Using Knowledge Servers to Promote Knowledge Integration in Software Engineering Environments

Ricardo de Almeida Falbo [1,2]
Crediné Silva de Menezes [2]
Ana Regina C. da Rocha [1]

[1] COPPE/UFRJ
Computer Science Department
Federal University of Rio de Janeiro
Caixa Postal 68511 - CEP 21945-970
Rio de Janeiro - RJ - Brazil
e_mail: {rfalbo, darocha}@cos.ufrj.br

[2] CT/UFES
Computer Science Department
Federal University of Espírito Santo
Fernando Ferrari Avenue
CEP 29060-900 - Vitória - ES - Brazil
e_mail: {falbo, credine}@inf.ufes.br

Abstract

One of the recurring problems in Software Engineering Environments (SEEs) is the integration of tools. Three dimensions have been considered in this context: data, control and presentation. However, as the knowledge-based support in SEEs increases, we believe that another dimension must be considered: the knowledge integration. In this paper we propose the use of Knowledge Servers to promote knowledge integration in SEEs, offering domain knowledge components to be reused and shared by the tools. The Knowledge Server architecture is designed based on domain ontologies and task models.

1. Introduction

As the complexity of the software development process augments, it becomes essential to provide computer-based tools to support software engineers to perform their tasks. To be effective, however, these tools must work together in a Software Engineering Environment (SEE).

SEEs are built to support complex software development processes, providing a framework to integrate tools along three main dimensions: control, data and presentation. With the increase of knowledge-based support in SEEs, it becomes necessary to consider also a *knowledge integration* dimension[1].

We have been studying the knowledge integration in TABA Workstation [2] during the last years. TABA is a meta-environment that aims to instantiate SEEs based on the requirements of different projects, application domains and technologies.

In this paper, we present our current approach to deal with the knowledge integration problem. Basically, we advocate the use of Knowledge Servers to promote knowledge integration in SEEs. A Knowledge Server is an infrastructure that makes domain knowledge components available for reuse and sharing. Its architecture is designed based on domain ontologies and task models.

In the following sections, we first discuss the integration of tools in SEEs and describe briefly TABA Workstation. After, we discuss the role of ontologies in knowledge integration. Then, we define Knowledge Server and present a Knowledge Server developed for TABA Workstation, showing how to build intelligent assistants with it's use. Finally, we present our conclusions.

2. Tools' integration in SEEs

Most of the architectures and definitions of SEEs treat integration of tools as a three-dimension issue: control, data and presentation. Nevertheless, with the growing complexity of software processes, it becomes necessary to offer knowledge-based support for helping software engineers in their tasks. In this context, it is important to consider a fourth dimension: *knowledge integration* [1].

Knowledge-based tools (or intelligent assistants) usually has to communicate with each other while working in a SEE. So, it is essential that they share a common vocabulary and the same interpretation for the terms used. This is the main goal of the knowledge integration: to establish the semantics of the information exchanged between the various tools in a SEE.

Many efforts have been made to provide intelligent support for software process activities, such as requirement specification, system design and risk analysis, among others. Analyzing some of these efforts, we can notice that the development of knowledge-based tools obeys the following strategy: the knowledge is acquired and modelled for a specific purpose, and embedded in an intelligent assistant. Even in those SEEs that embody some kind of knowledge-based support, the approach is the same. However, this approach presents the following problems:

a) the knowledge can hardly be shared or reused;
b) each tool is built based on a specific conceptualization and uses its own vocabulary, making the communication between tools difficult;
c) there is usually a large portion of knowledge that is common to several tools, leading to redundancy and inconsistency;
d) knowledge acquisition is made from scratch for each new tool to be built, increasing the costs and lowering the productivity of the development of knowledge-based tools.

This tendency must be reverted in order to achieve truly integrated environments. Only with the knowledge integration properly considered, it is possible to achieve fully integrated SEEs.

To share knowledge, however, it is necessary to change the way tools are built. In the current SEEs (fig. 1(a)), the environment knowledge is given by the knowledge embedded in each of its tools. Indeed, what is required is the construction of a knowledge model for the environment that could be used by each of the tools (fig. 1(b)). In this case, the environment offers the knowledge-based support for its tools.

3. Knowledge integration in the TABA Workstation

The knowledge integration has been studied in the context of the TABA Workstation [2]. TABA is a prototype of a meta-SEE that allows the specification and instantiation of SEEs according to the particularities of specific software processes, application domains and technologies. This involves the definition of a software development process, and the selection of tools to be provided. The instantiated environments are used by software developers in the development of specific software products.

The first initiative to promote the knowledge integration in TABA Workstation focused on the integration of knowledge representation technologies [1]. This approach proved to be useful and needful, but not enough. Actually, it is only a way to facilitate the integration. Since it can be used to implement specific and non-reusable knowledge, it cannot be considered a solution to the knowledge integration problem. In fact, it does not address the problems enumerated in section 2, which are the heart of the matter. Thus, we had to look for a more consistent approach for integrating knowledge.

4. Using ontologies to achieve knowledge integration

The merits of an analysis of knowledge on a more abstract level than that of representation languages have been recognized since the publication of [3]. According to Newell, it is necessary to consider a level of discourse (the knowledge level) where the knowledge and the problem solving could be dealt independently of their possible implementations (the symbolic level). In the knowledge level, problem solving agents are characterized in terms of the action they can perform, their knowledge and their goals. A formalism is only a symbolic system that encodes a knowledge body.

Our first effort was placed on the symbolic level. For a successful knowledge integration, however, it must happen at the knowledge level too. The key point in this issue is the knowledge modelling. It is essential to shift the focus on the development of knowledge-based tools to an approach focusing on the sharing of common knowledge. This approach should emphasize the *development for reuse*, more specifically, knowledge reuse.

The main impediment for the knowledge sharing originates from the lack of a basic domain conceptualization over which the various tools can be built. Therefore, the use of ontologies arises as the key point to deal with knowledge integration. An *ontology* is a specification of a conceptualization [4], i.e., a description of concepts and relations that can exist for an agent or a community of agents. Basically, an ontology consists of concepts and relations, and their definitions, properties and constrains expressed as axioms.

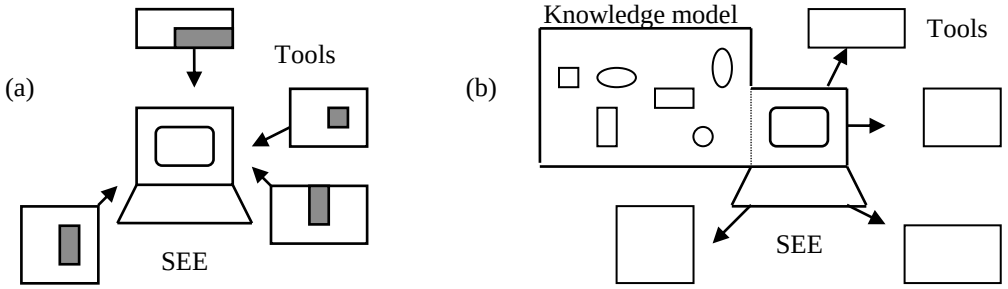


Figure 1 - (a) Existing Environments and (b) the knowledge integration in a SEE.

One of the main benefits of the use of ontologies in Knowledge-Based System (KBS) development is the opportunity to adopt a more productive strategy for the Knowledge Acquisition (KA). In the traditional

Knowledge Engineering, for each new application to be built, a new conceptualization is developed. In an ontology based approach, the KA can be accomplished in two stages. First, the general domain knowledge relevant to several applications should be elicited and specified as ontologies. These, in turn, are used to guide the second stage of the KA, when the particularities of a specific software application are considered. In this way, the same ontology can be used to guide the development of several software applications, minimizing the costs of KA and allowing knowledge sharing and reuse [5].

5. Knowledge Servers

To capture a knowledge model and its corresponding domain theory, we built an infrastructure to make knowledge components available to be reused and shared among tools. This infrastructure is a *Knowledge Server*. A Knowledge Server aims to provide an infrastructure for developing KBSs in a *specific domain*. That is, the Knowledge Server makes a knowledge framework about a domain of interest available. In the context of a SEE, a Knowledge Server serves as basis for building intelligent assistants, assuring a common semantics and allowing knowledge reuse and sharing.

When analyzing the knowledge involved in the design of a KBS, we can see that there are two basically distinct, although connected, problems: the task to be solved and the domain knowledge involved. The separation between the domain model and the model of the task to be performed improves the reusability of these models. Thus, to aid the construction of KBSs, a Knowledge Server must offer models of both domain and task.

Domain models must define a common vocabulary, with fixed semantics of the terms used in the domain. Therefore, we adopted an ontology-based representation for Knowledge Servers. The domain knowledge components provided by the Knowledge Server, called *knowledge modules* (KM), correspond to an ontology or one of its instantiation. While an ontology is a description of the concepts and relations used in a domain, an ontology instantiation is a set of statements about domain elements that are made using these concepts and relations. The knowledge base of an application committed with one or various ontologies is to be the conjunction of the corresponding knowledge modules, added to the specific application knowledge.

Task models are used to customize a general inference engine to specific classes of problems, like planning and diagnosis. From the task point of view, it is clear that types of problems in different domains share several common aspects. If the inference engine is a general-purpose one, the strategy to solve the problem has to be embedded as part of the knowledge base. Then, Knowledge Servers provide reasoner templates for the problem types that occur more frequently in the domain of interest. Instead of providing only representation systems and their general inference engines, there is a library of task models that can be instantiated and adapted to particular applications.

Since this work focus on the integration in the knowledge level, and not in the symbolic level, we do not discuss the use of multiple representation formalisms. In fact, we emphasize that the knowledge organization is the main factor for the knowledge integration in SEEs. Figure 2 shows the general architecture of Knowledge Servers composed by:

- a modular knowledge base, where each knowledge module (KM) is a reusable body of knowledge developed based on an ontology, and
- the inference engine of the chosen representation system, associated with templates that specialize it for the most frequently problem types in the domain.

The knowledge modules and the templates of problem types are the reusable components that the Knowledge Server offers to aid the process of building intelligent assistants for SEEs. In addition, these components are strongly connected: the knowledge roles

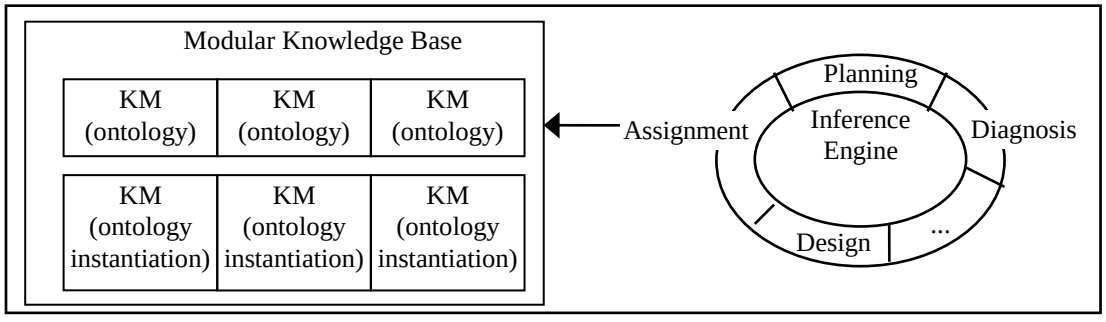


Figure 2 - The General Architecture of Knowledge Servers.

that appear in a template of a problem type must be fulfilled by the domain knowledge described in the

knowledge modules. Thus, with these components available, it is possible to change the way intelligent assistants are developed.

6. A Knowledge Server in the TABA Workstation

One important aspect to be considered in TABA Workstation is the software development process. The meta-environment and its instantiated SEEs need to handle some knowledge about software processes. This knowledge has to be shared along the Workstation. Due to this, we decided to build a Knowledge Server for software process domain.

The Software Process Knowledge Server (SPKS) was developed in the following manner. First, we built an ontology of software development process [6]. Given the complexity of this domain, we adopted a leveled approach for building ontologies [5]. The ontology of software process was developed on the top of domain ontologies for activity, procedure and resource. Each one of these ontologies originates a knowledge module and has a number of instantiations that also originates knowledge modules. The knowledge modules were implemented in Prolog, which inference engine is already integrated to the environment, as a result of the first initiative to promote knowledge integration in TABA Workstation [1]. Figure 3 shows part of the knowledge module for activity ontology. Figure 4 shows part of a knowledge module derived from an ontology instantiation for the Planning Activity.

```
subactivity(A,C) :- subactivity(A,B), subactivity(B,C).
superactivity(B,A) :- subactivity(A,B).
preactivity(A,B) :- output(S,A), input(S,B).
preactivity(A,C) :- preactivity(A,B), preactivity(B,C).
posactivity(A,B) :- preactivity(B,A).
```

Figure 3 – Activity Ontology Knowledge Module.

To handle task knowledge, we developed task models for planning and assignment, that are the main problem types that occur in the definition of a software process. For this to be done, we used the CommonKADS library [7]. The models were selected and adapted for the domain of interest. Then we developed templates in Eiffel, based on the adapted task models. Figure 5 shows the architecture of the SPKS.

This infrastructure was used to develop *Assist-Pro*, an intelligent assistant to support the definition of software processes. First, we selected task templates from the SPKS and adapted them to the problem in hands, trying to match knowledge modules with their knowledge roles. The knowledge roles that were not fulfilled had to be elicited. In the case of *Assist-Pro*, it happened only with a role concerning the adequability of life cycle models to project characteristics. Since *Assist-Pro* requires lots of knowledge about activities, methods, techniques, tools, life cycle models, etc, some ontology instantiation had to be done. TABA Workstation offers facilities to input this knowledge, generating automatically the corresponding knowledge bases in Prolog's format, like the one shown in Figure 4.

```
activity (Planning, ManagingActivity).
activity (SoftwareScopeDefinition, ManagingActivity).
activity (ResourceEstimating, ManagingActivity).
activity (EffortEstimating, ManagingActivity).
activity (Scheduling, ManagingActivity).
input (ScopeStatement, ResourceEstimating).
input (ScopeStatement, EffortEstimating).
input (EffortEstimation, Scheduling).
output (ProjectPlan, Planning).
output (ScopeStatement, SoftwareScopeDefinition).
output (ResourceEstimation, ResourceEstimating).
output (EffortEstimation, EffortEstimating).
output (TimelineChart, Scheduling).
subactivity (SoftwareScopeDefinition, Planning).
subactivity (ResourceEstimation, Planning).
subactivity (EffortEstimation, Planning).
subactivity (Scheduling, Planning).
```

Figure 4 – Planning Activity Knowledge Module.

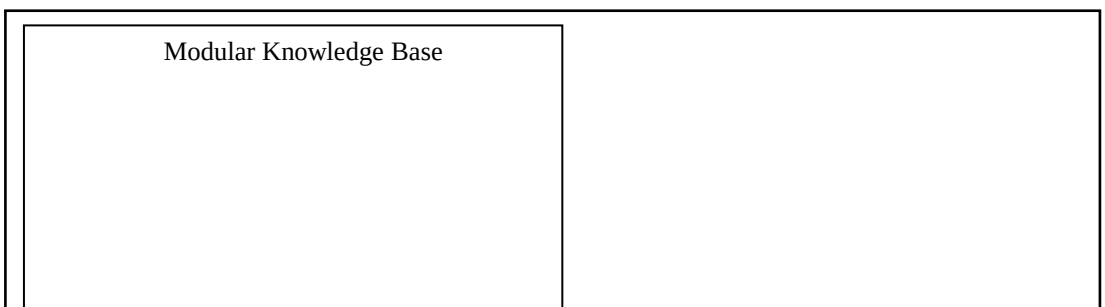
Process definition using *Assist-Pro* involves the following steps: (i) project feature definition, (ii) selection of a life cycle model, (iii) process decomposition, (iv) method and technique assignment, and (v) definition of inputs and outputs for each activity.

In steps (iii) to (v), *Assist-Pro* uses the knowledge modules provided by SPKS to assist software engineers defining processes. In step (ii), it uses its own knowledge base about life cycle model adequability. In this way, the SPKS offers facilities to be used in the construction of knowledge-based tools, such as *Assist-Pro*.

7. Conclusions

As the interest in providing knowledge-based assistance in SEEs increases, it becomes essential to consider tools' integration as a four-dimension issue, including, in addition to data, control and presentation, *knowledge integration*.

Knowledge integration cannot be achieved considering only the symbolic level. Providing services to represent and store knowledge in a SEE is useful, but it is not enough to promote knowledge integration. It is necessary to change the way knowledge-based tools are developed and to offer a framework that supports development *for/with* knowledge reuse. This framework is a Knowledge Server.



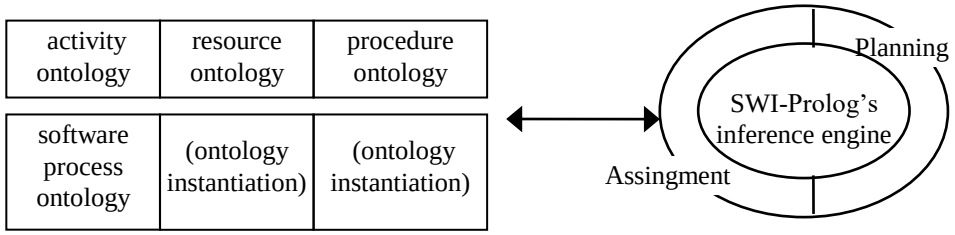


Figure 5 - The Software Process Knowledge Server.

Knowledge Servers make two main kinds of reusable components available: domain knowledge components and task templates. The knowledge is not any longer acquired and modelled with a specific purpose and embedded in an assistant. Unlike, it is available to the SEE and can be used in the development of several tools.

When using Knowledge Servers to promote knowledge integration in SEEs, it is possible to achieve Domain-Oriented SEEs (DOSEEs) [8], offering a library of reusable domain components in various levels of abstraction, including knowledge. Thus, a DOSEE can support the development of software systems in a particular domain (like medicine or law), embedding the knowledge of this domain in it, and aiding software engineers to develop software in a domain not familiar to them.

Knowledge Servers address the four problems enumerated in section 2 in the following manner:

a) *Difficulty to share and reuse knowledge*: Since the Knowledge Server makes knowledge modules (that operationalize ontologies and their instantiations) and reasoner templates (that operationalize task models) available, it is possible to reuse and share both domain and task knowledge.

b) *Communication problems due to different conceptualizations adopted by each tool*: Since the knowledge modules are derived from ontologies, there is a basic domain conceptualization and a well-established vocabulary underlying the development of tools, minimizing the communication problems.

c) *Knowledge redundancy and inconsistency*: Typically, the knowledge common to various tools is the domain knowledge, captured as ontologies and implemented in the Knowledge Server knowledge modules. Consequently, since the Knowledge Server makes these knowledge modules available, the knowledge redundancy and inconsistency in the SEE can be minimized.

d) *High costs and low productivity in the development of knowledge-based tools for the SEE*: Because the Knowledge Server offers reusable components for both domain and task, it is possible to adopt a more productive strategy in the development of knowledge-based tools for the SEE. In this approach, the knowledge acquisition can be accomplished in two stages. The first one is strongly supported by the Knowledge Server. The second one, although more closely supported by the experts, is still guided by the first stage. In this way, the productivity can be increased and costs reduced.

Acknowledgments

The authors acknowledge Kathia Oliveira for her comments and CNPq (The Brazilian Research Council) and CAPES for the financial support to this work.

References

[1] Falbo, R.A. and Travassos, G.H.; "Improving Tool's Integration on Software Engineering Environments Using Objects and Knowledge", Proceedings of the SCI'97/ISAS'97, Caracas, Venezuela, 1997.

[2] Rocha, A.R. et alli; "TABA: a heuristic workstation for software development", COMPEURO'90, Israel, May 1990.

[3] Newell, A.; "The Knowledge Level", Artificial Intelligence, 18, 1982.

[4] Gruber, T.R.; "Towards principles for the design of ontologies used for knowledge sharing", Int. J. Human-Computer Studies, 43(5/6), 1995.

[5] Falbo, R.A., Menezes, C.S. and Rocha, A.R.; "A Systematic Approach for Building Ontologies". Proc. of the IBERAMIA'98, Lisbon, Portugal, 1998.

[6] Falbo, R.A., Menezes,C.S. and Rocha, A.R.; "Using Ontologies to Improve Knowledge Integration in Software Engineering Environments", Proceedings of SCI'98/ISAS'98, Orlando, USA, July, 1998.

[7] Breuker, J., Van de Velde, W.; *CommonKADS Library for Expertise Modeling*, IOS Press, 1994.

[8] Oliveira, K., Rocha, A.R., Travassos,G., Matwin, S; "Towards Domain-Oriented Software Development Environment for Cardiology", CAiSE'98, 5th Doctoral Consortium, Italy, June 1998.