Learning How to Manage Risks Using Organizational Knowledge

Ricardo A. Falbo, Fabiano B. Ruy, Gleidson Bertollo, Denise F. Togneri

Computer Science Department, Federal University of Espírito Santo, Vitória - ES - Brazil
{falbo, fruy, gbertollo}@inf.ufes.br, togneri@terra.com.br

Abstract. In spite of being an important software activity, many software organizations present difficulties in managing risks. This happens mainly due to their low maturity level, and because Risk Management is a complex and knowledge intensive task that requires experienced project managers, which many times are not available. In order to overcome this barrier, novice software engineers must learn how to perform this task, and organizational knowledge concerning it can be very useful. In this paper, we present a knowledge management approach to support organizational learning in Risk Management.

1 Introduction

In order to survive, an organization must be committed with learning. Its success depends heavily on its tailoring and flexibility capabilities, and it can only be achieved through learning. But we shall not think learning as an individual process only. Organizations should learn with their own experiences, lessons learned must be registered and shared, and relevant knowledge must be institutionalized and reused, preventing repeating mistakes [1].

This holds for software development, i.e. software organizations must conduct themselves as learning organizations. Software development is a collective, complex, and creative effort, and in order to produce quality software products, software organizations should use their organizational software engineering knowledge. Several software process activities can be improved by offering knowledge management facilities to support their accomplishment, such as resource allocation, quality planning and requirement specification, among others. Risk Management is one of them. Risk Management is a complex and knowledge intensive task that requires experienced project managers, which many times are not available.

In this paper, we present an ontology-based knowledge management approach to support organizational learning in Risk Management. There are some related works (presented in section 5) that aim to offer knowledge management (KM) support for risk management, but none of them is centered on ontologies. For us, as pointed by Steffen Staab and his colleagues [2], ontologies are the glue that binds knowledge subprocesses together, and they open the way to a content-oriented view of KM, where knowledge items are interlinked, combined, and used. Thus, we first developed a risk ontology (presented in section 4). Based on this ontology and on a Risk Management general process (discussed in section 2), we built GeRis, a KM-based

tool for supporting Risk Management (presented in section 4). Since GeRis is built based on an ontology, the knowledge items handled by it are annotated with ontological tags that guide knowledge retrieve and dissemination. In fact, GeRis was developed in the context of an ontology-based Software Engineering Environment, called ODE [3], and it uses ODE's KM infrastructure (presented in section 3).

2 Risk Management and Knowledge Management

Several issues contribute to increase the complexity in software development. Technological innovations, delivery periods more and more tight, and high demand for quality products are examples of factors that become software development an uncertain task. In this context, risk analysis is an essential activity in software project management. Its main goal is to identify potentials problems before they happen, taking actions to reduce the probability and impact of them.

The need for risk management in software projects is unquestionable. Almost all norms, quality models and standards of project management claim that risk management is essential. As an example, we can mention CMMI [4], which treats risks analysis in the maturity levels 2 (Project Planning, and Project Monitoring and Control Process Areas) and 3 (Risk Management Process Area).

Generally, risk management includes the following activities:

- Risk Identification: attempts to establish threats (risks) to the project. Its goal is to anticipate what can go wrong in the project. The identified risks in previous similar projects can help the software engineer in that task;
- Risk Analysis: concerns analyzing the identified risks, estimating its probability and occurrence impact (exposure degree);
- Risk Assessment: aims to rank the identified risks and to establish priorities. The goal is to allocate resources only for the most important risks, without managing risks with low probability and low impact;
- Action Planning: concerns planning mitigation and contingency actions for the managed risks (those of higher priority). Mitigation actions aim to reduce probability or impact of a risk before it occurs. Contingency actions assume that mitigation efforts have failed, and are to be executed when a risk occurs;
- Risk Monitoring: as the project initiates and proceeds, managed risks should be monitored. Risks' exposure degrees could change, new risks could appear, or anticipated risks could be no more relevant. It is necessary to control managed risks, to identify new risks, and to accomplish the necessary actions and evaluate the effectiveness of them.

Managing software project risks allows avoiding problems. Disregarding risks is dangerous. It can bring several negative consequences, such as delaying the schedule, increasing costs, or even causing the project cancellation. But, in spite of being an important activity, many software organizations present difficulties in managing risks. This happens mainly because Risk Management is a complex and knowledge intensive task. In this context, managing organizational knowledge about risks is important to improve the accomplishment of this activity, and to allow organizational learning about risk management. In this way, inexperienced project managers can

learn while they work. Learning can be viewed as an evolutionary process that starts when workers execute a task. Supported by KM, project managers can use the organizational knowledge about risk management together with their own previous knowledge to perform the task. During the job, learning can take place in many ways: through the assimilation of the organizational knowledge by the project managers; by the growth of the organizational memory, when project managers relate their experiences; and by the arising of new knowledge, or improvement of existing knowledge, originated from the practice of the task execution. Thus, there is an integration of the project managers' knowledge with the organization's knowledge, in which both learn.

In sum, project managers and organization learn while the firsts work. Thus, to be effective, a knowledge management system should be integrated to the software process. Since Software Engineering Environments (SEEs) integrate collections of tools supporting software engineering activities across the software lifecycle [5], it is natural to integrate KM facilities into a SEE [6]. In this way, to actually offer KM-based support to risk management, we need to integrate the KM system into a SEE. This is what we did: we built GeRis, a KM-based tool supporting risk management that is integrated into ODE, a SEE that is presented following.

3 ODE: An Ontology-based software Development Environment

ODE (Ontology-based software Development Environment) [3] is a Software Engineering Environment (SEE) developed based on some software engineering ontologies. Since the tools in the SEE are built based on ontologies, tool integration is improved, because the tools share a common conceptualization. ODE's architectural style reflects its basis on ontologies. It has two levels, as shown in Figure 1.

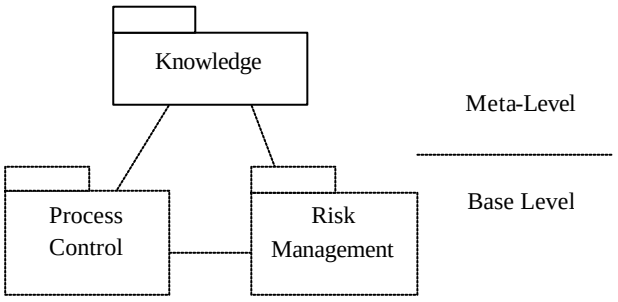

Fig. 1. ODE's Architectural Style.

The base or application level concerns classes that address some software engineering activity. The meta-level (or knowledge level) defines classes that describe knowledge about objects in the base level. The classes in the meta-level are derived directly from the ontologies, using the systematic approach to derive Java object frameworks from ontologies described in [7]. All classes derived directly from the ontology are prefixed by the character "K", indicating that they constitute knowledge in ODE. Meta-level objects can be viewed as instances of ontologies.

The classes in the base level are also built based on ontologies, but since an ontology does not intend to describe all the knowledge involved in a domain, new classes, associations, attributes and operations are defined to deal with specific design decisions made in the application level. Moreover, several classes in the base level have a corresponding Knowledge class in the Knowledge package. In this way, the meta-level can be used to describe base-level objects' characteristics.

Ontologies are also used to deal with knowledge management (KM) in ODE. In ODE's KM approach, ontologies are used to structure the organizational memory (OM), as well as to support the main knowledge services. Figure 2 shows ODE's KM infrastructure [6]. The *organizational memory* (OM) is at the core of the infrastructure, supporting knowledge sharing and reuse. Arranged around it, KM services are available, supporting activities of a general KM process, including knowledge capturing, retrieve, dissemination, use, and maintenance.

It is worthwhile to point that knowledge dissemination is a tool specific service. Knowledge dissemination is proactive in the sense that it is initiated by the system, without requiring the user to explicitly formulate a query. But it is not possible to provide proactive knowledge dissemination without knowing details about the task being done. Thus, knowledge dissemination is not provided by the environment. It is a service that must be implemented in each tool with KM support. In ODE, this service is developed using agents. Each tool with KM support must implement an agent (or a community of agents) that monitors developers using it. When the agent perceives that there are knowledge items that can help the user, it acts showing them. Both ODE's classes and agents are implemented in Java.

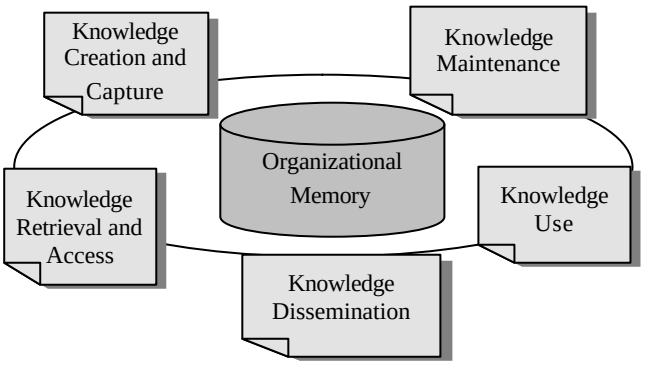

Fig. 2. ODE's KM Infrastructure.

ODE's OM contains four types of knowledge items: artifacts, instances of ontologies, lessons learned and discussion packages. Artifacts created during the software process are submitted to configuration management and become available in ODE's central repository, which is also accessible from the KM infrastructure. Instances of ontologies are used to index other knowledge items. They are also used as suggestions given to developers performing tasks related to the corresponding ontology. Discussion packages are created by packing messages exchanged in ODE's groupware applications. Finally, lessons learned register important findings made during software development.

4 A Knowledge Management Approach for Learning about Risk Management

Managing software project risks is a complex task that requires practitioners with high level of knowledge and experience. Ideally, an experienced project manager should be allocated to this task. But unfortunately, many times organizations do not have practitioners with this profile available for all projects. Attempting to attenuate this problem, we proposed a KM approach to support organizational learning about risk management that consists in providing a risk management tool, called GeRis, integrated to ODE. GeRis uses ODE's KM infrastructure to promote organizational learning for the accomplishment of this task. Using GeRis, inexperienced project managers could accomplish risk management using organizational knowledge and experience about risks accumulated in the software engineering environment.

GeRis was developed grounded on a risk management ontology. Thus, the concepts related to risks are consistently defined and, together to the knowledge items derived from the ontology, they can be used to support learning while the tool is used.

Using the ontology as basis, and capturing experts' knowledge about risk management, an initial knowledge repository (GeRis' meta-level) is built. This repository contains potential software project risks, their categories, and actions that can be taken.

As project managers use GeRis to manage risks, learning occurs, and knowledge share and reuse proceeds by two flows: initially GeRis provides knowledge-based suggestions to the project managers, who uses it to perform the desired task; after that, project managers supply knowledge to GeRis, as lessons learned gather when accomplishing the task, and the generated risk plans. Thus, project managers learn with the tool and with the practice, the organizational knowledge increases, and GeRis could offer a better support next time.

4.1 An Ontology of Software Risks

Before we can devise a strategy for supporting organization learning about Risk Management, we must understand what software risk means. There are several information sources (books, standards, papers, experts, and so on) talking about this subject matter, many times using different terms with no clear semantics established. In order to establish a shared conceptualization about software risks, we developed an ontology of software risks. Several books, standards, and experts were consulted during the ontology development process and a consensus process was conducted.

The method for building ontologies described in [7] was applied. This approach consists of: (i) defining the ontology competence, using competency questions (ontology requirement specification); (ii) capturing concepts, relations, properties and constraints expressed as axioms, using a graphical representation (ontology capture); (iii) formalizing the ontology using a formal language, such as predicate logics (ontology formalization); (iv) evaluating and (v) documenting the ontology.

Following, we present the software risk ontology, using UML as modeling language. It is worthwhile to enhance that we are using the enough subset of UML

adequate to represent ontologies, as discussed in [8]. Due to limitations of space, we present only part of this ontology, concerning the following competency questions:

1. What is the nature of a risk?
2. What are the identified risks in a project?
3. What are the manageable risks in a project in a given moment?
4. What is the probability of occurrence of a risk in a project in a given moment?
5. What is the impact of occurrence of a risk in a project in a given moment?
6. How will a given risk in a project be managed?
7. What actions can be taken to try to avoid the occurrence of a risk or to minimize the consequences of its impact?
8. What actions can be taken in the case of a risk occurs?
9. What were the actions effectively taken to manage a risk in a project?

To address these competency questions, the concepts and relations shown in Figure 3 were considered.

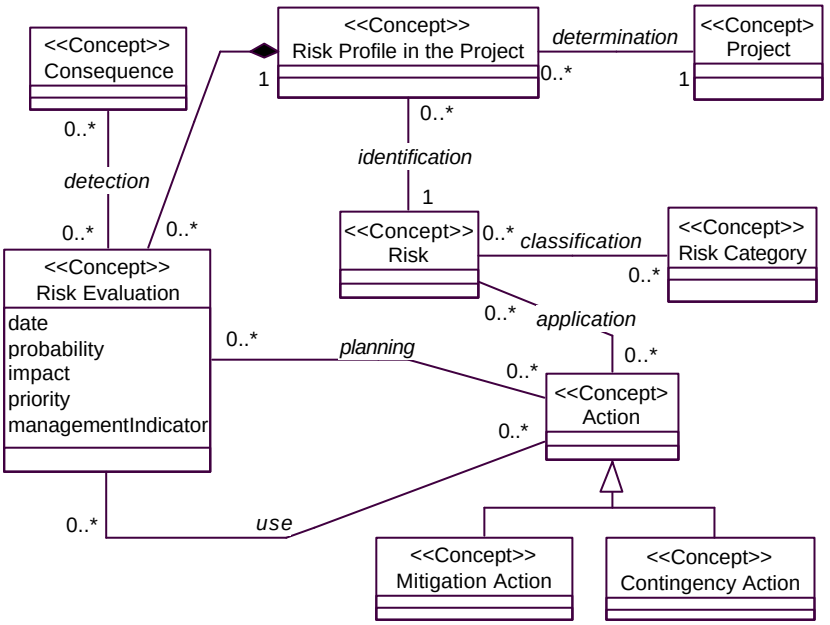

Fig. 3. The Software Risk Ontology .

A *risk* is any condition, event or problem whose occurrence is not certain, but that can affect the *project* negatively if it occurs. A risk can be classified into several *risk categories*, such as risks associated with product size, risks associated with the technology to be used to build the product, risks associated with the people that will do the work, and so on. During the software project planning, risks that can affect the project should be identified and evaluated with the objective of prioritizing and defining what risks will be treated and how they will be treated. Each risk associated to the project defines a *risk profile in the project*. Each project-risk profile is composed of several *risk evaluations*. The first risk evaluation is done during the project planning. Others are done in each one of the project milestones, when the risks

of the project should be reevaluated. A risk evaluation determines the state of the risk in a given instant of the project, i.e. it registers when the evaluation was done (*date*), the *probability* of risk occurrence and its *impact* in case the risk occurs in the course of the project, its *priority*, and if the risk is to be managed (*management indicator*).

Several *actions* can be applied for controlling a risk. These actions can be classified into: *mitigation actions*, which are accomplished proactively with the purpose of reducing the probability of a risk, its associated impact or both, and *contingency actions*, which are planned to be accomplished when an event or condition effectively occurs in the project, making the risk be a reality. During risk evaluation, mitigation and contingency actions can be planned or used. The last refers to actions effectively taken to control a risk.

Finally, if a risk occurs in a project, it has a *consequence* for the project. Therefore, the detection of the occurrence of a risk in a project leads to an analysis of its consequence.

Beyond the model presented in Figure 3, several axioms were defined in the software risk ontology. Those axioms are classified according to [9] as epistemological, consolidation and ontological axioms. The first class of axioms concerns the way concepts and relations are structured. Those axioms are captured by the lightweight extension of UML used to model ontologies [8]. For instance, the following axiom regards the multiplicity of the whole-part relation:

$$\forall(e, p1, p2)\ (compositionProfile(e, p1) \wedge compositionProfile(e, p2)) \rightarrow (p1 = p2)$$

In this epistemological axiom, the predicate compositionProfile formalizes the whole-part relation between Risk Profiles in Projects (*p1, p2*) and Risk Evaluations (*e*). According to this axiom, a risk evaluation (*e*) is part of only one Risk Profile in the Project (*p*), since it is a composition.

Consolidation axioms define constraints for establishing a relation or for defining an object as an instance of a concept, such as:

$$\forall(a,e,p,r)\ ((planning(a,e) \vee use(a,e)) \wedge compositionProfile(e,p) \wedge identification(p,r) \rightarrow application(a,r)$$

This consolidation axiom states that if an action (*a*) is planned or used in an risk evaluation (*e*) of a risk profile in a project (*p*), then that action (*a*) should be applied to the risk (*r*) identified in that profile (*p*).

Ontological axioms, in turn, concern the meaning of concepts and relations and allow new information to be derived from the previously existing knowledge. For instance, if an evaluation (*e*) has planned or used some action (*a*), then that evaluation (*e*) is considered controlled.

$$\forall(e)\ controlled(e) \leftrightarrow \exists(a)\ planned(a,e) \vee use(a,e)$$

4.2 Managing Software Risk Knowledge

Applying the concepts of Risk Management, KM and Organizational Learning, we built GeRis, a risk management tool. GeRis offers KM-based support to the main activities of the risk management process, using ODE's KM infrastructure.

Like any ODE's integrated tool, GeRis architecture follows the two-layer architecture style of the environment, as discussed in section 3. The classes regarding software risks of *Knowledge Package* were derived from the software risk ontology,

and their instances represent part of ODE's knowledge repository about software risk. Figure 4 presents the part of *Knowledge Package* model that addresses software risks. According to this model, GeRis allows registering risks that generally occur in projects, classifying them according to risk categories, and defining actions that can be used to mitigate or to threat a risk. This knowledge is entered in ODE's knowledge repository through a tool for instantiating ODE's software engineering ontologies.

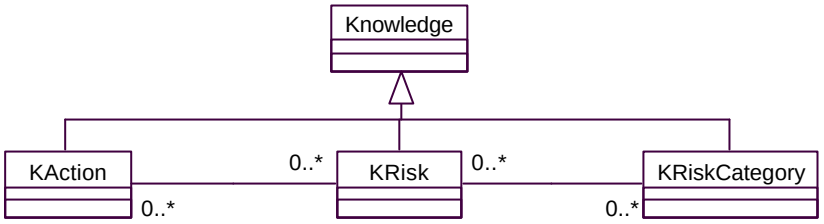

Fig. 4. Knowledge Package's classes regarding software risks.

Figure 5 presents GeRis' class diagram of the Base Level. Those classes are used to support the risk management process described in section 2.

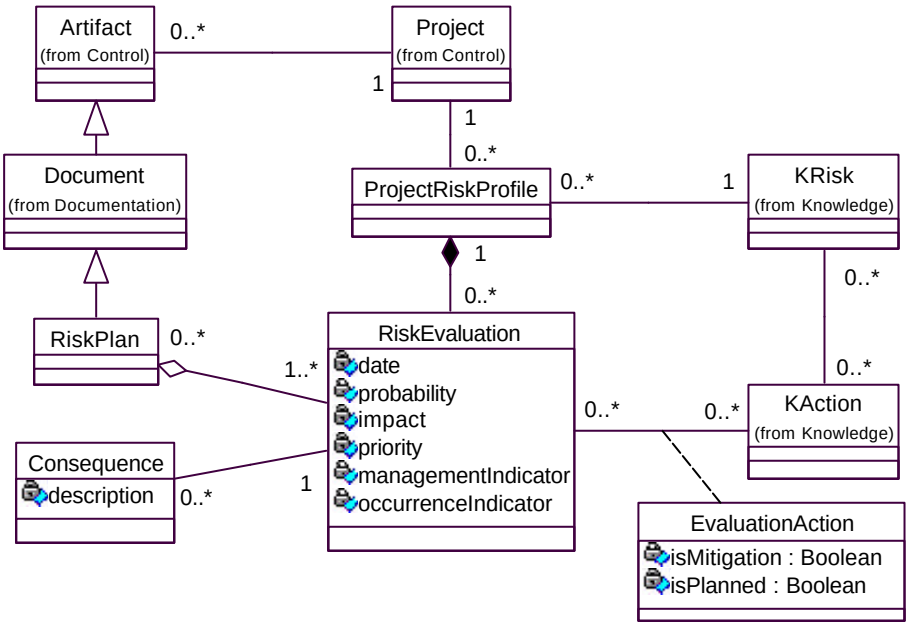

Fig. 5. GeRis Internal Model.

Each project's identified risk holds a risk profile, composed by evaluations. Each risk evaluation determines the probability and impact of the risk, its priority, mitigation and contingency actions planned and used, an indicator pointing if the risk is being managed or not, and an indicator pointing if the risk occurred, or not, in the evaluation date. If the risk occurred, consequences must be indicated. Finally, risk plans can be generated by GeRis, assembling several risk evaluations.

As discussed in section 3, ODE's KM infrastructure provides KM services. Except knowledge dissemination, those services are provided for the environment. Collect and approval of lessons learned, retrieval and maintenance of knowledge items are examples of general services available in ODE that can be used at any time. The dissemination service, however, should be implemented in each tool that uses the infrastructure, through software agents.

Risk planning in GeRis involves the following steps:

1. The project manager has to identify potential risks for the project. In this step, as shown in Figure 6, ontology instances are presented as suggestions, i.e. the risks stored in ODE's knowledge repository (instances of *KRisk*) are presented, so that the project manager does not need to start risk planning from scratch. Also the GeRis' software agent acts in this moment, making risk plans, lessons learned and package discussions available for the project manager (see the icon in right of Geris' menu in Figure 6). This is done based on similarity between projects, using ODE's project characterization that includes: staff features, such as team experience; problem features, such as problem complexity and application domain; and development features, such as development paradigm and software type (Real Time Systems, Information Systems, Web Systems, and so on).
2. Once the risks are identified, they should be evaluated, defining their probability and impact of occurrence. In this step, the project manager can look for similar projects to see what has gone right and wrong in past projects. Also, GeRis' software agent shows relevant knowledge items, such as risk plans for similar projects, and lessons learned.
3. The identified risks should be ranked and prioritized. Again, past experience can be used to support this step.
4. Finally, mitigation and contingency actions for the managed risks (those of higher priority) should be defined. An analogous support to that offered in step 1 is used, that is, ontologies instances are presented as suggestions (instances of *KAction*), and other knowledge items can be retrieved or disseminated by the agent.

In sum, during risk planning, GeRis uses ontology instances as initial suggestions for risk identification and action planning. Also, as a way for favoring learning, a software agent promotes knowledge dissemination. Basically, this service concerns making available other knowledge items from ODE's knowledge repository, including risk plans, lessons learned and discussion packages. From similar projects that had already their risks managed, and from lessons learned in performing risk management, GeRis gives other guidelines for risk planning, presenting identified risks and planned actions defined in previous projects, and reporting risk probabilities, impacts and priorities previously defined in those projects. In this way, project managers are guided by knowledge items of ODE.

We should reinforce that project managers can also use, at any moment, the general KM services, writing his/hers owns learned lessons or searching for knowledge items. These services are available from the *Knowledge Management* menu shown in Figure 6.

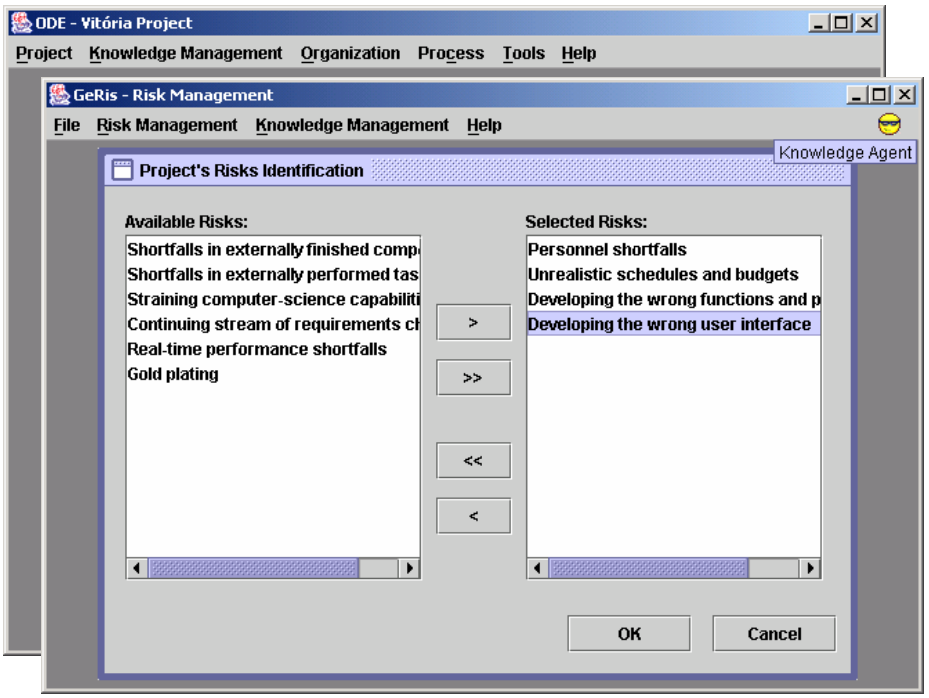

Fig. 6. Identifying Risks in GeRis.

Finally, during risk evaluation, the agent acts in a similar way as described early, making available information on: (i) how this risk was evaluated previously, (ii) which were the defined probabilities and impacts for it, (iii) an indication if the risk occurred, and (iv) its consequences. Besides, the learned lessons about evaluations of this risk can be readily made available.

As GeRis is used more and more, the organizational knowledge on risks evolves, being reused and enriched through the experience acquired by the project managers. Thus, along the time and with the amount of managed projects, the organization accumulates its own risk knowledge, generated from the collective learning of its members.

5 Related Works

There are several works in the literature that describe different approaches for supporting risk management, some of them integrated to software engineering environments. Barros et al. [10] presented a software risk management approach for the Odyssey environment. The risk management process is divided into two risk sub-processes: one for domain risk management, other for application risk management. The main goal of the risk management process for domain models is to identify the most common risks that occur in projects developed within a specific application domain, organizing this information and allowing its reuse in risk management

processes for specific software projects. Within this process, risks are identified and documented by risk patterns, a standard information structure that describes a potential recurring problem to the software development process or product. A risk pattern includes contextual information upon the forces that enable or inhibit the problem in a software project, presenting standard solutions to solve the problem or to minimize the effects of its occurrence. During the risk management process for application development, risks of a project developed within a specific domain are identified reusing the domain risks.

Comparing the Odyssey's approach with ours, we should observe two main aspects. First, they share some ideas. In both cases, the most common risks that occur in software development projects can be registered, shared and reused. Second, while Odyssey's approach uses risk patterns, ODE uses a KM approach, which allows handling several types of knowledge items, such as lessons learned, ontologies instances, discussion packages, and risk plans. The Odyssey's risk patterns are related to ODE's lessons learned, but none of the others ODE's knowledge items have corresponding ones in Odyssey.

Kontio & Basili [11] proposed a risk knowledge capture framework that was built upon the Riskit Method and the Experience Factory. The Riskit's framework enables the knowledge captured to be stored into an Experience Base. The knowledge in the Experience Base can be in various forms, such as raw and summarized data, experiment reports, and lessons learned. In those aspects, ODE's KM infrastructure is very similar to the Riskit framework approach. In fact, ODE's KM infrastructure is also based on the concept of Experience Factory that underlies the Riskit framework. But ODE's KM infrastructure uses other important technologies for KM, such as ontologies and software agents.

Farias et al. [12] presented a software risk management approach based on KM and developed RiscPlan, a tool to support this approach, which is integrated to TABA Workstation, a SEE. Comparing RiscPlan's approach with ours, we should observe that in both cases, the risk management process is supported by KM. But RiscPlan does not use ontologies to establish a common vocabulary, does not have offer proactive knowledge dissemination, as GeRis does through software agents, and does not support several knowledge item types.

6 Conclusions

KM facilitates access and reuse of knowledge, and consequently organizational learning, typically by using several emerging technologies, such as ontologies and software agents. In this paper we presented an approach for managing risk knowledge. In this approach, knowledge workers constantly create new knowledge as they work. Some benefits of this approach can be pointed out: (i) With KM integrated to the environment, it is easier for developers to create new knowledge. In this way, the organizational memory is always evolving. A major concern for KM in ODE is to capture knowledge during the software process without developers' extra effort. Thus, KM is actively integrated into the work process. (ii) In the case of GeRis, project managers are not passive receivers of knowledge, but are active researchers,

constructors, and communicators of knowledge. So that, learning can be improved. (iii) A KM system must provide the information workers need, when they need it. GeRis' agent monitors the project manager actions as he/she works, and inform him/her about potentially relevant knowledge for the task at hand. In this way, GeRis plays an active role in knowledge dissemination.

We are now working to put ODE and its tools, including GeRis, in practice in a software house. Especially concerning GeRis, this effort is being very important, since this software organization does not have a solid culture of accomplishing risk management. But they are looking for an ISO 9000 certification, and then they need to learn more about how to perform this task.

Acknowledgments

The authors acknowledge CAPES and CNPq for the financial support to this work.

References

1. D. A. Garvin, Learning in action: a guide to putting the learning organization to work, USA: Harvard Business School Press, 2000.
2. S. Staab, R. Studer, H.P. Schnurr, Y.Sure, "Knowledge Processes and Ontologies", IEEE Intelligent Systems, vol. 16, No. 1, January/February, 2001.
3. R.A. Falbo, A.C.C. Natali, P.G. Mian, G. Bertollo, F.B. Ruy. "ODE: Ontology-based software Development Environment", Proceedings of the IX Argentine Congress on Computer Science (CACIC'2003), La Plata, Argentina, 2003, pp 1124-1135.
4. *Capability Maturity Model Integration*, Version 1.1, Pittsburgh, PA: Software Engineering Institute, December 2001.
5. W. Harrison, H. Ossher, P. Tarr, "Software Engineering Tools and Environments: A Roadmap", in Proc. of the Future of Software Engineering, ICSE'2000, Ireland, 2000.
6. A.C.C. Natali, R.A. Falbo, "Knowledge Management in Software Engineering Environments", Proc. of the 16th Brazilian Symposium on Software Engineering, Gramado, Brazil, 2002.
7. R.A. Falbo, G. Guizzardi, K.C. Duarte. "An Ontological Approach to Domain Engineering". Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, SEKE'2002, pp. 351- 358, Ischia, Italy, 2002.
8. P.G. Mian, R.A.Falbo, "Supporting Ontology Development with ODEd", Proc. of the 2nd JIISIC, Salvador, Brazil, 2002.
9. R.A. Falbo, C.S. Menezes, A.R.C. Rocha. "A Systematic Approach for Building Ontologies". Proceedings of the 6th Ibero-American Conference on Artificial Intelligence, Lisbon, Portugal, Lecture Notes in Computer Science, vol. 1484, 1998.
10. M.O. Barros, C.M.L. Werner, G.H. Travassos. "Risk Analysis: a key success factor for complex system development". Proceedings of the 12th International Conference Software & Systems Engineering and their Applications, Paris, France, 1999.
11. J. Kontio, V.R. Basili, "Risk Knowledge Capture in the Riskit Method", SEW Proceedings, SEL-96-002, University of Maryland, 1996.
12. L.L. Farias, G.H. Travassos, A.R.C. da Rocha, "Managing Organizational Risk Knowledge", Journal of Universal Computer Science, vol. 9, no. 7, 2003.