order to transfer this item. The broker GUI create an object representation of the item and then asks the broker to store it in the PCM datasource.

The IPCMConnector creates a PCMRequest that it sends thanks to the PCMRequestSender.

The server of the PCM datasource returns a result as an xml structure that the ICPMConnector parses using the StoreItemSaxParser so as to give a report of the storing process.

<u>View PCM Item</u>

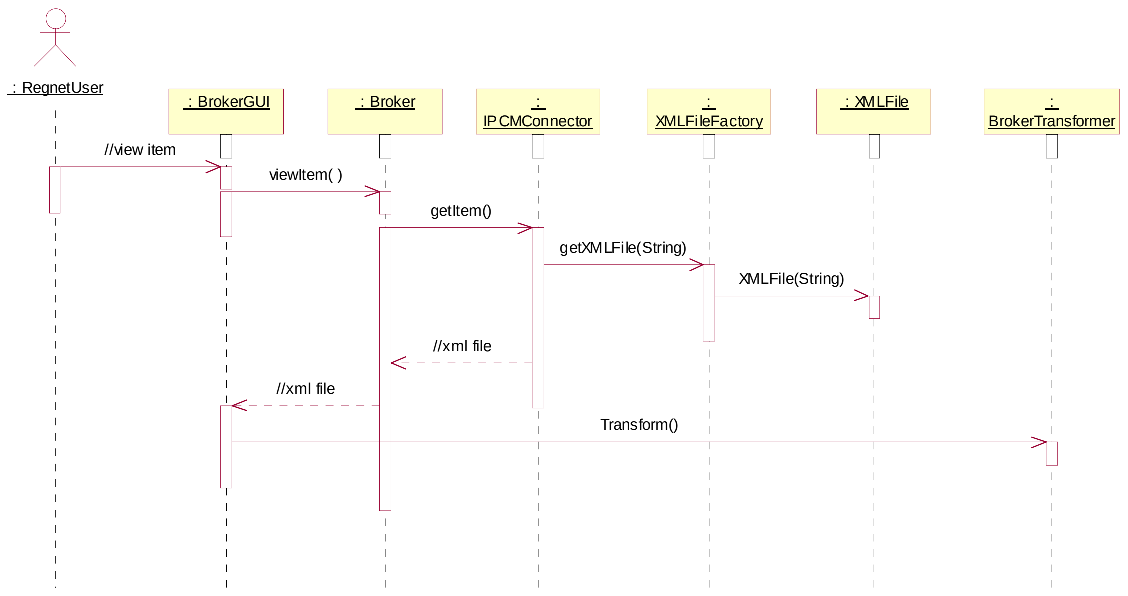


Figure 14 View PCM Item

This diagram illustrates a scenario in which a REGNET user wants to see the description of an item. He has previously choosen the item which he wants to consult.

Thanks to the broker GUI he asks the broker to give him a description of the desired object.

The broker invokes the IPCMConnector to retrieve the whole description of the item. Once the IPCMConnector has obtained the information it creates an xml file via the xml file factory and then return it to the broker.

Finally the broker GUI uses the BrokerTransformer in order to present the description to the user.

Involved classes

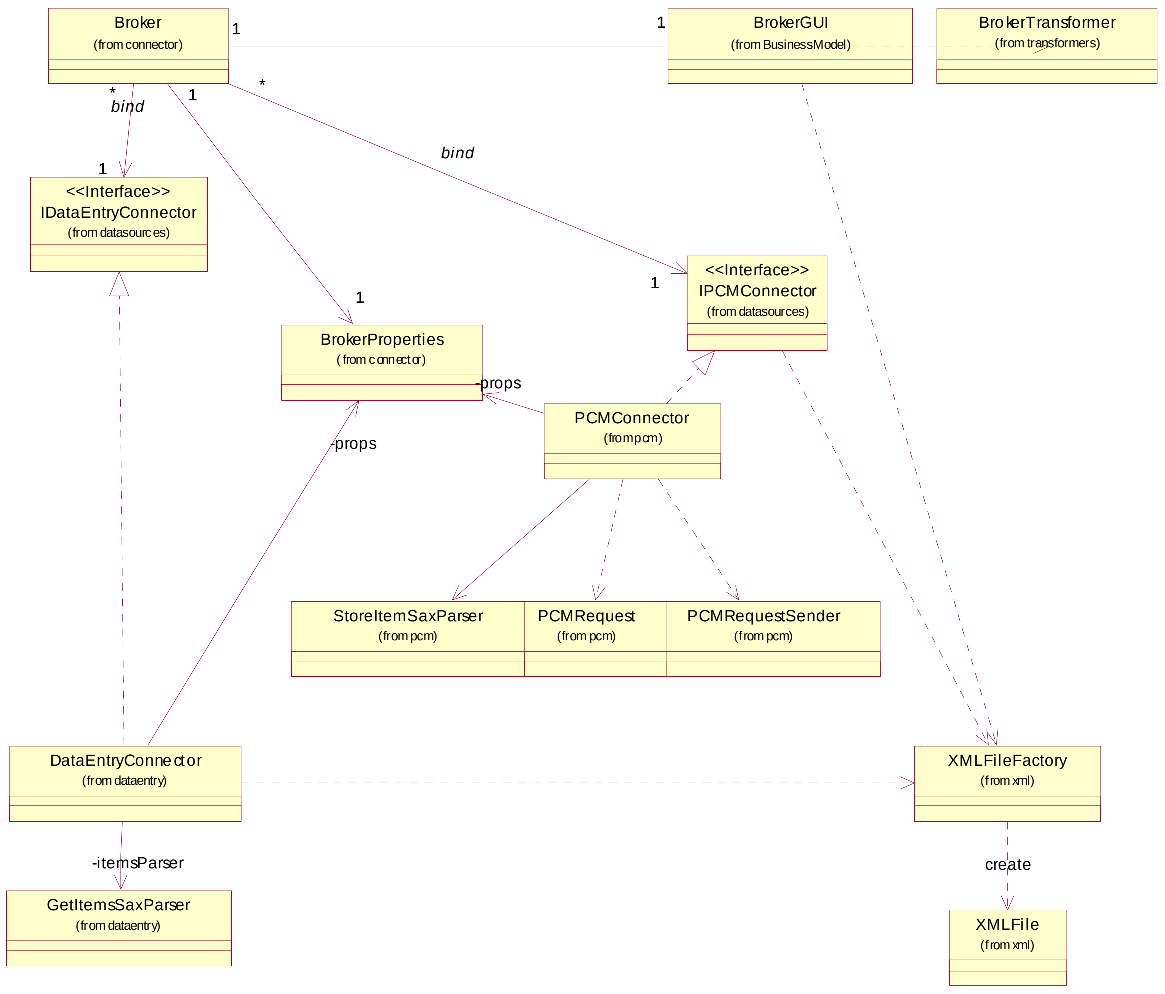


Figure 15 Classes overview

8 Epublishing

The following section describes the technical issues of the Macromedia Director integration into the EP component of the REGNET system. The evaluation and implementation of the demonstrator was based on Macromedia Director version 8.5.

8.1 Requirements

From the main objectives numerated in our task brief we identified the following requirements

Evaluation and demonstration of integration-options of the REGNET workspace / documents based on the REGNET DTD.

Evaluation and demonstration of XML Transformation options using the Macromedia Director plug-in infrastructure.

Evaluation and demonstration of Import / Export options of Macromedia Director templates into the REGNET ontology node.

8.2 General Functionality of Macromedia Director 8.5

This section describes the basic functionality of the Director application. Director uses a number of theatrical metaphors.

The basic thing that a user creates with Macromedia Director is a Director movie, or movie (for short). A movie has a beginning, a middle and an end. A Movie can be played, stopped and rewinded. The user can also splice movies together, or show a movie inside another movie.

When the user saves a movie, Director creates a file your disk. This file contains all the necessary information about the movie, so the user can always go back and change it. The movie file format is platform independent. This means, the user can create a movie on a Macintosh and then play/edit that movie on a Windows computer, or vice versa (as long as Director installed on both machines). Of course, the Director application itself isn't platform independent; if the user wants to run it on two different platforms, two different copies need to be purchased.

8.2.1 The Cast

Everything that appears in a movie is referred to as a cast member. That includes all of the images, whether they are snapshots of a cartoon character in motion, props on the stage, or background images. Titles and labels are also cast members. So are the buttons and the text areas. Even the sound effects and instructions to the computer (scripts) are cast members.

All cast members are stored in a cast, which may be viewed with the cast window (select Cast from the Window option on the menu bar). A cast member may be referenced by a number (reflecting its position in the cast) or a given name. Although something must be a cast member in order to appear in the movie, cast members don't actually appear in the movie until they are placed on stage.

8.2.2 The Stage

All the action in a movie takes place on the stage. A movie can be created by arranging the cast members on the stage.

What the user can see on the stage is actually a copy of the cast member known as a sprite. Each sprite has properties, such as its position on stage and its size.

8.2.3 The Score

The score is similar in appearance (and function) to timing sheets used in traditional animation. The main part of the score is a large table divided into rows and columns. Each column corresponds to a different instant of time or frame. Each row corresponds to a different channel. The numbered channels are the sprite channels, which show what sprites are on the stage. When a movie is played, Director does this by showing one frame after another until it reaches the end of the movie. The end of the movie is the last frame (column) that has anything in it.

8.2.4 Event Handlers

Playing a movie can be viewed as a succession of events. The movie starts; each frame is shown (entered and exited); the movie stops. Director also recognizes several other events: a mouse button is clicked; a key on the keyboard is pressed; a timer runs out. Normally, these events go by unnoticed. That is, unless the creator of the movie has written event handlers indicating that something else should be done.

An event handler is a sequence of instructions to the computer. When the named event occurs, the computer follows (executes) the instructions within the handler. Handlers are written in scripts using a language called Lingo.

8.2.5 Lingo

Lingo is a powerful scripting language that supports: branching and conditional statements; local and global variables; sequential and associative arrays; string, numeric, and boolean operators. It also supports a form of object-oriented class definition.

Director is used throughout industry to create prototypes -- and even some finished products -- for games, educational applications, and electronic books. But the thing that has done the most to establish Director as a key development tool is Shockwave.

8.2.6 Shockwave

Shockwave is a special movie format that can be played by any web browser that has the Shockwave plug-in (or, like the newer browsers, has an integrated Shockwave player). Shockwave allows you to embed your Director movies in your web pages, making them truly interactive. Numerous commercial web sites use Shockwave to enhance the web experience.

8.3 Implementation 1

We implemented the functionality by developing a movie using the Macromedia Director script language Lingo. The XML document's contents can be accessed through Lingo or convert the contents to a Lingo list that is meaningful to you and your movie. Once your movie has read an XML document, it can perform actions that you define based on the contents of the document.

When started, the movie is able to import REGNET XML files and the elements are collected in an own cast library called *REGNET1*. Furthermore a sample slide show is demonstrating only one possibility of generating movies by using the REGNET content. After stopping, the elements of the cast library can be used for any movie the user wants to generate.

For extending the functionality to parse XML and import files we used some free available *Xtras*.

Xtras are software components, to enance the functionality of Director. It is a software module written to work with (to "plug into") the internal software structure of most Macromedia products. Like a Netscape plug-in, an Xtra is designed as an adjunct, extending the base functionality of the program.

The Xtra, generally speaking:

- can extend the functionality of sprites, transitions, tools, and/or Lingo
- are more easily made cross-platform since the Macromedia Open Architecture is a software standard
- can be cross-*product* since the MOA is identical between

The result sets gathered from the Search&Retrieval component of the Regent system can be imported by using the **FileIO Xtra**. It is used for loading the REGNET XML file into a member field, which can then be parsed by other scripts.

For parsing the XML structure we used the **DOM Lingo Xtra**.

We also evaluated the XML Parser Xtra which is already included in the Macromedia Director 8.5 but was not powerful enough for our parsing requirements. The following figure shows the files needed for the generation of a slideshow.

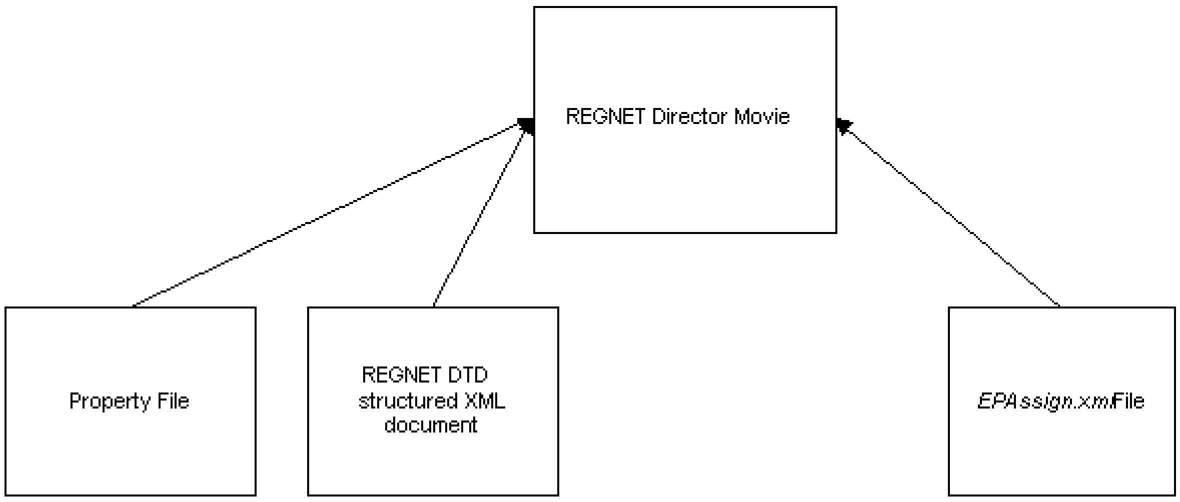


Figure 1: The REGNET Director Setup Movie files imported

8.3.1 Functionality

This section describes the general functionality of the REGNET Director movie.

8.3.1.1 Load Property File

Opens a file dialog with which the user can select a XML Properties file, which includes the elements that will be imported into the REGNET cast library.

Example:

```
<?xml version="1.0"?>
<included>
  <tag>
    <name>_created</name>
    <type>text</type>
  </tag>
  <tag>
    <name>title</name>
    <type>text</type>
  </tag>
  <tag>
    <name>dc-relation</name>
    <type>bitmap</type>
  </tag>
</included>
```

8.3.1.2 Load XML REGNET File

Opens a file dialog with which the user can select the REGNET XML file to be parsed. (The tag from which the parsing is started can be entered by the user in text field.)

8.3.1.3 Generate Slideshow

The EPAssign file is loaded and specifies which XML fields are assigned to which fields in the slide show. The user can dynamically choose the fields to be published in the slideshow. If the file cannot be found there, a default setting is used.

EPAssign.xml file:

```
<?xml version="1.0"?>
<assignpair>
  <REGNETtag>name</REGNETtag>
  <assignedto>number1</assignedto>
</assignpair>
```

8.3.2 Cast Libraries

The movie owns two cast libraries:

8.3.2.1 XML Parser 1.0.5

This cast library contains all the fields and scripts for the REGNET template movie. The fields control the workflow of the import and control the slideshow. The Lingo scripts are the implementation of the functionality.

8.3.2.2 REGNET Cast

The *REGNET1* cast library contains all the imported field of the REGNET DTD. The members can be exploited for any Macromedia Director presentation or movie by drag and drop.

8.4 Implementation 2

For demonstration purposes an interactive multimedia production was realised around the theme "Faydherbe's traces in Mechlin".

This implementation makes use of:

storyboard and scenario templates developed during the REGNET project (the methodology)

third party software, Macromedia Director (the tool)

The scenario templates represent several sequences that are typical for several Cultural Heritage presentation schemes. The combination and parameterisation of these sequences can produce tailor made electronic productions suited for use in different contexts.

Typical characteristics for this approach are the dynamic interaction on one page lay out (instead of sequential changing of full pages) and the multilingual capabilities at any time at any place during the navigation of the multimedia production.

For Web publications, the well-known and wide spread plug in “Shockwave” of Macromedia has to be downloaded and installed. This way the same multimedia production can be viewed through a browser.

8.4.1 The stage templates

Four templates were developed to cover all requested functionalities for an interactive guide city and location tour pertaining to traces of a Mechlin (Belgium) sculptor and architect.

Template 1 consists of:

main menu
language choice
image of the artist

Template 2: consists of:

the city map of Mechlin with indication of locations
a list with the names of the locations
language choice
help facility
back and main menu buttons
image of the indicated location
entrance button to enter the indicated location

Template 3 consists of:

the ground plan of the chosen location with indication of pieces of art
a list with the names of the pieces of art
language choice
help facility
back and main menu buttons
image and text of the indicated piece of art

Template 4 consists of:

contextual thematic text with hyperlinks
images activated by clicking on the hyperlinks
language choice
help facility
back and main menu buttons

8.4.2 The used casts

In order to offer language switching at any time at any place and to facilitate the separate development and editing of images, texts and other multimedia elements, the following cast structure was decided:

Internal casts (always present in the movie)

behaviours (for the event handlers)

imported images (jpeg from the REGNET database)

External casts (loadable into the movie)

push buttons with text (one in Dutch and one in English)

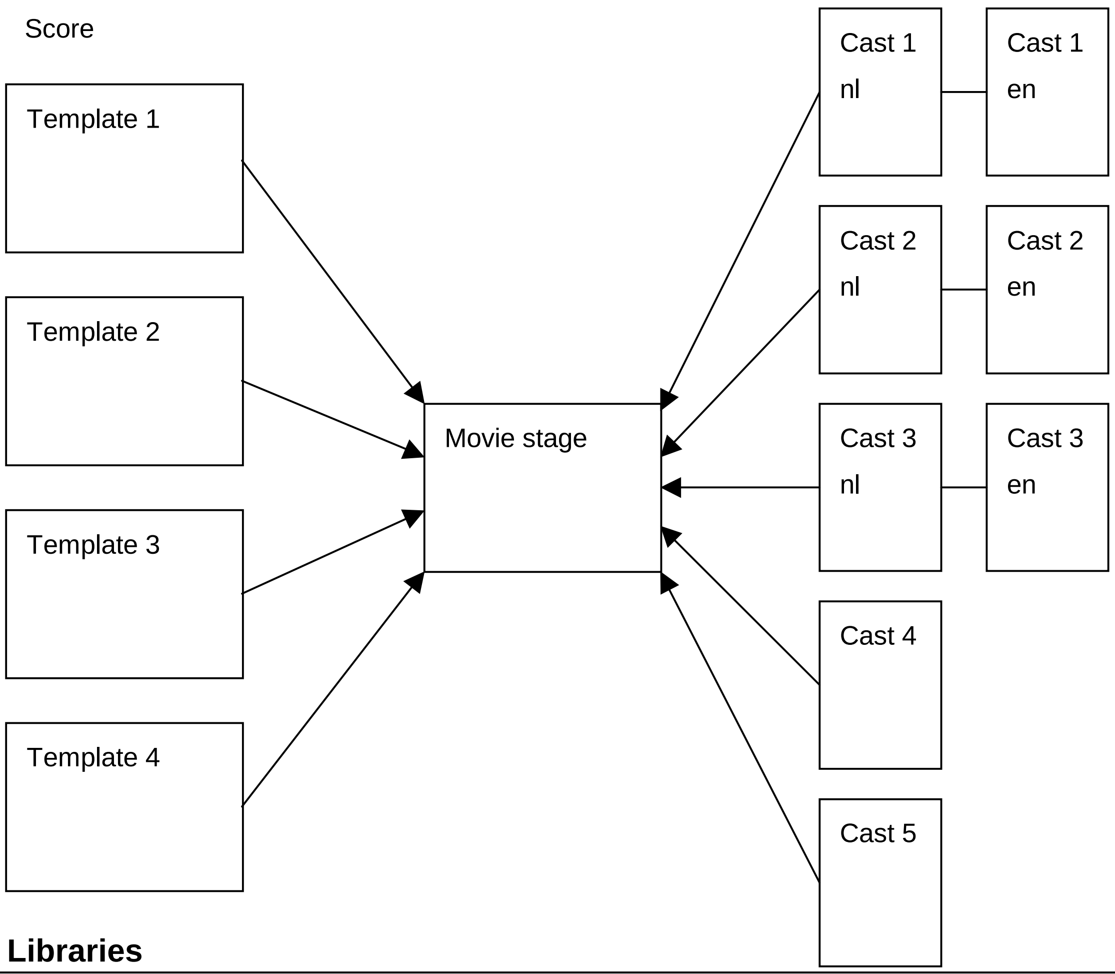
imported texts (from the REGNET data base, one in Dutch and one in English)

inserted texts (from the Director editor, one in Dutch and one in English)

8.4.3 The movie

Two types of movies can be generated, a full one and a compressed one. The first will be used in kiosks, public terminals and on CD. The compressed version will be used on the Internet (Shockwave plug in).

The following diagram reflects the implementation.



8.5 Libraries

8.5.1 Fileio Xtra

The fileio xtra is needed for importing the content of a file into a field of a cast library. It is already included in the Macromedia Director 8.5.

8.5.2 XML Parser Xtra

The XML Parser Xtra allows the Director developer to access the nodes of an XML document. A node can be a tag (similar to an HTML tag, also called an element), character data (text that does not appear inside the angle brackets of a tag), or a processing instruction (a special type of tag that passes data to the parsing application for special processing). You can extract information from the XML document by looking at its nodes with Lingo. This access to XML data allows users to incorporate XML documents into their movies, and selectively extract data from the documents to be used in whatever way the user wants.

8.5.3 DOM-Lingo Beta Release 2

DOM-Lingo is a binding of the Lingo scripting language used by Macromedia's Director 8.5 Shockwave Studio to the W3C Document Object Model (DOM) IDL interfaces, and a complete implementation of the DOM Level 2 specification.

DOM-Lingo includes all the parent scripts necessary to implement the DOM spec. With it, XML document trees can be created from scratch or generated from parsed XML documents, manipulated via the DOM in memory, traversed, and serialized/printed.

DOM-Lingo does not require the XML Parser Xtra. It includes a robust, non-validating XML processor written entirely in Lingo. The processor creates DOM-Lingo node trees from valid XML documents located in memory, on a local drive, or at a remote URL. The DOM-Lingo XML processor is currently the only comprehensive XML processor available for Director/Shockwave.

DOM-Lingo includes support for the TreeWalker and NodeIterator interfaces from the DOM's Traversal and Range modules for quick and efficient traversal of XML document trees. The NodeFilter interface is also supported to provide exacting specificity in node visibility. Both a brief technical overview and a complete implementation listing are available at this site.

DOM-Lingo is currently available in a Beta version. A licensing model might be applied at a later stage.

8.6 Conclusions

The following action points summarize the main outcome of the Macromedia Director analysis:

In order to be as independent as possible and due to effort constraints we used file import for data integration. The user is able to collect all the data needed in the personal Web space and download it. The user can then further develop the presentation. We have not evaluated whether a SOAP interface is supported by Lingo, in order to directly load the data from the REGNET repository into Director.

In Director, Plug-Ins are provided via Xtras. There are two XML Parser Xtras. The XML Parser Xtra doesn’t provide all the functionality we needed for parsing the REGNET files. Using the XML Parser Xtra requires that the structure and content of the documents is known during parsing. We wanted to be independent of a fixed DTD (e.g. REGNET DTD) to be flexible on changes of the REGNET DTD.

The DOM-Lingo Xtra supports the W3C Dom Level 2, which is an open standard specification. As parsing directly on the file does not work in the evaluated version, we also had to use the Xtra fileio, which provides a file dialog for file opening and import of files into member fields and further processing.

The REGNET Director Movie can be used as template and customized with additional REGNET data.

Macromedia Director supports the generation of a Shockwave exe, which can be generated for different platforms.

The development possibilities of Macromedia Director, i.e. the scripting functionalities of Lingo, are quite powerful for organizing and arranging presentations. However, the functionalities for processing XML documents and defining user interfaces are quite limited although this functionality is provided by external Xtras.

Due to that, a full integration of Director into the REGNET system would require the implementation of a dedicated Xtra which was not focus of the underlying analysis.

9 Eshop

9.1 Overview

E-shop is a system, which allows users to make purchases via Internet. The main difference of this system with others is that it allows users to search for products in different databases and not only in the local database. So that means that the system can use distributed databases which can be placed wherever the seller wants. This fact makes system more flexible relatively to the existed trade systems.

9.2 Structure and content of the total system

9.2.1 Functionalities

The main functionalities of the system are analysed below:

Search facility

Free Search :

The user can fill the following:

Name of the product, Category of the product,Supplier, Price range (between two values)

Search through the different categories

Shopping Cart

The user can add items to the shopping basket or remove items from this, to view any information related to the item (which is presented in different window than the current), to move the item in his wish list, change the quanity of the item and calculate the total amount of the order.Other functionalities such as updating the basket, changing the currency of the price (the default value is in Euro) and proceeding with a order are also supported.

Order

The system gives the user the appropriate forms in order to specify the delivery address even if it is different than that of the user.

Before the order completes, the system can present the order again to the user just for a last check. At this point, the user is able to verify its orders and specify the shippment costs.

E-payment

The system also supports the e-payment functionality. The users are given the option to specify the way of their payment. In the case of the credit card, the system provides the user with an e-mail as a confirmation of the order and the user completes the order by filling the card number, taking advantage of the Internet Secure Environment which is supported.

In the case of a deposit in a bank account, an e-mail is sent as a conformation of the order including all the necessary details of the payment and the bank account.

Order History

The user can view at any moment all of his orders.

Terms and Conditions

At this part, a document is provided to the user in order to inform him for any special conditions and implications that are related to the use of all if the functionalities.

Return to portal

An easy and direct way for the exit of the e-shop page

9.2.2 Technical features

As the e-shop is connected with the Product Catalogue management, SOAP technology is used to support a stable interconnection. The e-shop system retrieves all the required information from the PCM system (where all the data related to the items are stored). The PCM system also supports the e-shop functionalities as there is a field which showswhich items have been put in the e-shop system.

In the figure below there is a representaion of the inreactions among different PCMs and the e-shop system

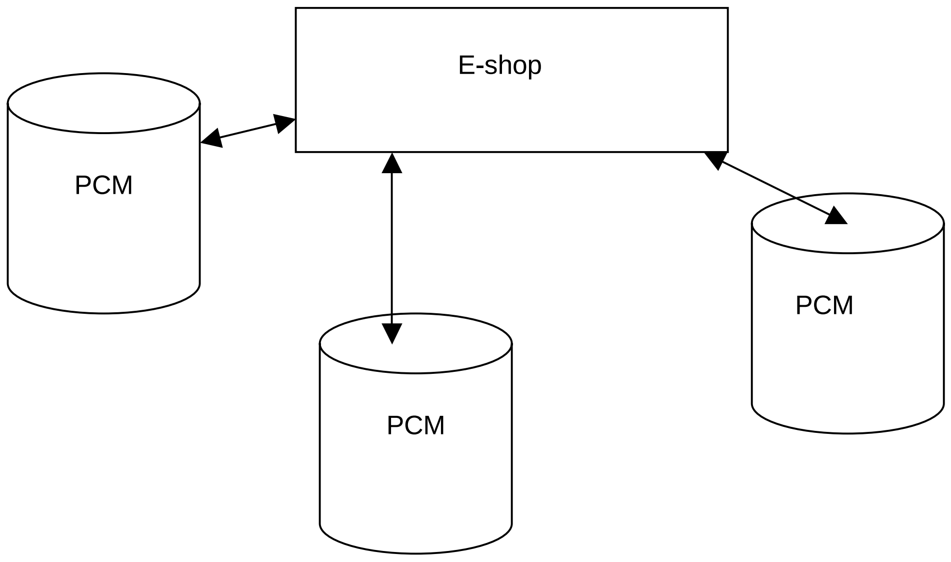


Figure 1: Interconnections among different PCMs and e-shop system (all are supported by the SOAP protocol)

Furthemore, has been established a connection between e-shop component and ontology system which contains all the personal information about users. So we have developed an ontology client that can connect e-shop with ontology system in order to take the full profile of registered users. With this connection there is not necessary for users to register within our system in order to gain the specifc e-shop functionalities. For that reason any potential user has to register firstly from the REGNET portal if he/she wants to use e-shops' functionalities.

10 E-Business

10.1 Overview

The E-Business (EB) gives possibility of advanced search of goods and services and also process orders on the goods and services for registered users (only for suppliers). Below are the main capabilities of the system:

10.2 Structure and content of the total system

10.2.1 Functionalities

The e-business component offers the following functionalities

Advanced search of goods and services.

The E-Business gives possibility in search by the name of the items and services, by the manufacturer, by its category, by the warehouse, and also by the price type and other attributes.

Drawing-up of contracts.

Registered users (only suppliers) have the possibility to draw up a contract. After the contract is drawn the supplier should get to the following page of order processing. On this page for order processing the supplier gets the information about the contract number, the name of goods or services, the vendor, the quantity or quality goods or services, the warehouse address and others.

Proccess of orders

After the contract is drawn the supplier should get to the following page of order processing.

After the finishing of the order processing, the system can send an email not only to the supplier who offers the specific items or services but also to the person who express his/her interest.

10.2.2 Technical features

B2B extension was built on SOAP technology. B2B scenario is a part of e-business. This sub-system interconnects with PCM (Product Catalogue Management), in order to find the necessary products and services to make a contract.

The series of interconnections between the B2B and PCM is presented in Diagram 1.

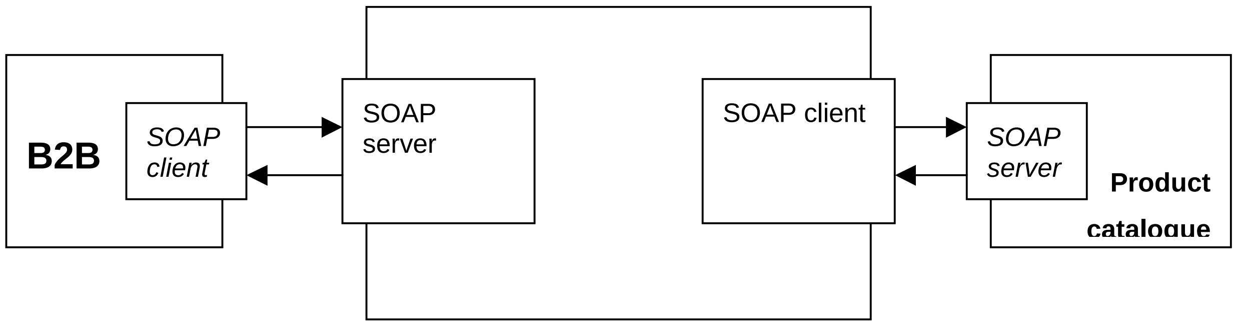


Diagram 1.- Series of interconnections between the b2b and PCM.

11 Product Catalogue Managment

11.1 Overview

This subsystem includes meta data describing products (real or digital) offered by content providers up to services offered by service providers (e.g. consultancy, digitising projects, etc). REGNET allows also search and retrieval of distributed product (and service) catalogues. Doing this the user will be able to compare products which supports his/her buying decision.

11.2 Structure and content of the total system

The PCM system is a strong information system which enables the management of all the items and services through the catalogues in an efficient way. The system can be accessed only by registered users.

11.2.1 Functionalities

The first step for a new user of PCM is to register himself in the portal. Then he/she has to apply to the administrator of the system with a request for the creation of a new catalogue. The creation of the new catalogue can be done with the administrator part of the PCM which gives administrator the ability of creating new catalogues. Also it is very important for every user to specify the warehouse that contains the products. Every user can have more than one warehouse.

Suppliers have the ability to insert items or new services to the catalogues, to edit the information of already existing items and to delete items from the catalogues. Moreover, he will have the ability to add information about the new warehouses to the PCM and to his own database.

Furthermore a user has the ability to add his items to the e-shop system in order to sell them. So when he wants to add a new item in a catalogue he can choose specifc field in order to add the item to the e-shop system. Furthermore users can provide a picture for each specific item.

Main functionalities are listed below.

Insert of items

The user can insert either new item or new service in the system. In case there is no warehouse for the specific items, the user is not allowed to insert the item.

Insert warehouse

The user must follow the procedure of filling all the required information about the warehouses.

Management of items and services

Any kind of modifications such as update, delete, view of the items or services which are stored in the system. For the most efficient management of the databases, all items and services must be categorized in catalogues and categories.

Insert items into the e-shop

The PCM system is connected with the e-shop system. Therefore, the user can put items from the PCM page to the e-shop sysrtem.

Administration functionality

There is an additional functionality for the administartor to modify the categories, the catalogues and the currency types.

11.2.2 Technical features

The most important aspect of PCM is the use of the SOAP protocol in the interconnections between the central (administrative database) and the remote (distributed) catalogues.

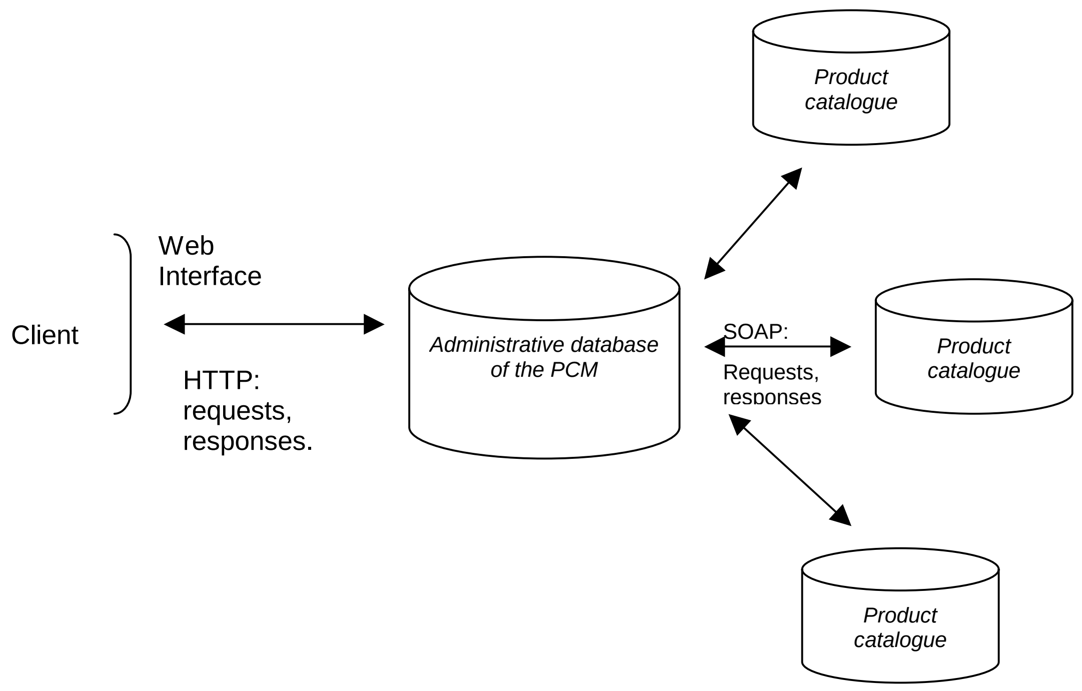


Figure 2: Interconnections among the PCM components

PCM uses two databases from the one logical model, which is presented on the picture 2. First one is (btob_cl) is a product catalogue. It provides the information about the products and contains three tables, the items, the stock and the shipping table. Total system may have many databases btob_cl in distributed locations. Rest of the tables from model belongs to the server's database.

12 Auction

Overview

Auction system was build with a purpose to provide clients with the opportunity to bid for different products. Moreover, some clients may apply to the auction administrator in order to obtain the right to display their own items for sale. The administrator has the authority to change users' rights and the ability to forbid an item to take part in an auction.

Auction system has a mechanism, which allows moving the auction's scheduled time to an overtime, which is no more than 20 minutes after the actual time appointed for termination.

Users with different rights have a different set of options in the users' menu and as a result different functional abilities.

12.2 Structure and content of the total system

12.2.1 Functionalities

The functionalities of the auction system is different for different users' groups. The most important difference is that the ordinary user doesn't have a right and menu option to insert a new item to the auction. On the contrary users, who have this privilege, can also obtain statistics for bids made regarding their items. This menu option is provided in the profile options.

Ordinary users have the ability to see and make changes in their own profile data, and can also see their bids in different auctions. All users have the ability to get information related with current and future auctions.

The searching system provides the ability for a multi-criteria search. These criterions can be the name of the product, its description, category, start and finish auction time, providing the option to search before, after or at the moment of the appointed date.

Auction system has its own administrative system. However, this system is unavailable for ordinary users, while permission is available only by the administrator's login and password. This system allows the administrator to manage the items, users and add different categories to the database. Nevertheless, it does not allow the administrator to remove existing categories, since a great number of items may exist at the auction and any removal may destroy existing records.

12.2.2 Technical features

The database was designed according to the technical requirements. The structure is presented in figure 1.

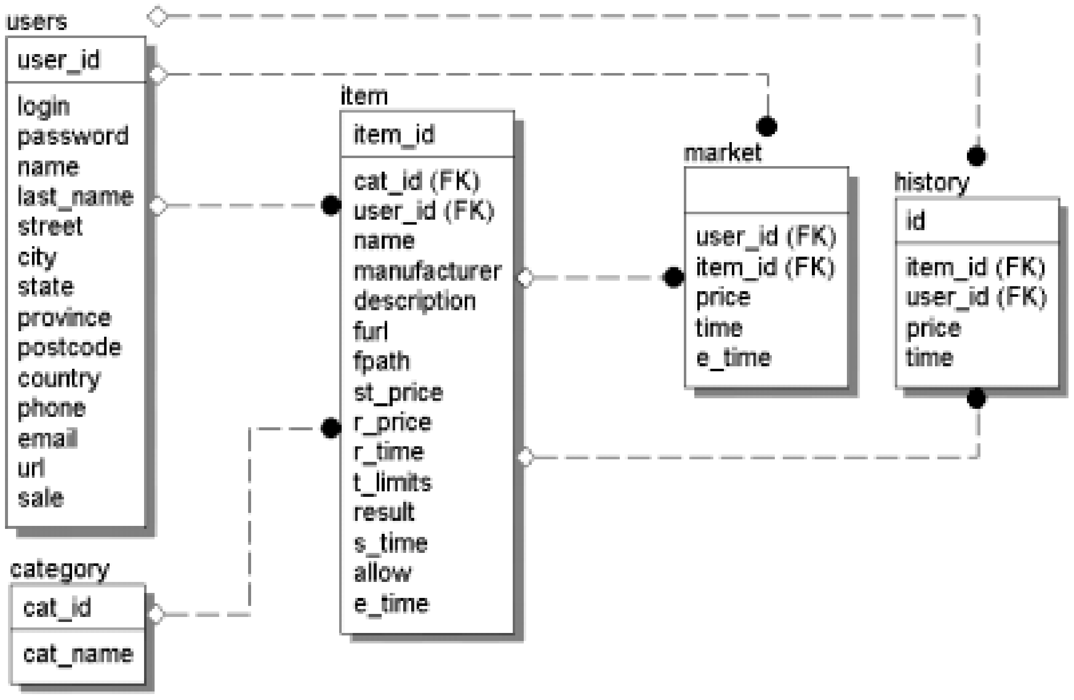


Figure 1 – Structure of the database "Auction".

The structure contains 5 tables with an optimal sets of attributes. Table **MARKET** is the main table, where current bids are stored. After over passing the time limits of the auction for a particular item, all the records related to this item are transferred to the table **HISTORY**, where auction's records are kept. These records can also provide statistics for bids and successful auctions.

Each item in table **ITEM** belongs to the owner (user_id). This table contains attributes such as start price (st_price), reserve price (r_price), time of registration (r_time), time to start the auction (s_time), time to end the auction (e_time), result (result), permission to start the auction (allow), and a standard set of attributes, which describe items.

13 Index+ gateway

13.1 Introduction

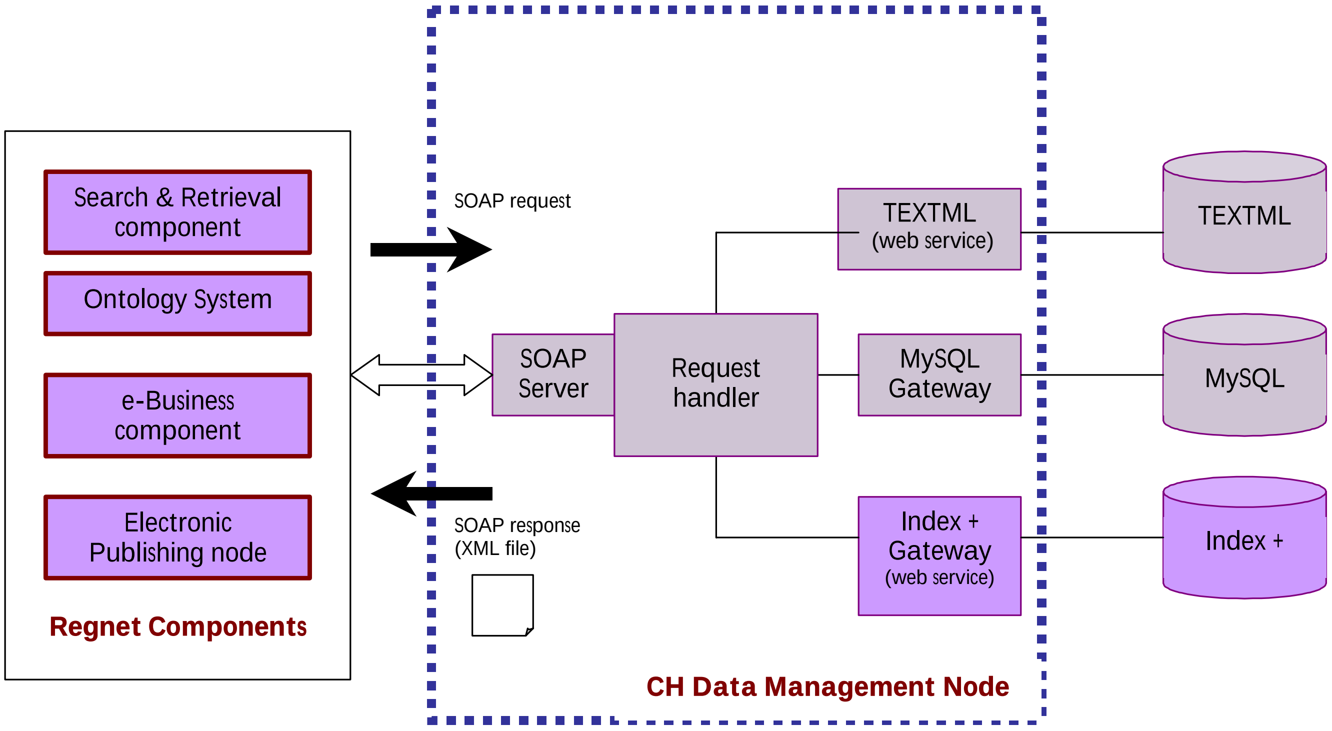
Index+ is a powerful software toolkit for creating systems to manage structured and unstructured text, data. It features fast searching, high storage capacity, a robust, network orientated, server-client architecture and a range of application development tools.

Index+ interfaces can be simple enough for inexperienced users, yet powerful enough for professional users. An Index+ database can be re-structured, copied or customised with a variety of application building tools. Index+ also provides interfaces to a range of programming environments.

Index+, created by System Simulation Ltd, has evolved as a toolset for building a range of applications for demanding clients. This range of applications and the continuous evolution of the Index+ system has ensured that it is very well suited to the requirements of the modern information economy. It enables the creation, development, management and exploitation of information and knowledge assets of any kind. System Simulation Ltd's business is now largely focused on partnerships with clients to optimise their use of information assets and maintain Index+ as a leading-edge technology.

Index+ has been designed to open systems principles. By meeting industry standards, Index+ provides users with flexibility in their choice of hardware and software components. Index+ is supported on a wide variety of hardware platforms and peripherals. Index+ can be integrated with third party hardware and software products, providing a consistent user interface across a wide range of environments.

Index+ is written in the C and C++ programming languages. Each major component of the Index+ software is implemented as a separate module with clearly defined interfaces and functions.



The Index+ server is the database engine at the heart of the Index+ system. It is a single threaded process handling all storage, retrieval, and searching. This ensures integrity for information held in the database.

The Index+ Access module is a layer for accessing Index+ databases within the Reference System. It provide essentially consistent interface to the Search&Retrieve component.

Index+ gateway key technical features:

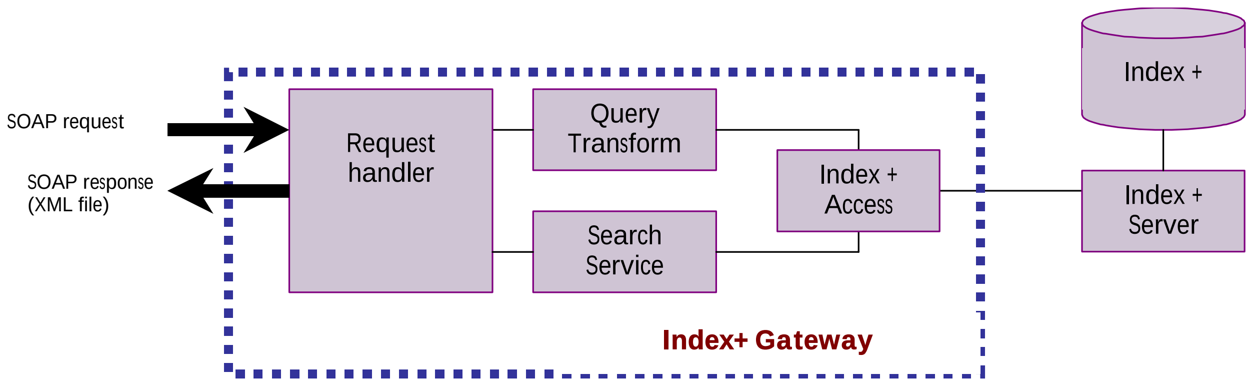
- Full-text indexing and searching
- Server-client architecture with multi-platform operability, support for multiple sites and security features
- Designed to be flexible
- Suitable for applications of any size, Index+ is scalable and can adapt as information requirements change
- May be moved between different hardware platforms without having to be rebuilt
- Search and retrieve information from remote databases.

13.2 Index+ Gateway component

The target of Index+ gateway is to transform all the request for the Index+ repositories. The index+ connection will be usefull in the project because of the clustering of REGNET with the OpenHeritage project.

The Index+ gateway component for the CH Management Node is designed to link directly to the REGNET Web server so allowing users to access Index+ databases from any Web browser.

The Index+ gateway is a web service which allows the Cultural Heritage Management Node to search and retrieve information from remote Index+ databases.



The interface that the Index+ Gateway has to realize is the same of the component for access the TextML database. Fore the search service the component has to provide the following functionalities:

- SearchRetrieve

  The main method of the Search Service. Returns query hits and the records found respectively.

- GetRecords

  Returns the records specified in a record list filtered by the optional doctypes list.

A full range of clients and other utilities for database configuration and management supports the Index+ server.

The Index+ gateway provides the interface to the Index+ server for the REGNET platform through its programmable interface to the basic services in the C language.

The Index+ Access module also provides a set of application libraries supporting higher-level capabilities in C, Visual Basic, C++ and an application developer's language called IPI.

13.3 Index+ features

Text searches look for the presence of a given word or phrase in the documents being searched. They can include proximity and partial matching requirements. Proximity specifies that the words being searched for must occur within a given span of words and optionally in a given order. Partial matching allows incomplete words to be entered together with wildcard characters. Wildcard characters can occur anywhere in the search specification.

Expression searches look for numbers or strings within a range of values. They can search for the results of calculations based on the contents of fields in one or more records. This provides Index+ with functionality common to conventional numerically orientated relational databases.

By default Index+ searches across all the fields in a data record. However, if required, searches may be restricted to a particular field or group of fields. Clients can be configured to allow the user to create complex queries by simply filling in forms or to give the user full flexibility and power of the Index+ retrieval language in a command-line interface

Index+ does not impose any pre-defined structure on the application. It can store and index large bodies of text as single fields or records as well as supporting the highly structured organisation typical of a relational database application. Both approaches can be used in a single database allowing the data to be structured to suit the needs of the application rather than the dictates of the underlying database system

13.3.1 Index+ Data structures

Index+ databases are configured from the following classes of objects - fields, links, record types and indexes. A field is a sub-category of information about a particular entity or object. For example, a record containing information about a particular artist is likely to have a field for the artist's name, a field for their date of birth, and a field giving information about the artist's work. Fields can hold variable length text, numbers or dates. The Index+ server can also handle structured data in the form of lists, arrays and arbitrarily nested structures.

Links provide one-to-one, one-to-many and many-to-many associations between records allowing complex data objects to be built up in the manner of object-oriented databases. Links can have an associated type so that retrievals can be made on the basis of link type. Index+ has different ways of implementing links to enable the most appropriate to be chosen for a specific link function.

Record types consist of collections of fields. They allow different kinds of records to be maintained separately in the same database. Each record type in a database can have a different set of fields and types of information or they can share fields, allowing users to search for field-specific information across multiple record types.

13.3.2 Text indexes

Text indexes divide text fields into a number of words or phrases using a word parser and then index the location of each word or phrase.

Text indexes are configurable. They can be set up to use particular character sets for parsing a body of text into individual words. This means that Index+ has no difficulty with non-English characters such as those with accents. Text indexes can be configured to be case sensitive or insensitive. They can contain information about which field contains the word and which words surround it to allow for field-specific or proximity searching. If there are words that do not need to be indexed, these can be held in a stoplist.

A database can contain many differently-configured text indexes. Typically a single field-specific text index may be used for the majority of the text fields with separate text indexes used for special cases such as part numbers, accession numbers and so on.

13.3.3 Expression indexes

Expression indexes support both range searching (retrieving all records with a field value between an upper and lower limit) and the retrieval of a set of records in a particular order. This can be used to index date, time, date/time and numeric fields, or to allow alphabetic browsing of structured text fields.

Expressions can be numerical or logical calculations and can utilise string manipulation operations. A versatile language allows the result of any expression which can be derived from a field or combination of fields to be indexed

Index+ has developed in response to the requirements of our users. It has many features to support the implementation of robust, easy to use, information systems.

13.3.4 Multiuser

Index+ servers running on UNIX platforms are full multiuser systems, using region locking on files. Index+ has been designed so that items in the database are kept locked for the shortest possible time, giving impressive performance statistics. The only limit on the number of users is the hardware available.

13.3.5 Multi-platform operability

Index+ can operate in a wide range of environments and configurations including:

- Single user MS-Windows
- UNIX server networked to MS-Windows PC clients
- UNIX server and clients with serially linked user terminals
- UNIX server and clients with LAN and user PCs in terminal emulation mode
- NT server configurations as for UNIX

The database files created by Index+ are portable between implementations on different platforms. This means that a database can be built on one machine and then distributed and run on a range of different systems without having to re-build the database.

The Index+ server is designed to be portable to any UNIX or POSIX compliant platform. It has been ported to a wide and growing range of platforms. We will port Index+ to new environments on request.

13.3.6 Network support

The Index+ server-client networking layer uses a network abstraction which means that Index+ can be run on a variety of networks. Several different network interfaces are supported on various platforms. In recent years there has been a strong trend among network software providers towards presenting applications with a single interface to networking, regardless of the network or the network software. This means that often a single Index+ installation can run on different underlying network platforms, just by changing the networking software: no change is necessary to Index+. This is particularly true of TCP/IP networks, the most common network for running server-client Index+.

Unless specially requested, Index+ networking clients for MS-Windows are now configured to use the WINSOCK.DLL v1.1 interface to TCP/IP. Index+ will run over any network software which conforms to this standard.

On UNIX systems, Index+ is built to use the 4.2/4.3BSD sockets interface, where available. This allows Index+ software to run alongside all the other standard TCP/ IP applications available under UNIX.

13.3.7 Multimedia support, electronic publishing

Index+ is a sophisticated tool for creating high quality presentations covering a large quantity of text, graphical and multimedia content served from an Index+ database.

Index+ can store any form of digital data, indexing it as required by use of an appropriate parser and index combination. This makes it ideal for supporting large scale repositories for managing digital data for electronic publishing on CD-ROM, online systems or other media. These repositories hold digital content in a general purpose form that can be re-used for multiple purposes thus offsetting the cost of digitisation.

13.3.8 Thesaurus support

The Index+ Thesaurus Editor supports the development and use of thesauri to BS 5723 (ISO 2788). Users can import standard thesauri such as the BSI Root thesaurus and then develop them further to meet their particular requirements.

The Index+ thesaurus provides a means of enforcing a controlled vocabulary in certain fields during data input as well as being an aid for searching. When searching, the thesaurus offers broader, narrower or related terms in order to refine a query.

13.3.9 Support and Training

We offer a comprehensive package of support arrangements tailored to user requirements, ranging from 24-hour pager support to telephone and email support. Training is provided for application builders, database administrators and end-users.

13.3.10 Migration tools

Tools for importing data from a wide range of third party files and databases to Index+ applications are available. We have extensive experience in this area.

13.3.11 Specialist requirements

Information management systems frequently require support for a range of specialist functions as well as the basic storage and retrieval facilities. The application building tools provided with Index+ are designed so that appropriately tailored systems are very easy to make, use and maintain. These facilities include:

- Interfaces to newswire systems
- Storage management for large text and image archives
- Interfaces to Point of Sale equipment
- Interfaces to EDI systems
- Interfaces to scanning and OCR software
- Script-based multimedia authoring
- Loan management
- Exhibition design

Index+ has been successfully integrated with RDBMS and other more specialized applications, for example workflow automation or accounting systems.

List of Figures

Table of Appendices