An Ontology-Based Approach for Semantic Integration

Conference Paper · November 2010

DOI: 10.1109/EDOC.2010.32 · Source: IEEE Xplore

2 authors:

Rodrigo Calhau
Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo (I…
13 PUBLICATIONS **35** CITATIONS
SEE PROFILE

Ricardo de Almeida Falbo
Universidade Federal do Espírito Santo
172 PUBLICATIONS **1,661** CITATIONS
SEE PROFILE

An Ontology-based Approach for Semantic Integration

Rodrigo Fernandes Calhau
Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department
Federal University of Espírito Santo (UFES)
Vitória ES, Brazil
rfcalhau@inf.ufes.br

Ricardo de Almeida Falbo
Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department
Federal University of Espírito Santo (UFES)
Vitória ES, Brazil
falbo@inf.ufes.br

Abstract - **One of the main challenges for enterprise application integration is the semantically correct integration of systems. There are many approaches that aim to address this problem, but most of them focus mainly on technical and syntactical aspects. This paper presents an Ontology-Based Approach for Semantic Integration (OBA-SI), which uses ontologies as conceptual models for defining mappings between applications in three layers: data, service and process. OBA-SI was applied for integrating two software tools supporting a Software Configuration Management process: a tool for controlling changes, and Subversion, a well known version control system.**

Semantic Tools Integration, Ontologies, Semantic Interoperability

I. Introduction

Enterprise information systems are generally composed of a multitude of applications supporting different, but interrelated enterprise processes. In order to properly support the whole organization's business process, these applications should be integrated. However, integration is a complex problem, especially for large and dynamic organizations. In general, each application runs independently and implements its own data and process models. These models are not shared between applications, leading to several conflicts, including technical, syntactical and, especially, semantic conflicts. This heterogeneity is considered one of the major difficulties of the integration problem [1].

An approach to deal with the heterogeneity problem when integrating different applications is to introduce mediators concerning both data and process. Data mediation consists of transforming sent data to received data keeping their semantics. Process mediation consists of solving operation and invocation heterogeneities, also keeping the corresponding semantics [1]. A key factor for integration is to have applications sharing a common understanding of the meaning of the handled terms and services exchanged by them. I.e., we need to have a shared representation of concepts and tasks in the domain of interest.

In this context, ontologies can be used as an interlingua to map concepts and services used by the different enterprise applications, in a scenario of access to data and services via a shared ontology.

Several studies have focused on the semantic integration of enterprise applications, several of them making use of ontologies [1]. Some cover the layers of data, service and process, such as [2,3,4], or only some of those layers, such as [5,6]. Some approaches use web services [2,3,4,5,6], and others is focused on an implementation using an ontology language, such OWL [5]. In this case, the ontology language is used for describing the semantics of data or services. However, many of these studies focus too much attention in technological aspects of integration solution. This ends up interfering in the semantic integration approach, since technological aspects permeate the proposed solution.

In this paper, we advocate that initially semantic integration should be independent on how the integration will be designed and implemented. As any software development process, semantic integration process should start at the conceptual level. Only after considering integration requirements and building a conceptual model of the integration, the technological aspects should be addressed (design phase).

In this paper we present an Ontology-Based Approach for Semantic Integration (OBA-SI) that concentrates efforts in the integration requirements analysis and modeling. In this approach, integration is addressed in a high level of abstraction, looking for getting semantic agreement between the tools at the conceptual level. A premise of the proposed approach is that the assignment of semantics and the use of ontologies must be independent of the integration solution itself. The proposed approach deals with integration in three layers: data, service and process. To accomplish this, conceptual and behavioral models of the tools to be integrated are compared in the light of a domain reference ontology, which is responsible for assigning semantics to the shared items of the tools.

The general scenario of the enterprise application integration also holds for software organizations. To support the whole software process, several tools are used, each one supporting one or more related activities. Thus, to illustrate our approach, we present a real case of its application in integrating the version control system Subversion (SVN) [7] and a tool for controlling changes, in order to support a Software Configuration Management (SCM) process [8].

This paper is organized as follows: Section 2 regards the theoretical background of the paper, discussing briefly the integration problem and the use of ontologies to deal with it; Section 3 presents the proposed approach (OBA-SI); Section 4 presents its application in the real case of integrating tools to partially support a SCM process; Section 5 discusses related works; finally, Section 6 presents our conclusions and future work.

II. Semantic Integration

Organizations are using an increasing number of applications to support their processes. In general, enterprise applications are standalone software entities that are defined in isolation, and are operated autonomously supporting specific parts of the whole business process [10,1]. This situation leads to problems such as data replication, inconsistencies, and repeated functions.

Enterprise applications are usually developed by different vendors, in different points in time, and thus they often have no concern with integration. An appropriate characterization of enterprise applications is that they are HAD (heterogeneous, autonomous and distributed) systems [9,1]. Heterogeneous refers to the fact that each application implements its own data model defining its relevant concepts in its own way. Autonomous means that an application may run independently of other applications. Distributed means that applications implement their data model in their own data repository and these are not shared with other application systems [1]. In this context, any form of integration between applications happens outside them, i.e., the applications are not aware of the fact that they are synchronized in any way with each other [9].

There are various dimensions that allow the integration notion to be characterized. Izza [1] synthesized them in four dimensions, namely: scope, distinguishing between intra and inter entreprise integration; viewpoint, distinguishing between user, designer and programmer views; layers, considering data, service and process integration; and levels, considering hardware, platform, syntactical and semantic levels.

In respect to integration layers, *data integration* deals with how the applications share data, while *service integration* addresses sharing services. *Process integration* views the enterprise as a set of interrelated processes and it is responsible for handling message flows, implementing rules and defining the overall process execution [1].

Regarding integration levels, the *hardware level* concerns differences in computer hardware, networks, etc. The *platform level* encompasses differences in operating system, database platform, etc. The *syntactical level* regards the way the data model and operation signatures are written down. Finally, the *semantic level* concerns the intended meaning of the concepts in a data schema or in operation signatures [1]. It is worthwhile to point out that each level depends on the prior one to happen. For instance, it is not possible to speak about semantics if there is no syntax.

Most integration approaches focus on the syntactical level [1]. However, integration at the semantic level is gaining more and more attention. While syntactical integration occurs at the structural level, involving technological aspects of integration, semantic integration occurs at the knowledge level [10], considering that applications must share the meaning of their terminologies.

Although there are works focusing on semantic integration, semantics is still an important missing element that characterizes most current approaches [1]. Semantics must be correctly dealt with in order to bring enterprise applications to their full potential. For this, ontologies may constitute an interesting way to deal with meaning and semantic heterogeneities.

Most authors consider that ontologies represent the heart of future of the integration tools [1]. Ontologies can be used to establish a common understanding about some universe of discourse, serving as an interlingua for communication between the applications.

According to their level of generality, ontologies can be classified in [11]: *top-level* or *foundational ontologies*, which describe very general concepts that are independent of a particular task or domain; *domain* and *task ontologies*, which describe, respectively, the vocabulary related to a generic domain or a generic task or activity; and *application ontologies*, which describe concepts depending both on a particular domain and task.

A domain ontology is a conceptual specification of the semantics of a certain domain that describes knowledge about it. It aims to restrict vocabulary interpretations so that its logical models get as near as possible to the set of structures that conceptualize that domain [12]. So a domain ontology can be used to establish a common conceptualization about certain domain, in order to support communication and tools integration in it.

Enterprise applications manipulate representations of entities in a domain of the real world. For two tools to properly communicate, they must reference the same entities in this domain [4]. As formal and consensual representations of real world entities, ontologies can be used as a representation of the world for the tools, facilitating the semantic integration between them.

Semantics and ontologies are important to application integration, because they provide a shared understanding of data, services and processes that exist in an integration scenario, and facilitate communication between people and information systems. Using ontologies we can organize and share enterprise information, as well as manage content and knowledge, allowing better interoperability between inter- and intra-enterprise information systems [3].

Nowadays, semantic integration is one of the most serious challenges for enterprise application integration. Despite there are already approaches regarding semantic integration, many of them are focused on only one layer of integration (typically data or service integration) and encompass technological aspects, such as the use of Semantic Web technologies for annotating data or web services [5,13]. They do not make the best use of ontologies for semantic assignment.

In fact, one of the major bottlenecks in semantic integration is how to make mappings and we claim that we should advance in the methodological aspects of integration. In the next section, we present an approach for semantic integration that considers data, service and process layers, and that is centered on the assignment of semantics through ontologies. This assignment is made at the conceptual level and it is independent of technology. In this approach, we use what Guizzardi [13] calls a *domain reference ontology*, i.e., an ontology that is constructed with the sole objective of making the best possible description of the domain in reality, with regard to a certain level of granularity and viewpoint. A reference ontology is to be a special kind of conceptual model, an engineering artifact with the additional requirement of representing a model of consensus within a community. It is a solution-independent specification with the aim of making a clear and precise description of domain entities for the purposes of communication, learning and problem-solving.

III. An Ontology-based Approach For Semantic Integration

The Ontology-Based Approach for Semantic Integration (OBA-SI), analogously to the software development process, advocates the use of a process composed of the following activities: analysis, design, implementation, and tests. OBA-SI, however, is centered in the first phase, integration analysis, giving only some guidelines to the design phase.

According to OBA-SI, semantics must be defined at the beginning of the integration process, in the analysis phase. This is because tools must agree semantically prior to design and implementation of a specific solution. In this way, OBA-SI seeks to be independent of technology and of a specific integration solution. During the analysis phase, semantic mappings are made between the conceptual models of the enterprise applications being integrated, using ontologies to assign meaning.

In the analysis phase, three artifacts are very important: (i) a reference domain ontology describing the integration domain; (ii) the conceptual models of the applications being integrated (both structural and behavioral models); and (iii) the process model of the business being supported by the tools. The reference ontology, as said before, is a solution-independent specification making a clear and precise description of the domain entities. It is used as a reference model to assign semantics during the integration analysis. The conceptual models of the applications being integrated can be known priori (for instance, the applications were developed by the organization and the conceptual models are available), or they should be excavated by means of reverse engineering techniques. The business process model, in turn, should describe the organization's processes involved in the integration scenario, indicating, among others, activities, workers, supporting tools, and inputs and outputs.

These models focus on different aspects of integration. The reference domain ontology along with the conceptual models of the tools is used for establishing semantics in both data and service layers. The business process model is used mainly for addressing process integration, although it is also useful in service integration.

As a result of the analysis phase, an integration model is produced, capturing the overall picture of the integration scenario. The integration model specifies: (i) the concepts to be represented, (ii) how the set of integrated tools will behave as a whole, and (iii) which process activities and enterprise applications' services are to be integrated. The integration model is built based on the integration requirements and the three artifacts mentioned before. In fact, the main purpose of the integration model is to establish mappings between the models of the tools being integrated. These mappings are discussed following.

A. Model Mappings

The purpose of a mapping is to relate model elements[1] that are considered to be equivalent. Mappings occur primarily at two levels: vertical and horizontal.

Vertical Mappings (VMs) are responsible for assigning meaning to the model elements based on the reference ontology. They are independent of the integration solution. Their goal is to assign meaning to the tools' entities through relating them to entities in the domain reference ontology. Since VMs relate model elements to ontology's concepts and relations, they are independent of integration scenarios and can be used in several of them.

Horizontal Mappings (HMs), on the other hand, occur between model elements focusing on the integration scenario. When establishing the HMs, model elements of the tools are mapped to model elements in the integration model, as shown in Figure 1. The HMs define how the tools will be seen by the integration mediator, and how the interaction between them will occur. Ideally, HMs should be based on the VMs. However, since HMs focused on a specific integration scenario, they are not always based on the VMs.

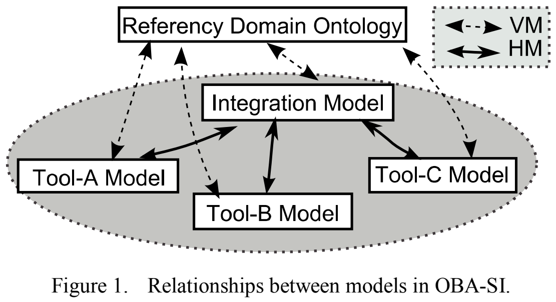

Figure 1. Relationships between models in OBA-SI.

[1] A model element can be any model element in the sense of UML, represented by the models being mapped, such as concepts, relationships or activities.

B. *The Integration Analysis Phase*

The analysis phase of OBA-SI encompasses the following activities: (i) integration requirements specification; (ii) conceptual models retrieval; (iii) semantics assignment to tools' structural conceptual models; and (iv) integration modeling.

During the integration requirements specification activity, integration requirements and goals should be defined. Initially, integration goals should be established. The main questions to be answered are: Why do we need to integrate some enterprise applications? Which are the business processes to be integrated? Based on the goals, requirements can be elicited. The integration scope should be defined, pointing which activities of the business process will be supported and which enterprise applications should be integrated to support these activities. These activities and the enterprise applications that will be used to support them compose what we call the integration scenario. For this task, business process models should be available or should be developed as part of the integration project. It is also important to define the domain where the integration scenario takes place.

Once defined the integration scenario, it is the time to retrieve the required conceptual models. For each tool involved in the integration scenario, we need to have both its structural and behavioral conceptual models. If they are not available, we should perform their reverse engineering. We need also to select a reference ontology regarding the domain of the integration scenario. If a domain reference ontology is not available, a new one must be developed.

The next step is to assign semantics to the tools' structural conceptual models, defining the vertical mappings between their concepts and relations and the concepts and relations of the reference ontology. These mappings should be validated by a domain expert.

Once the structural models are semantically annotated, integration modeling can start. The purpose of the integration model is to model structural and behavioral aspects of the integration at the conceptual level, taking into account the requirements previously specified. It is quite similar to a conceptual model in any development process. The main difference is that the integration model is built based on the ontology and on the tools' models. This way, the semantic-gap between the integration model and the tools' models will be smaller.

The integration model is the main output of the analysis phase of OBA-SI. It is built in a way that mixes a top-down and a bottom-up approach. Initially, a preliminary model is built based on the requirements, the business process model and the reference ontology, in a top-down approach. Next, this model is refined taking into account the conceptual models of the tools being integrated, in a bottom-up approach.

With a more refined version of the integration model in hand, we should establish structural and behavioral mappings. First structural vertical mappings (VMs) are established between the elements of the reference ontology and the elements of the structural part of the integration model. Next, we should establish horizontal mappings (HMs) between the elements of the tools' conceptual models and the elements of the integration model. Both structural and behavioral mappings should be done. Structural mappings concern the data layer, while behavioral mappings regard process and service layers.

Figure 2. illustrates an integration scenario, where two structural conceptual models are to be mapped to an integration model at the light of a domain ontology. Concepts of the tools and of the integration model are mapped to the concepts of the ontology, characterizing the VMs, such as the mappings *(b2, o2)* and *(a2, o2)*. HMs are then established based on the VMs, as is the case of the mappings *(a2, i2)* and *(b2, i2)*. However, there may be HMs, such as *(a3, i3) and (b3, i3),* that are not based on a VM. This may occur because sometimes the reference ontology does not cover all the model elements considered by the applications.

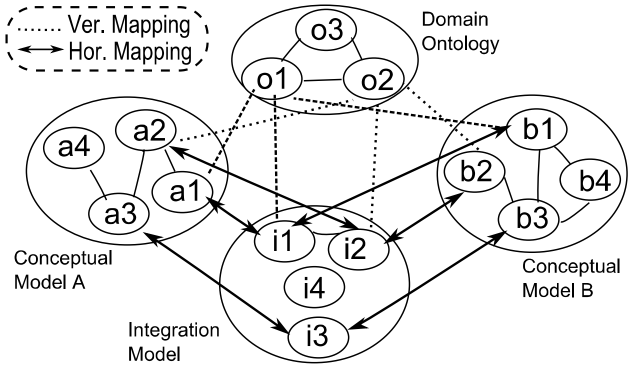

Figure 2. Vertical and Horizontal mappings.

Regarding the structural mapping (data layer), first mappings between concepts should be established. Once established the mappings between concepts, mappings between relations can be defined. The procedure is analogous, i.e., VMs and HMs are established and the corresponding relations are incorporated to the structural model of integration. However, as a concept is also defined in terms of their relationships, mappings between relations can influence the mappings between concepts, and so this is an iterative process.

Based on the structural mappings, the behavioral mappings (between tools' functionalities and the behavioral model of integration) can be established. These mappings are concerned with the process and service integration layers, and are also based on the structural mappings. As a result of the behavioral integration, functionalities of tools are linked to form a unique behavior.

The same way that equivalent concepts and relations are identified in the structural mapping, behavioral mappings look for identifying equivalent services / tasks. However, identifying these correspondences is not something straightforward. The analysis of the structure of the behavioral models, for example, can help identifying functionalities and tasks that occur in the same context. Related functionalities or tasks usually occur in similar situations, having, for example, similar functionalities/tasks that occur before and after. The analysis of inputs and outputs of services and tasks can also provide important tips about how to perform the mappings, since corresponding services and tasks usually handle the same concepts.

Before the behavioral mappings are made, it is advisable that inputs and outputs of tools' functionalities and process activities are mapped to concepts in the structural model of integration. Once these mappings are established, tasks/ functionalities can be mapped. This involves the mappings between the inputs and outputs of the corresponding features. First of all, the integration analyst should look for tasks and services that seem to have the same semantics. This is a completely subjective job. To assist in this analysis, the analyst can look at the inputs and outputs of the services being compared. If there are correspondences between inputs and outputs, the chance of having a mapping between the tasks / services is considerable. On the other hand, if the concepts mapped to the inputs and outputs of the two services are not the same, it is necessary to analyze the relationship between them.

Figure 3. shows some HMs between services, inputs and concepts of two tools (*A* and *B*) through an integration model. Each service input should correspond to a concept in the structural model of the tool. The relationship between the concepts that represent inputs and outputs can give hints to the mappings between them. However, in this case, what should be mainly considered is the role of inputs (and outputs) in the corresponding services. The mapping between the inputs *i1* and *i2* illustrates this. Notice that *i2* could be mapped to *i3*, since they represent equivalent concepts (*a2*, *c3*). However, in this example, the inputs *i1* and *i2* have similar roles in the corresponding services (Serv1, Serv2) and therefore they were mapped.

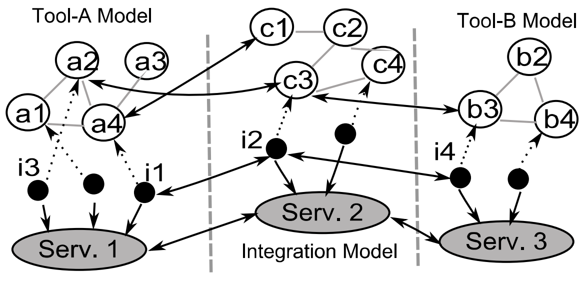

Figure 3. Mapping services and their inputs.

The same analysis done for the inputs also applies to outputs. The mapping between outputs is directly related to the consistency of the data manipulated by the integrated tools. As services are executed, data will be created and updated. Therefore, it is important to have means to ensure inputs and outputs map. The services and tasks considered in the mappings are part of the behavioral model of integration.

Figure 4. summarizes the model elements that are mapped in the OBA-SI analysis phase. In the figure, we show two models (A and B), representing both application models, integration model and the ontology. The mappings shown may represent both VMs and HMs. In the first case, one of the models must be an ontology. In the last, one of models must be the integration model and the other must be an application model. In addition to the mappings, we also show the order in which they occur: (1) concepts, (2) relations, (3) inputs/outputs to concepts, (4) tasks (services) and (5) inputs/outputs. Note that this is an iterative process, because the types of mappings are related one to another.

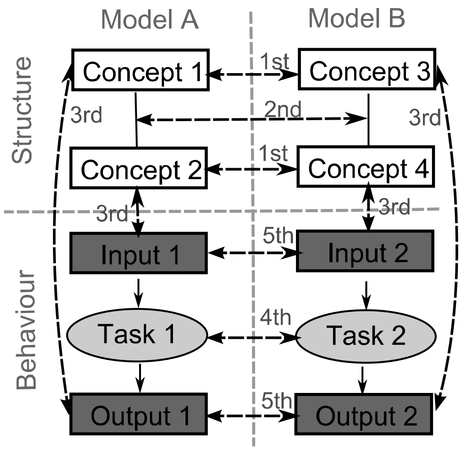

Figure 4. Mappings between model elements.

C. Design and Implementation Phases

An integration solution can be designed and implemented in several ways. OBA-SI is not committed to any specific integration solution. However, some guidelines should be followed in order to maintain the semantic consistency in these steps. First of all, the design should be based on the semantics established during the analysis phase. Thus, the main inputs for the design phase are the conceptual models of the tools and the integration model. These artifacts are independent of the computational aspects related to the integration solution. However, since the design phase takes into account such aspects, it is natural that these artifacts are changed, as in any software development process. So, the integration model as well as the conceptual models of the tools can also be changed to incorporate decisions regarding the integration solution.

Usually, integration of enterprise applications happens outside them. A way to integrate applications is to build mediators that glue the applications. In this strategy, the tools do not talk to each other directly, but through mediators. A mediator must have an overall view of the systems being integrated. Mediators are responsible for exchanging data and functionalities between tools, and the integration model provides the most important information to design them. The integration model is composed of both a structural model and a behavioral model. The structural model captures the concepts and relations that are to be manipulated by the mediator. The behavioral model, in turn, shows the process activities and the enterprise applications' services that are to be integrated by the mediator. Moreover, the orchestration of the tools' service invocations is also made through the mediator taking into account the behavioral model of the integration. Thus, it is important that the mediator implements the integration model in order to link the tools. Finally, the horizontal mappings (structural and behavioral) can be used to design the communication between the tools and the mediator. The implementation of this communication can occur inside the mediator or out of it, by means of adapters that connect the tools to the mediator, as in [8]. The horizontal structural mappings can be used to translate data between the tools and the mediator.

IV. INTEGRATING SOFTWARE CONFIGURATION MANAGEMENT TOOLS

In this section we present a real case in which OBA-SI was applied to solve an integration problem in the Software Configuration Management (SCM) domain at the Software Engineering Lab (LabES) at Federal University of Espírito Santo. The integration scenario involves Subversion (SVN) [7] and CM-ODE. The first is a well known free/open-source version control system that manages files and directories, and the changes made to them, over time. The second is a tool supporting change management in ODE (Ontology-based software Development Environment) [15]. Next, the main aspects of the application of OBA-SI in this integration project are discussed. Due to space limitations, we cannot present in details the whole integration process.

A. Integration Scenario

According to [8], the SCM is "the discipline of identifying the configuration of a system at distinct points in time for the purpose of systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the system life cycle". There are various systems supporting partially the SCM process, such as SVN, which is concerned with version control.

The following SCM process was defined to be followed by all projects developed at LabES:

- Plan Software Configuration Management
- <u>Identify Configuration</u> (<u>Select</u> and <u>Describe Configuration Items</u>)
- <u>Control Configuration</u> (<u>Manage Baselines</u>, <u>Manage Branches</u>, <u>Control Version</u>)
- <u>Control Change</u> (<u>Request Change</u>, <u>Evaluate Change</u>, <u>Perform Change</u>, Verify Change)
- Perform Configuration Audits
- Report Configuration Status

The integration scope includes the activities underlined in the business process presented previously. In order to support this integration scope, SVN and CM-ODE should be integrated.

B. Retrieving the Conceptual Models

Once defined the integration scenario, the required conceptual models are to be retrieved. Concerning the reference ontology, we used the SCM ontology proposed in [16]. Figure 5 shows a fragment of this ontology. As shown in this figure, *repositories* are organized in *branches* that contain *versions* of *configuration items*. Although not show in Figure 5, *versions* can be *variants* or *revisions*. *Items* are derived from *artifacts*, and can be versioned (*configuration items*) or *unversioned*. *Changes* are made in *versions*. When a *checkout* occurs, *copies* of the *versions* are generated. These *copies* are effectively changed and give rise to new *versions* of *configuration items* when a *check in* occurs. A *baseline* is a set of *versions* grouped for some purpose [16].

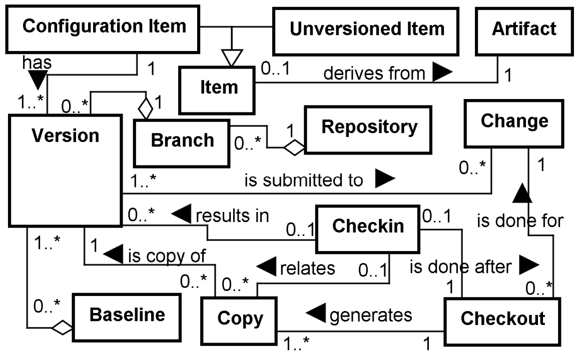

Figure 5. Part of Strutural Model of SCM ontology [16]

Regarding the tools' conceptual models, two different approaches were used. Since CM-ODE was developed at LabES, its behavioral and structural conceptual models were available. This was not the case of SVN. The conceptual models of SVN had to be excavated. Due to space limitations, the models of CM-ODE are not presented here. However, it is worthwhile to point out that its structural conceptual model is quite similar to the ontology, since it was built based on it. Concerning its behavioral model, CM-ODE supports, among others, the following activities: change request, assessing the impact of a change, and traceability.

With respect to SVN, first we had to excavate its structural conceptual model. This was done by analyzing schemas, metadata and the actual source code of SVN. Figure 6. shows the structural conceptual model resulting from this step. SVN *repository* is composed of *repository items* that can be *repository directories* or *files*. Each item in the repository can give rise to other items when changes occur. A *working copy* are generated when a checkout occur. Working Copies (WC) are composed of *WC files* or *directories* that represent copies of the items from the repository. The changes made in items in the working copy are recorded in the repository via a commit. At this moment a new revision is created in the repository as well as a set of new items.

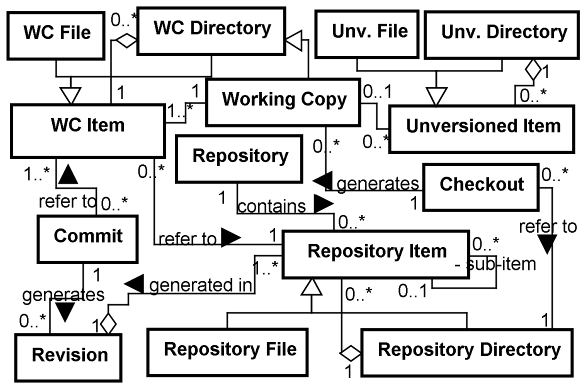

Figure 6. Conceptual model extracted from SVN

Concerning SVN behavioral model, it was extracted from the SVN reference manual [7], considering also our experience in using it at LabES. Following we present a fragment of the process defined, underlying the SVN functionalities used to support it:

1. Perform Change
   a. Checkout
   b. Change (Add, Remove, Copy, Update, Revert a WC Item)
   c. Check in
2. Manage Repository (Import, Export, Copy a Repository Item)

C. *Assigning Semantics to the Structural Models*

After retrieving the SCM ontology and the conceptual models of the tools, the next step is to assign semantics to the structural models of the tools. In this step, vertical mappings (VMs) between the structural models of the tools and the reference ontology are established.

Since CM-ODE was built based on the SCM ontology, its mappings were almost direct and easy to be done. Thus they are not presented here. However, with respect to SVN, the assignment of semantics was a hard task.

TABLE I. shows the vertical mapping established to link SVN concepts to the concepts of the SCM ontology. A *Repository Item* in SVN corresponds to a *Version* in the ontology; a *WC Item* corresponds to a *Copy* and so on. It is important to highlight that the term Revision is present both in the ontology and in SVN. Despite both models have concepts termed with the same word, in this case, they have different meanings. In the ontology, *Revision* means a version that substitutes another one. On the other hand, in SVN, *Revision* means a state of the repository generated after a commit.

It is also important to point out that, despite the concepts of *Branch* and *Baseline* of the SCM ontology being relevant for SVN users, SVN does not support them. For SVN, everything is seen as directories or files in the repository. The users are free to assign the semantics for their own. In practice, the users generally treat branches and baselines as copied directories. Another important point regards the concept *Revision* in SVN. It is an important concept in the SCM domain, because it represents the states of the software being developed (its configuration). However, as said before, this concept does not appear in the selected ontology.

TABLE I. VERTICAL MAPPINGS

Subversion	SCM Ontology
Checkout	Checkout
Commit	Checkin
Unversioned Item	Unversioned Item
Repository Item	Version
WC Item	Copy
Repository	Repository
Revision	-
Working Copy	-

Once performed the mappings between concepts, we performed the mappings between relationships in order to assign semantics to the relationships in the SVN model. Figure 7 shows an example of a VM between relations.

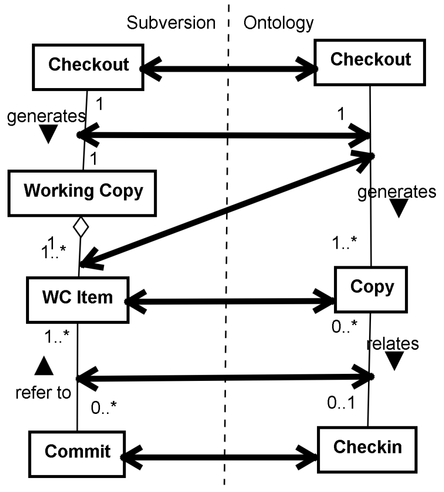

Figure 7. Example of vertical mapping between relationships.

The right of the figure shows part of the ontology. The left shows part of the conceptual model of SVN. Arrows indicate the mappings between relationships and also between concepts. The mappings between relationships are based on the mappings between concepts. For instance, the mapping between the relationships "*relates*" in the ontology and "*refers to*" in SVN was based on the mappings between the concepts *Copy* (ontology) and *WC Item* (SVN) and between the concepts *Checkin* (ontology) X *Commit* (SVN). Another mapping that occurs in Figure 7 is between the relation "*generates*" of the ontology and the relationships "*generates*" and "*composed of*" in the SVN model. Note that in this case, the two SVN relationships only have a meaning according to the ontology if they are considered together.

D. Modeling the Integration

Once the structural models were semantically annotated, integration modeling started. In this step, first, the structural conceptual model of the integration was developed. It is partially presented in Figure 8. This model is quite similar to the ontology. It is basically the conceptual model of the ontology plus some concepts related to SVN (represented in gray), other related to CM-ODE (not shown in the fragment of Figure 8) and some new concepts introduced in the integration model to address specific issues of the integration (represented in dark gray).

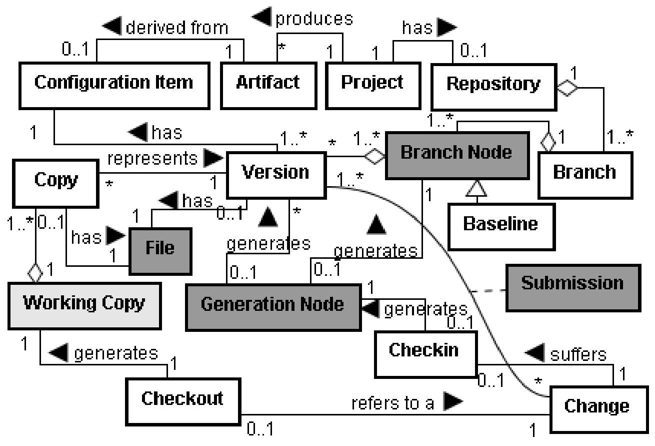

Figure 8. A Fragment of the Structural Part of the Integration Model .

Regarding the concepts added to the integration model, *Branch Node* represents a state of a branch composed by a set of versions. *File* represents a file that can be the content of a version in the repository, the content of a copy in a working copy or an unversioned file. *Generation Node* represents the event generating a branch node. It can result from a check in. Finally, *Submission* represents the versions selected to be changed in a change.

With the initial structural integration model in hands, HMs were performed. In this step, the structural models of SVN and CM-ODE were mapped to the integration model. These mappings were similar to the VMs, because most of the HMs were based on the VMs. The differences occur basically to treat tools' concepts that do not have a counterpart in the ontology. This is the case of *Working Copy* from both SVN and integration model. They are linked one to another by a HM, since they do not have VMs to the ontology. The HMs without semantic also occurred between the concepts Revision (SVN) and Branch Node (Integration), which represents the state of a branch.

Once established the HMs in the structural level, we went ahead to address behavioral aspects of the integration. Initially, a behavioral model of integration was created. It was based mainly in behavioral models of the tools and in the SCM process model defined at LabES, and involves the following activities:

- Create Configuration Item (CI)
- Control Configuration (Create Baseline, Create Branch)
- Control Change (Request Change, Evaluate Change)
- Perform Change (Checkout, Change, Check in)

With the behavioral model of integration in hands, we performed the HMs between its tasks and functionalities of SVN and CM-ODE. TABLE II. shows some of the behavioral HMs between the SVN functionalities and the integration model. E.g., *Create Branch* and *Create Baseline* (from integration model) were mapped to *Copy* (from SVN). The *Copy* functionality of SVN can be used to make copies of repository items (directly in the repository) generating other repository items. So, it can be used to create branches or baselines, since they are represented as directories in SVN. Besides, *Create Configuration Item* (integration model) was mapped to *Import* (SVN). This SVN functionality is responsible to place *Unversioned Items* under version control, generating new repository items.

TABLE II. BEHAVIORAL HORIZONTAL MAPPINGS

Tasks of Integration Model	SVN Functionalities
Create CI	Import
Create Branch	Copy (of a Repository Directory)
Create Baseline	Copy (of Repository Directory)
Checkout	Checkout
Check in	Commit

For assisting in the establishment of mappings between functionalities and tasks, we also performed mappings of their inputs and outputs. Figure 9. shows an example of these mappings in the case of the mapping between the function Import (SVN) and the task *Create CI* (integration model).

To perform the task *Create CI*, the following items should be informed: the artifact (*art*) from which the configuration item (*ci*) is derived, the branch (*bra*) where the *ci*'s versions should be located, and the file (*file*) that represents the *ci*'s content. As SVN is used to control the versions of the configuration items, the *file* should be added to the SVN repository using the functionality Import.

As shown in Figure 9, there are HMs between the inputs, outputs and the concepts from which they are instances. Regarding the mapping between the inputs, there are only two: (*file*, *dir*) and (*bra*, *target_dir*). In the first, the role of these objects is similar. Both are responsible to inform what is being placed under version control. However, the concepts they instantiate are not the same: *file* is an instance of *File*, where *dir* is an instance of *Unversioned Directory*. Nevertheless, they are related concepts. Unversioned Directories can be composed of Unversioned Files, which are mapped to the concept *File* in the integration model.

Regarding the mapping (*bra, target_dir*), they also have similar roles, since these objects inform where the configuration item will be stored. In addition, they instantiate concepts that are horizontally mapped (*Branch* and *Repository Directory*, respectively), reinforcing the existence of the mapping between the inputs.

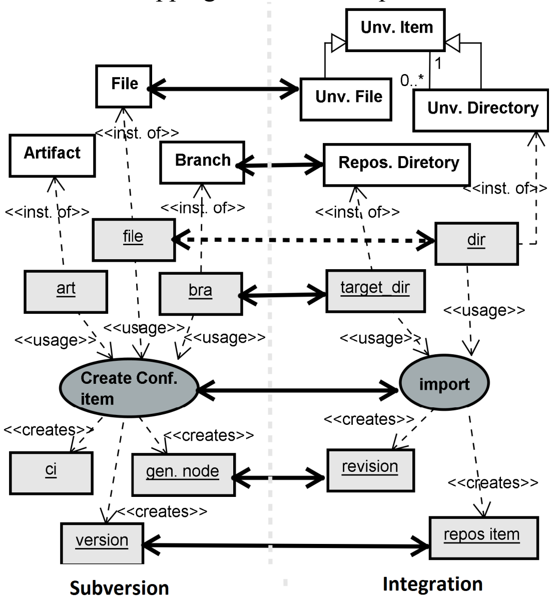

Figure 9. Example of input and output mappings.

E. Integration Design and Implementation

Once developed the integration model, we performed the design and implementation of an integration solution. This solution consists of a mediator that translates data exchanged between the tools and that controls the invocation of the tools' services. To access the SVN functionalities, the mediator uses the svnkit library (http://svnkit.com/), which provides the functionalities of SVN in Java, the same programming language used to implement CM-ODE and the mediator.

To translate data between the tools, the mediator stores a set of information about the mappings between instances of concepts, allowing identifying the corresponding data of one tool to the other. For example, from an instance of *Branch* in CM-ODE, the mediator stores the location of the corresponding directory in SVN. From an instance of *Version* in CM-ODE, in turn, the mediator stores the location of the corresponding file in the SVN repository and the revision to which it is related.

V. Related Works

As pointed by Guarino [11], regarding a temporal dimension, ontologies can be used at development time or at run time. OBA-SI uses ontologies at the development time of an integration solution, focusing on the conceptual level of the integration. Other approaches, such as ODSOI [2], use ontologies at run time and, because of that, they are more tied to the technological aspects. Following some of the approaches that are more similar to OBA-SI are briefly discussed.

Izza et al. [2,1] proposed ODSOI (Ontology-Driven Service-Oriented Integration). ODSOI aims to extend the web services technology by means of a semantic layer offering semantic services. This layer mainly defines the service semantics and performs semantic mediation in the context of Enterprise Application Integration (EIA). Also, ODSOI proposes an integration process. As OBA-SI, ODSOI covers data, service and process layers, although the first prototype of the architecture was restricted to data integration. Concerning the integration process of ODSOI, it is quite different from the one proposed in OBA-SI, since the first aims to integrate services at run time.

Tektonidis et al. [5] developed ONAR, an ontology-based framework for EAI. ONAR utilizes semantic web technologies in order to enrich the semantics of the exchanged information. Its integration process considers aspects of both development and run time. Concerning the development time, the ONAR integration process proposes two phases that are very in line with OBA-SI, namely the conceptualization phase and the association phase. In the conceptualization phase, an integration case is defined (analogous to our integration scenario), and entities that need to participate in the integration case are conceptualized in OWL ontologies. The set of the conceptualizations gives rise to the conceptual schema of the integration In the association phase, the analyst associates the concepts of the ontology to the resources of the information systems repositories that participate in the integration, in an approach similar to the one proposed by OBA-SI. However, there are also differences. First, for assigning semantics to model elements, OBA-SI considers domain reference ontologies in opposition to OWL ontologies. As pointed by Guizzardi et al. [17], a reference ontology should be produced, aiming at representing the subject domain regardless of computational requirements. The main goal of a reference ontology is to help modelers to externalize their tacit knowledge about the domain, to make their ontological commitments explicit to support meaning negotiation, and to afford the tasks of domain communication, learning and problem solving as best as possible. The same reference ontology may further give rise to different lightweight ontologies in different languages (such as OWL), thus satisfying different sets of non-functional requirements. Defining the most suitable language for codifying a reference ontology is then a choice to be made at the design phase, by taking both the end-application purpose and the tradeoff between expressivity and computational tractability into account. Second, concerning the association phase, OBA-SI gives several guidelines on how to perform them, while ONAR only says that concepts of the ontology should be associated to the resources of the information systems repositories. Finally, ONAR addresses integration in the data and service layers, while OBA-SI includes also aspects related to the process layer.

A very interesting approach is proposed in [5]. Among the approaches we have inspected, this is one of the most complete among the ones that consider integration analysis. A striking feature of this approach is that it uses MDA techniques. Like OBA-SI, this approach considers data, service and process layers, and it also occurs at a high level of abstraction, through platform-independent models (PIMs). The extraction of PIMs that represent the tools' models is done automatically through processing platform-specific models (PSMs) that implement the tools. In this approach, ontologies are also used to assign semantics to the models that represent the tools, such as in OBA-SI. The approach also has a general model to model the integration. This integration model is also independent of platform (PIM). One drawback of this approach is that it does not focus much attention on the semantic mappings. Thus, the semantic integration occurs in a simpler way, relating only concepts. In OBA-IS the semantic assignment seems to be more complete. It occurs between concepts, relations, services and activities, and their inputs and outputs.

VI. Conclusions And Future Work

This paper presented OBA-SI, an Ontology-Based Approach for Semantic Integration. OBA-SI proposes an integration process with focus on the conceptual level of the integration.

Although OBA-SI is based on semantic mappings to reach agreements, the mapping procedures themselves are not its focus. OBA-SI's main concern is the methodological steps of a semantic integration at the conceptual level. Thus, OBA-SI can be combined with other approaches, especially those concerned with semantic mappings, such as [4].

Regarding OBA-SI integration process, its emphasis is on the analysis phase. This task is eminently subjective and thus it is difficult to perform automatic mappings. Thus, the process is manual. We claim that this is not a problem at all, since OBA-SI is an approach for integrating existing organizational applications in an integration scenario previously defined. It would be a great limitation if we were interested in integrating services on the fly.

We are now working on integrating tools to support a project management process. With this, and probably other applications of OBA-SI, we expected to improve our approach. From its initial usage, we had more difficulties when dealing with behavioral mappings. While in the structural mappings we had the support of a reference domain ontology, the behavioral mappings modeling lacks the support of a process reference model. Thus, we are now studying how to use task ontologies [11] to support this activity. Since task ontologies describe generic tasks, they could be used to assign meaning to services and activities in a business process model, in the same way that the concepts of the domain ontology are used to assign semantics to structural model elements. So, we could perform vertical mappings between services and activities in a business process model based on the tasks defined in a task ontology.

Acknowledgment

This research is funded by the Brazilian Research Funding Agencies FAPES (Process Number 45444080/09) and CNPq (Process Number 481906/2009-6).

References

[1] Izza, S.; "Integration of industrial information systems from syntactic to semantic integration approaches", Enterprise Information Systems, Vol. 3, No. 1, February 2009, pp. 1-57.

[2] Izza, S.; Vincent, L., Burlat, P.; "Ontology-Based Approach for Application Integration", First International Conference on Interoperability of Enterprise Software and Applications – Interop-ESA´05, Geneva, Switzerland, February 2005.

[3] Bouras, A., Gouvas, P., Friesen, A., Pantelopoulos, S., Alexakis, S., Mentzas, G.; "Business Process Fusion based on Semantically-enabled Service-Oriented Business Applications", In: Interoperability for Enterprise Software and Applications Conference (I-ESA´06), March 2006, Bordeaux, France. Great Britain: ISTE, pp. 157–168.

[4] Pokraev, S; Quartel, D.; Steen, M. W. A.; Reichert, M; "Semantic Service Modeling: Enabling System Interoperability", Proc. Int'l Conf. on Interoperability for Enterprise Software and Applications (I-ESA'06), Bordeaux, France, March 2006.

[5] Tektonidis, D., Bokma, A., Oatley, G., Salampasis, M.; "ONAR - An Ontology-based Service Oriented Application Integration Framework", In: Proceedings of the 1st International Conference on Interoperability of Enterprise Software and Applications, Geneva, Switzerland, February 2005. London: Springer-Verlag, 65–74.

[6] Jin, D; Cordy, J.R.; "A Service-Sharing Methodology for Integrating COTS-Based Software Systems" 5th International Conference on COTS-Based Software Systems, Orlando, Florida, 2006.

[7] Collins-Sussman B., Fitzptrick B. W., Pilato C. M.; Version Control with Subversion, 2009. Available at http://svnbook.red-bean.com/

[8] IEEE, *SWEBOK - Guide to the Software Engineering Body of Knowledge*, 2004 Version, IEEE Computer Society, 2004.

[9] Bussler, C.; "The Role of Semantic Web Technology in Enterprise Application Integration", IEEE Data Engineering Bulletin, Vol. 26, Issue 4, 2003, pp 62-68.

[10] Park, J.; Ram, S.; "Information Systems Interoperability: What lies Beneath?", ACM Transactions on Information Systems, vol. 22, pg. 595-632, 2004.

[11] Guarino, N.; Formal Ontology and Information Systems. in: N. Guarino, (Ed.) Formal Ontology in Information Systems. pp. 3-15, IOS Press, Amsterdam Netherlands.

[12] Guizzardi,G.; *Ontological Foundations for Structural Conceptual Models*, Universal Press, The Netherlands, 2005.

[13] Guizzardi, G.; "On Ontology, ontologies, Conceptualizations, Modeling Languages and (Meta)Models", In *Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV*. IOS Press, Amsterdam, 2007.

[14] Falbo, R.A., Ruy, F.B., Moro, R.D.; "Using Ontologies to Add Semantics to a Software Engineering Environment". In: 17th International Conference on Software Engineering and Knowledge Engineering - SEKE'2005, Taipei, China, 2005. p. 151-156.

[15] Arantes, L.O., Falbo, R.A., Guizzardi, G. "Evolving a Software Configuration Management", 2nd Workshop on Ontologies and Metamodeling Software and Data Engineering, Brazil, 2007.

[16] Guizzardi, G., Lopes, M., Baião, F.A., Falbo, R.A.; "On the Importance of Truly Ontological Distinctions for Ontology Representation Languages: An Industrial Case Study in the Domain of Oil and Gas". In: 14th International Conference on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD), Amsterdam, The Netherlands, 2009. p. 1-13.