The Use of an Enterprise Ontology to Support Knowledge Management in Software Development Environments

Karina Villela
University of Salvador (UNIFACS)[1]
Rua Ponciano Oliveira, 126,
Rio Vermelho, CEP 40170–100,
Salvador - Ba, Brasil
kvillela@unifacs.br

**Gleison Santos, Lílian Schnaider,
Ana Regina Rocha,
Guilherme Horta Travassos**
COPPE/Federal University of Rio de Janeiro
68511, CEP 21945–970, Rio de Janeiro - RJ, Brasil
{gleison, darocha, ght}@cos.ufrj.br

Abstract

Software engineering is knowledge-intensive activity and knowledge is thought to be the most important asset in an organization. Therefore this paper presents an approach to support Knowledge Management in Software Development Environments that is strongly based on ontologies: Enterprise Oriented Software Development Environments. After describing the components of such environments, this paper focuses on the Enterprise Ontology and on three tools developed based on this ontology: a 'yellow pages' tool which shows the distribution of competencies in the organization, a tool to support the allocation of people to software projects and a graphic tool for representing and visualizing organizational processes.

Keywords: Ontology, Knowledge Management, Software Development Environment

1. Introduction

Software engineering is a knowledge-intensive activity [1]. Several knowledge representations and transformations are required throughout software projects and different kinds of knowledge are important to software engineers in this context, such as: domain knowledge, organizational guidelines, software techniques and methods, best practices and previous experiences. Furthermore, knowledge is considered the most important asset in an organization, having a significant impact on its competitiveness [1].

Knowledge Management can be defined as a systematic and active management of organizational knowledge assets, using appropriate technology and aiming at generating strategic benefits to the organization. This can involve promoting satisfactory communication and sharing of knowledge among individuals, obtaining relevant knowledge from internal and/or external sources, making available and distributing the obtained knowledge appropriately to satisfy the user's needs, generating new knowledge and eliminating outdated knowledge.

Therefore Knowledge Management should be integrated into Software Development Environments in order to develop and capture organizational knowledge relevant to the software engineering activity and to improve the flow of knowledge among software developers and project managers.

However, one of the great obstacles for Knowledge Management is the use of different vocabularies to describe

[1] This is the present workplace of the author after she finished her PHD thesis at the Federal University of Rio de Janeiro, where the work presented here was carried out.

the knowledge about a domain. Ontologies provide shared vocabularies used to facilitate communication, representation, search and storage in Knowledge Management Systems [2]. An ontology is an explicit representation of a shared conceptualization [3]. In this context, conceptualization refers to an abstract model of a world view with respect to a particular subject area. It is composed of a set of concepts, their definitions and their inter-relationships.

Ontologies have been used in Domain Oriented Software Development Environments (DOSDEs) [4,5] to support software developers in their activities by providing domain and task knowledge that can be used throughout the software development process. However, after the definition and building of DOSDEs for different domains [4] it can be seen that, apart from domain and task knowledge, other kinds of knowledge could also be necessary and useful during a software project, mainly knowledge regarding the enterprise working context.

This paper describes an approach strongly based on ontologies to support Knowledge Management in Software Development Environments (section 2). We call Enterprise Oriented Software Development Environments (EOSDEs) the environments obtained using this approach. The paper therefore focuses on the Enterprise Ontology (section 3) and some specific tools are presented to show how this ontology has been used in order to contribute to Knowledge Management in such environments (section 4). Finally, our conclusions are presented in section 5.

2. Enterprise Oriented Software Development Environments

An Enterprise-Oriented Software Development Environment (EOSDE) [6,7] supports the activity of software engineering, making it possible to manage knowledge that can be useful to software engineers when carrying out an organization's software projects. As well as domain knowledge, other kinds of knowledge are of interest to increase productivity and quality in this context. This includes knowledge about the organization itself, specialized knowledge about software development and maintenance obtained on previous software projects within the organization, and also knowledge about its clients.

Figure 1 gives an overview of the components of an EOSDE. The *Knowledge Management Infrastructure* is composed of the *Organizational Memory* and the *Knowledge Management Services/Tools*. *Knowledge Management Services/Tools* support the storage of data, knowledge and experiences in the *Organizational Memory*, promoting the dissemination and evolution of its contents. *Software Engineering Services/Tools* support the activities of software development and maintenance as well as the management of these activities. These services/tools must be able to provide software engineers with all the knowledge held by the organization which is relevant for the activity being carried out, using the *Knowledge Management Infrastructure*. A *Project Database* stores all data related to the software project.

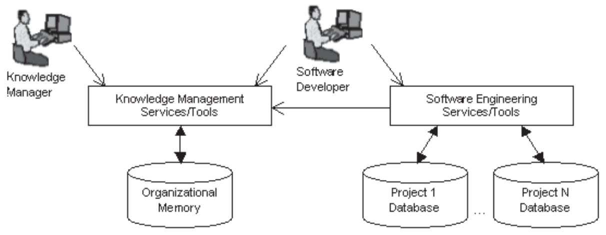


Figure 1: Overview of the EOSDE Components

An architecture has been devised for the Organizational Memory component (Figures 2a and 2b) taking into account whether the EOSDE is for Software Organizations (in which software engineering is a business activity) or for Non software Organizations (in which software engineering is a support activity for the running of the business). Each architecture component has its own goal and contains important knowledge. The arrows in Figure 2 indicate elements in the source components referring to elements in the target components.

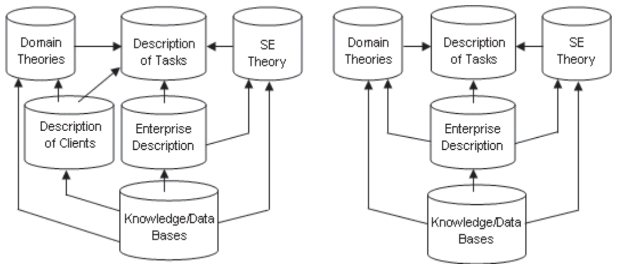


Figure 2: Organizational Memory - EOSDE for Software Organizations (a) and EOSDE For Non-Software Organizations (b)

The *Description of Tasks* component contains the description of generic tasks, such as to reserve and to configure, regardless of domain and organization. As proposed by ZLOT et al. [8], a task description consists of a high level description, a task ontology, the inference to solve the problem the task represents or its breakdown into sub tasks, as well as bibliographic references. The goal is to support software developers in understanding a problem (E.g. sonar configuration) through an understanding of the tasks that it consists of (E.g. the configuration task and its sub-tasks of selecting, proposing, verifying and reviewing).

The *Domain Theories* component organizes domain knowledge and it has been built on OLIVEIRA el al. [4,5]'s work. A *Domain Theory* is broken-down into sub-theories. The sub-theories, in their turn, can be broken-down into smaller sub-theories (E.g. the *Domain Theory of Acoustic Propagation* can be broken-down into the sub-theories of Acoustic Environment and Propagation Theory) or be composed of an *Domain Ontology* along with the mapping among the ontology concepts and the generic tasks which apply them (E.g. sonar is a concept of the Acoustic Environment ontology and it is mapped, for example, to the configuration task). Besides promoting the domain knowledge understanding, this component guides the registration and updating of the organization's knowledge map by offering a common vocabulary in the domain.

Despite the subject area, the *Software Engineering (SE) Theory* component is the same as the *Domain Theories* component. The goal is to make communication among software engineers easier and to guide the registration and updating of the organization's knowledge map regarding this subject area (E.g. a software engineer's knowledge about an OO analysis method can be registered into the organization's knowledge map by using links to the *SE Ontology* concepts of Paradigm, Activity and Method and to their respective instances). Althoff et al. [9] propose the use of Software Engineering Ontologies in Software Engineering Experience Environments.

The *Enterprise Description* component contains a description of the organization, identifying the generic tasks that are performed and the software engineering knowledge necessary in the context of the organizational structure and processes. If the organization develops and maintains software for its own use (Non-Software Organization), this component also sets which domain knowledge is required throughout the organizational structure and processes (E.g. the sonar configuration activity is part of a Navy's organizational process and it can be linked to the generic task of configuring and to the Sonar domain concept. Another Navy's activity is software design which can be linked to the generic task of designing and to the Design Pattern concept from the *SE Theory*, among others). The organizational process models allow the specification of the context in which a knowledge item was created and the application context for it. The organization's knowledge map is part of this component and defines the competencies each employee has and to which degree these competencies are held.

The *Description of Clients* component is specific of EOSDE created for Software Organizations that develop and/or maintain software for clients. It is similar to the *Enterprise Description* component, but it describes the client organizations. Possessing knowledge, even if only limited, about clients and their domains can give a strategic advantage to holders of this knowledge in competition for new projects.

As different organizations are structured in different ways, varying in the quantity of structural levels, the relationships among them and the names adopted, an *Enterprise Ontology* is fundamental to define a common vocabulary to guide the description of any organization for which an EOSDE can be generated or any of their client organizations. The use of an Enterprise Ontology to support Knowledge Management is mentioned in [10].

Knowledge/Data Bases component stores the knowledge and data relevant to the organization acquired and updated over the course of many software projects. Each knowledge item stored in the environment is associated to one or more concepts and instances of these concepts obtained from the EOSDE ontologies. This enables subsequent retrieval of different types of knowledge items based on the selection of concepts and instances, regardless of the specific tools used to record and read the knowledge items (E.g. a knowledge item which describes a lesson learned about the use of the Model-View-Controller architectural style will refer to the activity defined in the *Enterprise Description* in which the lesson was learned as well as to the concept of Architectural Style and to its instance defined in the *SE Theory.* Then this knowledge item can be retrieved, for example, by an EOSDE user who looks for knowledge items related to the Model-View-Controller architectural style, by selecting the appropriate concepts and instances from the *SE Theory*).

From this description of the EOSDE components, it can be seen that the use of ontologies is critical to make the retrieval of knowledge stored in the environment as well as communication among multiple users and tools more straightforward. When retrieving knowledge items, the purpose of ontologies is to supply vocabularies whose terms are used as indexes to access the knowledge items and also as links among multiple knowledge/data bases contents. By defining synonyms and acronyms for concepts, ontologies provide linguistic equivalents that may occur in text documents and can be used to access knowledge. As regards communication, the defined ontologies have the purpose of reducing terminological and conceptual mismatches. A common class model can be created based on a ontology and used by various tools as well as matches among classes from different models can be made through their association to the ontology terms.

Tools to support the description and updating of tasks and Domain Theories [5], organizations' structure and processes [6] as well as the capture of knowledge items [11] are provided by the *Knowledge Management Infrastructure* of EOSDEs.

It should be emphasized the *Organizational Memory* is expected to be incrementally fulfilled with knowledge according to the organization's objectives and projects in order to make the approach cost effective. For example, the first project involving an unknown domain or task provides the basic knowledge about it, which grows with each new project carried out in the domain or involving the task. In the same way, the structure and processes of an organization should be described as they are needed in the software projects.

OLIVEIRA el al. [4,5] and ZLOT et al. [8] have respectively dealt with *Domain* and *Task Ontologies*. Moreover, the need of a specific *Domain* or *Task Ontology* depends on the specific organization for which an EOSDE is constructed. The *Software Engineering Ontology* should be made up of many sub-ontologies such as the ones defined by FALBO et al. [12] and by KITCHENHAM et al. [13] and will be focus of our future work. This paper focus on the *Enterprise Ontology*, which is explained at greater length in the following section, after which the three tools based on this ontology are presented.

3. Enterprise Ontology

As mentioned previously, the Enterprise Ontology aims to supply a common vocabulary that can be used to represent useful knowledge on the organizations involved in a software project for the software developers. It can be useful for:

- supplying a structure to organize knowledge and guide knowledge acquisition in one or more organizations;
- allowing the development of generic tools based on its structure, reducing the effort required to construct software development environments for different organizations;
- promoting the integration among tools that manipulate knowledge related to the ontology by the sharing of databases created based on the ontology structure;
- facilitating the development of systems that manipulate knowledge on the organization (for example, a system that supports an organizational process). It can provide a common vocabulary to be used by developers and users, allow the reuse of knowledge on the organization to draw up a first version of the requirements and allow the identification of those who can give information about the system, and;
- assisting the identification of professionals with the appropriate competencies for discussing ideas about a subject, for guiding the execution of a task or for putting together a team to suit the characteristics of the project.

The process used to define the Enterprise Ontology is composed of: i) ontology purpose identification, in which the ontology scope is defined by describing the motivation scenes and identifying the general competence questions to be answered by the ontology; ii) requirements specification, in which the general competence questions are refined into more specific ones and grouped into sub-ontologies according to content similarity; iii) ontology capture, in which concepts, relations and restrictions are described in natural language and exemplified, and finally iv) ontology formalization, in which first order logic is used to formalize the ontology by defining constants, predicates and axioms. Ontology validation is a support activity, being carried out throughout all process. The resulting Enterprise Ontology combines new concepts with others defined by Fox et al. [14] and the TOVE project (TOronto Virtual Enterprise) [15].

Figure 3 shows the sub-ontologies of the Enterprise Ontology, which were defined to answer the questions on:

- how the organization is perceived in its environment;
- how the organization is structured and how the distribution of authority and responsibility is accomplished;
- who works in the organization and how the desired and possessed competencies have been distributed within it, and;
- how the organization behaves and the objectives it has.

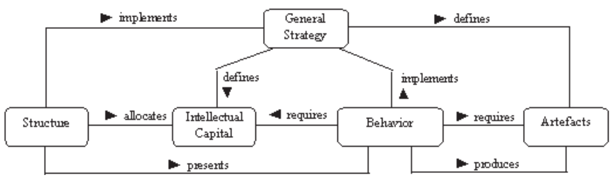


Figure 3: Sub-ontologies of the Enterprise Ontology

The **Intellectual Capital** sub-ontology deals with aspects such as: taxonomy of competence, interaction between experience and knowledge, availability of competencies and breakdown of knowledge domain. **People** are the basic components of an organization, executing the necessary activities for the fulfilment of the organization's mission. **Competencies** are characteristics that make people capable of carrying out activities that involve some degree of difficulty. They can be classified according to their nature into *knowledge, skill and experience*. **Knowledge** is the understanding of a subject obtained by thinking, using definitions, perception, analysis, comprehension or other ways of understanding. **Skills** are personal characteristics or acquired abilities not associated to specific activities or knowledge domains, for example: the ability to negotiate and leadership. **Experiences** are acquired through practice, in other words, the carrying out of activities. Examples are experience in defining client-server architectures and airport administration. Experiences usually involve the use of knowledge in practice. Finally, a **Knowledge Domain** organizes knowledge items according to content similarity. Table 1 and 2 respectively show the main relations and axioms of this sub-ontology.

Relation Predicate	Description
holding(p,c,l)	Person **p** holds competence **c** at level **l**
application(x,c)	Experience **x** involves the application of knowledge **c**
subdomain(d_1,d_2)	Knowledge domain **d_1** is part of knowledge domain **d_2**
superdomain(d_2,d_1)	Knowledge domain **d_2** contains knowledge domain **d_1**
domain_composition (c,d)	Knowledge **c** is part of knowl-edge domain **d**

Table 1: Relations of the Intellectual Capital sub-ontology

Axioms
$(\forall c)(\text{knowledge}(c) \rightarrow (\exists d)\text{domain\_composition}(c,d))$
$(\forall d_1,d_2)(\text{subdomain}(d_1,d_2) \leftrightarrow \text{superdomain}(d_2,d_1))$
$(\forall d_1,d_2,d_3)(\text{subdomain}(d_1,d_2) \wedge \text{subdomain}(d_2,d_3) \rightarrow \text{subdomain}(d_1,d_3))$
$(\forall d_1,d_2)(\text{subdomain}(d_1,d_2) \rightarrow \neg\text{subdomain}(d_2,d_1))$
$(\forall d)(\text{elementar\_domain}(d) \leftrightarrow (\neg\exists d_1)(\text{subdomain}(d_1,d)))$
$(\forall d)(\text{macrodomain}(d) \leftrightarrow (\neg\exists d_1)\text{superdomain}(d_1,d))$
$(\forall c,d)(\text{domain\_composition}(c,d) \rightarrow \text{elementar\_domain}(d))$
$(\forall c,d)(\text{domain\_content}(c,d) \leftrightarrow \text{domain\_composition}(c,d) \vee (\exists d_k)(\text{subdomain}(d_k,d) \wedge \text{domain\_content}(c,d_k)))$

Table 2: Axioms of the Intellectual Capital sub-ontology

The Structure sub-ontology deals with the organization of organizations, distribution of authority and responsibilities among organizational units, how they are broken down into positions, distribution of authority and responsibilities among positions, specification of functions and positions, staff allocation, definition of teams and definition of objectives.

An **Organization** can be defined as an organized group of people working together for the fulfilment of a mission. There are several ways to break down an organization, but the main components normally used are *functions*, *organizational units* and *committees*. A **Function** specifies the set of activities to be executed by the people who occupy it, their responsibilities and the required competencies as well as working conditions. An **Organizational Unit** is a grouping of organization components (for example: activities and people) which enables the organization to be economical and efficient. An organizational unit is related to other ones through cooperation or subordination relationships and it is structured in positions. A **Position** specifies activities, responsibilities and competencies in line with the purpose of the specific organizational unit and also determines the location of a person in the organizational structure. Each position relates to other positions through subordination relationships. An **Agent** represents a profile that allows the organization to accomplish its mission throughout the execution of activities and it can represent a function or position. Staff allocation involves selecting people for positions, taking into consideration people's functions and competencies and the functions and competencies required by the positions. People also take part in committees inside the organization. A **Committee** is a group of people with a specific goal that usually work together for a period of time until a specific goal is achieved, for example: a committee for planning a new product or a committee for guaranteeing security at work. Finally, **Objectives** are statements about the results to be reached in a fixed period of time and may be applied to the organization, organizational units or positions. Table 3 and 4 respectively show the main relations and axioms of this sub-ontology.

Relation Predicate	Description
organization_ composition1(u,r)	organizational unit **u** is part of organization **r**
organization_ composition2(c,r)	committee **c** is part of or-ganization **r**
function_plan (f,r)	**f** is one of the existing functions within the organi-zation **r**
unit_subordination (u_1,u_2,s)	organizational unit **u_1** re-ports to organizational unit **u_2** concerning scope **s**
unit_cooperation (u_1,u_2,s)	organizational unit **u_1** coop-erates with organizational unit **u_2** concerning scope **s**
unit_composition (p,u)	position **p** is part of organ-izational unit **u**
allocation_ opportunity(p,f)	people who occupy function **f** can be allocated to position **p**

Table 3: Relations of the Structure sub-ontology

Relation Predicate	Description
position_ subordination (p_1,p_2,s)	position **p_1** reports to position **p_2** concerning scope ***s***
responsibity(a,g)	agent **g** is responsible for carrying out activity **a**
additional_ competence(c,g)	competence **c** is required for agent **g** in addition to the ones required to carry out its activities[2]
contract(p,f)	person **p** occupies function **f** regardless specific type of work contract
allocation(p,s)	person **p** is allocated to position **s**
superobjective (t_2,t_1)	objective **t_2** is achieved by achieving objective **t_1**
subobjective(t_1,t_2)	objective **t_1** is a way to achieve objective **t_2**
preobjective(t_1,t_2)	objective **t_1** should be achieved before the achievement of objective **t_2**
posobjective (t_2,t_1)	objective **t_2** should be achieved after the achievement of objective **t_1**
organizational_ objective(t,r)	objective **t** guides the carrying out of activities within organization **r**
unit_objective (t,u)	objective **t** guides the carrying out of activities within organizational unit **u**
position_objective (t,p)	objective **t** guides the carrying out of position **p**'s activities

Table 3: Relations of the Structure sub-ontology (continued)

Axioms
$(\forall u)(organizational\_unit(u) \rightarrow (\exists r)organization\_composition1(u,r))$
$(\forall u,r_1,r_2)(organization\_composition1(u,r_1) \wedge organization\_composition1(u,r_2) \rightarrow r_1 = r_2)$
$(\forall c)(committee(c) \rightarrow (\exists r)organization\_composition2(c,r))$
$(\forall c,r_1,r_2)(organization\_composition2(c,r_1) \wedge organization\_composition2(c,r_2) \rightarrow r_1 = r_2)$
$(\forall p)(position(p) \rightarrow (\exists u)unit\_composition(p,u))$
$(\forall p,u_1,u_2)(unit\_composition(p, u_1) \wedge unit\_composition(p,u_2) \rightarrow u_1 = u_2)$
$(\forall p)(position(p) \rightarrow (\exists f)allocation\_opportunity(p,f))$
$(\forall p)(person(p) \rightarrow (\exists f)contract(p,f))$

Table 4: Axioms of the Structure sub-ontology

Axioms
$(\forall p,f_1,f_2)(contract(p,f_1) \wedge contract(p,f_2) \rightarrow f_1 = f_2)$
$(\forall t,r)(organizational\_objective(t,r) \rightarrow (\neg\exists u)\ unit\_objective(t,u) \wedge (\neg\exists p)\ position\_objective(t,p))$
$(\forall t,u)(unit\_objective(t,u) \rightarrow (\neg\exists r)\ organizational\_objective(t,r) \wedge (\neg\exists p)\ position\_objective(t,p))$
$(\forall t,p)(position\_objective(t,p) \rightarrow (\neg\exists r)\ organizational\_objective(t,r) \wedge (\neg\exists u)\ unit\_objective(t,u))$
$(\forall t_1,t_2,r)(organizational\_objective(t_1,r) \wedge preobjective(t_1,t_2) \rightarrow organizational\_objective(t_2,r))$
$(\forall t_1,t_2,u_1)(unit\_objective(t_1,u_1) \wedge preobjective(t_1,t_2) \rightarrow (\exists u_2)unit\_objective(t_2,u_2))$
$(\forall t_1,t_2,p_1)(position\_objective(t_1,p_1) \wedge preobjective(t_1,t_2) \rightarrow (\exists p_2)position\_objective(t_2,p_2))$
$(\forall t_1,t_2,r)(organizational\_objective(t_1,r) \wedge subobjective(t_1,t_2) \rightarrow organizational\_objective\ (t_2,r))$
$(\forall t_1,t_2,u)(unit\_objective(t_1,u) \wedge subobjective(t_1,t_2) \rightarrow unit\_objective(t_2,u))$
$(\forall t_1,t_2,p)(position\_objective(t_1,p) \wedge subobjective(t_1,t_2) \rightarrow position\_objective(t_2,p))$

Table 4: Axioms of the Structure sub-ontology (continued)

The **Artefacts** sub-ontology groups the concepts and relationships that define artefacts in terms of their nature and composition. An **Artefact** is anything produced by humans and not by natural causes that is able to exert different roles in an organization, such as the product of an activity. Artefacts can be composed by other artefacts and are classified according to their nature into *goods*, *documents* and *components*. **Goods** can be classified in *goods for use* and *goods for production*. Goods for production can in turn be classified into *hardware*, *software* and *device*. A **Component** can be a *hardware component*, a *software component* or a *spare part*. Table 5 and 6 respectively show the main relations and axioms of this sub-ontology. To understand Table 6, the more complex concept predicate must be explained: component(s,t), which means *s* is a component of type *t* whose possible values are *SoftwareComp, HardwareComp* or *SparePart*.

Relation Predicate	Description
subartefact(s_1,s_2)	artefact **s_1** is part of artefact **s_2**
superartefact (s_2,s_1)	artefact **s_2** contains artefact **s_1**

Table 5: Relations of the Artefacts sub-ontology

[2] See definition of the intellectual_resource(c,t) predicate in Table 7.

Axioms
$(\forall s)(\text{elementar\_artefact}(s) \leftrightarrow (\neg\exists s_1)\,\text{subartefact}(s_1,s))$
$(\forall s)(\text{macroartefact}(s) \leftrightarrow (\neg\exists s_1)\,\text{superartefact}(s_1,s))$
$(\forall s)(\text{goods}(s) \rightarrow \text{macroartefact}(s))$
$(\forall s_1,s_2,t_1,t_2)((\text{component}(s_1,t_1) \wedge \text{component}(s_2,t_2) \wedge \text{subartefact}(s_1,s_2)) \rightarrow t_1=t_2)$
$(\forall s_1,s_2)(\text{document}(s_2) \wedge \text{subartefact}(s_1,s_2) \rightarrow \text{document}(s_1))$
$(\forall s_1,s_2)(\text{software}(s_2) \wedge \text{subartefact}(s_1,s_2) \rightarrow \text{component}(s_1,\text{SoftwareComp}) \vee \text{document}(s_1))$
$(\forall s_1,s_2)(\text{device}(s_2) \wedge \text{subartefact}(s_1,s_2) \rightarrow \text{component}(s_1,\text{SparePart}) \vee \text{document}(s_1))$
$(\forall s)(\text{software}(s) \wedge (\exists s_1,\ldots,s_n)(\text{subartefact}(s_1,s) \wedge \ldots \wedge \text{subartefact}(s_n,s)) \rightarrow (\exists s_k)(\text{subartefact}(s_k,s) \wedge \text{component}(s_k,\text{SoftwareComp})))$
$(\forall s)(\text{hardware}(s) \wedge (\exists s_1,\ldots,s_n)(\text{subartefact}(s_1,s) \wedge \ldots \wedge \text{subartefact}(s_n,s)) \rightarrow (\exists s_k)(\text{subartefact}(s_k,s) \wedge \text{component}(s_k,\text{HardwareComp})))$
$(\forall s)(\text{device}(s) \wedge (\exists s_1,\ldots,s_n)(\text{subartefact}(s_1,s) \wedge \ldots \wedge \text{subartefact}(s_n,s)) \rightarrow (\exists s_k)(\text{subartefact}(s_k,s) \wedge \text{component}(s_k,\text{SparePart})))$

Table 6: Axioms of the Artefacts sub-ontology

The aspects covered by the **Behaviour** sub-ontology[3] include: activity as an action of transformation, taxonomy of activity, process and activity breakdown, adoption of procedures, taxonomy of procedures, method as systematic procedure, automation of procedures, organizational processes and related norms as well as organizational projects. An **Activity** is the action of transforming raw material and/or input artefacts into output artefacts, which may require competencies and the use of goods for production. An activity can be classified in *operational activity*, *managerial activity* or *quality control activity* according to its nature and into a *main activity* or a *support activity* according to its role in the fulfilment of the organization's mission. An activity can also be made up of a set of other activities. A **Process** is a set of structured activities which produce artefacts or services of value to the organization itself, for a client or for a business market. **Procedures** are instructions for executing activities and are classified into *methods*, *techniques* and *guidelines*. **Methods** as well as **Techniques** can be classified according to the type of activities they can support. **Guidelines** are further classified into *templates* and *norms*. A procedure may be supported by software tools. An organization has its behaviour defined by the set of processes executed within it and they may comply with norms. **Projects** are undertakings initiated by the organization which entail processes to guide their activities and have project teams allocated to them. Table 7 and 8 respectively show the main relations and axioms of this sub-ontology. To understand Table 8, the more complex concept predicate must be explained: activity_group(g,r), which means *g* is an activity group in which the activities follow the restriction *r* whose possible values are *Obligatory, Exclusive* or *None*.

Relation Predicate	Description
adoption(p,t)	Procedure **p** can be adopted in the carrying out of activity **t**
automation(s,p)	Software **s** can (semi-)automate procedure **p**
activity_breakdown (t,g)	Activity **t** is defined by the activity group **g**
grouping(t,g)	Activity **t** is part of the activity group **g**
preactivity(t_1,t_2)	Activity $\mathbf{t_1}$ is carried out before activity $\mathbf{t_2}$
posactivity(t_2,t_1)	Activity $\mathbf{t_2}$ is carried out after activity $\mathbf{t_1}$
description(m,t,g)	method **m** breaks down the activity **t** into the set of activities **g**
execution(p,t)	person **p** is responsible for carrying out the activity **t**
input-1(m,t)	raw material **m** is required by activity **t**
input-2(s,t)	artefact **s** is required by activity **t**
output(s,t)	Artefact **s** is produced by activity **t**
intellectual_ resource (c,t)	Competence **c** is intellectual resource to activity **t**
material_resource (b,t)	Good for production **b** is material resource to activity **t**

Table 7: Relations of the Behaviour sub-ontology

The **General Strategy** sub-ontology establishes the vocabulary to describe how the organization interacts with its environment, that is: its business domain, the artefacts/services it offers and the relationships with client organizations. An organization works in a knowledge domain which means it possesses intellectual capital related to the domain and executes activities which require knowledge from this domain. A **Service** is an abstract notion, an intangible product offered by an organization to satisfy the need or desire of a client or market, as opposed to an artefact which is a tangible product. A **Business Agreement** is an agreement among two or more

[3] Behaviour sub-ontology was defined based on the software process ontology defined by Falbo [12].

organizations which establishes a business relationship. Tables 9 and 10 show the main relations and axioms of this sub-ontology.

Axioms
$(\forall t)(\mathrm{quality\_activity}(t) \rightarrow \mathrm{support\_activity}(t))$
$(\forall t_1,t_2)(\mathrm{preactivity}(t_1,t_2) \leftrightarrow (\exists s)(\mathrm{output}(s,t_1) \wedge (\mathrm{input\text{-}1}(s,t_2) \vee \mathrm{input\text{-}2}(s,t_2))))$
$(\forall t_1,t_2)(\mathrm{subactivity}(t_1,t_2) \leftrightarrow (\exists g)(\mathrm{activity\_group}(g,*) \wedge \mathrm{activity\_breakdown}(t_2,g) \wedge \mathrm{grouping}(t_1,g)))$
$(\forall t,t_1,\ldots,t_n,s)(\mathrm{subactivity}(t_1,t) \wedge \ldots \wedge \mathrm{subactivity}(t_n,t) \wedge \mathrm{input\text{-}1}(s,t_i) \wedge (\neg\exists t_k)\mathrm{output}(s,t_x) \rightarrow \mathrm{input\text{-}1}(s,t))$
$(\forall t,t_1,\ldots,t_n,s)(\mathrm{subactivity}(t_1,t) \wedge \ldots \wedge \mathrm{subactivity}(t_n,t) \wedge \mathrm{input\text{-}2}(s,t_i) \wedge (\neg\exists t_k)\mathrm{output}(s,t_k) \rightarrow \mathrm{input\text{-}2}(s,t))$
$(\forall t,t_1,\ldots,t_n,s)(\mathrm{subactivity}(t_1,t) \wedge \ldots \wedge \mathrm{subactivity}(t_n,t) \wedge \mathrm{output}(s,t_i) \wedge (\neg\exists t_k)(\mathrm{input\text{-}2}(s,t_k) \rightarrow \mathrm{output}(s,t))$
$(\forall t_1,t,b)(\mathrm{material\_resource}(b,t_1) \wedge \mathrm{subactivity}(t_1,t) \rightarrow \mathrm{material\_resource}(b,t))$
$(\forall t_1,t,c)(\mathrm{intellectual\_resource}(c,t_1) \wedge \mathrm{subactivity}(t_1,t) \rightarrow \mathrm{intellectual\_resource}(c,t))$
$(\forall r,t)(\mathrm{template}(r) \wedge \mathrm{adoption}(r,t) \rightarrow (\exists s)(\mathrm{document}(s) \wedge \mathrm{output}(s,t)))$
$(\forall p,t)((\mathrm{operational\_method}(p) \vee \mathrm{operational\_technique}(p)) \wedge \mathrm{adoption}(p,t) \rightarrow \mathrm{operational\_activity}(t))$
$(\forall p,t)((\mathrm{managerial\_method}(p) \vee \mathrm{managerial\_technique}(p)) \wedge \mathrm{adoption}(p,t) \rightarrow \mathrm{managerial\_activity}(t))$
$(\forall p,t)((\mathrm{quality\_method}(p) \vee \mathrm{quality\_technique}(p)) \wedge \mathrm{adoption}(p,t) \rightarrow \mathrm{quality\_activity}(t))$
$(\forall g,m,t)(\mathrm{description}(m,t,g) \rightarrow \mathrm{adoption}(m,t) \wedge \mathrm{activity\_breakdown}(t,g))$
$(\forall g,m,t,t_1)(\mathrm{grouping}(t_1,g) \wedge \mathrm{description}(m,t,g) \rightarrow t_1 \neq t \wedge \mathrm{subactivity}(t_1,t))$
$(\forall p,t)((\exists g)\mathrm{description}(p,t,g) \rightarrow \neg\mathrm{elementar\_activity}(t))$
$(\forall p,t)(\mathrm{technique}(p) \wedge \mathrm{adoption}(p,t) \rightarrow \mathrm{elementar\_activity}(t))$
$(\forall s,t)((\exists p)(\mathrm{adoption}(p,t) \wedge \mathrm{automation}(s,p)) \rightarrow \mathrm{material\_resource}(s,t))$

* indicates any value of the property "Restriction"

Table 8: Axioms of the Behaviour sub-ontology

Relation Predicate	Description
business_domain (r,d)	organization **r** carries out activities which require knowledge from knowledge domain **d**
service_supply (r,s)	organization **r** provides service **s** to other organizations
artefact_supply (r,s)	organization **r** provides artefact **s** to other organizations
agreement_client (c,b)	organization **c** takes part in a business agreement **b** as a client
agreement_supplier (p,b)	organization **p** takes part in a business agreement **b** as a supplier of artefacts and/or services
agreement_subject1 (s,b),	service **s** is subject of the business agreement **b**
agreement_subject2 (s,b)	artefact **s** is subject of the business agreement **b**

Table 9: Relations of the General Strategy sub-ontology

Axioms
$(\forall c,p)(\mathrm{business}(c,p) \leftrightarrow (\exists b)(\mathrm{agreement\_client}(c,b) \wedge \mathrm{agreement\_supplier}(p,b)))$
$(\forall b,p,s)(\mathrm{agreement\_supplier}(p,b) \wedge \mathrm{agreement\_subject1}(s,b) \rightarrow \mathrm{service\_supply}(p,s))$
$(\forall b,p,s)(\mathrm{agreement\_supplier}(p,b) \wedge \mathrm{agreement\_subject2}(s,b) \rightarrow \mathrm{artefact\_supply}(p,s))$
$(\forall b,c,s)(\mathrm{agreement\_client}(c,b) \wedge \mathrm{agreement\_subject1}(s,b) \rightarrow \mathrm{service\_demand}(c,s))$
$(\forall b,c,s)(\mathrm{agreement\_client}(c,b) \wedge \mathrm{agreement\_subject2}(s,b) \rightarrow \mathrm{artefact\_demand}(c,s))$

Table 10: Axioms of the General Strategy sub-ontology

4. Enterprise Ontology Based Tools

Integration between Knowledge Management and Business Process Modelling has been the trend since 2001 [16]. The idea is to make Knowledge Management part of the existing business processes, revising them to accommodate Knowledge Management. The use of business process models as a dimension for organizing corporate knowledge makes the deployment of the required knowledge to the right person at the right time easier. Furthermore, business process models capture organizational knowledge on how to fulfil the organizations' mission.

Capability Management is a sub-area of Knowledge Management whose goal is to understand the competencies that an organization needs to accomplish its business objectives. It consists of identifying which individual abilities exist within an organization and comparing the required knowledge with the available knowledge to allow the filling of gaps to help achieve the strategic goals of the organization [17].

Moreover, the management of people allocation to software projects can be enriched through the use of the concept of "corporate yellow pages", where information on the competence profile of each professional of the organization is kept [18]. With this, each professional profile can be captured, mapped in accordance with some previously established criteria, stored and continuously updated. Searches on this database are necessary in order to allocate the most suitable people for the tasks.

In the following subsections three EOSDE tools (Sapiens, RHPlan and ProcKnow) based on the Enterprise Ontology concerned will be discussed to explain these concepts usage.

4.1 Sapiens

The analysis of corporate yellow pages is of great importance when there is a need to find experts in an organization. Most of time the desired competence exists somewhere inside the organization, however, it often takes time to identify, locate and gain access to the person who possesses it [19]. Yellow pages offer a way not only to organize and keep control of the competencies, but also to search for the people that possess them [20].

Sapiens is a 'yellow pages' software tool whose purpose is to allow software developers and project managers to quickly identify within a organizational structure the most appropriate professionals to solve a given problem, as well as to supply knowledge about the organizational structure itself. To this end, Sapiens contains a representation of the organizational structure with the competencies required in its positions; staff allocation, including the competencies of each professional; and also search and navigation mechanisms. In this way it is possible to create a culture of identification and dissemination of the existing knowledge as well as communication among employees and this can be used by the organization to know itself better and take greater advantage of its potential.

Sapiens is designed to be generic, independent of a specific organization or domain, making use of the Enterprise Ontology to allow the description of any organization and, when appropriate, deals with its clients, technical partners and suppliers. Sapiens can also offer support to the activities of the People Management Department.

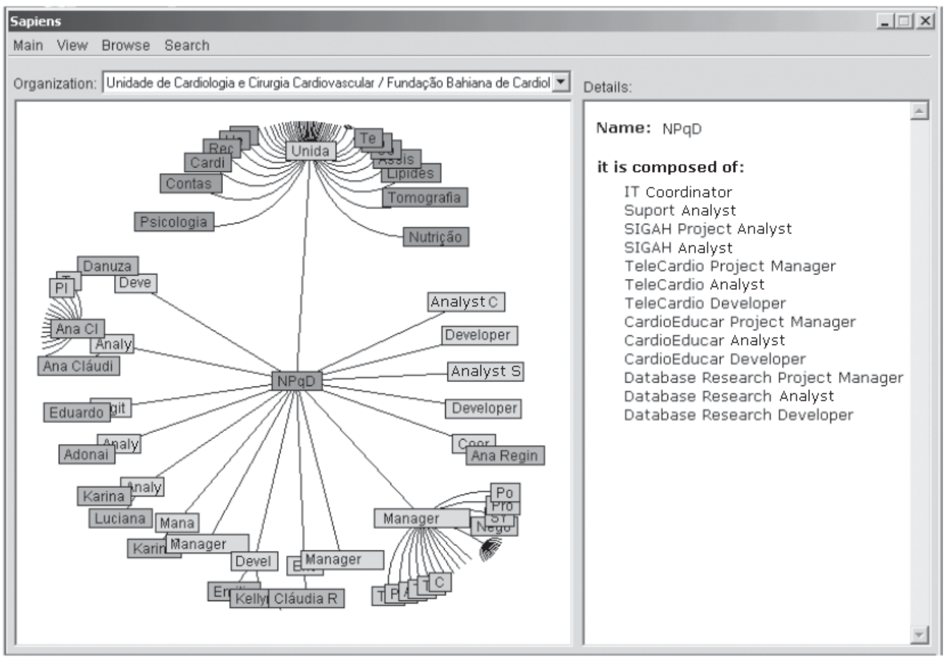
For each position in the organizational structure, it is possible to indicate which competencies are necessary (obligatory) or relevant (non-obligatory) for its performance. In a similar way it is possible to indicate which competencies a person possesses. The association between people and competence, as well as between position and competence, must take into consideration the level of the competence involved. Each competence is associated to a specific scale. For example, a scale for a specific skill could be made up of the following items: "Does not possess the skill nor took part in training", "Took part in training", "Capable with ability", "Capable with great ability".

For capturing information to fulfil Sapiens' database, we requested people from the organizations to fill in a form to collect their professional profiles and organization representatives to provide information about the organization's structure and the competencies required in each position. Professional profiles are verified for reliability at the beginning of a new project and updated according to the real performance by project managers using the RHPlan tool (see section 4.2). Sapiens can also be used by authorized people to update professional profiles at anytime. Through these mechanisms the reliability of the organization's 'yellow pages' is continually improved.

The organizational structure can be viewed using an organizational chart that shows the subordination relationships among organizational units and allows the visualization of each item details. A hyperbolic tree structure [21] (as shown in Figure 4), recommended for visualization of great amounts of organized data in a hierarchical form, is used to browse through the contents of the tool database by exploring the relationships between the items that make up this database. The initial root node is the organization itself. From this point of view the user can browse through its relationships with the other items in the database. When the user clicks on an item, data relative to it are shown (see Details box on the right side of Figure 4) and the focused item and its relationships with the other items become more evident. For example, when the user clicks on an organizational unit (such as the "NPqD" in Figure 4), the existing positions inside this unit appear in the centre and then the user can see who has been allocated to the positions and which competencies are related to each one.

In order to do searches on the database, Sapiens provides some search options. However, the user can create a totally new one if desired. Examples of search options include: "Who has a specific competence?", "Who occupies a specific position?" and "Which positions is a certain competence required in?".

The concepts and relationships described by the Enterprise Ontology have guided and restricted the construction of the class model used by all tool modules. Each class of this class model keeps a reference to the ontological concept that originated it. For implementation reasons, not all the relationships described in the ontology are mapped on the class model. When doing searches, the relationships described in the ontology become important to allow the identification of related concepts and, consequently, of related classes in the class model. Thus, ontology representation is much used in the search module, considering that ontologies are particularly useful for knowledge recovery and access [22].



Organization: Unidade de Cardiologia e Cirurgia Cardiovascular / Fundação Bahiana de Cardiol

Name: NPqD

it is composed of:

IT Coordinator
Suport Analyst
SIGAH Project Analyst
SIGAH Analyst
TeleCardio Project Manager
TeleCardio Analyst
TeleCardio Developer
CardioEducar Project Manager
CardioEducar Analyst
CardioEducar Developer
Database Research Project Manager
Database Research Analyst
Database Research Developer

Figure 4: Visualization of the Organizational Structure through the Hyperbolic Tree

When a professional profile is being recorded or updated, the knowledge a certain employee has is also associated to one or more concepts of the EOSDE ontologies and their instances in order to facilitate this information recovery. For example, knowledge about the Use Case Points technique is associated with the concept "Technique" and its instance "Use Case Points".

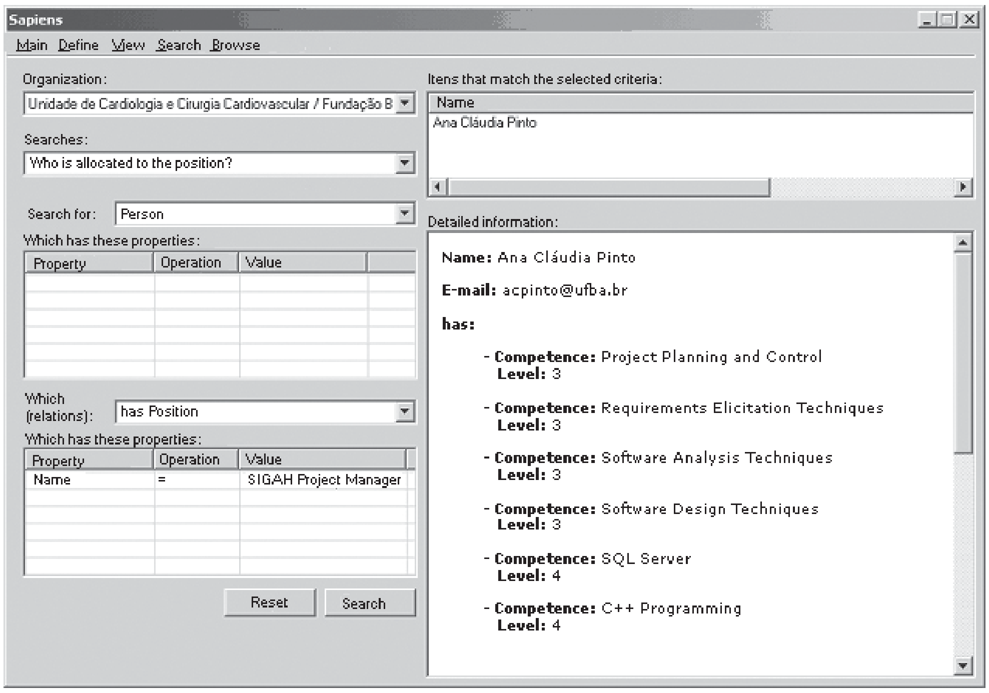
In turn, Sapiens' previously defined search options have been created on the basis of the existing relationships among Enterprise Ontology concepts. Each pre-defined search contains a description, an item to be looked for (based on an ontology concept) and a related item (based on another ontology concept related to the first). In case the user does not wish to carry out one of the listed search options, the existing concepts in the Enterprise Ontology are shown. Consequently, when choosing one of these concepts its relationships are listed and the user can select the related concepts, creating his/her own search. Figure 5 shows Sapiens' search form.

Another use of the Enterprise Ontology is in the report exhibition. The data are shown in form of XML/HTML pages (as shown to the right of Figure 4 and 5) created using only the existing relationships among the ontology concepts and classes used by the tool.

4.2 RHPlan

RHPlan is a software tool to support the planning of human resources for software projects based on organizational knowledge about the corporate competencies and allocation of staff. The resource planning activity is carried out during project planning, when the appropriate competencies to perform the project activities need to be identified so that people can be allocated to the project. The knowledge used by project managers to do this must be shared across the organization so that the organization can learn from its previous successes and failures.

Like Sapiens, RHPlan's class model is based on the Enterprise Ontology. Both manipulate the organization's knowledge map and benefit from the same mapping infrastructure of the ontology concepts for physical model classes. The RHPlan tool also uses the concepts defined in the sub-ontology of Behaviour to describe processes, activities and the necessary competencies for the accomplishment of activities.



Organization: Unidade de Cardiologia e Cirurgia Cardiovascular / Fundação B

Searches: Who is allocated to the position?

Search for: Person

Which has these properties:

Property	Operation	Value

Which (relations): has Position

Which has these properties:

Property	Operation	Value
Name	=	SIGAH Project Manager

Itens that match the selected criteria:

Name
Ana Cláudia Pinto

Detailed information:

Name: Ana Cláudia Pinto

E-mail: acpinto@ufba.br

has:

- **Competence:** Project Planning and Control
  Level: 3
- **Competence:** Requirements Elicitation Techniques
  Level: 3
- **Competence:** Software Analysis Techniques
  Level: 3
- **Competence:** Software Design Techniques
  Level: 3
- **Competence:** SQL Server
  Level: 4
- **Competence:** C++ Programming
  Level: 4

Figure 5: Sapiens' Search Form

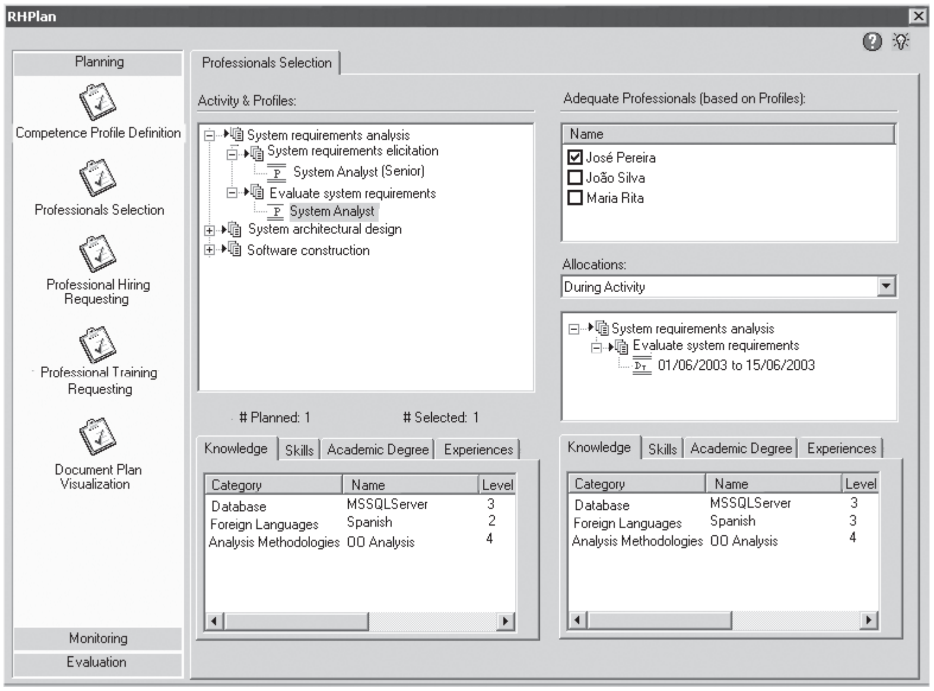
A screenshot of the RHPlan tool can be seen in Figure 6. It illustrates the selection of professionals for a software project, but on the vertical bar on the left all the activities supported by the tool for planning human resources are listed: definition of profiles needed for the execution of each process activity, selection of professionals, request for hiring or training for professionals when the available professionals in the organization do not fit the desired profile, and visualization of the human resources plan.

The first step in planning the human resources for a software project is the identification of the necessary competencies to perform each project activity. After this, the right professionals can be selected by comparing the necessities of each activity and the competencies of each professional in the organization (Figure 6). On the left the user clicks on a professional profile specified for carrying out an activity and then can see the academic degree and the competences required for it. The professionals whose characteristics match the profile defined are shown on the right side. It is assumed that a professional has a compatible profile when each of his/ her competence matches a level equal to or higher than the desired level for the activity. The user has access to the academic degree and the competences of a professional by clicking on it. On analysing the profile of the professionals presented by the tool, the project manager should be able to select the most appropriate individuals to take part in the project.

During the project a manager can monitor human resources by checking the execution of previous activities and allocating or reallocating selected professionals for the next ones. Periodically, throughout the software project life cycle, and after its conclusion, the human resources can be evaluated. RHPlan does not support these evaluations, but once poor performances are identified, corrective actions can be adopted, assigning a new resource to the activity or providing training to the currently allocated professional.

In addition to knowledge about available competencies inside the organization, analysis of past experiences is also of great importance in helping project managers to plan and control human resources. Lessons learned (successful or otherwise) in other projects are very important to avoid committing the same errors and to remember the successes. Two icons, located in the upper right hand corner, allow available knowledge to be consulted as well as knowledge acquisition through an interface to a knowledge acquisition tool called Acknowledge [11]. The knowledge acquired can be indexed by ontological concepts and their instances, which are later used to help in the required knowledge retrieval.

The carrying out of a *post mortem* evaluation is also of great importance for the correct allocation of professionals to new projects [23] because, by evaluating the whole project after its conclusion, the project manager has a clear view of which professionals might have been responsible for the success or failure of a specific task.


RHPlan

Planning: Competence Profile Definition; Professionals Selection; Professional Hiring Requesting; Professional Training Requesting; Document Plan Visualization; Monitoring; Evaluation

Professionals Selection

Activity & Profiles:

```
System requirements analysis
  System requirements elicitation
    System Analyst (Senior)
  Evaluate system requirements
    System Analyst
System architectural design
Software construction
```

Planned: 1   # Selected: 1

Knowledge | Skills | Academic Degree | Experiences

Category	Name	Level
Database	MSSQLServer	3
Foreign Languages	Spanish	2
Analysis Methodologies	OO Analysis	4

Adequate Professionals (based on Profiles):

Name
☑ José Pereira
☐ João Silva
☐ Maria Rita

Allocations: During Activity

```
System requirements analysis
  Evaluate system requirements
    01/06/2003 to 15/06/2003
```

Knowledge | Skills | Academic Degree | Experiences

Category	Name	Level
Database	MSSQLServer	3
Foreign Languages	Spanish	3
Analysis Methodologies	OO Analysis	4

Figure 6: Professionals Selection in the RHPlan tool

4.2 ProcKnow

Both organizational processes and clients' processes knowledge can be useful for organizations that develop and maintain software. Individuals need to understand their role within a larger process and organizations need to understand their processes as a whole to be able to improve them [24]. Furthermore, the objective of software developed for client organizations is to support their processes. Another point is the increased attention which has been given to knowledge built into processes to achieve a smooth integration of Knowledge Management with business processes.

ProcKnow tool aims at allowing the description and visualization of processes executed by an organization. Its process models are able to provide the context in which certain knowledge is used, making it easier to understand both the activity and the knowledge required to implement it. The graphic representation of process models can include activities and their inter relationships, actors, input artefacts and raw material, output artefacts, required and developed knowledge, either formalized or not in the organization, as well as required goods for production used as a resource for the execution of activities.

ProcKnow's main screen is represented in Figure 7. A process model is usually composed of various diagrams. An activity or process represented in a diagram can be detailed in a lower level diagram when necessary, giving rise to nested diagrams. The left window displays the tree of diagrams which make up the process model. The right window initially displays the main diagram of the model but the user can select another diagram to be edited or viewed. The elements in a diagram can be described in the edit mode by clicking on the element and filling in a specific form for each element type. ProcKnow also provides features for visualizing the process model and navigating among its different levels of abstraction, supplying upon request details about the elements represented in the diagrams. The user can have access to formalized knowledge by clicking on its representation in a diagram. Thus, knowledge formalized and available in a digital file can be associated to its representation in a diagram. Moreover, the description of an actor can be linked to a position or function in the organizational structure, to an organizational unit, to a person or even to groups of positions or people in the organization.

The class model of the ProcKnow tool includes classes to deal with the process models described by users, such as: Model, Diagram, Graphic Element (specialized in Graphic Object and Graphic Association), Logical Object, Logical Association and Attribute. It also includes classes to deal with the types of these elements being provided by the tool, such as: Model Type, Diagram Type, Element Type (specialized in Object Type and Association Type), Attribute Type, Grouping Rule and Association Rule. The concepts and relationships of the Enterprise Ontology have provided the contents for the instances of the latter set of classes (E.g. instances of the class "Object Type" are "Knowledge", "Document", "Process" and "Activity"). The use of concepts and relationships of the Enterprise Ontology as instances in the ProcKnow's class model (Element Type instances) makes it possible to identify physical classes used by the ontology-based tools (E.g. Activity and Knowledge classes) which are related to diagram elements in a ProcKnow user's model (E.g. the *Activity* 1 and *Knowledge A* diagram elements represented in Figure 7). This link is made through the references kept by Element Type instances to the related ontological concepts and the mapping among ontological concepts and physical classes created based on them. This allows the definition of mechanisms for tool integration.

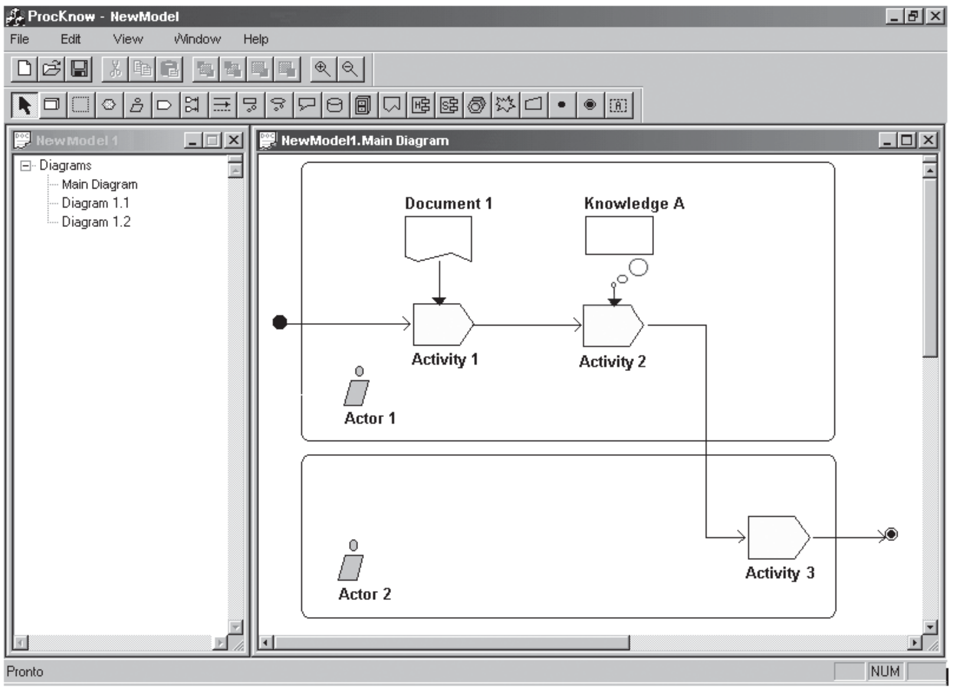


Figure 7: ProcKnow main screen

CONCLUSION

This paper has initially presented an approach to support Knowledge Management in Software Development Environments strongly based on ontologies: the Enterprise-Oriented Software Development Environments (EOSDEs). Such environments are composed of knowledge and data bases, sets of tools and services, knowledge about one or more application domain, experience in software development and maintenance built up by an organization over time and knowledge about the organization itself. They can also contain knowledge about client organizations when suitable. However, it is important to note that the knowledge in EOSDEs can and should evolve over time since the effort involved in building an EOSDE with all the desired knowledge at once may be prohibitive.

After establishing the context, the paper has focused on describing the Enterprise Ontology as well as some EOSDE tools based on it (Sapiens, RHPlan and ProcKnow) in order to show how this ontology has been useful in supporting Knowledge Management in these environments. The Enterprise Ontology has provided a structure to organize knowledge and guide knowledge acquisition in one or more organizations, has allowed the development of generic tools based on its structure and has promoted the integration among these tools by the sharing of a database created based on the ontology structure. In addition to the tools presented here, other EOSDE tools also make use of the Enterprise Ontology.

Currently, EOSDE tools offer automated support to: (i) definition of the organizational structure [6,11], (ii) adaptation of the organization's standard processes for a specific project, (iii) organizational knowledge acquisition [11], (iv) project planning, monitoring and control including time, costs, risks and human resources [11], (v) planning and execution of configuration management activities, (vi) requirements management, (vii) planning and monitoring of corrective actions, (viii) measurement and analysis activities, and (xix) post-mortem analysis.

Since the end of 2003 EOSDEs and its tools, including Sapiens and RHPlan, have been used in Brazilian software companies. An experimental study was planned and executed in 2004 to evaluate the software processes and the automated support provided by EOSDE [25] with respect to their relevance from the point of view of developers. As far as the EOSDE automated support is concerned, 10 organizations have been surveyed regarding developers perception in using this automated support. From 16 developers in different organizations, 50% said EOSDE always reduced the effort required to carry out project activities, 40% said they saved effort for some activities by using the environment. The automated support was considered sometimes adequate by 50% developers and by 50% always adequate.

They also stated that the EOSDE automated support facilitated the dissemination of best practices and supported decision making. The initial results are promising: three companies obtained ISO 9000:2000 certification and one company has just achieved CMMI (Capability Maturity Model Integration) [26] level 2. Nonetheless, further evaluations are needed to address other important aspects such as costs.

As regards the use of Sapiens and RHPlan, we have observed that interest in both tools is proportional to the size of the organization and their projects: the bigger the organization the greater the interest. A possible explanation for this is cost-benefit: the formalization of the organization's knowledge map is laborious for small organizations in relation to the benefit they can obtain from it. In small organizations, the number of professionals available for allocation to software projects is limited and the manager in charge of this probably already possesses tacit knowledge about them. On the other hand, in bigger companies this information is harder to manage and an organization's knowledge map becomes a clear necessity.

ProcKnow tool is planned to be available for EOSDE users at the end of this year.

In the future, we plan to specifically evaluate the *Knowledge Management Infrastructure* of EOSDEs, taking into consideration the different mechanisms and types of knowledge provided by these environments, and attempt to determine their individual benefits and costs. Through this evaluation it should be possible to evaluate the role of the EOSDE ontologies in the improvement of an organization's software processes.

As for approach limitations, the current *Knowledge Management Infrastructure*, through its tool for edition of Domain Theories [5], allows the evolution of Domain Ontologies, including the Software Engineering (SE) Ontology, anytime. However this evolution is only perceived in new EOSDEs generated from the EOSDE infrastructure, because new classes to hold the ontology instances are automatically created based on the new ontology definition. Furthermore, an Enterprise Ontology evolution is not automatically reflected in the tools based on it.

Acknowledgment

The authors wish to thank CNPq and CAPES for the financial support granted to the project Enterprise Oriented Software Development Environments.

References

[1] I. Rus, M. Lindvall. Knowledge Management in Software Engineering. *IEEE Software*. 19(3):26-38, 2002.

[2] D. O'Leary. Enterprise Knowledge Management. *IEEE Computer*. 31(3):54-61, 1998.

[3] M. Uschold, M. Gruninger. Ontologies: principles, methods and applications. *The Knowledge Engineering Review.* 11(2):93-136, 1996.

[4] K. Oliveira, C. Galotta, C. Menezes et al. Defining and Building Domain-Oriented Software Development Environments. *In Proceedings of the International Conference on Software & Systems Engineering and their Applications.* pages 1/8-8/8, 1999.

[5] K. Oliveira, F. Zlot, A. R. Rocha et al. Domain-oriented software development environment. *Journal of Systems and Software*. 72(2):145-161, 2004.

[6] K. Villela, F. Zlot, , G. Santos et al. Knowledge Management in Software Development Environments. *In Proceedings of the International Conference on Software & Systems Engineering and their Applications.* pages 1/8-8/8, 2001.

[7] K. Villela, K. Oliveira, G. Santos et al. Cordis-FBC: an Enterprise Oriented Software Development Environment. *In Proceedings of the Workshop Learning Software Organization*. pages 91-96, 2003.

[8] F. Zlot, K. Oliveira, A. R. Rocha. Modeling Task Knowledge to Support Software Development. *In Proceedings of the International Conference on SE & KE.* pages 35-42, 2002.

[9] K. Althoff, A. Birk, S. Hartkopf et al. Managing Software Engineering Experience for Comprehensive Reuse. *In Proceedings of the International Conference on SE & KE*. pages 10-19, 1999.

[10] A. Abecker, A. Bernardi, K. Hinkelmann et al. Toward a Technology for Organizational Memories. *IEEE Intelligent Systems*. 13(3):40-48, 1998.

[11] M. Montoni, R. Miranda, A. R. Rocha et al. Knowledge Acquisition and Communities of Practice: an Approach to Convert Individual Knowledge into Multi-Organizational Knowledge. *In Proceedings of the Workshop Learning Software Organization*,

pages 110-121, 2004.

[12] R. Falbo, C Menezes, A. R. Rocha. Using Ontologies to Improve Knowledge Integration in Software Engineering Environments. *In Proceedings of the International Conference on Information Systems Analysis and Synthesis*. pages 1/8-8/8, 1998.

[13] B. Kitchenham, G. Travassos, A. Mayrhauser et al. Towards an Ontology of Software Maintenance. *Journal of Software Maintenance: Research and Practice*. 11(6):365-389, 1999.

[14] M. Fox, M. Barbuceanu, M. Gruninger. An Organization Ontology for Enterprise Modeling: Preliminary Concepts for Linking Structure and Behaviour. *Computers in Industry*. 29:123-134, 1996.

[15] M. Uschold, M. King, S. Moralee et al. The Enterprise Ontology. *The Knowledge Engineering Review*. 13(1):31-89, 1998.

[16] D. O'Leary. How Knowledge Reuse Informs Effective System Design and Implementation. *IEEE Intelligent Systems*. 16(1):44-49, 2001.

[17] J. Stader, A. Macintosh. Capability Modeling and Knowledge Management. *In Applications and Innovations in Expert Systems VII*. pages 33-50, Springer-Verlag. Berlim, 2000.

[18] T. Dingsoyr, E. Royrvik. Skills Management as Knowledge Technology in a Software Consultancy Company. *In Lecture Notes in Computer Science - Advances in Learning Software Organizations*. 2176:96 103, 2001.

[19] V. Basili, M. Lindvall, P., Costa. Implementing the Experience Factory concepts as a set of Experience Bases. *In Proceedings of the International Conference on SE & KE*. pages 102-109, 2001.

[20] M. Alavi, D. Leidner. Knowledge Management Systems: Emerging Views and Practices from the Field. *In Proceedings of the Hawaii International Conference on System Sciences*, page 7009, 1999.

[21] P. Pirolli, S. Card, M. van der Wege. The Effect of Information Scent on Searching Information: Visualizations of Large Tree Structures. *In Proceedings of the Advanced Visual Interfaces Conference*, pages 161-172, 2000.

[22] D. O'Leary. Using AI in Knowledge Management: Knowledge Bases and Ontologies. *IEEE Intelligent Systems*. 13(3):34-39, 1998.

[23] A. Birk, T. Dingsoyr, T. Stalhane. Postmorten: Never Leave a Project Without It. *IEEE Software*. 19(3):43-45, 1998.

[24] E. Oh, A. Hoek. Adapting Game Technology to Support Individual and Organizational Learning. *In Proceedings of the International Conference on SE & KE*. pages 347-362, 2001.

[25] A. R. Rocha, M. Montoni, G. Santos et al. Reference Model for Software Process Improvement: a Brazilian Experience. *In Proceedings of the World Conference for Software Quality*, 2005.

[26] Capability Maturity Model Integration (CMMI) Version 1.1 - Staged Representation. Software Engineering Institute, Carnegie Mellon University, Mar 2002.