Integrating Tools for Supporting Software Project Time Management: An Ontology-based Approach

Glaice Kelly da Silva Quirino, Ricardo de Almeida Falbo

Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department – Federal University of Espírito Santo (UFES) – Vitória, ES – Brazil.

gksquirino@inf.ufes.br, falbo@inf.ufes.br

Abstract. *Project Management is a complex process involving several activities and a large volume of information. There are several tools offering partial solutions for supporting this process, increasing the need for integrating some of them, in order to provide a fuller support to the Project Management process. This paper presents an integration initiative aiming at semantically integrating dotProject, a web-based project management application, to ODE, an Ontology-based software Development Environment. This integration initiative focuses on the project time management, mainly for supporting the following activities: definition of project activities, allocation of human resource to these activities, and scheduling. This initiative was conducted partially following OBA-SI, an Ontology-Based Approach for Semantic Integration, and it was done using a domain ontology built from a Software Process Ontology Pattern Language.*

Keywords. *Project Management, Semantic Integration, Ontologies, Semantic Interoperability, Ontology Pattern Language.*

1. Introduction

Project Management is a process that aims at establishing and evolving project plans, defining activities, resources and responsibilities for the project, as well as providing information to assess the actual achievement and progress against the plans, in order to control the project execution [ISO/IEC 2008]. In order to support this process, several tools are needed, such as tools for project control, software process definition, resource allocation and scheduling. Ideally, these tools should work together, exchanging data and services. The use of several tools to support the same process without some degree of integration between them brings many problems such as rework and inconsistency.

ODE (Ontology-based Software Development Environment) [Falbo et al. 2005] is a process-centered Software Engineering Environment (SEE), which has been developed grounded on ontologies. ODE has several tools, some of them supporting the Project Management process, such as tools to support software process definition, resource allocation, estimation, and risk analysis. However, there are still project management activities that are not supported by ODE, such as project scheduling and tracking.

In order to increase the support offered by a SEE, two main approaches are typically used: (i) developing new tools already integrated to the SEE; (ii) integrating to

the SEE tools already available. In the first approach, the same group develops new tools as integrated pieces of the SEE, continuously expanding its functionality. In the second approach, the focus is on integrating tools produced by others. The main challenge in this case is to establish a common understanding of the meaning of the terms used by the tools, and to solve semantic conflicts between these tools and the SEE in which they should be integrated.

In the context of the ODE Project, the first approach was predominant in the first years of the project. This approach was in line with the main design premise originally established for the project, namely: if the tools in a SEE are built based on ontologies, integration can be more easily achieved, since the same set of ontologies is used for building different tools supporting related software engineering activities [Falbo et al. 2005]. However, more recently, we realized that adopting only this approach is not enough. Nowadays there are many free software tools available, and sometimes it is more appropriate to integrate an existing one to ODE, instead of developing a new tool. Moreover, this is especially important for allowing software organizations continue to use some tools to which they are already accustomed, preserving their organizational culture. Thus, we started to work also in integrating existing tools to ODE. The first effort in this direction was done to integrate Subversion (a version control system) to ODE [Calhau and Falbo 2010]. Since ODE is an ontology-based SEE, the integration approach adopted focus on the use of ontology 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, as pointed by Jasper and Uschold (1999).

Aligned to this new direction of the ODE Project, we decided to improve the support to Project Management in ODE by means of integrating a tool that could provide functionalities for scheduling software projects. Since the current version of ODE runs in the Web platform, we looked for a web-based free project management tool that provides such functionality, and, after comparing some of them, we decided to select dotProject[1]. dotProject is a free open source web-based project management system, which basically provides functionalities for managing tasks, schedules and communication with contacts. As pointed in the main page of dotProject[1], it does not provide all the functionalities required for managing projects.

However, each application (ODE and dotProject) 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. As pointed by Izza (2009), this heterogeneity is considered one of the major difficulties in the integration problem. In this context, the adoption of an approach that helps reduce the complexity of this task is important.

In [Calhau and Falbo 2010], Calhau and Falbo developed OBA-SI (Ontology-Based Approach for Semantic Integration), an approach to semantic integration of systems that concentrates efforts on requirements analysis and conceptual modeling. In this approach, semantic integration is performed in a high abstraction level, promoting semantic agreement between the systems at the conceptual level. For this, ontologies are used to assign semantics to items shared between systems, proposing an integration

[1]http://docs.dotproject.net/index.php?title=Main_Page

process independent of technology and covering three layers of integration: data, services and process. Once OBA-SI is very aligned with the premises of ODE Project (using ontologies for building and integrating tools), we decided to adopt it in our integration initiative. However, since dotProject does not provide an API (Application Programming Interface) providing services, we decided to address integration only in the data layer.

In this paper, we present the semantic integration of dotProject to ODE, following OBA-SI. First, we defined the integration requirements, defining the integration scenario. Our focus is on integrating functionalities supporting the Project Time Management process, as defined in the PMBOK [PMI 2008], which involves the following activities: define project activities, sequence activities, estimate activity resources, estimate activity duration, and develop schedule. Second, we developed an ontology addressing this universe of discourse to be used to map the concepts and relations of both systems. This ontology, called Project Time Management Ontology (PTMO), was built by assembling patterns of the Software Process Ontology Pattern Language (SP-OPL) [Falbo et al. 2013]. Besides, we retrieved the structural conceptual models of the tools to be integrated. ODE's conceptual model was already available; dotProject's conceptual model, on the other hand, had to be excavated. With the ontology and the conceptual models of the tools to be integrated in hands, we established mappings between them, in order to assign semantics to the structural models of the tools. As a result, we achieved an integration model, which were used to design a mediator. Finally, we implemented the mediator application, integrating dotProject to ODE.

This paper is organized as follows. In Section 2, we present a brief review of the literature on topics relevant to the context of this work, namely: Project Management and Systems Integration. In Section 3, we present the PTMO ontology, discussing how it was built from SP-OPL. In Section 4, we present the semantic integration of dotProject to ODE, using OBA-SI. Section 5 discusses related works. Finally, in Section 6, we present the final considerations of this paper.

2. Project Management and Semantic Integration

According to the PMBOK [PMI 2008], "Project management is the application of knowledge, skills, tools and techniques to project activities in order to meet project requirements". It is a complex process that involves several sub-processes, among them the project planning is a major one. The PMBOK groups planning related activities in the Planning Process Group, which involves the processes performed to establish the project scope, define its goals, and develop the course of actions required to attain them. This group includes processes for Scope Management, Time Management, Cost Management, Quality Management, Human Resource Management, Communication Management, and Risk Management, among others.

As pointed in the introduction of this paper, ODE provides partial support for some of these processes, namely: Scope Management, Time Management, Quality Management, and Risk Management. Thus, we have worked to improve this support by means of developing new tools to ODE, or integrating existing tools to it. In this paper our focus is on the Project Time Management process, which includes five very inter-

related activities [PMI 2008]: define project activities, sequence activities, estimate activity resources, estimate activity duration, and develop schedule. ODE provides only partial support to this process, since it does not help in developing schedules. To improve this support, we decided to integrate dotProject to ODE.

However, when integrating these systems, conflicts arise. They were developed by different groups, in different points in time, and they have no concern with integration. Thus, they can be said heterogeneous, autonomous and distributed systems [Izza 2009]. Heterogeneous refers to the fact that each application implements its own data model defining its concepts in its own way. Autonomous means that each application runs independently of the other. Distributed means that they implement their data model in their own data repository and this repository is not shared with the other tool [Izza 2009].

In particular, there are semantic conflicts, and integration in the semantic level should take the intended meaning of the concepts in a data schema or in operation signatures into account [Izza 2009]. Basically, semantic conflicts occur because applications do not share a common conceptualization. In this context, ontologies can be used to deal with meaning and semantic heterogeneities. Ontologies can be used as an interlingua to map concepts and services used by the different systems, in a scenario of access to data and services via a shared ontology [Jasper and Uschold 1999]. Moreover, semantic integration should occur at the knowledge level [Park and Ram 2004], considering that applications must share the meaning of their terminologies.

Among the various approaches for integrating systems that consider semantics to integrate systems, there is OBA-SI (Ontology-Based Approach for Semantic Integration) [Calhau and Falbo 2010]. This approach considers that the integration process is analogous to the software development process, consisting of phases of requirements gathering and analysis, design, implementation, testing and deployment. OBA-SI focuses on the integration analysis phase, in which the semantics may be set. During this phase, semantic mappings are made between the conceptual models of the tools being integrated, using an ontology to assign meaning. This ontology should be a reference ontology, i.e., a solution-independent specification making a clear and precise description of the domain entities for the purposes of communication, learning and problem-solving [Guizzardi 2007].

The integration process of OBA-SI begins with integration requirements elicitation phase, when the goals and requirements must be established. In this phase, we need to define the activities of the business process to be supported, the systems to be integrated to support them, and the domain where the integration takes place. The output of this phase is the integration scenario. Once defined the integration scenario, integration analysis can be performed. Figure 1 shows the activities involved in this phase. First, the conceptual models of the tools to be integrated should be retrieved, as well as a reference ontology of the domain where the integration takes place should be selected or developed. Following, concepts and relations of the conceptual models of the tools to be integrated should be mapped to the concepts and relations of the ontology. These mappings are said vertical mappings. Once the structural models are semantically annotated, the integration model is developed. This model is mainly based on the domain ontology, but it can also include elements of the tools being integrated that do

not have a counterpart in the ontology. These elements, if present in both tools, should be directly mapped, by means of horizontal mappings [Calhau and Falbo 2010].

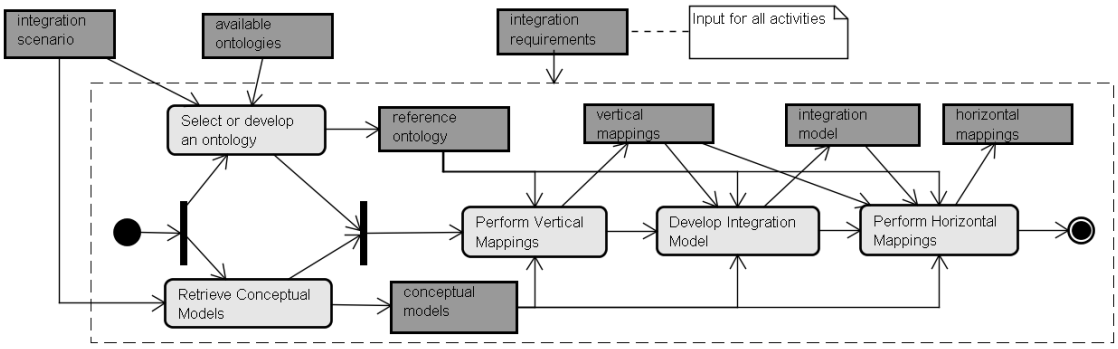

Figure 1. OBA-SI Analysis Phase

With the integration model in hand, an integration solution can be designed and implemented. There are several ways to design and implement an integration solution, and OBA-SI does not commit to any specific integration solution, although it proposes some guidelines for maintaining the semantic consistency in these steps.

In order to integrate dotProject to ODE adopting OBA-SI, we needed a reference domain ontology regarding project time management. Since several planning-related ODE's tools were developed based on the version of the Software Process Ontology presented in [Falbo and Bertollo 2009], we decided to use it. However, this ontology was reengineered in [Bringuente et al. 2011] to become aligned to the Unified Foundational Ontology [Guizzardi et al. 2008], and more recently it was defined as an Ontology Pattern Language (OPL)[2] [Falbo et al. 2013]. Thus, we decided to build a Project Time Management Ontology (PTMO) by assembling patterns of the Software Process Ontology Pattern Language (SP-OPL).

3. Using the Software Process Ontology Pattern Language to Develop a Project Time Management Ontology

SP-OPL is an OPL for the Software Process application domain. The main problem areas addressed by SP-OPL are Standard Software Process Definition, Project Process Definition and Scheduling, Resource Allocation, and Software Process Execution. In the next section, we discuss how we developed PTMO from SP-OPL.

SP-OPL has three entry points[3], depending on the focus of the ontology engineer. Considering our purposes, our entry point was the SPP (Software Process Planning) pattern, which considers the planning of the project process from scratch (i.e., without being based on a standard software process). The SPP pattern represents how a software process is planned in terms of sub-processes and activities, as well as it deals

[2] An OPL aims to provide holistic support for using Domain-related Ontology Patterns (DROPs) in ontology development for a specific application domain. It provides explicit guidance on what problems can arise in that domain, informs the order to address these problems, and suggests one or more patterns to solve each specific problem [Falbo et al. 2013].

[3] Each entry point allows the ontology engineer to focus on certain problems (and thus using certain patterns), disregarding others [Falbo et al. 2013].

with activity sequencing [Falbo et al. 2013]. Once defined the project activities, it is necessary to schedule the project and define the human roles required for performing the activities. To handle these aspects, the patterns PSCH (Process Scheduling) and HRP (Human Role Planning) were selected. The first defines the time window for project processes and activities, while the second defines the human roles responsible for performing a project activity [Falbo et al. 2013].

Human resource allocation was treated by reusing the PTD (Project Team Definition) and TDHRA (Team-dependent Human Resource Allocation) patterns. The PTD pattern regards the human resources that are member of a project team; the TDHRA pattern deals with allocating human resources to project activities, considering team allocation constraints. These patterns are in the Resource Allocation group of patterns [Falbo et al. 2013].

Finally, regarding process execution, we reused the PAET (Process and Activity Execution and Tracking) and HRPAT (Human Resource Participation and Tracking) patterns. The first registers the occurrences of processes and activities, taking into account the planned processes and activities, allowing to track the execution against to what was previously planned; the second registers the participation of human resources in activity occurrences, taking into account the existence of a prior allocation of these resources to planned activities [Falbo et al. 2013].

Figure 2 shows the conceptual model of the PTMO, resulting from the assembly of these patterns. This conceptual model is written in OntoUML, a UML profile that enables modelers to make finer-grained modeling distinctions between different types of classes and relations according to ontological distinctions put forth by the ontology of endurants of the Unified Foundational Ontology (UFO-A) [Guizzardi 2005]. Thus, the stereotypes shown in Figure 2 represent types of classes and relations as defined in UFO-A.

Project Processes are defined for a *Project*. There are two types of *Project Processes*: *General Project Process* and *Specific Project Process*. The first is the overall process defined for the *Project*. It consists of *Specific Project Processes*, thus allowing defining sub-processes. The second is composed by *Project Activities*, which may be, *Simple Project Activity* or *Composite Project Activity*. These activities are to be performed by human resources playing certain *Human Roles*. For example, a requirements specification activity defined for a project requires a requirements engineer to perform it. Once the project processes and activities are defined for a project, it is possible to establish the start and end dates for them, giving rise to *Scheduled Processes* and *Scheduled Activities*, respectively.

A *Human Resource Allocation* is the assignment of a *Scheduled Activity* to a *Human Resource* for playing a specific *Human Role*. A *Human Resource Allocation* depends on a *Project Team Allocation*, which allocates the *Human Resource* to the *Project Team* and indicates the role he/she will play in this team.

When scheduled processes and activities are executed, they generate *Process* and *Activity Occurrences*, respectively. Analogously to project processes, there are two types of processes occurrences: *General Process Occurrence*, which corresponds to the execution of the process as a whole, and the *Specific Process Occurrence*, which

corresponds to the execution of a particular project process. Similarly, there are two types of Activity Occurrences: *Simple Activity Occurrence*, which is an atomic action, and *Composite Activity Occurrence*, which is composed of other activity occurrences. Finally, when activities are performed (Activity Occurrence), they involve various *Human Resource Participations*, which refers to the participation of a single *Human Resource*.

Considering the *start* and *end dates* of *Scheduled Processes* and *Activities*, *Human Resource Allocations*, *Process* and *Activity Occurrences*, and *Human Resource Participations*, it is possible to track the project progress, contrasting what was planned (scheduled) with what actually happened (occurrences and participations).

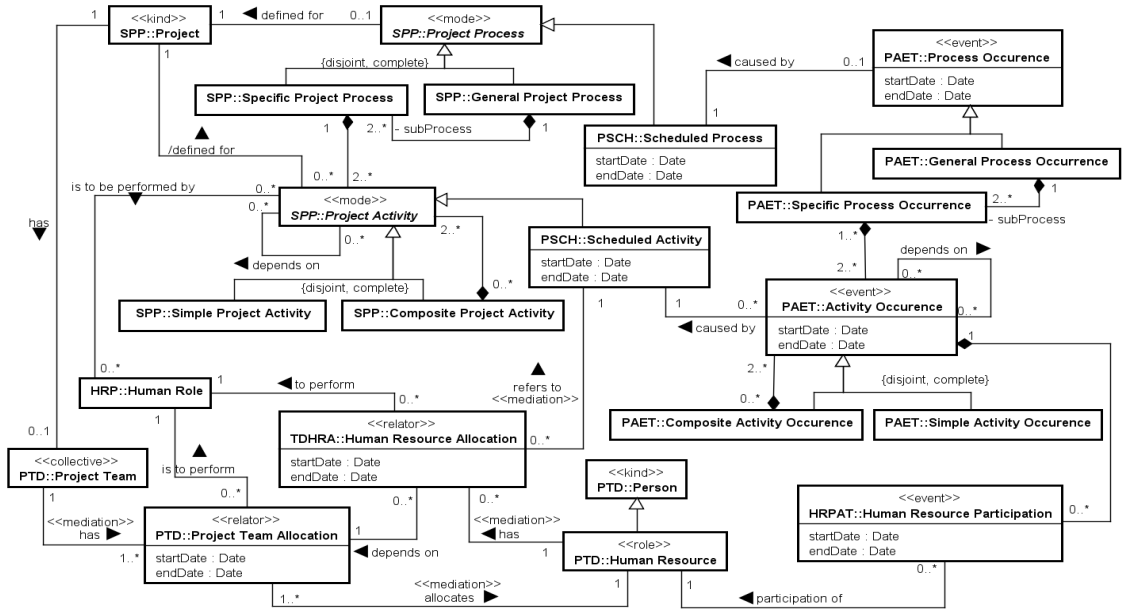

Figure 2. Project Time Management Ontology

4. Semantic Integration of` Software Project Management Tools

Once defined the integration scenario and the reference ontology, the required structural conceptual models had to be retrieved. Two different approaches were used. Since ODE was developed at NEMO, its structural conceptual model was available. On the other hand, the conceptual model of dotProject had to be excavated.

Figure 3 presents a fragment of ODE's structural conceptual model. It is presented only partially, due to space limitations. In ODE, a *General Project Process* is defined for a *Project*. This *General Project Process* is decomposed into *Specific Project Processes* that, in turn, are decomposed into *Activities*. During project activity definition, several process assets (resources, artifacts required and produced and so on) are defined for each activity, as well as sub-activities and dependencies between activities. All this information is registered in the *Activity Definition* class, which register also the scheduled start and end dates for the activity. For each activity, human resources can be allocated (*HRAllocation*), according to the demands informed during the process definition (*HRDemand*). When an activity is initialized, its actual start date is registered in the *Activity Execution* class, which represents the actual occurrence of

the previously planned activity. When a human resource spends some hours performing an activity to which she has been allocated, the *Expended Effort* must be registered.

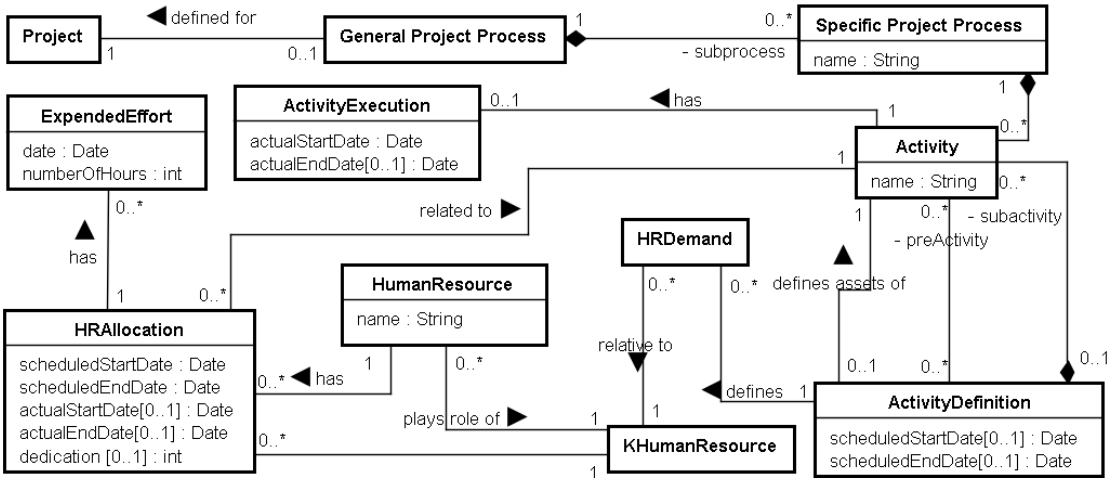

Figure 3. A fragment of ODE's Class Diagram

With respect to dotProject, we had to excavate its structural conceptual model. This was done by analyzing its database schema database and its graphical interface. Figure 4 shows a fragment of the structural conceptual model resulting from this step. As this figure shows, in dotProject, a *Project* has *Tasks*, to which *Contacts* can be allocated. *Tasks* can have sub-tasks and may depend on other tasks. Any events associated to a task can be registered by means of *Task Logs*.

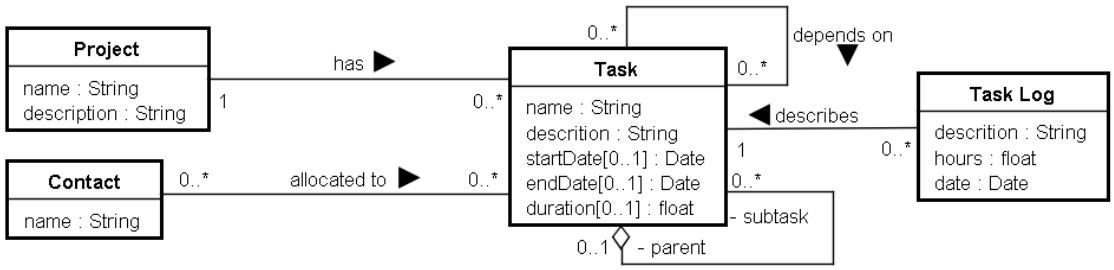

Figure 4. A fragment of dotProject's Class Diagram.

After retrieving the conceptual models of the tools, the next step is to assign semantics to their concepts and relationships by mapping them to concepts and relations of the reference domain ontology. These mappings, said vertical mappings [Calhau and Falbo 2010], allows comparing the concepts of the systems involved. Table 1 shows part of vertical mappings established to link the concepts of ODE and dotProject to the concepts of the PTMO ontology.

Project in ODE and dotProject are directly mapped to the concept of *Project* in PTMO, as well as *Human Resource* in ODE and *Contact* in dotProject that are directly mapped to the concept of *Human Resource* in PTMO. However, most of the concepts used in the tools are not directly mapped to a concept in PTMO. Contrariwise, in most cases, we need to consider attributes or relationships between classes to establish the same semantics of a concept in PTMO. For instance, the concept of *Simple Project Activity* in PTMO corresponds to a *Task* that does not have subtasks associated to it in

dotProject (Task.subactivity = null). In ODE, in turn, there are two concepts (*Activity* and *ActivityDefinition*) that map to the concept of *Project Activity* in PMTO. In order to know if a project activity is a simple or a composite project activity, it is necessary to see if the *ActivityDefinition* defines sub-activities for the corresponding *Activity*.

Table 1. Vertical Mapping of concepts

PTMO Ontology	ODE	dotProject
Project	Project	Project
Human Resource	Human Resource	Contact
Human Resource Allocation	HRAllocation	---
Project Activity	Activity + ActivityDefinition	Task
Simple Project Activity	Activity + ActivityDefinition, if ActivityDefinition.subactivity = null.	Task, if Task.subtask = null.
Composite Project Activity	Activity + ActivityDefinition, if ActivityDefinition.subactivity != null.	Task, if Task.subtask!=null
Scheduled Activity	Activity + ActivityDefinition, if (ActivityDefinition.scheduledStartDate != null and Activity.ActivityExecution = null).	Task, if (Task.startDate!=null and Task.startDate > currentDate)
Activity Occurrence	Activity + ActivityExecution	Task, if (Task.startDate != null and Task.startDate ≤ currentDate)

These types of mappings (direct and indirect) can also be observed in the case of vertical mappings between relationships, as shown in Table 2. The relationship "*Human Resource Allocation – refers to – Scheduled Activity*" in PTMO is directly mapped to the relationship "*HRAllocation – related to – Activity*" in ODE; whereas for mapping the whole-part relation between *Composite Project Activity* and *Project Activity* in PTMO to ODE, we need to cross two associations: "*ActivityDefinition – defines assets of – Activities*" and "*Activity – is sub-activity of – ActivityDefinition*".

Table 2. Vertical mapping of relationships

Ontology	ODE	dotProject
Human Resource Allocation – refers to – Scheduled Activity	HRAllocation – related to – Activity	Contact – allocated to – Task
Human Resource – has – Human Resource Allocation	Human Resource – has – HRAllocation	
Composite Project Activity – is composed by – Project Activity	ActivityDefinition – defines assets of – Activity; and Activity – is sub-activity of – ActivityDefiniton	Task – parent – Task

Once the structural models were semantically annotated, integration modeling started. In this step, first, the integration model was developed. The integration model is basically the conceptual model of the ontology plus some concepts arising from dotProject and others coming from ODE that do not have a counterpart in the ontology model. Due to space limitations, we do not present the integration model here.

Regarding the concepts added to the integration model, *Activity Occurrence Log*, for instance, was added to represent the dotProject's class *Task Log*, since, through *task logs*, it is possible to register the participations of human resources in activities (*Human Resource Participation* in PTMO).

With the integration model in hands, horizontal mappings were performed. In this step, the concepts that do not have a counterpart in the ontology model, and thus were introduced only in the integration model, were mapped. For instance, *Activity Occurence Log* in the integration model was mapped to *Task Log* in dotProject, and the relationship "describes" between *Activity Occurence Log* and *Activity Occurence* was mapped to the relationship "describes" between *Task Log* and *Task*.

Once established the horizontal mappings, the integration analysis phase is concluded, and we can start to design and implement the integration solution. For ODE and dotProject to communicate, it is necessary that the shared elements are translated. For doing that, we develop a mediator, which is responsible for translating data between the systems, as shown in Figure 5. The mediator is located inside ODE, making easier the access to ODE's database. In order to access dotProject's database, we implemented an interface for external communication, called dpClient[4], which behaves as an API to dotProject, since did not find any API available for dotProject that fits the purpose of our work.

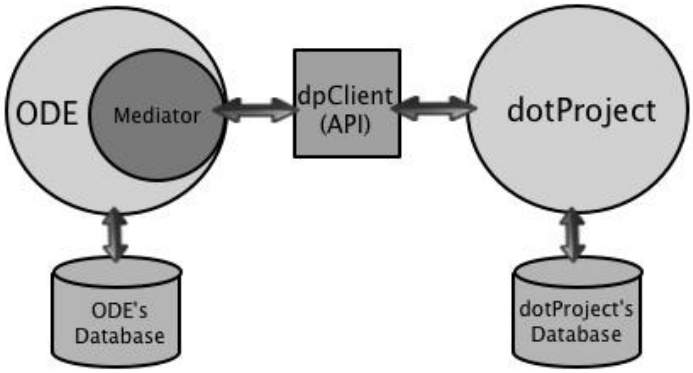

Figure 5. General Architecture of the solution for integrating dotProject to ODE

5. Related Works

Ontologies have been recognized as an important instrument for semantically integrating software applications [Izza 2009]. In the context of project management, Cheng et al. (2003) have used the Process Specification Language (PSL) Ontology for integrating Primavera P3, MS Project, Vite SimVision and 4D Viewer. Analogously to OBA-SI, the integration process used for building a distributed integration infrastructure also involves mappings between the concepts and relations of the involved systems and the concepts and relations of the PSL ontology. Moreover, there were also direct and indirect mappings, such as in our case (see examples given in the previous section). Although there are several similarities, there are also differences. The PSL Ontology deals with types of activities and activity occurrences. PTMO coverage, in turn, is

[4]https://github.com/glaice/dpclient

wider. It deals with the concepts of commitments (Project Process and Project Activity) and appointments (Scheduled Process and Scheduled Activities), in addition to the concept of occurrences (Process Occurrence and Activity Occurrence) as defined in UFO-C [Guizzardi et al. 2008]. Thus, it is possible to make finer distinctions, especially because the concepts of commitments and appointments are very important in project management. Regarding the technological solution for the integration, Cheng et al. develop wrappers for each application. The PSL wrappers are used to retrieve and transfer information between the applications, using PSL files. Not all scheduling and resource information is exchanged between the applications, since the granularity of the information may be different. Analogously, in our approach, the mediator is responsible for translating information from ODE to dotProject, using PTMO as an interlingua. However, in our case, changes made in dotProject do not reflect in ODE, since we have implemented the information exchange only from ODE to dotProject.

Concerning the methodological aspect, another work of semantic integration using OBA-SI is presented in [Calhau and Falbo 2010]. In their work, Calhau and Falbo integrated the version control system Subversion (SVN) to ODE. Access to SVN is worked by means of the svnkit library[5]. To translate data between the tools, a mediator stores information about the mappings between concepts and relationships of the tools being integrated and an ontology about the Software Configuration Management domain. Since in this work we also followed OBA-SI, the approach is quite similar.

6. Conclusions

This paper presented an initiative of semantically integrating dotProject, a web-based project management system, to ODE, an ontology-based Software Development Environment. This initiative was conducted partially following OBA-SI [Calhau and Falbo 2010], an Ontology-Based Approach for Semantic Integration, and it was done using a Project Time Management Ontology (PTMO), which was built from the Software Process Ontology Pattern Language [Falbo et al. 2013]. For implementing an integration solution, we developed a mediator responsible for exporting data from ODE to dotProject, allowing visualizing schedules in ODE, and thus providing a more complete support to the project management process in ODE.

We should highlight some limitations of our work. First, semantic integration is worked only in the data layer. Moreover, it occurs only from ODE to dotProject, i.e., data from ODE's database are passed to dotProject, but changes in dotProject's database are not reflected in ODE. Ideally, the integration should occur in both directions, and in other integration layers, especially in the service/message layer [Izza 2009]. Thus, there is room for adding new features to this work, or even integrating other tools in order to provide a wider support to the Project Management process.

Acknowledgments - This research is funded by the Brazilian Research Agencies FAPES/CNPq (PRONEX Grant 52272362/11).

[5] http://svnkit.com/

References

Bringuente, A. C. O., Falbo, R. A., Guizzardi, G. (2011), "Using a Foundational Ontology for Reengineering a Software Process Ontology". Journal of Information and Data Management, vol. 2, n. 3, pp. 511-526.

Calhau, R.F., Falbo, R.A. (2010), "An Ontology-Based Approach for Semantic Integration. Proceedings", Proc. 14th IEEE International Enterprise Distributed Object Computing Conference, Vitória, Brasil.

Cheng, J., Gruninger, M., Sriram, R. D., and Law, K. H., (2003), "Process Specification Language for Project Scheduling Information Exchange", International Journal of IT in Architecture, Engineering and Construction, vol. 1, n. 4, pp. 307 - 328.

Falbo, R. A., Ruy, F.B., Moro, R. (2005), "Using Ontologies to Add Semantics to a Software Engineering Environment". In: Proc. 17th International Conference on Software Engineering and Knowledge Engineering - SEKE'2005, Taipei, China.

Falbo, R. A., Bertollo, G. A (2009), "Software process ontology as a common vocabulary about software processes". International Journal of Business Process Integration and Management (IJBPIM), v. 4, p. 239-250.

Falbo, R. A., Barcellos, M.P., Nardi, J.C., and G. Guizzardi (2013), "Organizing Ontology Design Patterns as Ontology Pattern Languages," Proc. 10th Extended Semantic Web Conference, Montpellier, France.

Guizzardi, G. (2007), "On Ontology, Ontologies, Conceptualizations, Modeling Languages and (Meta) Models", In: Vasilecas, O., Edler, J., Caplinskas, A. (Org.). Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV, IOS Press, Amsterdam.

Guizzardi, G. (2005) Ontological Foundations for Structural Conceptual Models, University of Twente.

Guizzardi, G. Falbo, R.A. Guizzardi, R.S.S. (2008) "Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology", Proceedings of the XI Iberoamerican Workshop on Requirements Engineering and Software Environments, Recife, Brazil.

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

ISO/IEC (2008), ISO/IEC 12207: Systems and software engineering — Software life cycle processes, 2th edition.

Jasper, R., Uschold, M. (1999), "A Framework for Understanding and Classifying Ontology Applications", Proceedings of the IJCAI99 Workshop on Ontologies and Problem-Solving Methods, Stockholm, Sweden.

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

PMI (2008), A Guide to the Project Management Body of Knowledge (PMBOK Guide), 4th edition, Project Management Institute, Inc.