

# THE ECOLEAD PLUG & PLAY COLLABORATIVE BUSINESS INFRASTRUCTURE

Ricardo J.Rabelo<sup>1</sup>, M.Mar Rodrigo Castro<sup>2</sup>, Alex Conconi<sup>3</sup>, Michele Sesana<sup>3</sup>  
<sup>1</sup> *Federal University of Santa Catarina, BRAZIL, [rabelo@das.ufsc.br](mailto:rabelo@das.ufsc.br)*  
<sup>2</sup> *Software AG Spain, SPAIN, [mrodrigo@softwareag.es](mailto:mrodrigo@softwareag.es)*  
<sup>3</sup> *TXT e-Solutions, ITALY, {[alex.conconi](mailto:alex.conconi); [michele.sesana@txt.it](mailto:michele.sesana@txt.it)}*

This chapter presents a distributed and open ICT infrastructure called *ICT-I* that has been implemented in the ECOLEAD project to help members of Collaborative Networks in doing businesses and collaborations more efficiently. *ICT-I* design relies on the service oriented architecture paradigm, and it has been implemented with web-services. Its services are used on demand and pay-per-use models. It is flexible to support an easy entrance of new services and the withdrawn of others. So far the type of organizations envisaged by the proposed *ICT-I* are the members of virtual breeding environments, virtual organizations and professional virtual communities. This paper details the *ICT-I* reference framework and the derived services. A general overview about its main value and assessment is given in the end.

## 1. INTRODUCTION

Previous book chapter has motivated for the need of collaborative business infrastructures as well as has introduced the concepts that were applied in the design of an infrastructure within the ECOLEAD project. This chapter focuses on the presentation of the related implementation aspects, showing the so-called *plug & play collaborative business ICT infrastructure* (or simply *ICT-I*) derived for the three classes of CNOs in ECOLEAD: virtual organizations breeding environment (VBE), virtual organizations (VO) and professional virtual communities (PVC).

*ICT-I* acts as a collaborative bus or platform that allows different and distributed CNO members and involved applications to better collaborate with each other. Regarding the CNO requirements for an *ICT-I* and the envisaged business models associated to current trends on software development, it has been designed as a *distributed* middleware and implemented as a set of services. Besides being devoted to CNOs, its main value-added relies on the fact it is *plug & play*, i.e. once CNO members want to collaborate, they get plugged into the CNO “world” and can further “play” with by accessing the available supporting services, seamlessly. The *ICT-I* designing principles aimed to build it as a transparent (as much as possible), open and easy-to-use platform, and that can be able to support the requirements associated to CNO in terms of processes, knowledge, computing resources, people and information (Rabelo et al., 06).

From the technological point of view, it has essentially been designed applying the SOA (*Service Oriented Architecture*) paradigm and the SaaS-U (*Software-as-a-Service-Utility*) model. *ICT-I* is security embedded (Sowa & al., 2007) and it is not

locally deployed. Its services are accessed remotely, on-demand and paid-per-use, following a diversity of *business models* (Borst & al., 05). As it was mentioned in the previous book chapter about the ECOLEAD ICT-I, it is an evolving infrastructure as long as its new services can be added and others be withdrawn from the *Services Federation*, a logical aggregation of services providers (Rabelo and Gusmeroli, 2007).

ICT-I implementation is actually the last stage of a derivation process, i.e. an activity where a particular instance is derived from an abstract concept. In this case, the *ECOLEAD ICT-I Reference Architecture* is the most abstract definition of “all” functionalities that can be developed to support “any kind” of CNOs, and that can last as long as ICTs evolve. This Reference Architecture has been detailed described in the previous book chapter.

The first stage of the derivation is the consideration of the *ECOLEAD ICT-I Reference Framework*, which is an instance of the Reference Architecture. At this stage those functionalities are already viewed as *services*, but they are still at an abstract level as they are independent of implementation technologies and computing platforms. This Reference Framework is depicted in section 2.

The second stage of the derivation is the *ECOLEAD ICT-I Framework*, which instantiates those reference services to particular ICTs; in this case to web-services technology. Although this particularization, the specification of the services are made in a complete but also abstract way, using the UML standard, which in turn makes the specification of the ICT-I Framework still platform independent. Even though, particular collaborative business ICT infrastructures can be implemented using different web-services-related technologies depending on the envisaged CNOs to support.

The third and last stage of such derivation is the implementation of the ICT-I's services themselves. In the ECOLEAD project, a set of web-services has been used and a subset of them - from the ICT-I Reference Framework - has been effectively implemented. Therefore, these services represent the ECOLEAD ICT-I itself and they are detailed in section 3 of this book chapter. The implementation of the ICT-I services tried to be the most generic as possible although ECOLEAD has focused on VBEs, VOs and PVCs.

Section 4 concludes this book chapter presenting a general assessment of the developed *ECOLEAD plug & play collaborative business ICT infrastructure*.

## 2. ICT-I REFERENCE FRAMEWORK

As clarified in the previous section, the reference framework is a possible derivation from the reference architecture. Therefore, the list of particular derived services reflects a view (including the services' names) about the CNO needs in terms of a supporting collaborative infrastructure. In the CNO context, this view means an infrastructure that endows *services* to enable *people* to collaborate and negotiate, *systems* to interoperate and adapt, *knowledge* and resources to be discovered and shared, and *processes* to be interconnected and synchronized. Figure 1 shows this particular view of the ICT-I Reference Framework, which tried to be the as wide as possible to cover the most different types of CNO manifestations as possible.

Looking at the ICT-I Reference Architecture (stressed in the previous book chapter), this figure 1 shows an “explosion” of the Reference Architecture’s boxes. For instance, “human collaboration services” in the reference architecture are here seen as a CSCW issue, which in turn could offer groupware, officeware and product development supporting services. As it was said before, this explosion reflects one vision in terms of needs for human collaboration. Other visions can exist for that.

For the ECOLEAD project, only some of these services have been implemented, either from scratch or from adaptations of existing open source tools (see section 3). Services in grey scale represent the developed services, whereas the black ones are supported only at a basic level. Services in white represent those that were not implemented but that can be easily added to the framework in the future.

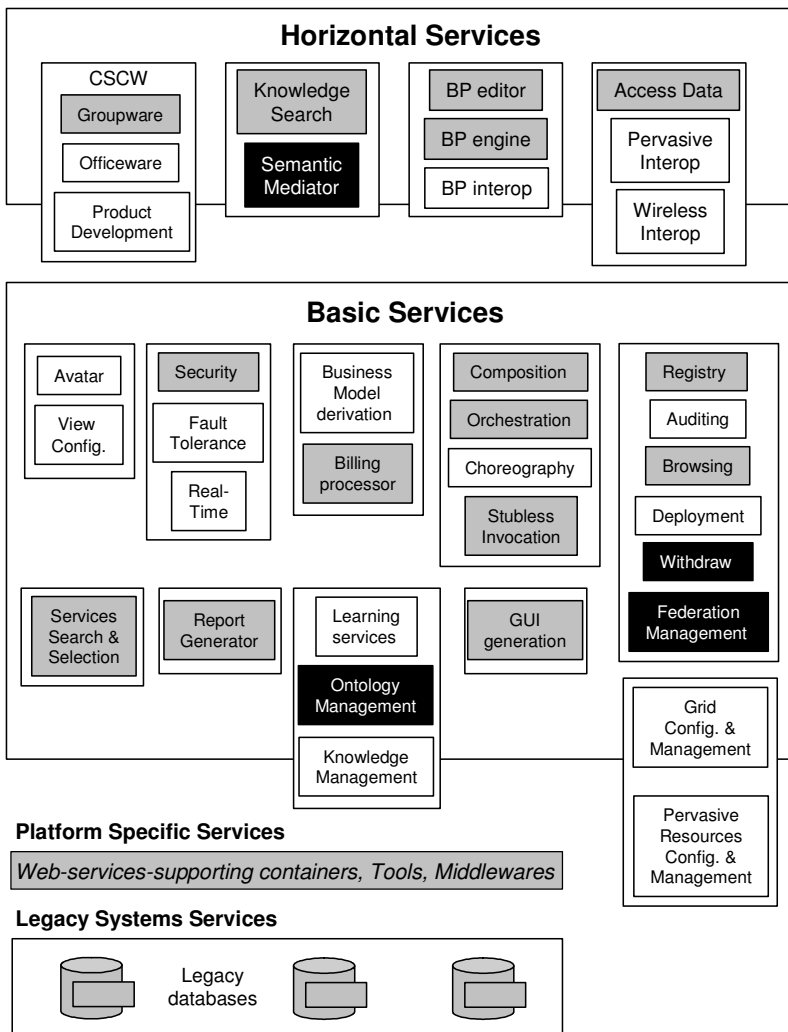


Figure 1 – ICT-I Reference Framework

In the following sections all services are briefly described. Their description is very general as their detailed scope, functionalities and behaviour can only be specified when they are derived for particular CNOs.

## 2.1 Horizontal Services

The following list corresponds to the horizontal services considered in the ICT-I framework. Horizontal Services can be described as offering general purpose and common services to the business level (i.e. at VOM, VBE and PVC), which represents the high-level CNO-related applications. Although important and useful, horizontal services aren't essential to guarantee the ICT-I functioning. The list below also reflects a vision about each of the presented functionality, i.e. the scope, functionalities and the behavior of each one can vary from derivation to derivation (and further implementation). For instance, in the ECOLEAD view, a groupware service should offer a number of functionalities which were considered useful. In other derivation, such list can be smaller or larger. Yet, the behavior, scope and implementing technologies used to each can vary from implementation to implementation.

- *Groupware*: sometimes referred as CSCW (Computer Supported Cooperative Work) services, they offer means to people to work in group, distributed, collaboratively. They include: *Document Management*, *News*, *Mailing List Management* (integrated with) *Discussion Forum*, *Calendar*, *Wiki*, *Information Syndication*, *Instant Messaging*, *Task List*, *Voice Conference*, and *Notification*.
- *Officeware*: back-office-supporting services (e.g. word processor, draw editor, spreadsheet).
- *Product Development*: supporting services that allow CNO members to develop new products collaboratively as well as to manage them along their development and life cycle.
- *Knowledge search*: service that allows searching for a given existing knowledge in the CNO members, which can also consider semantics.
- *Semantic mediator*: support service that will mediate the differences among diverse ontologies that may exist inside one CNO or in diverse CNOs. This service is strongly connected to the basic service *ontology management* (see below in the Basic services).
- *BP editor*: editor of business processes.
- *BP engine*: supporting engine for the BP editor and BP monitoring.
- *BP interoperability*: support for multiple BP modeling languages and business processes models.
- *Access Data*: service to allow gathering of information from CNO members' databases. In resume, this service represents the ICT-I efforts for dealing with legacy systems, i.e. as the way to access (public) information they provide.
- *Pervasive interoperability*: services responsible for supporting interoperability (platforms and communication protocols) among different/diverse pervasive networks which CNO members can have to deal with along the VO/VT life cycle in order to access data.

- *Wireless interoperability*: services responsible for supporting interoperability among different wireless protocols which CNO applications can have to deal with along the VO/VT life cycle in order to access data.

## 2.2 Basic Services

Basic Services can be described as domain-independent services that are used mainly by other services. These services represent the core of the ICT-I, comprising services as discovery, selection and orchestration of services, and security. The same considerations about different views and derivations mentioned in section 2.1 about horizontal services are valid to the basic services.

- *Avatar*: service that can act on behalf of users in some situations (entire or partial business processes) during their absence or busyness, or even in repetitive activities (Zambiasi and Rabelo, 2007).
- *View configuration*: supporting service through which users can configure the avatar' behaviors and user preferences.
- *Security*: security services. This service comprises authentication, authorizations and accounting (AAA).
- *Fault tolerance*: supporting services that can implement fault tolerance mechanisms to recover from faults in the network also during the execution of some transactions/communications. This is one facet of the wider issue Quality of Service (QoS).
- *Real time*: supporting services that can allow the execution of some services in very well defined / tight time slots. This is another facet of QoS.
- *Business model derivation*: service that can express the type of business model to be applied to the ICT-I services for billing purposes.
- *Billing*: to allow services billing.
- *Composition*: to support web-services composition. Composition can be tackled from different point of views, involving orchestration and choreography, depending on the derivation process. Some standards can be used for that (e.g. BPEL).
- *Stubless invocation*: this service allows seamlessly services invocation without the need of having proxies or other stubs no matter which B2B framework (e.g. WebSphere, WebLogic, BizTalk, .NET, AXIS).
- *Registry*: to publish web-services in given services repositories.
- *Browsing*: to browse registered web-services in given services repositories.
- *Deployment*: to aid developers in the web-services deployment. It is also useful to allow a better / more rational (or even replicated) services allocation over distributed services repositories.
- *Auditing*: to support auditing (from several perspectives, such as quality control, privacy & federation rules, actual amount of accesses versus generated billing) in the services federation.
- *Withdraw*: to support the unregistering of given services from the services repository.
- *Federation Management*: global service to support all the issues related to the management of the services federation, basically comprising services & ICT-I life cycles, operational & security rules, and evolving mechanisms.

- *Search & Selection*: to allow the searching and further selection of registered services that fit a set of specifications, which can comprise more intelligent levels (semantics-driven) of searching and selection.
- *Reporting*: to support the generation of reports, usually for billing purposes.
- *Knowledge management*: to support means to manage the knowledge produced in the CNOs along their life cycle.
- *Ontology Management*: to support the use of multiple ontologies expressed in different ontology tools. This service is an essential base for the horizontal service *Knowledge Search*.
- *Learning*: to support the generation of new knowledge out of what is stored in the CNOs' knowledge base (Loss & al, 2007).
- *GUI generator*: to generate (dynamically and automatically) web-based GUIs, also useful for context awareness when on-the-fly adaptations are required (Gesser, 2006).
- *Grid*: general service to allow deployment of grid platforms (e.g. *Globus*) and to configure them in a way to both support high processing and hardware sharing. This aims at extending the collaboration level among CNO members, usually limited to information sharing (Pinheiro and Rabelo, 2005).
- *Pervasive computing*: general service to help in the management of the interoperation with different/diverse pervasive networks in a way to enable high-level systems to access the information provided by these networks for any kind of purpose.

In any case, all these services (Basic and Horizontal) can both call other services and be called by other services. As such, they are both services consumers and providers.

### 3. ICT-I FRAMEWORK AND IMPLEMENTED SERVICES

This section aims at giving an overview about the implemented ICT-I services. It is important to point out that thanks to the intrinsic flexibility provided by a SOA design and to the derivation processes, ICT-I is not only open, but also a scalable infrastructure. In this way, it is possible to evolve it by providing new services, many different (competing) implementations of the same services, or improving existing ones.

It is also important to highlight that *Interoperability* is not the focus of the ECOLEAD ICT-I. In fact, it is seen as an *enabler* for collaboration. Therefore, particular solutions are restrained only to the essential aspects required to support the services, benefiting from existing results. On the other hand, ICT-I strongly relies on ICT *de facto* standards in order to mitigate interoperability obstacles.

Although not mandatory – considering that a web-service can be implemented in any language and in a diversity of environments – ICT-I's services have been coded in Java, besides using some other basic ICTs, like *XML*, *SOAP*, *WSDL* and *UDDI*, and *AXIS*, *JBoss* and *Jonas* as the main containers.

ICT-I Reference Framework are referred to categories of services that should be implemented applying given technologies. In the current implementation of ICT-I services, horizontal services usually have some direct interaction with their users

(via user interfaces) also with the end-users, whereas most of the basic services are directed invoked by other services. In some cases, there is also some interaction with systems administrators. Other implementations of the same service category may present totally different behaviors, depending on the business rules that should be observed in a given derivation. This means that multiple different implementations of the same category of service may exist and hence co-exist in the Services Federation.

In the next sections the implemented services are more detailed described. As said before, they can be implemented in different ways, with a sort of functionalities and associated different behaviors. That's the reason why their description in the next sections is not homogeneous. The complete UML specification and examples of screenshots of all services can be found in (Ratti and Rabelo, 2007), (Rodrigo-Castro and Ratti, 2007) and (Sowa & al., 2007). The security framework, which empowers ICT-I with a very flexible environment allowing dynamic and controlled assignment of access right, s is specifically and detailed presented in another chapter of this book.

### 3.1 On-demand collaboration services

The on-demand collaboration services represent an integrated suite of the collaboration tools that can be used for mutual communication and information interchange between people inside the CNO. The following collaboration functionalities are included in the implemented version:

- *Instant Messaging / Chat*
- *Discussion Forum*
- *Mailing List Management*
- *Calendar*
- *Wiki*
- *Document Repository / Content Management System*
- *News & Announcement System*

*Liferay Portal Server* was used as the platform for integrating different collaboration tools. The graphical user interface for all collaboration services is implemented as *portlets* following the *JSR-168 Java Portlet* specification. Their usage enables flexibility and makes it possible to customize the user interface to match particular needs of the different distinct user groups.

Through the advanced integration options, the basic collaboration functionalities are further augmented with knowledge management and data reuse features. The system keeps history records of the ongoing communication and integrates with *Knowledge Search* service to allow finding information easily.

In contrast to other ICT-I basic and horizontal services, the collaboration services are GUI-intensive, meaning that the core part of implemented functionality focuses on the interaction with user through the graphical user interface. Thus, the exposition of the collaboration functionality through the Web Services interface is not always possible / reasonable. The functionality is exposed to other horizontal and vertical services through the Web Services and Java API interfaces.

#### *Architecture*

Figure 2 shows the basic architecture of the on-demand collaboration services. It

follows the 3-tier model, which separates the presentation layer, the application logic layer, and data storage layer.

The application logic layer consists of two main parts. One part is represented by the application logic objects for the collaboration services implemented by Liferay.

Components called *connectors* are used to integrate the application logic with Liferay and other external collaboration tools (*Wildfire* for instant messaging and *Asterisk* for voice conferencing). The connectors accommodate interfaces of particular collaboration tools to the needs of the Integration and Adaptation layers. The dependency of the CSCW service related to a particular tool implementation shall be minimized so that the tool can be replaced by other implementation in the future. *Spring* framework is employed as a bean container that is responsible for application objects initialization and their binding. It is based on *IoC* (inversion of control) / dependency injection approach, which promotes decoupling and better reuse of the application components.

The presentation and application logic layers are deployed within the servlet container (usually *Apache Tomcat*) in the form of web applications.

*PostgreSQL* database is used as the data storage. *Hibernate* is used as ORM (object-relational mapping) tool so that the application can access the database in an object oriented way instead of using SQL statements directly.

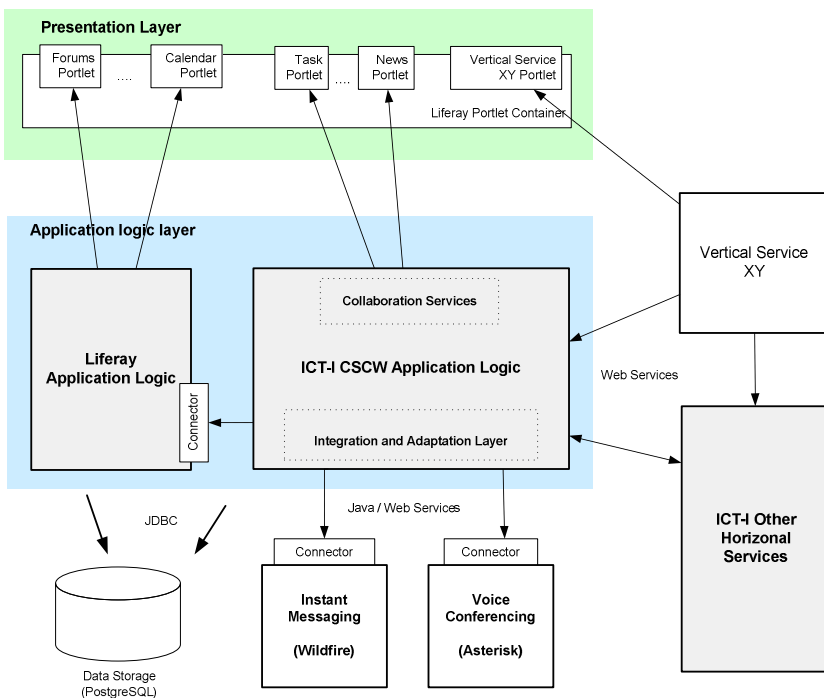


Figure 2 - CSCW services Architecture

Figure 3 shows an example of multi-user chat rooms that can be used for multilateral discussions about specific topics. The chat rooms can be persisted so that a



discussion about particular topic can be realized in several sessions over a longer period of time.

### 3.2 Knowledge search service

*Knowledge Search* (K. Search) corresponds to a general purpose service that provides support for searching knowledge (and information) available in CNOs in a more precise way (Tramontin-Jr and Rabelo, 2007). This knowledge is the one that is exchanged among CNO partners when they use collaborative tools (forum, chat, wiki, e-mail, files, etc. [see previous section]) and that are further stored in distributed information sources. The knowledge gets shareable only after the user explicitly set it as public.

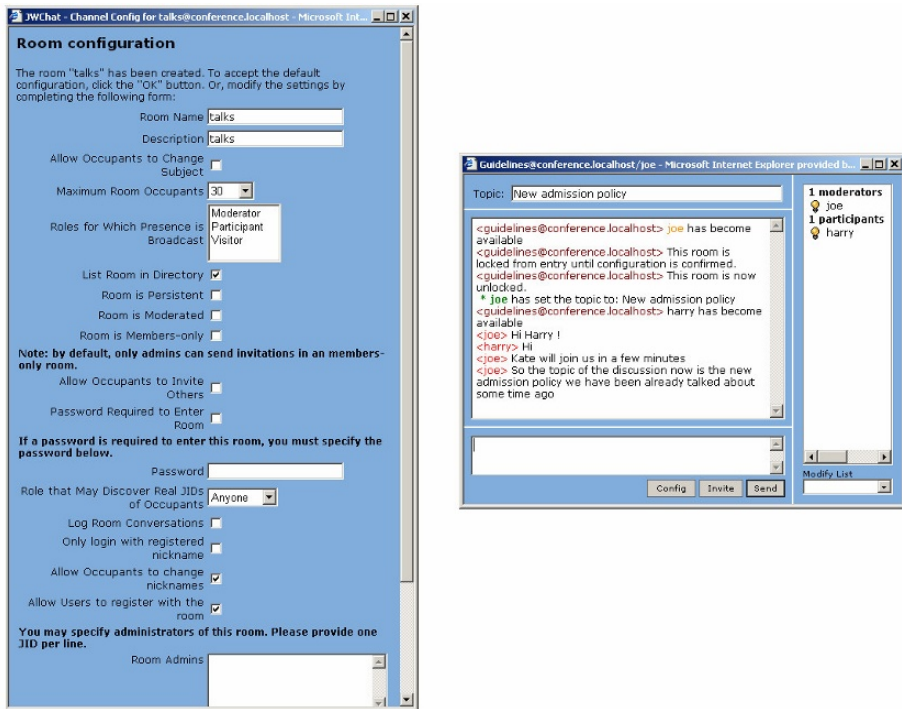


Figure 3 - Chat room configuration and chat room windows

K. search services provide four main functionalities:

1. **Ontology management:** it is considered a preparatory (off-line) stage for the other functionalities. It should be used by domain experts to build and manage ontologies for representing knowledge in a given CNO. As the process of automatic annotation depends on the instances of the ontologies, this functionality can also be used by CNO-related applications to add or update instance values related to CNO members, like profile information. This functionality is divided into two sub-functionalities: *ontology browsing* and *ontology edition*. While the first one allows users to go through current

ontologies, the edition provides means to add, change or delete elements of the ontology.

2. Document Indexing: system indexes its documents that can be further searched more quickly. Semantic annotations are generated transparently and automatically considering the ontologies used.
3. Simple Search: CNO partners can define semantic queries according to the ontologies adopted by the respective CNO. The results will be related to the contents that have been previously indexed.
4. Federated Search: users taking part in several CNOs perform federated searches by choosing the *context* of the search (among the CNOs they are involved in), defining the semantic query, and then performing the search itself. All the required translations are made transparently according to the ontologies of each involved CNO, and the search results are translated back to the ontologies used in the original query.

### Architecture

The K. Search services' implementing architecture is presented in

Figure . K. Search services can be accessed either by client applications (vertical services) or directly by end-users, and they are implemented on top of KIM Platform<sup>1</sup>. KIM covers some of the required features specified for the K. Search functionalities: *automatic semantic annotation*, *semantic indexing* and *retrieval of content*, and query and *navigation over the formal knowledge* (ontology and instances). The architecture is composed of the following components:

1. The services themselves (K. search services);
2. A Portlet for general purpose searches;
3. KIM Platform;
4. Client (CNO) applications that access K. Search.

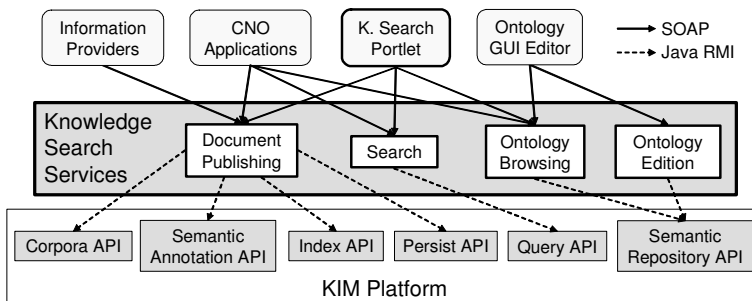


Figure 4 - K. Search architecture

- **Document Manager:** it implements the functionality *Document Indexing*. This service provides basic CRUD<sup>2</sup> operations on documents using *KIM Corpora* and *Persist* APIs. When documents are created / updated, they are semantically

<sup>1</sup> KIM is composed of APIs (Java RMI), some user interfaces, a general purpose ontology and a knowledge base (<http://ontotext.com/kim/>)

<sup>2</sup> Create, read, update and delete.

annotated and indexed using KIM Semantic Annotation and Index APIs, respectively. When a document is published, the security API is invoked to obtain the user’s context (i.e. the CNO he is involved in). This information is transparently attached to the document, which will be further used for filtering during searches.

- **Search Engine:** it implements the *Simple Search* functionality, using KIM’s Query API. It provides four operations:
  - Search for entities (instances of the ontology);
  - Search for documents containing semantic annotations related to a given entity;
  - Search for documents based on semantic queries;
  - Search for documents using keywords (like a traditional search).
- **Ontology Browser:** It is implemented on top of KIM’s Semantic Repository API.
- **Ontology Editor.** Implemented on top of KIM’s Semantic Repository API. The K. Search Portlet is developed as an ordinary JSR-168 portlet.

Figure shows an example of a search, highlighting the correlated terms found out via the (previously defined) ontology.

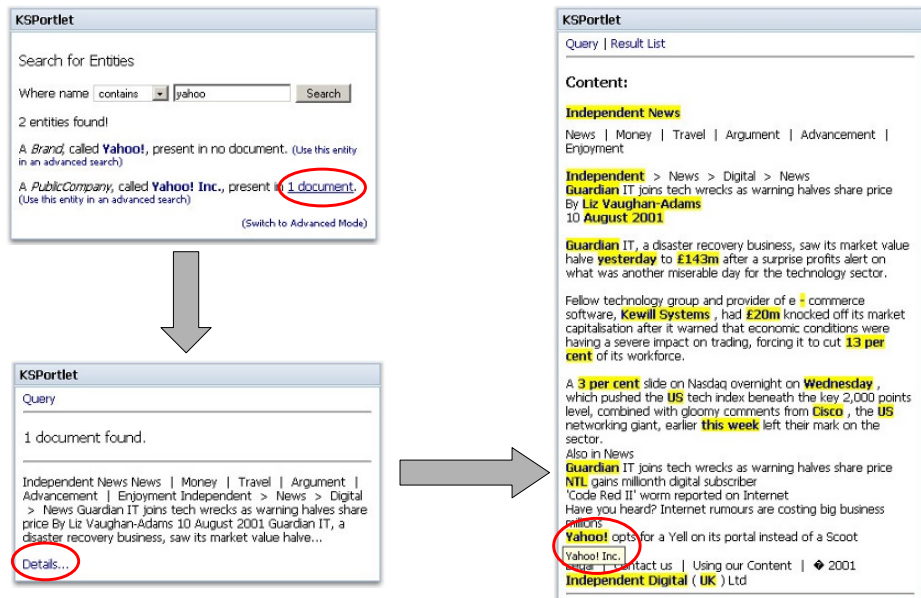


Figure 5: Simplified Mode of the K. Search portlet

### 3.3 Business Process management service

The goal of this horizontal service is the provision of an advanced user-centered /

*interactive Business Process Management* service (iBPM). The overall idea behind this service is to identify needs for complex business processes management, which may require human interactions (Ratti and Gusmeroli, 2007).

The current state of the art does not provide any tools or standard language able to manage the two separate domains: human and automatic. Currently, what exists is a support for human based activities, supported by the concept of workflow, and the automatic execution activities, namely business processes. Two different languages are widely adopted and recognized as standards: xPDL, for workflow management, and BPEL, for business process execution and orchestration. What is missing is a combination of the two languages, in order to allow the next generation processes, which involves both human and automatic activities. The new language, called CBP (Collaborative Business Process) comprises the two above paradigms.

The analysis performed has highlighted the need of maintaining the semantic and the structure of the two languages as much as possible. For this reason, one language has been chosen as basic (and reference) language, complemented with the tags of the other. So it has been decided to use xPDL as basic language, improved by BPEL constructs which are needed for managing the new approach.

Some semantic elements are still the same in the new schema: for instance, transactions, process and basic xPDL activities with BPEL ones. The inherited tags are not changed through the porting process from one language to the other. Figure 6 shows the “fusion” of the two languages.

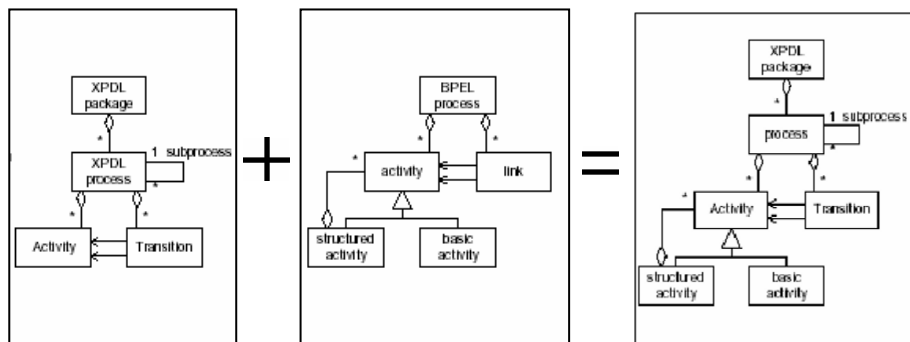


Figure 6 - BPEL and xPDL integration

In order to support these services, the iBPM provides two main tools: an editor and an engine. The former is required for modeling the CBP file, while the latter will run instances of such processes. The work performed in this activity has been focused on the extension of existing tools (for editing and executing processes) in order to be able to manage the new standard. For this reason it was decided to use existing tools able to support human interactions and workflows, and then to build upon them the supporting features for managing BPEL processes.

The selected solutions are two of the mostly used workflow editor and engine: *Jawe*, the editor, and *Shark*, the engine. They are Java client-based applications, which are able to store information inside XML files and to save local data inside any relational database management system. They have been extended using Java as default language in order to save file in CBP and to import BPEL processes.

*Architecture*

The architecture of the module is very simple. The work performed can be seen as a plug-in of the existing modules for loading BPEL processes and save / load CBP file. In order to provide a more comprehensive usage of the engine, a completely web based module is available for users. This has also been modified during developments stage. The editor will export CBP file, which can be loaded by the engine. Then the external application shall invoke such deployed processes in order to manage the execution of CBP file. Nevertheless, the original BPEL processes will still run under their engine. Figure 7 shows an example of how processes are edited.

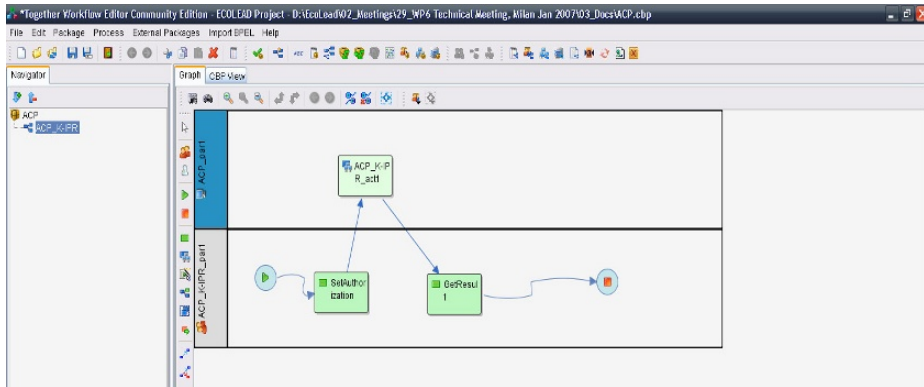


Figure 7 - iBPM editor

**3.4 Data Access service**

CNO data access facilitates the access to different information sources and databases provided by the different CNO members. It is a software layer, which offers a web service interface for accessing the public data from distinct data sources. It accesses the physical data sources and transforms the physical data into a model that different organizations can understand, providing data abstraction for the VO members. This process is carried out through the steps described below.

First of all, the identification of the domain model that contains the data to be shared between the different organizations should be done. This domain model is known by the organization members and includes the structure and semantic of the information. Data transformation, from the specific data structure of the members to the domain model, is done by the data access middleware.

Interoperability between organizations is facilitated by the use of this service in the data interchange. The incorporation of new members or new services is also made easier, as the meaning and structure of the data is already known. Any data change at low level is transparent to the others ICT-I or client services. Therefore, this data layer provides loose coupling between the services and the underlying databases or information sources.

One very relevant type of data source case is the (relational) databases that belong to the CNO members. A direct access to the companies' database is usually

not possible but this can be achieved – once agreed – with the Data Access web service interface. The middleware accesses the given database and transforms the data to XML, according to the pre-defined model. By using this service, organizations are provided with access to the VO members' legacy databases by using the standard http protocol and SOAP messages.

### Architecture

There are two different stages for this service: *Authoring* stage and *Run-time* stage. In the first one, it is necessary to analyze the information that is going to be accessed, to understand its meaning and to define its structure (the so-called *domain model*). Afterwards, it is necessary to define the links and meta-information required for relating the existing and physical data with the domain model.

Once deployed in the server, the middleware and the previously defined configuration files, the Data Access service can be used as any other web service. In order to use this service, it is however necessary an adaptation to concrete cases. This means the necessity of analyzing the data to be shared by the different members and of defining the common data structure (domain model) in XML, which corresponds to the authoring stage. The correspondences between the domain model and the existing data sources have to be established afterwards. All these information are stored in configuration files to be further used in the *run-time stage*, where information can indeed be accessed seamlessly. In general, this reveals the strategy adopted to implement the EAI (Enterprise Application Integration) approach. These processes can be grouped in the following steps, as illustrated in Figure 8:

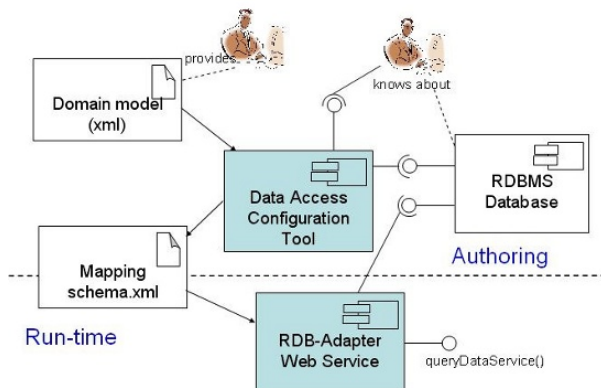


Figure 8 - Data Access services

Two roles can be identified. The first one corresponds to a business expert, who knows about the domain being modeled. The other role corresponds to people with knowledge on the concrete implementation of the information in the data bases. Both roles are necessary to configure this service via the Configuration Tool.

1. Identification and definition of the model that represents the information to be shared by the different members, in XML schemas.
2. Configuration of the adapters to be used. It is necessary to identify the relations between the domain model and each concrete data source. Concrete

parameters for each data source are provided and associated to the suitable adapter. Three types of adapters have been designed: web service adapter, XML files adapter and Relational Database (RDBMS) adapter. The most complex is the last one, and in order to assist the developer in this process a tool is provided. It is a stand-alone tool (*Data Access Configuration Tool*) where the mapping between the DBMS and the “Domain model schema” can be done relatively easily.

3. Deployment of the middleware and the configuration files in the corresponding web server. This web server should have access to the data source for which it is configured, and it will offer a Web Service interface.

Figure 9 illustrates a step during the mapping phase, indicating the correspondence between the domain reference model and a given data base model.

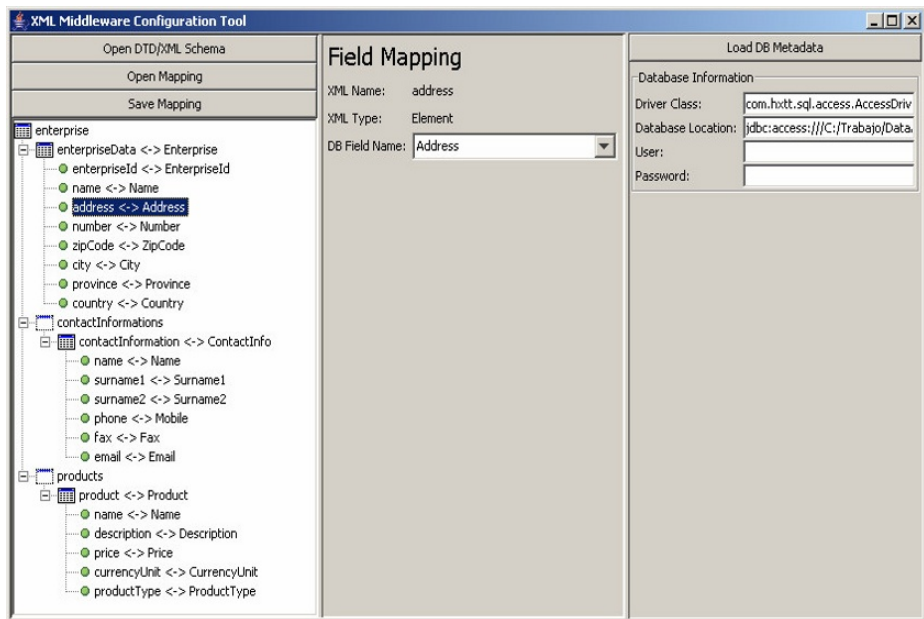


Figure 9 - XML middleware configuration Tool

### 3.5 Registry and Discovery services

ICT-I Registry and Discovery services provide support for working with the ECOLEAD repository of Web Services. ECOLEAD provides a UDDI v2 (OASIS standard<sup>3</sup>) server in order to register all the developed web services as well as to be able to find and use them in an easy and standard way. The core ICT-I Registry and Discovery includes the following components:

- UDDI Repository: stores the web services’ meta information and related information necessary for its characterization;

<sup>3</sup> <http://www.uddi.org/>



- ICT-I Services Registry: supports registration of the web services in the UDDI server;
- ICT-I Services Discovery: supports the web services finding process.

These services are categorized in two profiles (Figure 10):

- for services and applications, it is possible to use these functionalities by the use of the corresponding application programming interfaces (APIs), which are included in the UDDI specification provided by OASIS.
- for end-users (developers), it also provides two portlets in a portal for making accessible these web services to users that need to know which services are available.

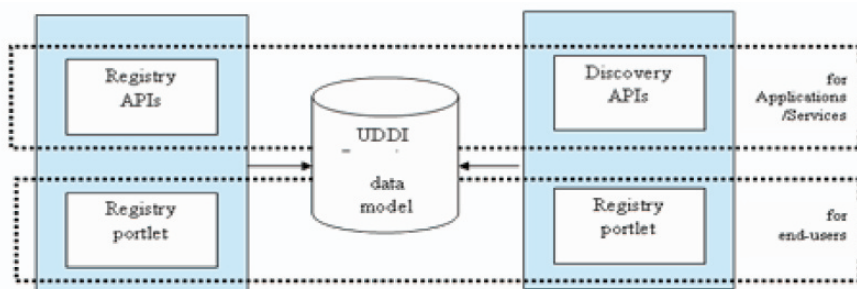


Figure 10 - UDDI Registry & Discovery services

This UDDI server supports a publicly accessible Universal Business Registry. *Tamino XML Server v.4.2.1*, a native XML database from Software AG Company, is used in the UDDI server implementation.

A functionality has been developed to allow end-users (mainly developers) to easily work with UDDI services and facilities. Two portlets have been developed according to the JSR-168 standards: the first one provides browsing facilities (Figure 11), and the second one provides the registration of services in the repository, according to a pre-defined classification. So far, there are not any smart mechanisms and performance criteria for services discovery and selection. On the other hand, this classification can be taken into account for that in the future.

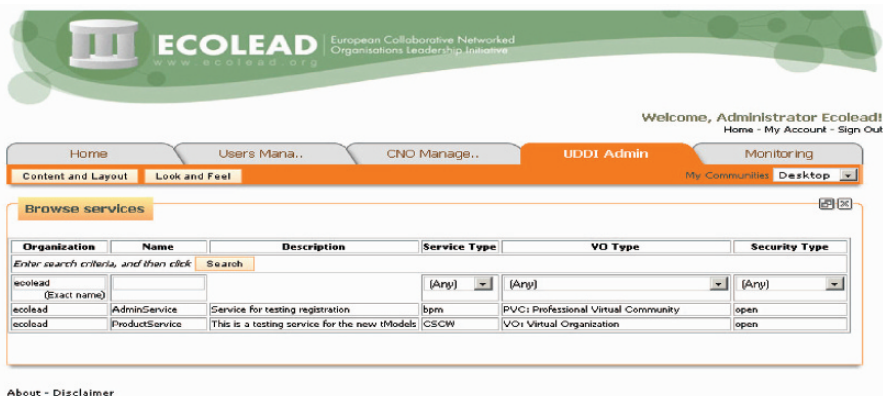


Figure 11 - UDDI Browsing service



### 3.6 Stubless Invocation

The purpose of a stubless invocation is to allow client applications (e.g. an organization) to effectively use any service of the Services Federation seamlessly, whatever the technology / environment services had been developed and deployed. This goes to the direction of extending the collaboration level among CNO members in a way they can also *share services*. So far there was a hard restriction as current implementations of the web-services specifications by different vendors were not fully compatible hence creating interoperability problems when one tried to invoke another one. Thanks the use of the Apache WSFI (*WS Framework Invocation*) standard specification and its implementation, plus a layer that prevents programmers / client-applications from knowing implementation details and creating proxies manually, the implementation done in the ICT-I makes this automatically and dynamically (Piazza and Rabelo, 2007).

#### Architecture

Inside this general framework, an entity can be a service provider or consumer. Actually, stubless is not a web service itself. From the consumer view point, Stubless process acts as a *layer* in the form of an API for services invocation support, making the differences among different implementations totally transparent. Using this API, different CNO members can share different services regarding their particular business processes' logics, and without changing the standard way of working with services (registry, publish and invoke). Figure 12 presents the conceptual architecture of the Stubless invocation process.

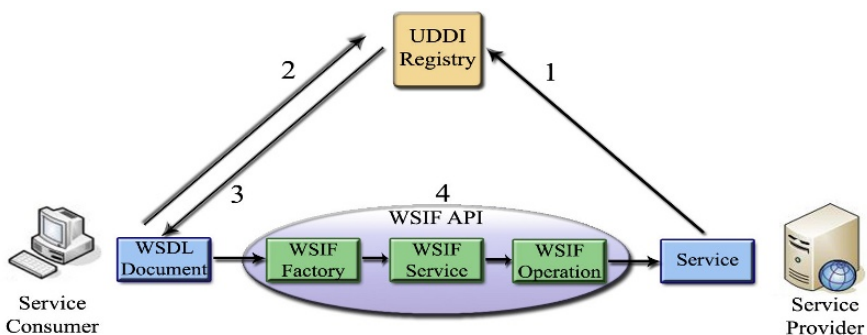


Figure 12 - Stubless Architecture

A stubless invocation is characterized by the following steps:

#### Service Provider Side

1 – Publish a service with the respective WSDL address endpoint

#### Service Consumer/Client Side (Stubless invocation)

2 – Search for a service on a registry

3 – Load the WSDL description of the service

4.1 – Create a WSIF factory for the service

4.2 – Create the WSIF service based on the WSDL document

4.3 – Use the WSIF service to get the operation to be invoked

4.4 – Create the message and invoke the service based on a particular binding. The invocation is handled by the WSIF Provider that is an implementation of a WSDL binding that can run a WSDL operation through a binding-specific protocol (like SOAP, EJB, JMS, JCA and Local Java).

### 3.7 Billing services

This service operationalizes the pay-per-use model as long as given web-services are accessed/used within previously agreed business models. This service is fed with data from the *Audit Handler* service (*Interceptor* sending *UsageRecords*). The bill is based on a period of Web Services Usage from WSC entities (i.e. Customers). The billing is performed in batch time, with support for simple business models (Data volume and Usage Time). Billing summary reports are automatically generated in HTML thanks to the Reporting Service (see next section). Figure 13 illustrates a bill generated under these two models.


Ecolead Billing Summary Report (Per company, per volume)	
	
Company: FT Billing details ( <i>Volume</i> ) for period: 01/02/2006 to /01/03/2006	
Bytes Sent from ICT Vertical Web Services	302300
Bytes Received by ICT Vertical Web Services	48908
# of Request/Response pairs (connections)	5038
Fixed cost	300 €
Global connection cost	500 €
Total volume cost	1000 €
Total costs	1800 €
TAX (5%)	90 €
Total	1890 €

Figure 13a – Volume oriented


Ecolead Billing Summary Report (Per company, per usage time)	
	
Company: FT Billing details ( <i>Usage</i> ) for period: 01/02/2006 to /01/03/2006	
Usage time ( Request/Response CPU time)	0:30:10
# of Request/Response pairs (connections)	340
Fixed cost	150 €
Global usage time cost	50 €
Total usage cost	200 €
TAX (5%)	10 €
Total	210 €

Figure 13b – Usage Time oriented

The main operations performed by the Billing web service interface are the storing Request/Responses pairs sent by the Audit Handler Interceptor, and the transformation of this information in start-stop records according to CDR format handled by *Radius* aware tools. The CDR Radius format is adapted with new attributes and new start-stop records are generated in the Radius log files.

### 3.8 Reporting services

The Reporting Service delivers a Report to a WS Client entity. The report is can be generated in some standard formats and based on predefined templates ('Detailed Billing Usage' or 'Usage Billing Summary'). The Reporting Service mainly offers an operation that computes a bill and delivers it in the HTML format.

### 3.9 Interception Handler

The scope of this service is to allow interception of messages at SOAP level without

disturbing or being intrusive in the web service chain (WS Client <-> intermediates <-> WS Provider). Figure 14 presents a Use Case diagram that explains the high level functions of this Service:

- Intercept and store temporarily Requests, Responses and Fault Messages that flow between a WSC and a WSP;
- Send bunches of SOAP messages to a Recording Service offering a function of storing and converting to appropriate format to do accounting and billing.

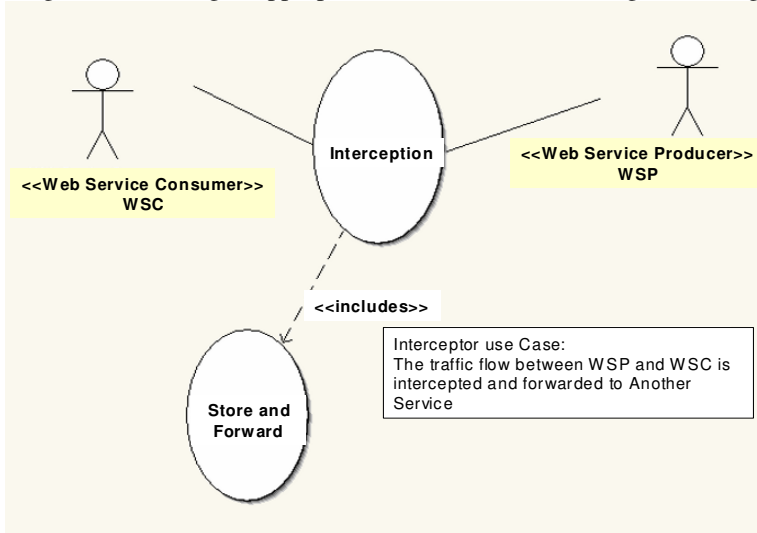


Figure 14 – Use Case Handler service

Interceptor is mainly built on top of the two following objects:

- The *Usage Record* object contains information relative to the SOAP message intercepted,
- The AuditHandler Object (interceptor), instantiated and deployed in the runtime infrastructure where the WSPs are deployed.

The number of items sent each time is a tradeoff between performance penalty (by sending one by one) and accurate/just in time information to the recipient (by sending a too big bunch of items). Handler Interceptor objects can be customizable at launch time or dynamically, at runtime.

### 3.10 Services composition

This service allows defining how two or more web services can interact. Services composition is a critical part of making web services effective. By using this service, it is possible to set a business process as a flow of executable processes, and to launch this business process by using an engine that interprets it by means of the web services' interfaces. BPEL is the business process language and it is a standard *de facto* for that.

This service is executed at two phases / using two modules: authoring (using the BPEL designer) and runtime (using the BPEL engine). The BPEL designer supports

the process of writing how a given set of web services is going to be connected, specifying the conditions of the flow and defining the inputs and the outputs, generating a “BPEL sequence” afterwards. The BPEL engine is the module that interprets and executes the “BPEL sequence”. In the current implementation, an extension was added-on for ActiveBPEL engine, which is deployed and installed inside the engine in order to be transparent to the end-user. This functionality consists on the use of the UDDI repository for searching web services and for getting the URL to access it dynamically. Figure 15 illustrates this process.

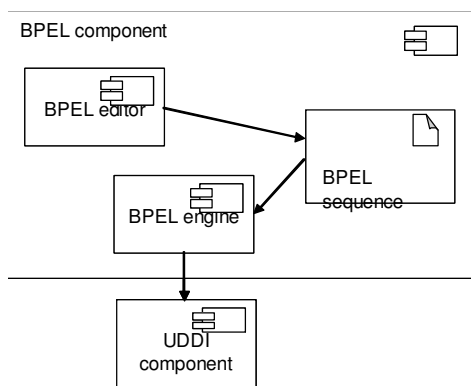


Figure 15 - Services composition Component diagram

A BPEL Editor is needed for the authoring stage. There are different products in the market and Active Endpoints product has been the one used. They have an ActiveBPEL engine (associated to ActiveBPEL™ Designer<sup>4</sup>) that is open-source. The designer is an integrated visual toolset for rapidly creating, testing and deploying composite applications based on the BPEL v1.1 standard. At the runtime stage, then a BPEL engine is required, which is in charge of interpreting and executing the BPEL sequence. To this engine, one module that allows the activation of the searching functionality in a UDDI server has been integrated. During the BPEL sequence execution, the engine checks if the Web Service used in the sequence has been registered in the UDDI server. If it is there, then it reads from the UDDI server the proper URL. This process is totally transparent to the end-user and the system administrator can easily configure this. Figure 16 shows an example of the interface to define a composition.

## 4. CONCLUSIONS

This paper has presented the Collaborative Business ICT infrastructure (ICT-I) that was developed in the ECOLEAD Project for supporting CNOs in collaborating and doing businesses more effectively. It has been conceived based on the service

<sup>4</sup> <http://www.active-endpoints.com/products/activebpeldes/index.html>

oriented architecture paradigm / web-services technology, providing organizations with a transparent (mostly), platform-independent, easy-to-use, secure-embedded, lean, distributed, scalar, on-demand and pay-per-use ICT-I.

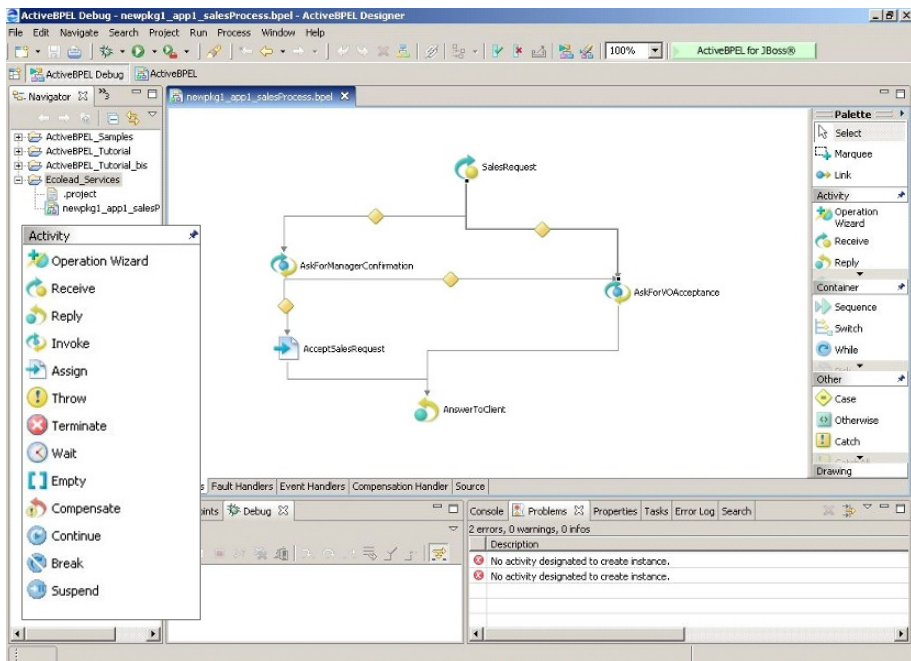


Figure 16 - Services composition editor interface

About the *plug & play* and *transparency* ICT-I features, they are materialized in the following general form. Considering the intrinsic current limitations of existing technologies, *Plug & Play* is concretized in the sense that users can access ICT-I services just using a browser and doing a small configuration of some security issues in the first time they use it. This means that neither local deployments nor intensive configuration operations are required. In the same way, other applications and services can execute ICT-I services just by invoking them. *Transparency* is also supported in the sense that both users and client-applications/services don't need to know anything about ICT-I, i.e. ICT-I services can be discovered, composed and used/executed no matter where they are, how they are and which technology has been used in their implementation, so this is transparent for clients.

It is important to point out the capability of ICT-I to be integrated with B2B functionalities, which are nowadays also getting web-services-based. This means the possibility of having a “complete” and global environment where B2B transactions-based and collaborative-based processes can be glued transparently.

The derivation process allows the instantiation of particular ICT-Is for envisaged CNOs. The services that have been described correspond to the derivation carried out in the ECOLEAD project, and it comprised the requirements of three types of CNOs: Virtual organization Breeding Environments (VBE), Virtual Organizations

(VO) and Professional Virtual Communities (PVC). The implemented services reflect a functional vision upon the Reference Architecture classes of services. As such, different implementations and functionalities can be developed in different derivations and/or extensions of existing services. This task can be facilitated as developers can take the UML specifications of each service presented in the Reference Framework and can specialize and implement them as they wish. In this respect it is worth to note that the value of ICT-I goes beyond the provided services; the reference architecture itself, in fact, is an important asset that makes the development and integration of new services easier, more robust and globally coherent.

The implemented services are actually at prototype level and have been only validated in the scope of ECOLEAD project. This means that services present some limitations, most of that due to their intrinsic complexity. In fact, there are several open and challenging problems around web-services technology and the developed services, and their solving were not the central focus of this work.

From the technological point of view, the chosen web-based technologies have proven to be a good choice in terms of integration and interoperation usual complexities. Regardless the fact that the development environment was settled in advance and hence most of the ICT-I developers have used the same one, the integration of the services (vertical, horizontal and basic) was carried out relatively smoothly. Once all services' interfaces, parameters, UDDI, etc., were properly defined (which used to take a reasonable time to agree on) and set up, all invocations – and hence the CNO processes – were executed without problems even having services physically deployed in several countries.

From the conceptual point of view, ICT-I, as a distributed WS-based infrastructure, seems to be a feasible approach. On the other hand, the complexity of developing web-services-based applications has to be pointed out. In fact, the related standards are capable of coping with relevant interoperability problems, so implementations can be concentrated on the services themselves. However, to become a reasonable expert on web-services, it takes time. The involved platforms, containers, development environments, the several concepts around UDDI, WSDL, SOAP, portlets, etc., are very time consuming to learn and require a relatively solid ICT background to implement services in a good and fast way. Another point refers to the identification of limitations in some specific technologies, e.g., the relative rigidity of LifeRay when more flexible GUIs are required.

From the performance point of view, it could be observed that some services took a relatively long time to execute. On one hand, this was also due to the intrinsic complexity of some services and the required processing time needed by some of them. On the other hand, the “deployment map” of the whole ICT-I (including the services repositories and registration repositories) was set up essentially considering the current partners' convenience and facilities as QoS considerations were not hard taken into account. Anyway, ICT-I is flexible enough to be set-up with any deployment map.

ICT-I will have its development continued. A number of new services and improvements in some already developed is on the way, also trying to take advantage of plenty of outcomes currently being researched and provided by other projects and initiatives.

## **Acknowledgments**

This work was mostly supported by the European Commission under the project IST FP-6 IP ECOLEAD project ([www.ecolead.org](http://www.ecolead.org)). The Brazilian participation was also supported by the Brazilian Council for Research and Scientific Development – CNPq ([www.cnpq.br](http://www.cnpq.br)) in the scope of IFM project ([www.ifm.org.br](http://www.ifm.org.br)). Special thanks to Mr. Rui Tramontin and Mr. Carlos Gesser (UFSC), Mr. Philippe Gibert (France Telecom), Mr. Roberto Ratti (TXT), and Mr. Walter Woelfel and Mr. Stanislaw Mores (Siemens Austria) for their collaboration in the conception and implementation of the ICT-I services.

## **5. REFERENCES**

- Borst, I.; Arana, C.; Crave, S.; Galeano, N. (2005). Technical Report (Deliverable) D62.2 ICT-I Business Models, in [www.ecolead.org](http://www.ecolead.org).
- Gesser, C. E. (2006). An Approach for the Dynamic Integration of Web Services in Web Portals. MSc Thesis, Federal University of Santa Catarina, Brazil, in [www.gsigma.ufsc.br](http://www.gsigma.ufsc.br).
- Loss, L.; Pereira-Klen, A.; Rabelo, R. J. (2007). Towards Learning Collaborative Networked Organizations. Proceedings PRO-VE'2007 - 8th IFIP Working Conference on Virtual Enterprises, in *Establishing the Foundation of Collaborative Networks*, Springer, pp. 243-252.
- Piazza, A. and Rabelo, R.J. (2007). An Approach for a Seamless Interoperability among Heterogeneous Web-Services Platforms for Collaborative Networked Organizations, MSc Thesis, Federal University of Santa Catarina, Brazil, in [www.gsigma.ufsc.br](http://www.gsigma.ufsc.br).
- Pinheiro, F. R.; Rabelo, R. J. (2005). Experiments on Grid Computing for VE-related Applications. Proceedings PRO-VE'2005 - 6th IFIP Working Conference on Infrastructures for Virtual Enterprises, in *Collaborative Networks and their Breeding Environments*, Springer, pp. 483-492.
- Rabelo, R. J.; Gusmeroli, S.; Arana, C.; Nagellen, T (2006). The ECOLEAD ICT Infrastructure for Collaborative Networked Organizations. Proceedings 7th IFIP International Working Conference on Virtual Enterprises, in *Collaborative Networked Organizations and the Services Oriented Economy*, Springer, pp. 451-460.
- Rabelo, R. J.; Gusmeroli, Sergio (2007). A Service-Oriented Platform for Collaborative Networked Organizations. Proceedings 8th IFAC Symposium on Low Cost Automation, 2007, Havana, Cuba.
- Rabelo, R.; Nagellen, T.; Arana, C. (2005). Technical Report Deliverable D61.1a (v2) – Reference Framework for a Collaborative support ICT infrastructure, in [www.ecolead.org](http://www.ecolead.org).
- Ratti, R.; Gusmeroli, S. (2007). Interactive User-Centered Business Process Management Services, in Proceedings PRO-VE'2007 - 8th IFIP Working Conference on Virtual Enterprises, in *Establishing the Foundation of Collaborative Networks*, Springer, pp. 487-494.
- Ratti, R., Rabelo, R.J. (2007). Technical Report Deliverable D61.1c – ICT-I Reference Framework, in [www.ecolead.org](http://www.ecolead.org).
- Rodrigo, M.; Arana, C.; Rabelo, R. (2005). Technical Report Deliverable D61.3a – First Prototype ICT-I Infrastructure for collaboration, in [www.ecolead.org](http://www.ecolead.org).
- Rodrigo, M.; Ratti, R. (2007). Technical Report Deliverable D64.1d – ICT-I Integrated Prototype, in [www.ecolead.org](http://www.ecolead.org).
- Rodrigo, M.; Ratti, R. (2007). Technical Report Deliverable D64.1c – ICT-I Integrated

- Prototype, in [www.ecolead.org](http://www.ecolead.org).
- Sowa, G., Śnieżyński, T. and Wolański, M. (2007). Technical Report Deliverable D61.4b - Security Framework and Architecture, in [www.ecolead.org](http://www.ecolead.org).
- Tramontin-Jr., R.; Rabelo, R. J. (2007). A Knowledge Search Framework for Collaborative Networks. Proceedings PRO-VE'2007 - 8th IFIP Working Conference on Virtual Enterprises, in Establishing the Foundation of Collaborative Networks, Springer, pp. 573-582.
- Zambiasi, Saulo P.; Rabelo, R. J. (2007). Virtualization of Collaborators in the Manufacturing: a Model based on Agent Bots, Proceedings VIII SBAI - The Brazilian Symposium on Intelligent Automation [in Portuguese].