B2B Interoperability using Business-Driven Integration: Conceptual Framework and Case Study

Ugo Barchetti, Anna Lisa Guido, Luca Mainetti
Department of Innovation Engineering, University of Salento, Italy

Abstract

The interoperability among companies and the business-to-business (B2B) message exchange are key aspects of the growth of businesses: in the next years the companies that are able to cooperate with each other will be winners. The use of B2B messages modifies a company's method of operation and, of course, its internal business process. In this paper, we propose a conceptual framework to solve the two types of problems that arise in the B2B field: one is more methodological, aimed at choosing guidelines oriented to the integration of business processes and the definition of business messages; the other is more technological, aimed at choosing an infrastructural solution more suitable for the development of message exchanges.

The paper also defines a case study in order to show the feasibility of the conceptual framework. Using a specific case study, the paper will show the methodological and technological choices and how ontologies are the glue for different standards and notations involved in business-driven integration problems. Using BPMN notation, we design in this paper a business process that involves different business messages. The paper also presents a REST implementation of the business message exchange.

1. Introduction

In the B2B field, interoperability plays a key role and several approaches have followed each other in time and brought about the birth of new standards and new implementations for solving interoperability problems in a specific supply chain.

The experience of several research projects in the interoperability field (Moda-ML [1], Trame [2], Dialogo [3]) shows that notation to define business processes, the interchange format for message exchange and technologies are different in different supply chains. In this context, it is important to make an abstraction process that allows us to face the interoperability problem from different points of view (the business process level, business messages level and technological level) and provides a solution that is efficient and independent of the specific supply chain.

The abstraction process leads to the definition of a methodological and technological framework (figure 1).

The main goal of the framework is to link together several analysis levels: at the high level it is important to define a methodology and a set of tools for defining and representing business processes, at the intermediate level the goal of the framework is to define an interchange format compliant with the business messages standard and at the low level it is important to define a technology useful for the message interchange. The glue between the different layers is the ontology that represents in an explicit and formal way the methodological choices of the conceptual and logical layers in order to automate the transaction in the physical one.

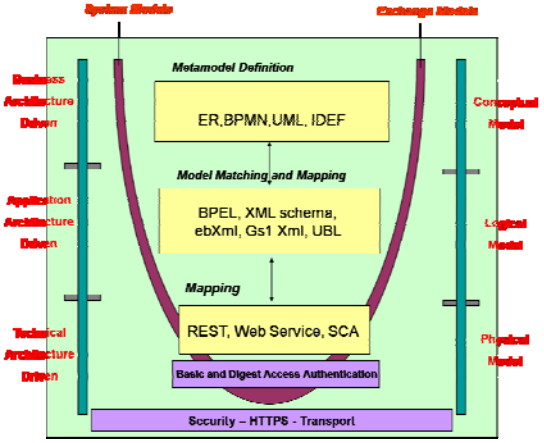

Figure 1. Business-driven integration conceptual framework

In the proposed conceptual framework, due to the special data that might be used, it is important that the implementation will use several peers: companies in a B2B context exchange reserved information as each participant wants to reserve its own information and companies do not like a central server to receive a message that two companies want to exchange. In order to manage the complexity of several business companies that exchange information, it is important to have a central server but its main feature is the discovery.

In this paper, we explain the problem that we face in several research projects that allow us to define the conceptual framework in figure 1. We will explain the framework, and define how ontologies will be used as the glue between the conceptual and logical layers. We will discuss a case study where we use a REST implementation in the physical layer.

In section 2 we discuss the state of the art related to each problem listed in the paper. The open issues

are in section 3 and in section 4 the paper presents details of the proposed conceptual framework. In section 5 there is a case study complex enough to show the effectiveness of the framework and section 6 integrates the metamodel that represents the notation used to describe the business process with the metamodel that represents the standard used in the definition of the business documents. In section 7 the paper presents several details of the case study and in section 8 it presents the business message exchange using peer-to-peer comunication and REST.

2. State of the Art

The state of the art of this paper is orienting to three main aspects (each related to a specific layer of the proposed framework). Another important aspect is related to the ontologies as a glue between different layer.

From the business process perspective, the most important business process design notations used in both scientific and commercial circles are IDEF[4], UML[5] and BPMN[6]. Among these notations, BPMN seems to be the best way to represent the company's business process and the interaction between actors inside the companies and outside the companies.

With regards to the e-business vocabulary, there was an attempt by OASIS to introduce a generic vocabulary/standard useful to each supply chain partner. The standard is ebXML [7]. ebXML belongs to the ebXML framework and the main goal is to define the components that will define a specific business message. There has not been, up to the present moment, a methodology to represent neutral core components in e-business vocabulary [8]. For this reason, the ebXML core component is not used by companies because each company must define its own vocabulary (using core components) and share it with other companies of the supply chain. This problem gave birth to UBL (Universal Business Language), an OASIS initiative that defines in its 2.0 version 31 business documents for different areas [9] such as Sourcing, Ordering, Invoice and Fulfilment.

Finally, as it regards the architectural solutions oriented to the enterprises, we consider SOA (Service Oriented Architecture). A simple SOA definition can be: 'loosely-coupled architecture designed to meet the business needs of the organization' [10]. The reference model provided by OASIS [11] for SOA underlines some fundamental concepts that any implementation is based on [12]. According to the SOA reference model was created the SCA (Service Component Architecture) architectural style. Taking also into account the concepts expressed by SOA reference model, moreover, was created web services as a possible SOA implementation. Finally, came REST, an architectural style that takes into account on the one hand the concept expressed by the SOA reference model and on the other one the most widespread technology: the Web. The REST architecture, created by Roy Fielding, seems to be very promising because it takes into account experiences acquired over the years in B2B electronic exchange and strong points of HTTP protocol that made it the most used in the world.

The glue between conceptual and logical layer and between logical layer and physical layer is the ontology.

Gruber [13], defines ontology as "the study of being as such". Ontologies are a simple to use and a flexible tool useful to describe a complex domain. Ontologies was used to represent BPMN and to represent the UBL vocabulary. BPMN was represented using ontologies in [14] and UBL was partially represented in the iSurf project[15]. In this representation there was defined the elements that define the 31 business messages of the UBL standard but not the 31 messages. Both UBL and BPMN ontologies were made up using OWL[16] language.

3. Open issues

The notation for the business process analysis, the business vocabularies and the architectural solutions presented in the section 2 answer to several problems that companies may face in their daily activities but does not provide a complete and integrated solution for business-driven integration problems [14].

The main problems are not only to define methodological and architectural solutions for the business messages exchange but to link together the methodological and technological approaches in order to provide a solution that answers the needs of all the business partners: in a few words, business partner may share a common language in order to have a communication helpful for the business.

In order to identify a conceptual framework, it is important to provide a way to define business messages that, on the one hand, are not very complex for the final user, and, on the other, able to guarantee a business message representation that is easy to understand for the company's information system.

There are three main open issues in the definition of the solution for the integration issues.

The first is the selection of the notation for the business process design. The business messages come from the more or less complex business process of the company. The business messages may be input or output tasks for execution in the company but, very often, they are not accurately defined in the business process design and its source/target is not well identified. It would appear that BPMN is the best way to represent business messages: it has the modelling primitives useful to define the presence of a message but it does not provide the possibility to

define in a formal way the exchanged message. It is easy to understand that the formal definition of a business process is not a task for the business expert but, at the same time, the definition of the process, made up by the designer, must also include the definition of the business message.

The second problem is to define a language for a formal definition of a business message but it is difficult to adapt these vocabularies to different application domains from which they are thought. Business vocabularies may be very complex, so it is very hard to customise the language, or else it will be very generic compared with the specific application context, so the customisation task may be very difficult. The third problem is that the software architecture can manage the business process exchange efficiently but without thinking (as is right) of the message complexity or the different ways of representing the message. From the previously defined open issues, it appears a double gap (methodological and technological) has to be managed in order to obtain a solution for business-driven integration (Figure 2).

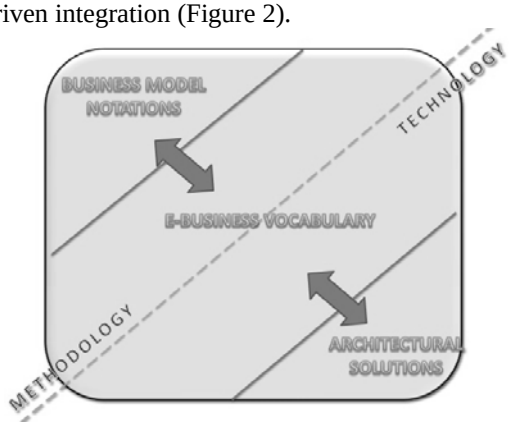

Figure2. Representation of the methodological and technological gaps

E-business vocabulary covers the methodological gap between the definition of the business process and the definition of the business messages. The e-business vocabularies provide inflexible solutions for non-expert users and, very often, it is very hard to link together the business process analysis and the specific e-business vocabulary. Another aspect is the technological gap owing to the need to use a specific business language for the selected architectural solution. This gap is because of the several technological solutions proposed in the e-business systems that are each based on a specific protocol and on complex rules (for example, ebMS vs AS2).

4. Business-driven integration conceptual framework

From the previous consideration it is possible to think that a solution to the open issues is to individualise a possible integration among the various levels that makes the solution feasible, operational, and efficient.

Many technological and methodological solutions are available for solving B2B problems but, to our knowledge, there is no overall solution to the methodological and technological problems that, at the same time, allows maintaining the separation of concerns between methodological and technological level.

The proposed conceptual framework allows us to take structured decisions for each layer.

Clearly most standard and technologies already exist, so it is not useful to add complexity to what has been already designed and implemented. It needs to go towards a simplification of the use of standard, notation and technologies in order to support the interoperability among companies. The two key aspect of our proposed conceptual framework are:

- to provide a high degree of freedom in the business process design and in the formalization of the specific business message;
- to suggest to the companies the use of only one technology of interchange that is flexible and easy to integrate with the company's information system.

The focus is not on the definition of a new solution but on the integration of existing standard technologies with the goal to support the interoperability.

The proposed conceptual framework is based, in fact, on the integration among three different layer, very important in the business to business interoperability (Figure 1):

- the first (Business Architecture Driven) is oriented to the business process design (where it is possible to see different business process notations);
- the second (Application Architecture Driven) is oriented to the definition of the business messages exchanged between companies
- the third (Technical Architecture Driven) is oriented to the technological aspect of the business message exchange.

Naturally, in the high level of the conceptual framework, the business experts define the business process of the companies and identify the messages that the companies must exchange with other partners. In the low level, business messages are defined in a specific way through e-business vocabulary. In the low level of the horseshoe- shape, there is the need to use a specific software architecture to develop the business message exchange. The position of the different layers in the horseshoe shows the decreasing liberty of the designer that, according to the business process design perspective, is free to design the process following its business needs. There are many

constraints (the choice of the specific vocabulary for the business process exchange may define well-defined constraints) until the technical level, where the specific architecture defines design and implementation constraints of well-defined systems.

Clearly the lingua franca between the different layer is the XML but, as clarified in the related works, XML is not the solution of all the problem: it is important to define a metamodel that all the companies must understand in order to communicate. Our idea is that the use of ontologies, thanks to their syntactic and semantic expressivity, helps in the definition of the metamodel. In the proposed conceptual framework, we think to an ontological metamodel to use as a glue between different layers. The ontological metamodel defines both the business process and the message exchanged between business processes. The metamodel may be shared by all the companies that want to exchange messages but the main role of the metamodel is to force each company not in the use of a specific notation/e-business vocabularies but in the task to obtain a business message starting from the business process design (and thus related to it) and not starting from the personal needs of the designer.

Starting from the metamodel it will be possible to obtain the specific model (within the specific application context). This model will be used in the chosen specific technological infrastructure as reference to the underlying level.

In the detail, selecting the business process notation to use, and selecting the business message interchange dictionary, it is possible to obtain an ontological metamodel where there will be, with proper semantics, both the representation of the notation and the representation of the specific dictionary to be used for the definition of the business messages. Therefore, the designer will not only have the possibility to define the business process but also to represent the specific business message exchanged in one model obtained from the developed meta-model (naturally the designer will be equipped with a set of support tools). The obtained model will use a well-defined technological infrastructure that allows business message interchange. From the analysis of open issues, it will be useful to choose a well-defined technological infrastructure that also considers specific partners' needs (within the supply chains).

The proposed conceptual framework is clearly the starting point for a software framework that we will design and implement in the next step of our research work. Surely, it is necessary to implement in the conceptual layer of the framework, all the tools useful to design a business process and to define the correct business message. To do so, it is important to follow a meta-model and a tool that glosses over the complexity of the metamodel and helps the designer to define the business message and translate it in the selected e-business vocabulary (REF PR 6). Finally, it is important to define all the technological infrastructure that allow us to translate the business message to the selected technological infrastructure and to make possible the messages exchange.

In order to understand the proposed conceptual framework, we show here a choice that it is possible to make in each level. It is important to note that the choices here presented do not force the design and development of the framework that we will develop in a next step of our work, but is only a possible solution.

On the highest level of the architecture, it is possible to select the BPMN notation for the business process design. The notation is useful to represent a business process where it is possible to define accurately (but not formalised) the business messages exchanged between different actors. There is an ontological meta-model of BPMN [14] that defines each notation detail. In the second level, it is possible to use the UBL standard (there is also an ontological description of the UBL standard http://ontolog.cim3.net/cgi-bin/wiki.pl?UblOntology). Using the import features of the semantic web technologies, it will be possible to link together the two meta-models in order to provide to the company the possibility to design both the business process and the formal description of the business messages.

To this point, the obtained model will be used in a specific REST architecture: in this way, it is possible to use a well defined protocol used in the Internet (the http) in order to guarantee the message exchange. REST architecture allows us to change the perspective of the business messages. With REST, the business messages are resources that will be used only by authorised companies; another important characteristic is that the system becomes stateless and in this way the client of the application has the task of updating the resource that she/he uses. The main advantages in the use of REST come from the operation supported by the protocol (for example, PUT and GET) that allows us to obtain resources (CRUD Operation Create, Read, Update and Delete that are most often used), and offers the chance for each partner to define its own resources and to store them using a navigational tree available through http. This allows each company to provide information useful for accessing its resources using a formal description of the navigational tree deployed in its server. It is possible to use a XSD schema as resource representation. This makes it necessary to have a XSD schema repository containing the resource representation of each company. This repository can be an UDDI registry where companies deploy their REST services and deploy the schema to access the resources.

5. Introduction to the case study

To show the effectiveness of the proposed conceptual framework, we propose the business process in figure 3 (using BPMN notation).

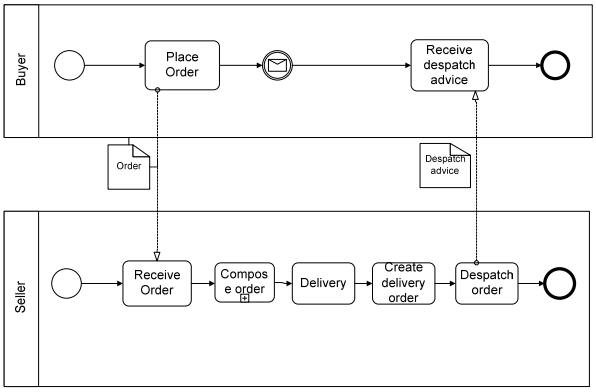

Figure 3. BPMN representation of the request order process

The represented business process shows a generic order made up of a company (peer); the peer that will receive the business message will know which internal business process to use in order to manage the incoming business message.

There is an important difference between the business process that describes the business message exchange between peers and the management of the business message in the company that receives it.

The first business message (exchanged between peers) is a generic business message and the peer that receives it may not be able to manage it. To understand the problem, it is possible to think of a business message that defines that a peer wants to buy a product/service from another peer. The business process that manages the business message exchange will be a generic business process but the company that will receive the business message must understand, from the body of the message, whether the business message is related to a product or to a service and thus manage its internal business message in an appropriate way.

This important aspect has as a consequence the implementation (in each peer) of a Yellow Pages useful for sending the message, inside the company, to the appropriate department to manage it.

In the business process of figure 3, two users (a buyer and a seller) exchange business documents. The buyer sends an order and the seller receives the order, elaborates the order and sends the product/service and a dispatch order to the buyer.

The business process defines two business messages: the order from the buyer to the seller and the dispatch order from the seller to the buyer as confirmation of the dispatch of the order. It is important to highlight that, in the definition of the business process, the business analyst does not define the details of the business message: this important step will be made by the tool that manages the proposed conceptual framework.

The designer defines the properties that characterize each business document in the appropriate task. There will be only one design: the underlying structure will define the UBL message and will send it in the UBL format to the right actor of the business process: this is transparent to the companies involved in the process.

The integration between the business process design and the business message definition is the point of contact between the first two levels and the third level of the conceptual framework. In the next section we will highlight the research aspect of the formal ontologies that allow us to realize the integration discussed above.

6. Business process and business message integration

The proposed conceptual framework suggests creating an infrastructure that allows us to integrate the selected business process notation and the selected business message language. In this paper we select BPMN as the notation to design the business process and UBL as the language for the business message representation. We use ontologies in order to implement the glue between these two aspects.

Using the MOF (Meta Object Facilities – www.omg.org) approach, the integration between different layers has been formed through a metamodel integration between BPMN and UBL. A metamodel can be defined as an explicit and formal representation of design primitives.

Using the BPMN metamodel already defined and the UBL metamodel realized in the iSurf project (http://www.srdc.com.tr/isurf/) the aim was to integrate semantically the two metamodels in order to create only one metamodel. A new business process, with the business message, will be the model of the metamodel and will be the input for the last layer of the conceptual framework, which consists of the technological infrastructure that enables the use of the business messages.

As regards the metamodel related to the BPMN notation, it includes all the primitives of the standard using concepts and relationships between concepts.

As regards the UBL metamodel, it is important to note that, by the choice of the authors of the iSurf project, it contains only the base concepts used in the business messages but it does not contain the 31 business messages as in UBL 2.0.

The conceptual framework proposed in this paper supposes that the business messages are pre-defined in the infrastructure in order to allow the business process designer to define an instance of the business

message and not to have to redefine the business message structure each time. As a consequence of this, it has been important to extend the UBL component ontology, adding the 31 business messages of the UBL standard.

We introduce into the UBL ontology a new class *_Message* that has as subclasses the 31 business messages (_Order, _Catalogue, DespatchAdvice …). The class _Message is related by the Object Property isOfType and representConcept to the subclasses _MessageType of TypeDefinition and _MessageConcept of Concept.

In order to clarify the idea of the ontological approach, we show in figures 4 and 5 the MOF representation of BPMN and UBL. It is important to note that in this paper we use MOF in order to clarify the integration process between BPMN and UBL: in order to realize the ontologies we use the editor protégé.

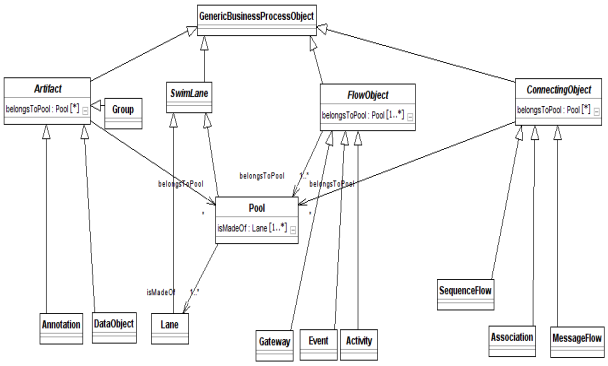

Figure 4. MOF representation of the BPMN metamodel

The MOF metamodel presented here (both of the UBL and of the BPMN) shows only the main elements of the notation without providing all the details of each element.

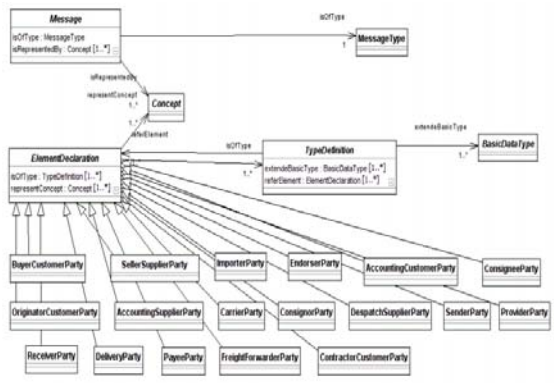

Figure 5. MOF representation of the UBL metamodel

In the UBL metamodel we also show the subclasses of the classes “elementDeclaration” useful for the integration.

Starting from these two metamodels, we realize, using the import mechanism of the ontologies, the integration between the two representations. The integration is performed using three object properties that allow us to link together BPMN primitives and UBL primitives.

In the specific case we add the Object properties “has Actor”, “hasRole” and “representBusinessDocument”, which allow us to link together the BPMN pool and the actor of UBL, the Lane of BPMN and the roles of UBL and finally the property “representBusinessDocument” allows us to define a business message compliant with UBL in a BPMN DataObject.

The MOF representation of the integration is in figure 6.

At this point we have a complete metamodel linking together the BPMN and UBL.

The designer, using an appropriate tool (under development), must describe the specific business process in which the messages will be. In this way, the designer will create a model that will be used:

- From the underlying layer that will extract the business messages in order to allow the message exchange;
- From the high layer in order to validate the correctness of the outgoing messages and the consistency of the exchanged messages related to the business process flow made by the business process designer.

The model will be described using the same ontological language of the metamodel; the model will be read and understood by the software agents.

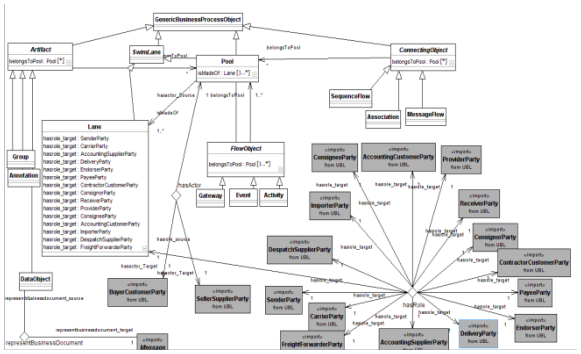

Figure 6. MOF representation of the integrated metamodel

7. Peer-to-peer communication for business message exchange

The proposed conceptual framework has been developed through peer-to-peer communication;

nevertheless, a central server is foreseen with the following roles:

- Taking into account all the peers that want to exchange messages;
- Taking into account the public resources of each peer shared with external peers.

In this way, every time a peer wants to send a message or to access another peer's public resources, it is necessary to know how the sending and receiving of data are structured, which operations are allowed with the resources and in which way the peer can reach the other peer. Then any peer can request from the central server the resources structure of the destination peer with which it wants to exchange messages, having cleared the way to interact with the public resources of the destination peer. Finally, any peer will be able to identify the message incoming from the other peer and to use it correctly.

Now, it is important to model the interaction dynamics of different peers in order to exchange and to manage business messages. Therefore, it is important:

- To identify and to design a solution able to provide a gateway between the central and distributed systems.
- To design a logical reference architecture for the peers that want to interface with each other.
- To identify the resources that guarantee a minimal set of functionalities in order to exchange REST-based messages.

Therefore, a distributed architecture has been designed in which there is a central server (like a UDDI Registry) with the following features:

- Storing the list of all the peers that want to exchange messages;
- Providing the needed information to obtain and use the resources made available by any peer (URL, the resources access schema, etc.).

The information stored on the central server should be useful to any peer that wants to send messages and to access the resources of Peer B, to manage the following information:

- The resources structure
- The allowed resource operations
- The ways to reach the target peer

Then, in this scenario, Peer B will only ask the central server for the resource structure of Peer A and it successively sends the messages, having cleared the interaction way with the resources made available by Peer A.

Thus, this scenario shows that the proposed solution is re-usable through duplication of the same configuration for each involved peer that will have a client REST implementation of the other peer needed to send a message and own server REST implementation needed to receive the message.

In this context we have supposed a scenario where the business messages are described through UBL. Then we have supposed simulating the previously described process (figure 3) where the transport infrastructure is needed to exchange order and dispatch advice messages between the two involved peers.

According to the previous discussion, we can identify some elements of the transport infrastructure. The user that requests the service represents the first element. Then it is necessary to define a client adapter based on pages and objects able to send B2B messages. There is correspondingly a server adapter that deals with receiving the client requests and giving a proper response based on resource representation. Using this system, the server implementation is hidden from the client. We have defined a minimal set of features made available by any peer, particularly:

- Allowing user registration;
- Allowing the updating of registered user profiles;
- Allowing the reception of an external message;
- Allowing the internal users to visualize the external received messages;
- Allowing the internal users to visualize the sender details and the related message contents.

Taking into account that the overall system will be both a client sender and server receiver, it will be able to provide accounting features for external users, to send messages, to visualize received messages for internal users and to retrieve sender information from receiving messages through proper security mechanisms. Then the resources for internal interaction will be the following:

- Local users list: resource "/local/users";
- Local user profile: "/local/users/[company name]";
- Received messages list: resource "/receivedmessages";
- Received message details: resource "/receivedmessages /[Message ID]".

After having identified the involved resources in the system, it is necessary to understand the actions associated with the resources. Then it is necessary to define the REST resource representation.

The resource "/local/users/[company name]" represents the abstraction of the concept of the internal user (peer's user). This resource is available only to the internal user of the system and is able to retrieve the sender details of a message by referencing the resource through a GET operation. The other operations are not allowed because an internal user (receiver) cannot modify the sender information. The same operation can be performed on the "/local/users" resource. This operation (GET) returns a representation of a list of users that have sent messages to the peer. The resource "/receivedmessages /[Message ID]" represents the

abstraction of the concept of the received message. The internal user (receiver) knows through this resource the message sender, the sending date, the object and the message content. The GET operation is the only allowed operation. The "/receivedmessages" resource returns to the internal user a list of all the received messages of the peer. For this resource it is possible to identify some parameters to filter the list (i.e. the company name of the user, etc.).

While the external users (the senders) can interact with the system through the following resources:

- External User: Resource "/users";
- External User Profile: Resource "/users/[company Name]";
- Message: Resource "/users/[company Name]/messages";
- Business Message: Resource "/users/[company Name]/message[msgId]file/";

the "/users" resource can be referenced by external users allowing the registration operation of an external user through the POST method and a representation containing the related user data. To update the external user profile and to retrieve the personal information the "/user" resource is used. It has two method implementations: GET (returns the data supplied during the registration phase) and PUT (updates the user profile). Then the "/messages" resource is used to send messages from external users to the target peer; the only allowed method is POST with the required data for the creation of a new message. Through the analysis of the semantic associated with the different resources, it is possible to define a resources hierarchy/tree in the URL server domain. Thus, we define a general schema representing these resources (figure 7) and their related operations.

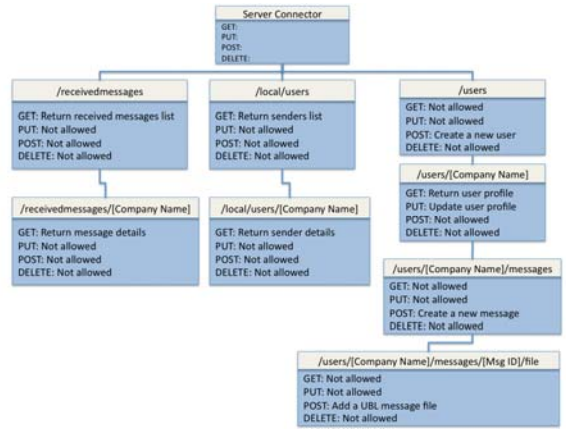

Figure 7. Schema of the resources

Afterwards, it is necessary to define a strategy that allows the representation of the resources structure and provides all the information useful to the external peer to interact with these resources. This information will be stored in the central server as previously described. It uses XML as a tool able to describe the resources structure. The XML resources structure and a Jar (client REST application) work as a collector between the client (sender peer) and server (receiver peer). The XML-based structure can also be useful during the integration phase in the own Web application through the use of XSL offering an appropriate degree of automation. Then the previous architectural choice provides any other kind of elaboration by the peers that retrieves the resource structures from the central server.

8. External resources representation

According to the architecture provided by REST, HTTP methods and the resources previously identified, the XML file will have the following structure.

There is a root node, "resources", with a "baseuri" attribute that will hold the IP address and the active port of the server REST application. Within this node will be n "resource" nodes according to the number of resources that a peer wants to make available to external users; each "resource" node will have the following attributes:

- "Protected": to define protected resources;
- "Name": the resource name;
- "Uri": the resource-related address that is concatenated with the "baseuri" attribute value of the root node to create the complete address needed to interact with the resource.

Each "resource" node will have an "implementedmethods" node that, in turn, will be made up of n "method" nodes according to the number of http methods made available for the interaction with the resource. Each "method" node will have the following attributes:

- "Label": a label to give a meaning to the performed action of the method;
- "Type": identifies the name of the http method used (i.e. GET, POST, etc.).

Some http methods need parameters (i.e. POST and PUT), thus within the "method" node it is possible to create n "param" nodes according to the number of parameters needed. The "param" node is made up of the following attributes:

- "Name": identifies the parameter name;
- "Type": identifies the parameter type (i.e. text, integer, etc.);
- "uriPart": a Boolean value that identifies if the parameter will be sent in the URL.

Figure 8 shows an example of an xml file based on the previously described structure.

```
<resources baseuri="http://localhost:8084/RestletServer/">
  <resource protected="false" name="users" uri="users">
    <implementedmethods>
      <method label="Create User" type="POST">
        <param name="RagioneSociale" type="text" uriPart="true" />
        <param name="FirstName" type="text" uriPart="false" />
        <param name="LastName" type="text" uriPart="false" />
        <param name="Mail" type="text" uriPart="false" />
        <param name="Place" type="text" uriPart="false" />
        <param name="Username" type="text" uriPart="false" />
        <param name="Password" type="password" uriPart="false" />
      </method>
    </implementedmethods>
  </resource>
  <resource protected="true" name="user" uri="users/{RagioneSociale}">
    <implementedmethods>
      <method label="User Profile" type="GET">
        <param name="RagioneSociale" type="text" uriPart="true" />
      </method>
      <method label="Update User Profile" type="PUT">
        <param name="RagioneSociale" type="text" uriPart="true" />
        <param name="FirstName" type="text" uriPart="false" />
        <param name="LastName" type="text" uriPart="false" />
        <param name="Mail" type="text" uriPart="false" />
        <param name="Place" type="text" uriPart="false" />
      </method>
    </implementedmethods>
  </resource>
  <resource protected="true" name="messages" uri="users/{RagioneSociale}/messages">
    <implementedmethods>
      <method label="Send Message" type="POST">
        <param name="RagioneSociale" type="text" uriPart="true" />
        <param name="Date" type="text" uriPart="false" />
        <param name="Object" type="text" uriPart="false" />
        <param name="File" type="file" uriPart="false" />
      </method>
    </implementedmethods>
  </resource>
</resources>
```

Figure 8: resources structure

Then we define an interaction scenario that uses a publish/subscribe model as shown in figure 9. In this scenario there are seven steps that allow the publication of the own resource structure on the central server and the interaction between two peers. The goal of this scenario is to send a business message from Peer A (sender) to Peer B (receiver). First of all, it is necessary for any peer using REST to map its own internal resources and consequently provide a client REST application that can be used not only to identify the URL of the server REST application, but also to describe the resources mapping for external users that want to interact with the peer. Particularly, if Peer A wants to send a business message to Peer B, it is necessary to follow these steps:

1. Peer B publishes its own references and its own business services/features on a UDDI Registry;
2. Peer A searches for the service on the UDDI Registry;
3. Peer A finds Peer B (service provider/receiver of the business message) and retrieves the URL needed to download the client REST application;
4. Peer A takes the download of the client REST application and the Peer B resources structure;
5. Peer A can access the public resources without any authorization or can request an account to access Peer B's private resources (the exchange of a business message is a form of access to a private resource);
6. Peer B sends the credentials to Peer A through which Peer A can access Peer B's private resources;
7. Peer A accesses Peer B's private resources, transferring the business message.

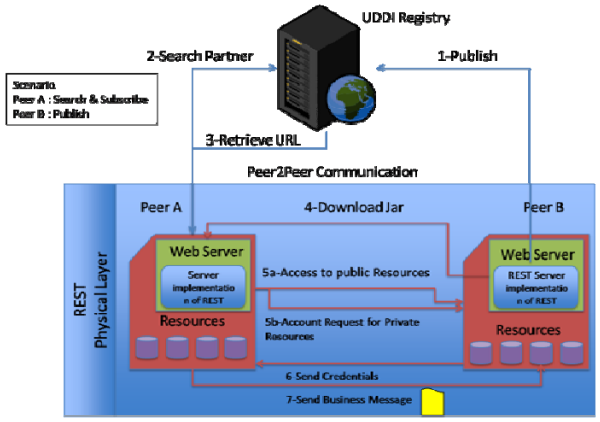

Figure 9: Publish subscribe model

In the same way and with a symmetrical approach Peer B can send a business message to Peer A by accessing Peer A's private resources.

9. Conclusions

The proposed conceptual framework answers several open issues that arose regarding B2B in several research projects and is an abstraction process from several research projects. The conceptual framework does not depend on the specific supply chain and has two main characteristics:

1. It uses formal ontologies to describe both the metamodel and the model of a specific case study. The model is the input of a technological layer that is able to read the model it and to take the business message exchange in the http protocol.
2. It separates the problems list related to the transport of a generic business message from the problem list related to the syntactic and semantic validation of the business process.

The proposed conceptual framework allows the cabling of the business process validation in the information system of the company that uses the metamodel.

Starting from the conceptual framework, we implement both a metamodel and the technological infrastructure, which allow us to validate, using a case study, the proposed solution. We pass through the horseshoe- shape, but it is useful to validate the conceptual framework with other case studies and to propose a set of tools that helps the designer in the task of defining business processes and business messages within the business process.

10. References

[1] Gessa N, Vitali F, Cucchiara G, De Sabbata P, Fraulini N, Marzocchi M, Imolesi T, Mainetti L. "Moda-Ml, An

Interoperability Framework For The Textile-Clothing Sector". In: Iadis International Conference Www/Internet 2003. Algarve (Portugal), November 2003, pp. 61–68.

[2] Barchetti U, Bucciero, A, Mainetti, L, Santo Sabato, S. "A Framework To Support Interoperability And Multi-Channel Delivery Among Heterogeneous Systems: Trame Project". Iceis 2008. Barcelona, Spain, 12–16 June 2008 insticc, pp. 179–189, ISBN/ISSN: 978-989-8111-36-4.

[3] Barchetti U, Bucciero A, Capodieci A, Capone L, Mainetti L, Santo Sabato S. "Multi-Modal And Multi-Channel E-Services Framework Meets Modern Freight Management Systems Requirements: Dialogo Project". In: Mccsis '08. Amsterdam, the Netherlands, 2008, pp. 25–27.

[4] Draft Federal Information Processing Standards Publication 183 "Integration Definition for Function Modeling", 1993 (IDEF0).

[5] Eriksson HE, Penker M. "Business Modeling with UML: Patterns at Work", John Wiley.

[6] OMG. "Business Process Modeling Notation Specification", 2006.

[7] Hofreiter B, Huemer, C. "B2B Integration – Aligning ebXML and Ontology Approaches", Proceedings of the First EurAsian Conference on Information and Communication Technology, 29–31 October 2002, pp. 339–349.

[8] OASIS "The Framework for eBusiness", 2007.

[9] OASIS Service Component Architecture/Assembly (SCA-Assembly) T.C. "Service Component Architecture Assembly Specification Version 1.0", March 2007.

[10] Microsoft SOA electronic book "Service Oriented Architecture (SOA) in the Real World", 2007.

[11] OASIS SOA Reference Model TC. "OASIS Reference Model for Service Oriented Architecture", April 2008.

[12] Venktraman N. "IT-Enabled Business Transformation: From Automation to Business Scope Redefinition", Sloan Management Review, 1994, pp. 73–87.

[13] Gruber T. "A Translation Approach to Portable Ontology Specification. Knowledge Acquisition", 1993, pp. 199–200.

[14] Paiano R, Guido AL. "A Design Tool for Business Process Design and Representation". In: Semantic Web Technologies and e-Business: Toward the Integrated Virtual Organization and Business Process Automation. Idea Group Publishing. Setùbal, Portugal, 11–13 April 2006, pp. 375–380. ISBN: 1-59904-192-8.

[15] Yarimagan Y, Dogac A. "A Semantic-Based Solution for UBL Schema Interoperability Export", IEEE Internet Computing, 2009, Vol. 13, No. 3, pp. 64–71.

[16] W3C. "Web Ontology Language Reference", 2004.

[17] Dorn J, Grun C, Werthner H, Zapletal M. "A Survey of B2B Methodologies and Technologies: From Business Models towards Deployment Artifacts". In: 40th Annual Hawaii International Conference on System Sciences, 3–6 January 2007.

11. Acknowledgements

We would like to thank the Fondazione Politecnico Consortium of Politecnico di Milano for their help with the identification of the issues faced by the Cariaggi Project funded by the Italian Ministry of Economic Development.

