Talea: an Open Source framework for e-business integration

Susanna Longo[1], Elena Cigliano[1], Andrea Vagliengo[1], Guido Levi[1]

[1] CSP Innovazione nelle ICT s.c.a r.l. - Torino - Italy

{longo, cigliano, vagliengo, levi}@csp.it

Abstract. *This paper describes Talea, a software platform developed within a Piedmont Region (North-West of Italy) project and released following the OSS paradigm. Within the regional policy where Open Source is a key lever for the spreading of ICT, open standards and knowledge and for the increase of local innovation and competitiveness, Talea represents a concrete step as free software "building block" on which even smaller companies will have the possibility to customize knowledge-intensive services. Aimed at supporting the customer-supplier interaction in an Enterprise Application Integration perspective, Talea can be viewed as a generic matchmaker for e-business, supporting a flexible matching between services provision and request. Talea architecture has been designed in order to facilitate easy customization and multi-device access.*

1.Introduction and project scenario

This paper describes Talea, an Open Source experimental software platform aimed at supporting the development of knowledge-intensive services for B2B integration.

The context in which Talea has been conceived allows to understand the motivations underlying its architecture, functionality, and licensing model. Talea has been released within Diadi 2000 (www.diadi.it), a regional project co-founded within the European Structural Funds framework and aimed at increasing the competitiveness of Piedmont Region. The beneficiaries are strictly limited to small/medium enterprises.

Today over 95% of Piedmont small/medium enterprises are micro-firms with less than 10 employees, traditionally reluctant to the introduction of technological innovation. Nevertheless, their business and their survival in the market depends on networking and collaboration dynamics, which are still usually based on face-to-face interaction. Each actor provides a specific task within a value network and this complex and wide networked business needs just as smart management as large enterprise business.

DIADI 2000 strategy, in line with the regional strategy for the development of a knowledge-based economy, is aimed to progressively move the local entrepreneurial system from the model of value networks into the Digital Ecosystem paradigm, where companies cooperate and are interconnected as to be competitive. In this perspective, Diadi 2000 works on 3 domains: culture, business models and technology.

The Enterprise Application Integration process [Alonso et al. 2004], [Trastour et al. 2003] must start from a change in cultural attitudes. At the same time, small companies have a need for mid-term results and is thus necessary to analyze the

sustainability of an economy based on knowledge-intensive services. Furthermore, some concrete steps are needed to make Opens Source software "building blocks" available in order to allow even smaller ICT developer companies to implement knowledge-intensive services and user companies to experiment the impact of ICT on their business processes.

Therefore Diadi 2000, making Talea Open Source software platform available, intends to motivate enterprises to exploit ICT technologies as real added value to their business. Talea can be viewed as a generic ontology-based brokerage system for e-business. It supports a flexible matching between services and goods provision and request[1]. In line with the above mentioned constrains imposed by Structural Funds allocation and DIADI 2000 objectives, Talea has been released, following the OSS paradigm, for all enterprises who may want to use it as first free "building block" for the development of knowledge-intensive services supporting B2B efficiency in the value chain.

To favour the customization and experimentation of Talea-based services in the real market, DIADI 2000 is 50% financing 8 Pilot Projects. Developed by Piedmont small/medium enterprises, Pilot Projects are aimed at customizing some new knowledge-intensive services dedicated to support collaborative business in the most several domains, such as social and medical, pharmaceutics, tourism, agronomy, small local trade, goods delivery and sustainable transfer, house up-keeping.

For instance, one of the Pilot Project is aimed at increasing the competitiveness of the small local trade in comparison with the large-scale retail trade, through the creation of a strong network of small local trade within a specific neighbourhood. It applies Talea brokerage system for the integration of the 3 key components of the small local trade: the local shops' supply, the delivery of goods, the need for sustainable transfer. The implemented system will in fact also be able to optimize the transport routes, in line with sustainable and environmental development plans.

All Pilot Projects include an experimentation phase with real users and will close on July 2006, as to test and consolidate the new services to be transferred to the real market.

2. What is Talea

Talea is a free "building block" for the development of knowledge-intensive services supporting the increase of efficiency in customer-supplier interaction (Figure 1).

Talea is now available at http://talea.csp.it. Its exploitation requires the customization of the framework to specific domains. In the following we will refer to the person (or the team) who performs such a customization as the *customizer*. The result of the customization is an application that can be exploited by the final user (from now on, simply, *user*) to provide and consume *resources* (products or services). Talea users can be both providers and consumers. Providers are small/medium enterprises, while consumers can be individuals, agencies or enterprises.

---

[1] The current prototype does not support automatic negotiation: when customers reserve products/services, Talea simply notifies the corresponding providers about such a reservation (by e-mail, sms, or fax).

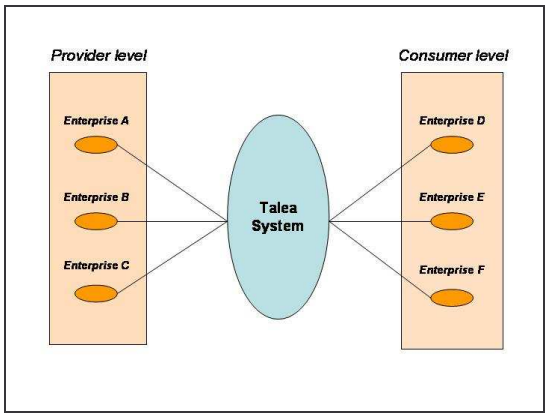

Figure 1. Talea as a mediator in business processes

The design of the Talea framework has been focused on the *customizer*, in order to support customization at three different levels: the presentation layer, the business logic, and the application domain. Such a customization is supported by three main mechanisms:

- an infrastructure that supports the matching between provider and consumer (Talea Backend);
- a customization mechanism that supports the description of the particular application domain (Talea Ontology extension);
- a programming pattern that supports the easy implementation of new features, to extend functionality (Generator/Performer Pattern).

The customization of the presentation layer requires the design of the final user interface, handled by the Frontend and providing multi-device access (see Sect. 3); the customization of the business logic involves the implementation of new features to extend the functionality of the Talea Backend (see Sect. 3), while the customization of the knowledge about the domain is achieved through the extension of the Talea Ontology (see Sect. 4).

3. Architecture

Talea architecture (see Figure 2) was designed in order to facilitate the integration of different business processes, with particular attention to make easier any domain-specific customization. This principle suggested a number of criteria for the design of the system architecture, as follows:

- the split of the system in two main separated parts: a Backend, containing the business logic, and a Frontend, managing the presentation layer;
- the implementation of principles of service-oriented applications (i.e. eXtensible Markup Language (XML [www.w3.org/XML/])-based communication and representation);
- the exploitation of specific design patterns that guarantee a high modularity;
- the use of standard-based XML messages for the communication of modules.

The separation in two independent parts (Backend and Frontend) was introduced to enable a flexible implementation of multi-channel and multi-device access to the system. The current prototype supports two main categories of client: on the one hand, browser-enabled clients, such as Personal Computers, Personal Digital Assistants, eXtensible HyperText Markup Language (XHTML [www.w3.org/MarkUp/])-enabled mobile phones, and so on; on the other hand, browser-less clients, such as stand-alone applications, Multimedia Home Platform (MHP [www.mhp.org/]) clients, and so on. The former can exploit XHTML markup to populate interfaces, while the latter has no browser capabilites and it needs to extract contents directly from XML objects. This is the reason why the Backend core is isolated and the Frontend can act as a "meta-client", manipulating XML content through eXtensible Stylesheet Language (XSL [www.w3.org/Style/XSL/] transformations to generate the final XHTML markup. The *customizer* could implement a new Frontend or modify an existing layout rewriting XSL transformations.

Moreover, in order to guarantee the generality of the approach, Talea core modules do not make any assumption about domain-specific semantics: the business logic embedded within the system modules is totally independent with respect to the specific semantics of resources or users. Besides, one of the main goals of the design was to make the implementation of the high-level operations as more modular as possible. For this reason, there is a class for each high-level operation, with the aim to isolate its management. In particular, the platform modules are based on the *generator/performer* design pattern, which greatly simplifies the task of adding new features.

Each Talea core functionality corresponds to a XML message format, devoted to a specific request/response type; each request type is handled by a specific *Performer* module. The Performer that manages the user request is dynamically selected at run-time, on the basis of the request type.

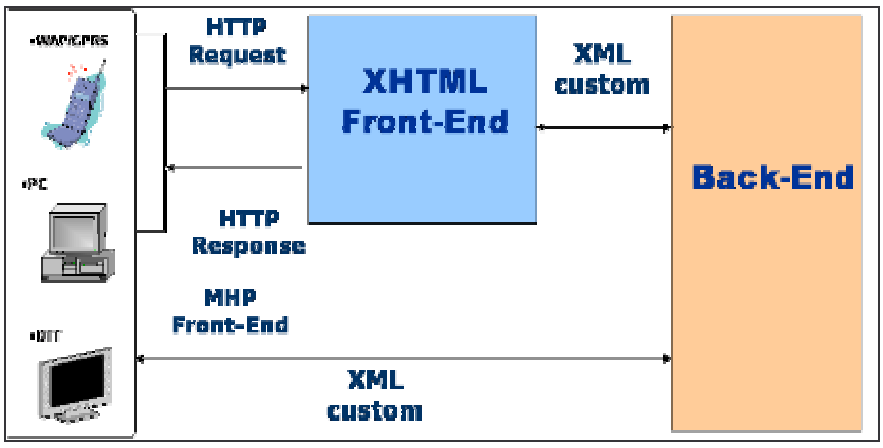

Figure 2. The Talea architecture

The Frontend modules, based on the same pattern, manage HyperText Transfer Protocol (HTTP [http://www.ietf.org/rfc/rfc2616.txt]) requests and they generate the corresponding XML message. In particular, for each request type (encoded in a parameter within the HTTP request), a *Generator* module creates the XML message (based on the HTTP request attributes), to be sent to the Backend.

The interaction between Frontend and Backend follows the request/response protocol: the Frontend sends an XML message to the Backend; the Backend forwards the message to the specific Performer; the Performer handles the request and sends the Frontend a response message.

The creation of a new Performer module enables the *customizer* to add a new high-level feature to the system. For instance, in order to add new advanced search capabilities, the creation of a new Performer module that encapsulates the new search strategy should be required. Moreover, the *customizer* could use two public interfaces, called *PreAction* and *PostAction*, to customize the pre-conditional (and post-conditional) logic of the Performer. In this case, some Aspect-Oriented architectural concepts are applied, in order to enable the *customizer* to enrich the Performer capabilities without changing its code. It is also possible to attach a XML pipeline before and after the Performer execution: this feature facilitates the extension the logic of the Performer, without any impact on its implementation. The associations between each Performer and the additional modules implementing the PreAction/PostAction pipelines are stored in a XML configuration file.

4. The Ontology

Within the Talea framework, the ontology describes the semantics of the matchmaking process, since it represents the relations between users and services. The ontology plays a major role in the customization task, as the *customizer* has to describe the knowledge about a specific business domain.

The ontology included in the Talea framework is expressed in Resource Framework Description (RDF/RDFS) format [www.w3.org/RDF] and represents the top-level classes: the customization process consists in the extension of this top-level. Moreover, the exploitation of a standard format, i.e. RDF/RDFS, guarantees the possibility of exploiting standard tools, such as Protégé [http://protege.stanford.edu/], to extend the ontology.

Talea Ontology defines two classes, together with theirs properties, and the relations between them. In particular the *User* class describes a generic Talea final user; the *Service* class describes a generic Talea service[2]; finally, the *provide* and *consume* relations represent the link between users and services. The *customizer* can define *User* and *Service* subclasses by exploiting the Protégé editor.

In the Authentication Server the information about each user includes the ontology classes she belongs to; if a user belongs to several classes at the same time, she will be able to provide/consume the union of the resources that instances of those user classes are allowed to provide/consume.

The link between each user and its role (i.e., the class she belongs to) supports the personalized navigation, based on the ontology structure.

The knowledge of the domain represented by the ontology is exploited by the Semantic Engine in order to support personalized navigation, tailored to the user role,

---

[2] The *User* and *Service* classes in the Talea Ontology are generic concepts, because theirs features depends on the particular application domain.

and semantic search for available resources, as we will describe in the following section.

5. The Role of the Semantic Engine

Talea Semantic Engine is the Backend module that performs semantic search and exploits the ontology to provide the user with personalized navigation. The Semantic Engine can be viewed as composed by different layers (Figure 3). In order to access the knowledge described by the ontology (in RDFS format), the Semantic Engine uses the Sesame RDF Query Language (SeRQL [www.openrdf.org]). In particular, the Semantic Engine exploits the set of Application Programming Inteface (API) for RDF Schema querying and inferencing provided by Sesame for the implementation of the Talea Semantic API.

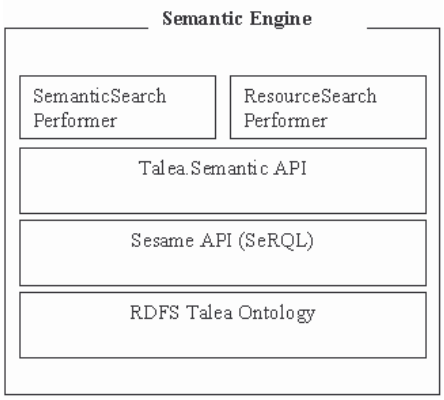

Figure 3. The Semantic Engine

The goal of the Talea Semantic API implementation is to facilitate the business logic customization by providing a set of parametric ontological queries. Such queries can be used by each Perfomer to access the semantics of resource categories (i.e. its properties, its subclasses, and its corresponding table class), thus avoiding the direct use of SeRQL.

The ontology-driven personalized navigation is particularly useful for devices with limited display capabilities (like smartphones or palmars), since it can reduce the amount of information that the user has to provide in order to perform her search: it enables her to operate only on those resources specified by the categories she belongs to.

Each user browsing action triggers a semantic search. More precisely, the request corresponding to a user action specifies the categories the user belongs to and the clicked item. The response depends on the *clicked_item* position in the ontology, i.e.:

- At the very beginning the user is asked to choose if she wants to insert (provide) a resource, or if she wants to look for (consume) a resource. In this interaction step, the user action (*clicked_item*) corresponds to the selection of a relation (*provide* or *consume*), thus the response is the list of service categories provided or consumed by each *user_class*.

- While browsing the ontology, usually the user clicks on service categories; in these cases, *clicked_item* is a class, thus the response is the list of its direct subclasses.
- Finally, *clicked_item* can be a leaf (a class with no subclasses); in this case the response is the list of its properties; the user can express requirements about the values of such properties, that will be used to query the resources database.

In this way the Semantic Search Performer supports the navigation by looking for the "meaning" of the clicked item and providing the user with the next navigation step, on the basis of such a meaning.

The instance search into the database is performed by the Resource Search Performer: the request corresponding to a resource search is translated by the Performer in a Structured Query Language (SQL) query to the database.

When a user performs a resource search, i.e. she looks for instances of a particular class, the salient properties of such instances are returned by the Semantic Search Performer; in this way the user can set a list of conditions on the values of those properties (e.g., type of resource, date and time, and so on), by filling in a form. The resource search response is a list of instances that match the conditions list.

The resource search/insertion functionality can be used independently from the personalized navigation and also in its place, when the ontology structure is too complex to support a user friendly navigation.

6. Implementation Details

Talea is implemented in Java and exploits the Java Servlet technology, running on an Apache Tomcat Web Server. The resources databases is implemented in MySQL and exploits the Java Database Connectivity (JDBC) Connector, while the users data are stored in a LDAP Server (OpenLDAP). The generation of user interfaces for browser-enabled devices are based on XML/XSLT technologies, while a MHP client has been developed for the Digital Terrestrial Television user interface. Apache Cocoon has been used as server-side web development tool in the presentation process. Moreover, the system relies on the already mentioned Protégé and Sesame, the former as ontology editor and the latter to support the semantic engine.

7.Conclusions and Future Work

In this paper we have introduced the first release of the Talea experimental system that represents a starting point for the implementation of knowledge-based services supporting e-business practices.

Small and medium enterprises already use DataBase Management Systems (DBMS) and/or Enterprise Resource Planning (ERP) for the management of their resources. In these terms, Talea has not been implemented to substitute those systems, but to provide a middleware enabling interoperability by means of a semantic matchmaking process, as well as favouring the spread of open standard and knowledge in the business context.

Concerning further implementation, we expect to make Talea evolve towards a more general and powerful framework, usable in real application domains. A new

release will involve a Web Service architecture and semantic orchestration: the idea is to wrap the Performer modules in Web Services interfaces and to exploit semantic orchestration in order to provide macro-functionality [Bruijn et al. 2005], [McIlraith and Martin 2003]. The semantic aspect will be improved by introducing a repository for instances storage, in order to support resource semantic search (no more SQL-like). This will require the *customizer* to use gateway API for interaction with existing DBMS or ERP. Finally, an intelligent brokering solution will be studied, possibly based on intelligent agent negotiation.

More generally, future actions and regional policies will be aimed at using Open Source as a key lever to move further steps towards a knowledge-based economy. Moving from Talea experience and from the Pilot Project that will be implemented, we intend to grow a network of local small/medium enterprises to be involved in research projects for the development of "OSS building blocks", the access to which will be widely promoted as opportunity for the increase of local competitiveness.

References

Alonso, G., Casati F., Kuno, H., and Machiraju V. (2004), "Web Services - Concepts, architectures and applications", Springer, Berlin Heidelberg New York

Bruijn, J., Fensel, D., Keller, U., and Lara R. (2005), "Using the Web Services Modelling Ontology to Enable Semantic eBusiness", Communications of the ACM (CACM) - Special Issue on Semantic eBusiness

"Diadi 2000 Project", http://www.diadi.it/

McIlraith, S. and Martin, D. (2003), "Bringing Semantics to Web Services", in IEEE Intelligent Systems, 18(1)

Peltoniemi, M. (2005), "Business ecosystem: a conceptual model of an organization population from the perspectives of complexity and evolution", in EBRC Research Reports 18

"Protégé: An Ontology Editor and Knowledge-Base Framework", http://protege.stanford.edu/

"Sesame: an Open Source RDF Database with Support for RDF Schema Inferencing and Querying", http://www.openrdf.org

"Talea Framework", http://talea.csp.it/

Trastour, D., Bartolini, C., and Preist, C. (2003), "Semantic Web Support for the Business-to-Business E-Commerce Pre-Contractual Lifecycle", in Computer Networks: The International Journal of Computer and Telecommunications Networking, 42(5), Special Issue on The Semantic Web: an Evolution for a Revolution, North Holland/Elsevier