Knowledge Management Support for Distributed Agile Software Processes

Harald Holz[1] and Frank Maurer[2]

[1] University of Kaiserslautern, Department of Computer Science,
D-67653 Kaiserslautern, Germany
holz@informatik.uni-kl.de
[2] University of Calgary, Department of Computer Science,
Calgary, Alberta, Canada, T2N 1N4
maurer@cpsc.ucalgary.ca

Abstract. Agile Software Development has put a new focus on the question of how to share knowledge among members of software development teams. In contrast to heavy-weight, document-centric approaches, agile approaches rely on face-to-face communication for knowledge transfer. Pure face-to-face communication is not feasible when applying agile processes in a virtual team setting. In this paper, we argue that the right approach for virtual software development teams using agile methods lies between a radical "none but source code" standpoint, and the multitude of documents proposed by heavy-weight development standards. This paper introduces work on developing a system for the task-based capture and pro-active distribution of recurrent information needs that typically arise for developers, as well as potential ways to satisfy these information needs. Our approach facilitates an incremental organizational learning process to capture and maintain knowledge on what documentation/information is actually needed, such that documentation is created on an "as needed" basis.

Keywords: task-oriented knowledge management support, virtual agile teams

1 Introduction

At first sight, Agile Software Development and Knowledge Management (KM) do not seem to fit well together. As pointed out by Cockburn [5], one of the main characteristics of agile methodologies is their attempt to shift the company's organizational and project memory from external to tacit knowledge, i.e. written documentation is replaced by informal communication among team members. While this might relieve team members from time-consuming documentation activities that are not directly relevant to their current development tasks, the absence of explicit documentation leads to a number of problems:

- Subject matter experts in larger teams find themselves spending much time in repeatedly answering the same questions.
- Team members find themselves in situations where they know that they have had a certain problem before, but cannot remember its solution.

S. Henninger and F. Maurer (Eds.): LSO 2002, LNCS 2640, pp. 60–80, 2003.

- There is no direct knowledge exchange between members of different teams if they do not belong to the same community.
- Important knowledge is lost as soon as experienced developers leave the project or company.

Although the last point is partially mitigated by the strong focus on pair programming and shared code ownership as advocated by Extreme Programming [4] and other agile methods, the other issues still remain. Thus, the advantages of following agile, light-weight methodologies have to be balanced against the disadvantages of the absence of documentation and the lack of knowledge management.

Knowledge sharing is particularly difficult in case of virtual agile teams where team members are not co-located and have less or no opportunity for face-to-face communication. On the other hand, virtual teams often use information technology, e.g. e-mail, newsgroups, on-line chat rooms or Wiki webs, to exchange information. This provides opportunity for knowledge management tools to capture the knowledge that is shared.

One of the main reasons why agile methodologies reduce the emphasis on documents other than source code is that the cost of creating and, in particular, keeping them up-to-date with the continuously changing requirements and source code (or project state/environment) do not pay off. This maintenance problem of keeping externalized knowledge bases up-to-date was also the reason why many knowledge management approaches failed in the end[1].

However, for any software development project there are a number of information sources that contain useful knowledge and need not be actively maintained by the development team; typical examples are e-mail/newsgroup postings discussing technical issues, lessons-learned stories maintained by a central process group, or web sites maintained either within or outside the company containing material about technologies that are used by the project. Thus, even during development activities that occur within software projects that follow a light-weight process, information is often available that could help team members to successfully perform their tasks.

Since studies have shown that people often are not aware of information that might be relevant to them[2] [8] we are investigating ways to pro-actively provide developers with access to information specific to their current tasks and preferences. In the following, we present a systematic, bottom-up approach on capturing recurrent information needs (including potential ways to satisfy those needs) that typically arise for team members as they are performing software development tasks. Depending on a characterization of their current situation (i.e. current activities, individual preferences and skills etc.), developers are provided with those modeled information needs that are triggered by the characterization; in particular, corresponding information items are retrieved for each of these information needs, which are assumed to satisfy the information needs in required detail.

In order to illustrate this approach, we have constructed a system, called the *Process-oriented Information resource Management Environment* (*PRIME*). PRIME pro-

---

[1] In fact, the knowledge base maintenance problem was never solved in the 1980'ies for expert system approaches, and prevented them from wide-spread adaptation in industry.

[2] While pair programming might increase the chances that at least one of the two developers is aware of relevant information, it does not solve the problem in principle.

vides a technical infrastructure for the task-specific capture of information needs and their distribution to developers, as well as feedback communication.

The remainder of this paper is structured as follows: In Section 2, we discuss existing approaches that help in knowledge sharing in distributed agile teams. Section 3 gives an overview on the functionality provided by PRIME, and presents the concepts underlying our approach. Related work on process-oriented knowledge management is discussed and compared with PRIME in the last two sections.

2 Communities of Practice

Agile methods value people and communication over tools and documentation. In its last consequence, using agile methods implies that knowledge management initiatives need to focus on establishing and supporting Communities of Practice [17]. In a software development environment, communities of practice can loosely be defined as groups of people who work on similar topics, use similar approaches & technologies and have similar information needs. As we concentrate in the paper on the knowledge management aspect of agile teams, the last issue is of most interest to us. Thus, the questions to answer are:

- How can we identify communities of practice?
- What kind of support can we provide to them?
- How can we reduce the knowledge maintenance problem?

In this paper, we address these issues from the specific perspective of virtual agile teams (VAT). An agile team is a software development team whose work practices are inspired by agile methods like Extreme Programming, Scrum, Feature-Driven Development, Adaptive Software Development, Agile Software Development and others. A virtual team consists of geographically dispersed members working on a common goal. Team members either belong to the same company, to a virtual enterprise or to volunteer efforts (e.g. open-source projects). A VAT combines these two aspects. In the following, we deal with the question of how knowledge sharing can be supported in VATs.

2.1 Identifying Communities of Practice

We see several ways how a community of practice can be identified in a VAT setting. First, a team working on the same project implicitly is a community of practice. Team members share several information needs, e.g. what is the state of the work, where can I find information about technologies used in the project etc. Second, technologies and tools used implicitly define a community of practice, e.g. people developing EJB-based applications can share their insights with each other. A similar argument holds for tools used by a group of people. A third category is based on the type of task that a group of people is performing, e.g. software testers from various projects may be able to share knowledge.

Communities of practice based on technology or tool use are orthogonal to project-oriented or task-type-oriented communities. This is immediately visible by looking on various technology or tool oriented newsgroups or web sites where people from different organizations share their questions and answers.

2.2 Supporting Communities of Practice in Virtual Teams

Communities of practice often rely on face-to-face meetings where colleagues exchange knowledge via informal communication. This is exemplified by special interest groups of organizations like IEEE or ACM, IT conferences or quality circles in companies. While agile methods are creating project-oriented communities of practice by stressing pair programming and (often) shared ownership, they are not explicitly tapping into communities based on technologies, tools and task types.

In a virtual environment, communities of practice are based on web sites and/or newsgroups. They can either be centered on technologies (e.g. http://www.theserverside.com/home/index.jsp centers around J2EE development), tools (e.g. http://www.jboss.org/ focuses on the JBoss server), or task types (http://www.testing.com is a web site focusing on software testing). Project-oriented portals are addressing knowledge management issues for distributed teams (e.g. http://sourceforge.net/ host more than 46000 open-source projects).
The following technologies seem to be useful for knowledge management in virtual teams:

- Expert directories: Listings of people and their skill sets that allow finding experts in specific technologies, tools or processes. This can be combined with some kind of evaluation mechanism and subcontracting mechanisms (e.g. http://www.elance.com/ is a marketplace for finding free lance contractors). Within companies, especially consultancy companies, expert directories are often combined with retrieval mechanisms for finding people that worked on specific projects.
- Web sites: Lots of knowledge is freely available on web sites. Web sites often maintain FAQs and cross-list other sources of information for a given topic.
- Newsgroups and mailing lists: While web sites often serve as information providers, they often do not support interactive questioning. Newsgroups, on the other hand, allow a user to ask questions. Often, somebody else from somewhere in the world will post the answer quickly. Newsgroups and mailing lists provide a similar functionality: broadcasting information to many recipients. Newsgroups are following a pull approach: a user has to actively read a newsgroup to find if she can help somebody else. Mailing lists, on the other hand, push questions into the inboxes of all subscribes and bring questions to their attention.
- Information retrieval on the Web: The problem with the Web is to find the gems of relevant knowledge in the vast amount of information available. Search engines and web directories can be of help in that. Search engines use information retrieval algorithms to get high precision and recall on searches. Their limitations stem from the syntactical nature of these algorithms. The semantic web initiative tries to overcome this by using ontologies, inference engines and human modeling effort.
- Collaborative filtering: Communities of practice can be used to determine relevance of information for others. The underlying assumption here is that if two people belong to the same community, they are interested in the same information. Amazon.com, for example, is using this approach for sales support: people browsing to one specific book are shown similar books. Sharing a bookmark list within a community of practice (e.g. a project team) reaches a similar effect.

2.3 Reducing the Knowledge Maintenance Problem

Maintaining knowledge sources in areas were knowledge is changing quickly is a costly undertaking. On the other hand, we argue that there is a vast amount of knowledge freely available and already maintained by some group with an interest in it. Hence, the knowledge maintenance problem can be reduced for a VAT by maintaining the information needs that it has instead of maintaining resources that fulfill these needs. Thus, our approach models the information needs of a VAT and uses existing resources for fulfilling these needs. The reduction in knowledge maintenance effort is based on the assumption that it is much less costly for a VAT to maintain a list of questions than to maintain a list of questions and their answers.

The questions that the VAT maintains can be parameterized. The parameters can be bound before the question is send to a knowledge source with context-specific values. E.g. a question can be: "Give me information about EJB technology provided that the EJB-Skill-Level is ?x and the EJB-Server is ?y". Before the question is send to a knowledge source, the skill level can be bound to "low" and the EJB-Server can be bound to "JBoss". These two pieces of information can be extracted from the current task of the project. Obviously, this requires a model of tasks and task-specific information needs; such a model is described in the remainder of this paper.

3 PRIME

Our approach is based on the assumption that developers are willing to maintain their individual lists of current development tasks (called to-do lists) in an application provided by their company. Tools like our MILOS-ASE[3] support this functionality for virtual agile teams. Tasks are represented by a short textual description, together with some (optional) scheduling information (e.g. a due date). They are assigned to iterations of the agile process.

While developer is working on one of his tasks, certain *information needs* arise for him that need to be satisfied in order to successfully perform the activity[4]. These information needs range from simple questions (e.g. "*How do I launch VisualAge in this company?*") to more questions that usually are more complicated to answer (e.g. "*What issues need to be addressed when using Serialization together with EJB?*").

Even in software organizations that follow a light-weight process, typically there are several information sources available which potentially contain information that can be used to satisfy the employees' information needs. These information sources can be either human subject-matter experts (e.g. experienced colleagues) that employees can contact, or any electronic information system that is accessible by employees. Furthermore, information sources might either be maintained outside the company (e.g. newsgroups, mailing list archives, tool vendor websites, etc.), or they might be internally maintained within the organization (e.g. the company's document management system (DMS), bug tracking systems, lessons learned systems, etc.). For software organizations, the existence of external information sources is an important factor, as a considerable amount of relevant up-to-date technical knowledge (in the

[3] http://sern.ucalgary.ca/~milos

[4] In the following, we use the terms 'task' and 'activity' synonymously.

form of documents, newsgroup postings, etc.) is created and made available outside the organization via Internet technology.

Rather than trying to capture or maintain this knowledge for use within the (virtual) software organization, our approach focuses on connecting developers with knowledge sources that are recommended by those communities of practice addressed by the developer's current task. This functionality has been realized in PRIME, a system to capture, distribute and satisfy task-specific information needs. PRIME has been linked with the project support system MILOS [20], the predecessor of the web-based MILOS-ASE system which has been tailored to support distributed agile software projects.

As a basic functionality, MILOS allows each developer to maintain a basic to-do list (see Fig. 1). In addition, each developer is provided with the PRIME Information Assistant component that enables him to access preferred information items, as well as to initiate an automatic retrieval of information items in order to satisfy an actual information need that was expected to arise for him during his current task. In correspondence to these two concepts, the Information Assistant window is divided into two panes, which are explained in the following two sections.

3.1 Personal Task-Specific Information Needs

The pane labeled 'Private InfoNeeds' allows the developer to associate information items (in the form of bookmarks/favorites) with each task on his to-do list that he considers as useful for this task (see Fig. 1). Furthermore, the PRIME Information Assistant allows users to post task-specific questions or information requests to a forum that is used by all members of the virtual team as a means to support each other by posting answers to a colleague's questions (Fig. 2 & 3). Supporting each other is expected by all members of an agile team and helps building a community spirit. The system creates a link to the corresponding question/answer thread and maintains this link as another task-specific information item, providing the user with immediate access to his postings.

The system stores all tasks together with their associated information items; during future tasks, users can search the set of former tasks and associated links via keyword search on their textual descriptions, in order to make use of formerly found information items during later tasks.

3.2 Recurrent Information Needs

The Information Assistant pane labeled 'Global InfoNeeds' allows the developer to browse the set of recurrent information needs that have been modeled in anticipation of likely information needs that might arise for him during the selected task. In particular, the developer can initiate predefined queries that can be performed on available information sources which are supposed to satisfy one selected information need (cf. Fig. 1).

The set of retrieved information needs depends on the characterization the developer has given for his task so far. In PRIME, a characterization consists of

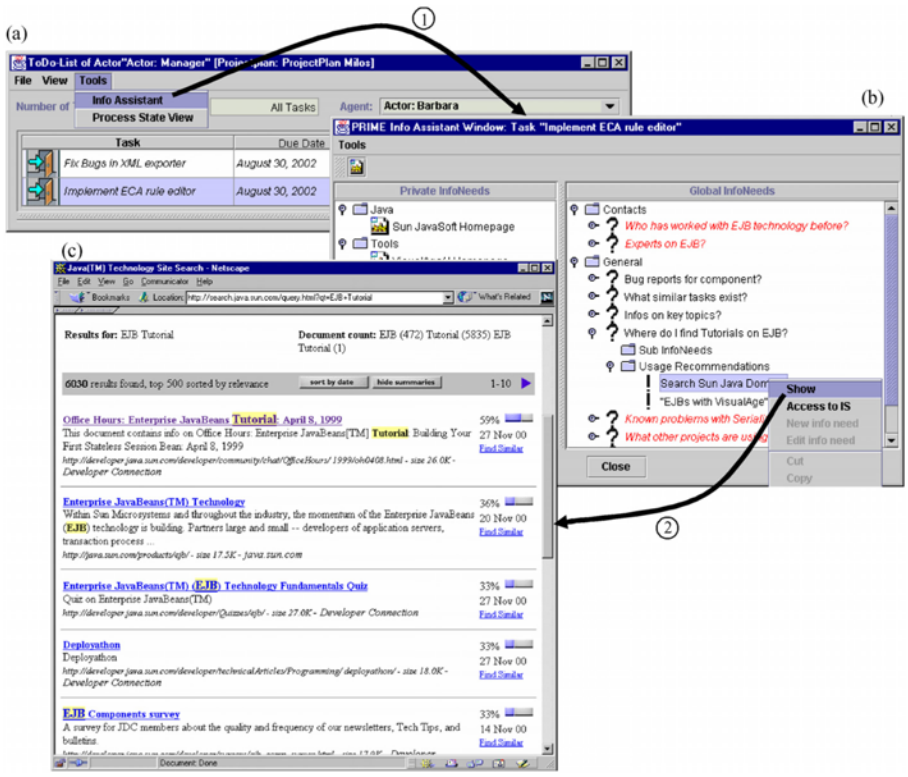

Fig. 1. Snapshot from a to-do list (a), the Information Assistant (b) launched for a selected task (1), and a query execution for an information need to find an EJB tutorial (c): the developer has selected the question "Where can I find a tutorial on EJB?" in the Information Assistant. Issuing the "Show" command on a corresponding IS usage recommendation (2), a browser opens and presents her a list of links which have been retrieved from the Javasoft homepage to the topic "EJB Tutorial". The developer can now refer to the hyperlinks to access the information items.

(i) a classification of the task (i.e. selecting a task type that fits the current task, e.g. “black-box testing”), and
(ii) choosing values for attributes that can be used to further describe the task (e.g. tools or technologies that will be used for executing the task, software components handled during the task, or other key topics that the developer would like to obtain information about).

Figure 4 shows the characterization editor provided by PRIME, which allows developers to characterize their tasks as well as other task-related entities. Task characterizations can be based on entities specified in an organization-specific ontology that is created and maintained by members of different communities that have emerged over time within the virtual organization. Typical entities listed in such an ontology are different task types and tools/technologies, in correspondence to the kinds of communities of practice identified in Section 2.1.

The main purpose of this ontology is to provide each community of practice with the means to systematically define and organize sets of recommended information

resources concerning the entities defined in the ontology. Fig. 5 depicts a simplified excerpt from a task-type hierarchy with associated recurrent information needs.

In the following section, we describe the structure of recurrent information needs in more detail.

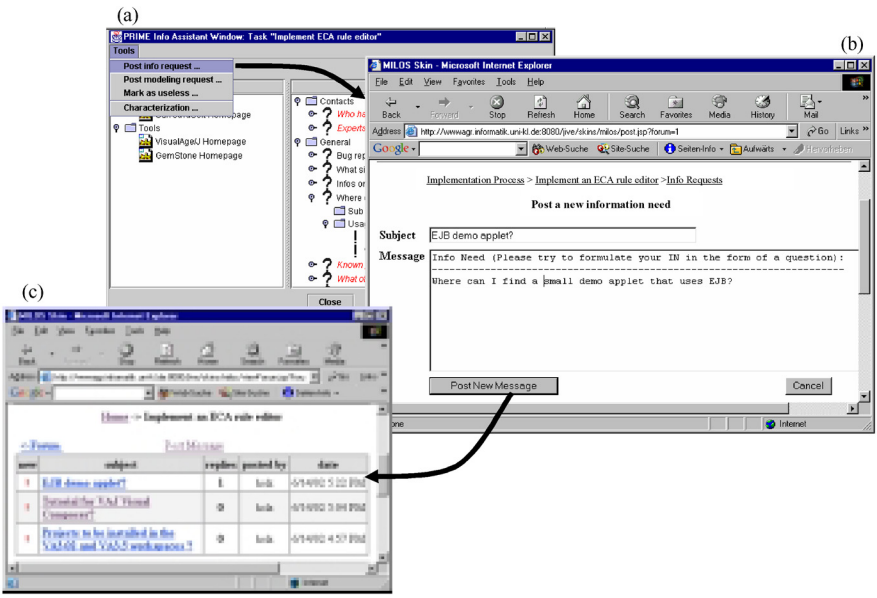

Fig. 2. Snapshot from an agent's Information Assistant (a) that allows agents to post their request (b) directly to a forum (c). For each task, a message forum is provided that maintains the agents' information requests and the replies posted by colleagues. The agents' requests are posted to the corresponding forum by the Information Assistant, after having been extended by a link to the activity during which the information need occurred.

3.3 Representing Recurrent Information Needs

We assume that information is represented as *knowledge items* (or *information items*). A knowledge item is any document (e.g. a MS-Word document, a web page, or an email) that is available to an agent (developer). We say that *information is provided to an agent* if a set of knowledge items is presented to him.

Knowledge items can be obtained by accessing/querying *information sources* that are available to the organization; typical examples of information sources are databases, Document Management Systems (DMS), Web search engines (e.g. Google, AltaVista, etc) or even experienced colleagues. It should be noted that a knowledge item retrieved from an information source can reference another knowledge item or even another information source. An example of such a knowledge item would be an e-mail of a colleague, in which he recommends to consult a particular database.

In the following, we introduce a number of concepts to represent knowledge about available information items that might be useful for agents during activities of a certain class.

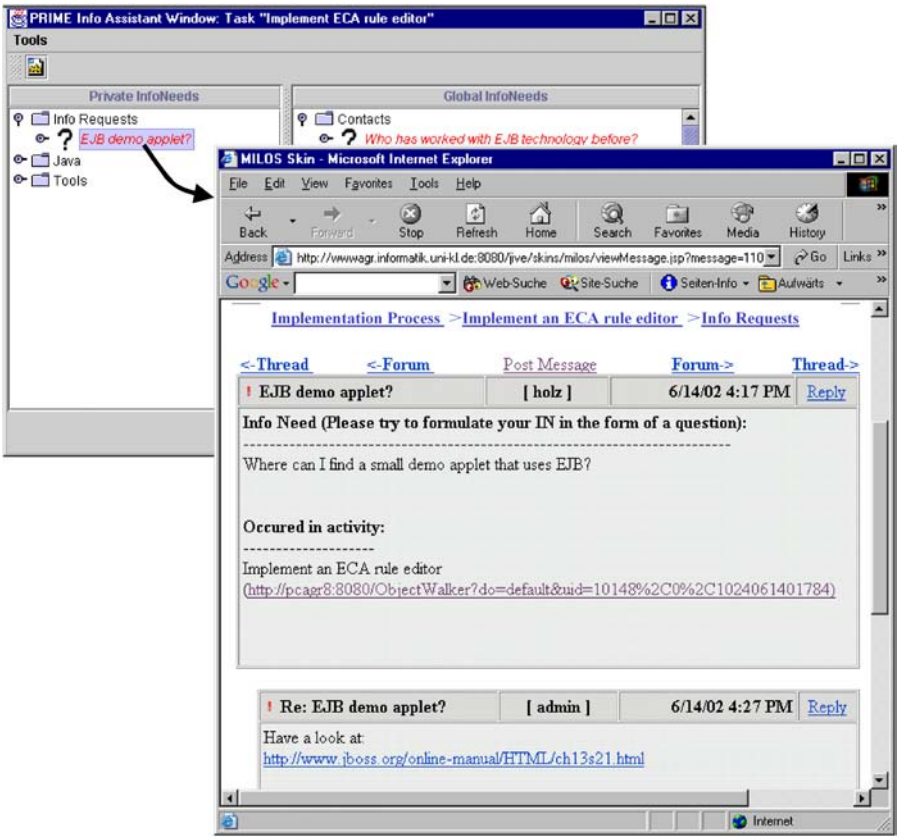

Fig. 3. Snapshot from the Information Assistant for a selected task (a) and thread stored within the task-specific information request forum (b). The Information Assistant maintains a link to the request posted by the agent in the context of a particular task (cf. Fig. 2). Thus, the agent is provided with direct access to the communication thread, i.e. to answers posted by colleagues.

An information source (IS) is represented by the following aspects:

- **name:** the name by which the information source is commonly referred to within the organization
- **contents description:** a short text that explains what information is stored here, and how the information source can be used
- **access:** specifies where the information source can be found or accessed (e.g. a URL in case of an online resource, or contact information about a colleague/expert)
- **query interface:** specifies the information source's interface for automated retrieval, if available (e.g. a CGI script to retrieve items from the information source). This interface will be used to execute queries that have been specified within information needs (see below)
- **quality/cost aspects:** a set of aspects describing quality and cost aspects of the information source (e.g. access cost in case of commercial information services)

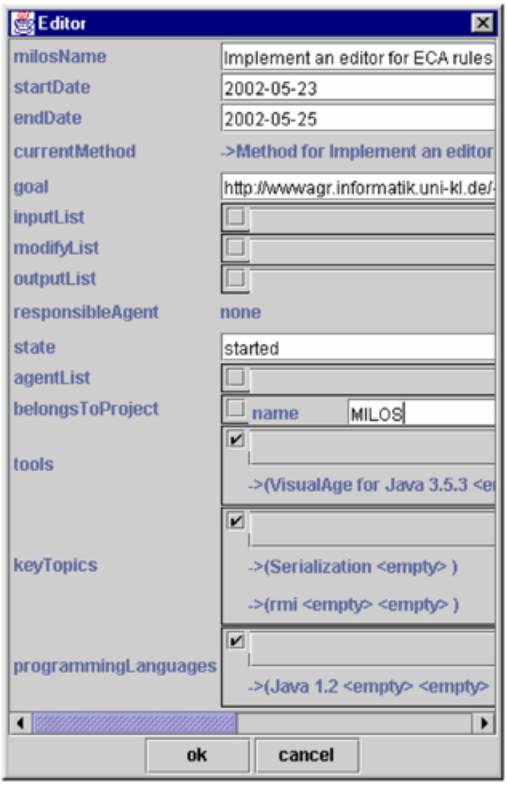

Fig. 4. Snapshot from the characterization editor that allows users to set/change attribute values for a selected task. Here, the user has specified 'VisualAge for Java' as one of the tools that is used during this task, 'Serialization' and 'RMI' as the tasks's key topics, and 'Java 1.2' as the programming language used for coding.

Table 1. Example for an information source representing a Java JDK1.2 language specification document that can be browsed by humans or searched automatically.

IS aspect	**Value**
name	Java JDK1.2 language specification
contents description	Official language specification for Java JDK 1.2
access	http://java.sun.com/products/jdk/1.2/docs/api/
query interface	http://search.java.sun.com/search/java/
quality/cost aspects	high reliability, freely available

Table 1 shows an example for the representation of an information source.

It should be noted that an individual document can also be represented as a special case of an information source (i.e. an information source that contains only one document).

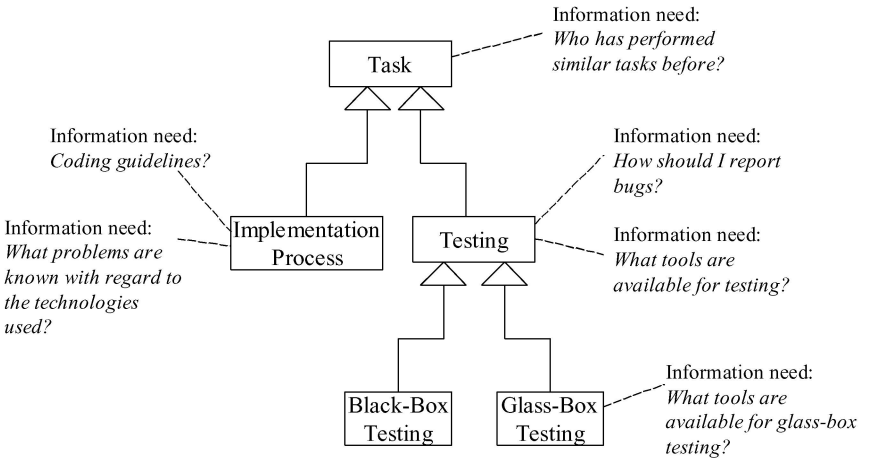

Fig. 5. Excerpt from a task-type specialization hierarchy (depicted as a UML class diagram). Information needs associated with a task type are supposed to be inherited by its sub-types.

Which information sources contain useful information for an agent during his work typically will depend on certain activity and agent characteristics. Furthermore, software engineering activities often undergo changes during their enactment (e.g. schedule changes, product feature changes etc.). As a consequence, the set of information sources that contain useful information changes during an activity's enactment, in correspondence to the activity's changing characteristics.

Consequently, a static list of information sources as a means to provide agents with useful knowledge items is often inadequate for software engineering activities. This leads to the concept of situation-specific *information source recommendations* in order to capture (meta-)knowledge that a certain information source might be useful to agents during activities of a certain type. An IS recommendation is represented by the following aspects:

- **information source:** the information source being recommended
- **task type:** the class of activities for which the information source might contain useful information
- **activity constraints:** specifies conditions on activity characteristics. The information source is only recommended if the conditions hold
- **role constraints:** restrict the recommendation to those agents performing a certain role in the activity
- **skill constraints:** specify conditions concerning the agent's skill profile that must hold in order for the information source to be recommended.

Table 2 shows an example of an information source recommendation.

So far, information source recommendations only describe which information sources are generally considered to be useful during an activity; they do not describe explicitly for what purpose they are considered to be useful, i.e. what information needs might be satisfied by their contents. In order to capture this knowledge, we introduce the concept of information needs.

Table 2. The information source "Java DK1.2 language specification" (see Table 1) is considered useful for all agents taking part in an implementation activity in the role of a "programmer", but only if the implementation language is Java 1.2 and the agent is not already known to be a Java 1.2 expert.

IS recommendation aspect	Value
information source	Java JDK1.2 language specification
task type	Implementation Process
activity constraints	programming language is Java 1.2
role constraints	useful for role 'programmer'
skill constraints	programmer is not a Java 1.2 expert

An information need (IN) encompasses a situation where an agent requires certain information in order to successfully carry out a given activity. We assume that information needs are being expressed in form of a question (e.g. "*Where can I find a tutorial on EJB?*"). These questions are supposed to be of the kind "*Where can I find pieces of information on ..., because it might help me to solve problem x?*", rather than "*What is the solution to problem x?*". In that way, information needs describe goals that, when achieved, enable agents to successfully perform their activities, which in turn are intended to achieve a certain project objectives.

Whether a certain information need arises for an agent will depend on certain activity and agent characteristics (e.g. the technologies that have to be used, the agents experience, skills etc.). Hence, a captured, expected information need should include a specification of the situations in which they typically occur, as introduced for the capture of information source recommendation.

Information needs potentially can be satisfied by accessing one or more information sources via their interface (e.g. send e-mail to a colleague, launch a tool to open a document, or query an information system). As a result, the information source returns one or more information items (e.g. a human answers by e-mail or the Document Management System returns a set of documents). The interpretation of these information items is supposed to either satisfy the information need directly, or help to satisfy it by referring to another information source that might contain the information required to satisfy the information need.

In order to provide a template for the description of a way to access an information source to potentially satisfy an information need under certain conditions, we introduce the concept of an *information source usage recommendation* (*IS usage recommendation*). IS usage recommendations are represented by the following aspects:

- **information source recommendation:** specifies the information source that
- potentially contains information to satisfy an information need, as well as the conditions (in terms of activity, skill and role constraints) when the information source is recommended to be accessed to satisfy the information need.
- **usage direction:** either is a short text explaining in natural language where to find the desired information, or it is a query specification. In the latter case, the query is specified by the following aspects:
  - **comment:** is a short text explaining the query's semantics to the human reader.
  - **queryCommand:** contains a query expression that can be sent to the information source via its query interface (see above).

Building on the concepts introduced so far, a recurrent information need is represented by the following aspects:

- **question:** a textual representation that describes the information need.
- **information source usage recommendations:** a list of information source usage
- recommendations, describing alternative ways to potentially satisfy the information
- need under certain conditions.
- **task type:** the class of activities during which the information need is expected to arise.
- **activity constraints:** specifies conditions on activity characteristics. The information need is only expected to arise if the conditions hold
- **role constraints:** describes for which roles the information need is expected to arise.
- **skill constraints:** specifies conditions concerning the skill profile of an agent participating in the activity. The information need is only expected to arise if the conditions hold.
- **sub-information needs:** references a set of sub-information needs; satisfying these information needs is assumed to provide information that helps in satisfying the referencing "parent" information need.

Figure 6 shows a screenshot from the Information Need Manager interface to define information needs.

We assume that the definition of recurrent information needs will be triggered by either of the following situations:

1. A subject-matter expert posted an answer to an information need, but still finds himself being repeatedly asked to answer this question again by different colleagues. Consequently, he would like to have users being automatically referred to the already documented question/answer thread.
2. Instead of searching the set of former tasks in order to find useful information items for his current task, users might request to have certain information items offered to them on a regular basis (e.g. “Offer me the Java Language Spec. whenever I perform an implementation task that includes coding in Java”).

Figure 7 summarizes the relationships between the different constructs introduced above.

Information source recommendations and information needs are used to differentiate conceptually between two strategies:

- providing access to an information item or source, without stating explicitly what information need(s) it is intended to satisfy
- presenting explicitly formulated information needs in form of a question, together with information items that potentially provide answers to the question. Thus, the question denotes the purpose for offering the information items.

In summary, information source recommendations are used whenever

(i) it is obvious what the corresponding information source is used for (e.g., a language specification will always be used for reference), or

(ii) there are so many different ways of usage that it would be too cumbersome to explicitly list all of them.

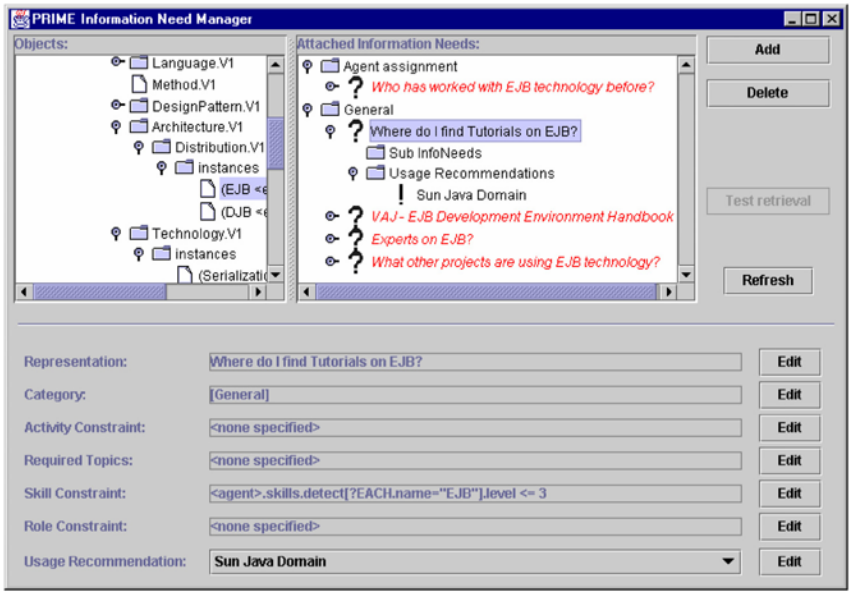

Fig. 6. Snapshot from the Information Need Manager interface: from the tree in the upper-left part of the window, the user has selected the entity 'EJB' from the domain ontology. The tree in the upper-right part displays the information needs associated with that entity, grouped under appropriate categories (rendered as folders). The attribute values of the selected information need are shown in the lower part of the window. For example, the skill constraint is shown, formalizing that the selected information need should only be offered to developers whose skill level concerning 'EJB' technology the less than or equal 3.

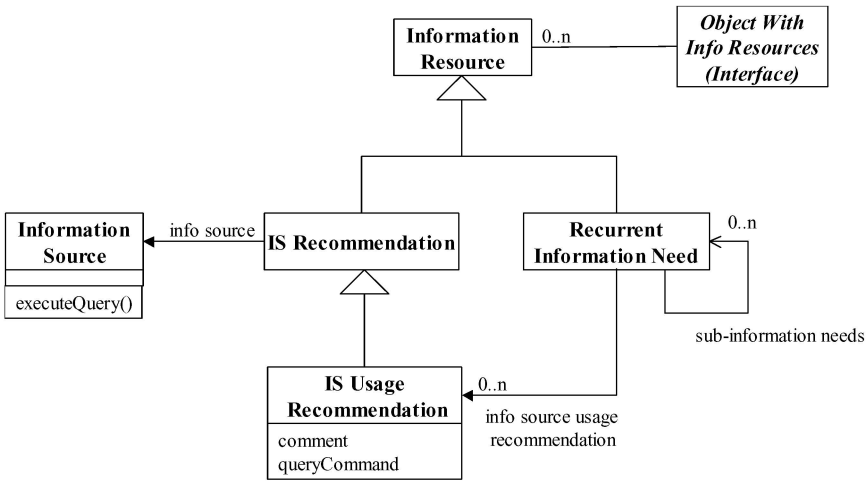

Fig. 7. UML class diagram depicting the relations between the concepts introduced to represent recurrent information needs.

In contrast, the representation of information needs allows capturing in more detail

- *what* information might be useful (expressed as a question)
- *where* and *how* this information can be found (i.e. a list of information sources that potentially contain the information, together with direction on how to access them)
- *when* it might be useful (i.e. constraints on certain activity characteristics available at enactment time)*when* it might be useful (i.e. constraints on certain activity characteristics available at enactment time)
- *to whom* it might be useful (i.e. constraints on performers' roles and skills).

4 Related Work

Most work on integrating Knowledge Management and process support has been done in the field of business processes (see [3] for a recent overview). In the following, we discuss and compare PRIME to related state-of-the-art approaches.

TIDE [18] is a web-based system that facilitates task-based retrieval of documents. A task in TIDE describes a yes/no question that a user is trying to answer; it is represented by a set of weighted slot/value pairs that characterize the question, where all values are terms (words or word stems). The weight of an attribute "loosely ... represents the importance or frequency of the value of that slot in relevant documents." [18]. Furthermore, a task (question) references a set of sub-questions that provide evidence towards answering the parent. This task hierarchy "corresponds to a Bayesian Network, which encodes the probabilistic relationships between questions" [18]. The weights and the task hierarchy are maintained in a task model that users instantiate for their concrete tasks.

The task representation is used to retrieve task-specific relevant documents via the vector space model [12]. Each document is characterized by a vector of weighted terms. A weighted keyword query is derived from a user's task representation by recursively collecting the terms from the question's sub-questions and computing weights for these terms according to the sub-questions' importance to the parent question. TIDE's method of query derivation allows the relevance criterion to be adapted to reflect changes in the users' opinion on relevance by weight modifications.

Compared to PRIME, two main differences can be identified: first, TIDE restricts the notion of a user's task to answering yes/no questions, whereas activities in PRIME reflect arbitrary tasks. Second, the determination of relevance in TIDE is computed by a weighted term query approach; in PRIME, relevance computation is two-step process: (i) determination of relevant information needs based on a symbolic activate/trigger model[5] using boolean expressions, and (ii) launching a well-formed query command to an appropriate information source as defined by the information needs. Thus, in TIDE relevance can only be expressed heuristically in terms of a probabilistic model; in PRIME, relevance can be formulated as a logical fact, which allows it to enforce that certain important documents are always retrieved in specific situations. Also, TIDE's weight-based relevance model will be difficult to maintain, as it is not trivial to identify which weights have to be changed in what way for an intended up-

[5] Information resources are activated by entities present in the activity representation; activated information resource trigger if their constraints are satisfied. Only triggered resources are presented to the agent performing the activity.

date of the relevance criterion. However, the TIDE system might be an interesting extension to PRIME, as TIDE's notion of a task actually corresponds more closely to PRIME's notion of an information need (formulated as a yes/no question): by mapping TIDE's tasks to PRIME's information needs, the information need's query command could be used to trigger TIDE's retrieval mechanism.

EULE [9] is a system that provides computer-based guidance for office workers at Swiss Life. It introduces a formal knowledge representation language that covers data and process aspects, as well as legislation and company regulations relevant for office tasks dealing with life insurance. Users are guided through a sequence of activities to perform their tasks and are being provided with access to relevant documents (contracts, letters, client data etc.); for each activity, users are requested to enter taskspecific data into forms that are presented to them by EULE. Depending on the data entered, new activities might be triggered because of certain laws or regulations. EULE uses deduction to create appropriate instances of rights and obligations, which are represented as concepts; its inference engine couples description logic and deductive database technology.

For each activity, EULE can present an explanation to the user why the activity has to be performed. Furthermore, letters that have to be created during certain activities can be generated automatically from the user's data (in combination with the company's databases). The system was introduced at Swiss Life in mid-1999, and is reported to be highly accepted by employees. Perhaps most interestingly for the approach presented in this thesis, a field study with EULE has been conducted at Swiss Life with positive results: team heads "noticed a considerable relief from the support they usually need to give their team members whenever they encounter a situation they do not know how to deal with" [9].

Because of its inflexible workflow enactment model (build/compile/execute lifecycle), EULE is inadequate to support software development processes[6]. Furthermore, EULE is not designed to provide users with information from external information sources. In addition to the documents related to an activity (contracts, letters, etc.), users are given access to textual representations of laws and regulations that are relevant to the user's current activity. In particular, relevance of information is determined strictly deductively in EULE. In PRIME, relevance can also be computed deductively (by means of information need preconditions formalized in F-Logic); but additionally, information can be retrieved via soft-matching mechanisms (e.g. standard information retrieval approaches [12], similarity measures [11] etc.). Depending on the query command specified within an information need and the retrieval mechanisms supported by available information sources, soft-matching can be used to find relevant information whenever this seems appropriate.

Schnurr et. al. describe an approach based on OntoBroker[7] for Reactive Agent Support [13, 14]. OntoBroker is used to define a domain ontology and to manage an archive of ontology-annotated documents. In addition, OntoBroker scans the documents for facts, stores them in a database, and infers facts from the database using a built-in inference engine.

In OntoBroker, business processes are defined as SGML nets (a special kind of Petri nets). Processes are represented by transitions; predicates based on document contents define when a transition may be executed. Queries (called context-based

---

[6] In fact, EULE is a single-user system and does not support workflows performed by project teams.

[7] OntoBroker is a commercially available F-Logic interpreter (www.ontoprise.com).

views) to the database can be associated to transitions and places. The approach focuses on strongly-structured processes, as the planning of activities and remodelling of SGML nets is not supported. Furthermore, the approach is restricted to F-Logic-based queries to one central repository of annotated documents. For PRIME, F-Logic-based queries to an OntoBroker repository only form one of many possibilities to retrieve information; alternatively, it can provide information retrieved from standard information retrieval systems, relational databases, or case-based reasoning systems. Especially the latter are considered to be of prime importance for experience management within software organizations [15].

Furthermore, the proposed SGML net-based approach does not facilitate an explicit representation of activities. Rather, the activity states are implicitly defined by the state of document attributes. As a consequence, queries can only reference attributes of the document currently being modified by an activity (i.e. transition).

In [16], Wargitsch et. al. present the OM-based flexible WFMS WorkBrain that provides integrated access to different information and communication services. These include a CBR system (storing former workflow cases), a workflow issuebased information system (WIBIS), a mechanical design database, an electronic product catalogue, a know-how database for engineering solutions as well as a traditional DMS.

WorkBrain supports both structural planning and enactment tasks: e.g. workflow construction is supported by retrieving similar former workflow cases, whereas enactment tasks are supported by retrieving documents created in former workflows. However, only the WIBIS system is process-oriented in the sense that processes are used to organize issue threads. Access to the other information systems can not be tailored to specific tasks; in particular, generic queries that are instantiated for concrete tasks cannot be modeled. While the OM is comprised of different information sources that have been made available, no process-specific usage is supported and no automatic query execution takes place, i.e. the OMIS is passive.

The KnowMore framework [2] outlines a three-step deployment process for their workflow-enabled information delivery system. First, a commercial business process modeling tool is used to define a process representation that can be enacted by a workflow engine. Second, knowledge-intensive tasks (KIT) within this process model are identified; these are enriched with KIT variables and with conditional, generic queries. KIT variables represent slots that have to be filled during process enactment, whereas queries represent potential information needs.

During workflow enactment, the generic queries are instantiated with workflow parameters in the context of concrete tasks. After instantiation, the queries are executed by computer agents which encapsulate knowledge on how to retrieve information from a particular information source. The results can automatically be integrated into document templates that specify the input fields that have be filled with retrieved information. In addition, KnowMore users can be presented with explanatory information on the values chosen/retrieved for the template's input fields. The retrieval results are updated whenever the context in which they have been retrieved changes.

Like OntoBroker/SGML, the KnowMore approach focuses on strongly-structured processes and the automated integration of retrieved information, both of which are inadequate for software development processes. With KnowMore, only the enactors (but not the planners) of workflows are supported by the automated information retrieval, and the set of information needs is defined statically in the process model.

KnowMore and PRIME also differ in their main strategy for knowledge delivery. The KnowMore system always automatically executes the whole set of information needs currently regarded as relevant, and then post-processes the results. In PRIME, the agent is given the possibility to choose from a set of offered information needs the one that she considers as relevant in her current situation. The approach implemented in KnowMore is indented to support (automatically) filling in the structured document template, whereas PRIME is intended to support creative processes handling informally specified documents. In particular, the objective of information needs in PRIME is not to fill in the slots (i.e. attributes) of a document's characterization object. On the contrary, the attributed are used to retrieve information items that help a human agent to successfully perform a creative activity.

DECOR[8] [1] builds upon the KnowMore framework, but addresses weaklystructured, knowledge-intensive processes which can not be planned fully in advance. Similar to PRIME, an Information Assistant is proposed that observes the workflow and interprets modelled information needs specified in the process model in order to offer relevant information. The main focus of the DECOR project is to provide a practice-driven, "total solution" for the integration of information retrieval into workflow-embedded, knowledge-intensive tasks. To this end, the project utilizes available, consolidated modeling methods and information technology in combination with research results from the KnowMore approach. However, continuous information need evolution as facilitated with PRIME is not reported to be addressed by DECOR.

Another system that shares some similarities with PRIME is Answer Garden 2 (AG2) [19], which supports astrophysicists in data analysis tasks. AG2 provides an integrated interface that allows users to locate and use about one thousand software components, their associated documentation, tutorials, frequently asked questions, data analysis recipes, or to ask a specific scientific community for help. It relieves users of the burden to remember the different data analysis tools, data formats, interfaces, and help systems, and provides shared recipes on how to use them. In particular, AG2 facilitates the collection and dissemination of organizational knowledge by building a database of commonly asked questions that 'grows "organically" as new questions arise and are answered' [19]. However, AG2 does not allow for different types of tasks, explicit task characterizations, or proactive, situation-specific distribution of those commonly asked questions.

5 Conclusion

Whereas face-to-face communication between team members might have the highest bandwidth for knowledge exchange, there are circumstances when this is either not feasible (as e.g. for VATs) or not always desirable (e.g. because of a communication overload for experts). In addition, a considerable amount of explicit knowledge is available on the Internet in the form of newsgroup postings, technology reports, web sites dedicated to certain tools/technologies, etc. Hence, additional support should be made available to team members to promote the use of information sources that are readily available.

A similar situation appears in open source projects, where newcomers face the problem of catching up with the knowledge of experienced project members, part of

[8] Delivery of context-sensitive organizational knowledge.

which is reflected in mailing list archives. We believe that capturing and distributing proactively typical information needs of newcomers with regard to certain system components, technology used, etc. will greatly relieve experienced group members from having to answer the same standard questions repeatedly; at the same time, newcomers are relieved from sifting through large FAQ and mailing lists.

In this paper, we presented a system to capture and distribute task-specific knowledge about available information resources in the form of explicitly represented recurrent information needs. Depending on the characterization of a currently selected task, a set of information needs is retrieved and presented to the user in the form of a list of corresponding textual questions. From this list, the team member is assumed to choose one that corresponds best to his current information need. For this chosen information need, the predefined information source usage recommendations are executed (i.e. the specified query commands are instantiated and sent to appropriate information systems, or contact information for human subject-matter experts is displayed) to provide the user with information items that potentially satisfy his information need.

In particular, PRIME allows a smooth introduction of Knowledge Management services into the every-day work practice of members of a VAT. To begin with, the system can be used by team members to maintain task-specific bookmarks (i.e. URL links to favourite documents), providing users with a task-oriented way to organize and quickly access their documents. As valuable information is already available on the Internet or in the organization's document repository, we argue that our approach ameliorates the knowledge acquisition bottleneck problem that let many KM initiatives fail in the beginning. In addition, PRIME's forum component[9] serves as platform for task-specific communication with the experienced colleagues. That way, additional task-specific information items can be captured on the fly.

Up to this stage, no modelling of a domain ontology is required. Furthermore, initial modelling of recurrent information need can start from basic task characterizations that consist of keyword lists to name any tools and technologies handled during the task; corresponding query commands can then search the available information sources for appropriate keyword (combinations). Only when

- users start to express an interest in being provided with certain bookmarks on a systematic basis during a certain class of activities, or
- subject-matter experts (or community-of-practice members) decide because of repeatedly asked questions that users should be provided with certain information during a class of activities, or whenever they are handling a certain tool/technology,

the need arises to capture and formalize these requests in the form of a corresponding entity in the domain ontology together with a set of explicit information resources, and to provide access to the requested information from appropriate information sources. Because of the personal gain achievable by explicitly modelled and automatically retrieved information needs, we argue that people will be willing to invest some time in modelling efforts (or posting modelling requests). Essentially, the explicitly represented information sources proposed in this work can be seen as a special kind of "markers and props" described in Cockburn's Manifesto For Software Development [6], which people use to "inform, remind and inspire themselves and each

[9] Based on Jive (www.jivesoftware.com)

other in getting the next move on the game", and which serve to "inform and assist the players of the next game".

Currently, the implementation of PRIME has reached a stage where students have started to use it during their implementation activities on MILOS. From our experience gained so far, future extensions of the work presented here will need to address the effort required for information needs modelling. As an alternative to the explicit representation of logical preconditions, we intend to adapt and integrate techniques known from Collaborative Filtering (see e.g. [7]) or Case-Based Reasoning (see e.g. [11, 10]) with our mechanism for situation-specific information need retrieval. Usage of this technology could provide team members with information items that colleagues found useful who had 'similar' information needs, or with former information needs that (other) team members had during 'similar' situations. It is to be hoped that such extensions could narrow the current gap between the low effort required to maintain the users' personally preferred information resources during their activities, and the relatively high effort required to model fully specified, generic information needs. One of the graduate students in Calgary is currently working on a comparison between text retrieval approaches and the ontology-based approach presented here.

Acknowledgements

The work on MILOS was supported by the DFG (as part of SFB 501: "Development of large systems with generic methods"), NSERC, and The University of Calgary, with several research grants. We would like to thank Empolis knowledge management division[10] for providing us with their CBR middleware 'Orenge'.

References

1. A. Abecker, A. Bernardi, S. Ntioudis, G. Mentzas, R. Herterich, C. Houy, S. Müller and M. Legal. The DECOR Toolbox for Workflow-Embedded Organizational Memory Access, In: Proc. of the 3rd Int. Conf. on Enterprise Information Systems (ICEIS 2001), Portugal, July 7-10 2001, Vol. 1, 2001.
2. A. Abecker, A. Bernardi and M. Sintek. Enterprise Information Infrastructures For Active, Context-Specific Knowledge Delivery. ECIS'99 - The 7th European Conference on Information Systems, Copenhagen, Denmark, June 1999.
3. A. Abecker, K. Hinkelmann, H. Maus and H.-J. Müller (Hrsg.). Geschäftsprozessorientiertes Wissensmanagement, Springer, 2002.
4. Beck, K.: Extreme Programming Explained: Embrace Exchange. Addison-Wesley, 1999.
5. Cockburn, A.: Agile Software Development Joins the "Would-Be" Crowd. Cutter IT Journal, Vol. 15, No. 1 (2002) 6-12.
6. Cockburn, A.: Human's and Technology Manifesto For Software Development. http://member.aol.com/acockburn/manifesto.html.
7. H. Liebermann. Letizia: An Agent that assists web browsing. In Proc. of the 13th Int. Joint Conference on Artificial Intelligence, San Francisco, CA, Morgan Kaufmann, 1995.
8. Mahe, S., Rieu, C.: Towards a Pull-Approach of KM for Improving Enterprise Flexibility Responsiveness: A Necessary First Step for Introducing Knowledge Management in Small and Medium Enterprises. In: Proceedings of the International Symposium on Management of Industrial and Corporate Knowledge (ISMICK '97), Compiegne, 1997.

[10] www.empolis.com

9. U. Reimer, A. Margelisch and M. Staudt. A Knowledge-Based Approach to Support Business Processes, AAAI Workshop on Bringing Knowledge to Business Processes, 20-22 March, 2000.
10. M. M. Richter. CBR: Past and Future - A Personal View. Invited Talk, International Conference on Case-Based Reasoning (ICCBR-2001), Vancouver, British Columbia, Canada, 30 July - 2 August 2001. http://wwwagr.informatik.uni-kl.de/~richter/
11. M. M. Richter and K.-D. Althoff. Similarity and Utility in Non-Numerical Domains, Mathematische Methoden der Wirtschaftswissenschaften, Physika-Verlag, pp. 403–413, 2001
12. G. Salton and M. McGill. Introduction to Modern Information Retrieval, McGray-Hill, 1983.
13. H.-P. Schnurr, S. Staab and R. Studer. Ontology-based Process Support. Workshop on Exploring Synergies of Knowledge Management and Case-Based Reasoning (AAAI-99). Technical Report, Menlo Park: AAAI.
14. S. Staab and H.-P. Schnurr. Smart Task Support through Proactive Access to Organizational Memory. Journal of Knowledge-based Systems 13(5). Elsevier, 2000.
15. C. Tautz. Customizing Software Engineering Experience Management Systems to Organizational Needs, Ph.D. thesis, Universität Kaiserslautern, 2000.
16. C. Wargitsch, T. Wewers and F. Theisinger. An Organizational-Memory-Based Approach for an Evolutionary Workflow Management System - Concepts and Implementation, Proceedings of the 31st Annual Hawaii International Conference on System Sciences, 1998, p174-183.
17. Wenger, E., Snyder., W.M.: Communities of Practice: The Organizational Frontier. Harvard Business Review , Jan-Feb (2000) 139-145.
18. M. Wolverton. Task-Based Information Management, ACM Computing Surveys, Vol. 31, Number 2es, 1999.
19. Ackerman, M.S., McDonald, D.W., "Answer Garden 2: Merging Organizational Memory with Collaborative Help." Computer Supported Cooperative Work (CSCW 96), (Boston, MA, 1996), 1996, pp. 97-105.
20. Maurer, F., Dellen, B, Bendeck, F., Goldmann, S., Holz, H., Kötting, B., Schaaf, M.: Merging Project Planning and Web-Enabled Dynamic Workflow Technologies. IEEE Internet Computing May/June 2000, pp. 65-74.