KNOWLEDGE PROVISIONING: A CONTEXT-SENSITIVE PROCESS-ORIENTED APPROACH APPLIED TO SOFTWARE ENGINEERING ENVIRONMENTS

Gregor Grambow[1], Roy Oberhauser[1] and Manfred Reichert[2]
[1]*Computer Science Dept., Aalen University, Aalen, Germany*
[2] *Institute for Databases and Information Systems, Ulm University, Germany*
{gregor.grambow, roy.oberhauser}@htw-aalen.de, manfred.reichert@uni-ulm.de

Keywords: Context-Sensitive Applications: Context-Aware Systems: Semantic Technology: Knowledge-Based Systems: Process-Aware Information Systems: Software Engineering Environments.

Abstract: Software development is a complex, dynamic, and highly intellectual process that provides automation challenges in the areas of process and knowledge management. Moreover, the ability to support the context-sensitive provisioning of knowledge is further exacerbated by the rapidly changing technologies, processes, knowledge, practices, methods, and tool chains that software engineering involves. Thus, the effective and timely provisioning of knowledge and its concrete utilization in the software development process remains problematic. Reasons for this include the need to ascertain the context, to be aware of the process, and to reason and select the appropriate knowledge to provision while abiding by human and other constraints. For such dynamic knowledge and process environments, this paper describes an approach for realizing a knowledge-based system that automatically provisions knowledge aligned with both the actual context (user, process, and project) and with automated workflow governance. To demonstrate the feasibility of the approach, a scenario-based application of the implementation to the software engineering domain is shown.

1 INTRODUCTION

Software engineering (SE) projects continue to face challenges and struggle with unique difficulties (Brooks, 1986)(Yourdon, 2004)(Jones, 2010). Today, software development is typically projectized, involves a dynamic and complicated collaborative team process, and faces numerous challenges including process governance and knowledge support. In support of software development, SE environments (SEEs) consist of an ever-changing milieu of technologies, processes, knowledge, practices, methods, and tools.

Process management can foster project efficiency (Gibson et al., 2006) and product quality (Heravizadeh, 2009). For example, (Müller, 2006) and (Lenz & Reichert, 2007) demonstrate that process automation is beneficial in industries that exhibit repeatable and standardized processes, which current business process management systems (BPMS) support. Yet SE is characterized by multiform and divergent process models, unique projects, a creative and intellectual process, and collaborative team interactions, all of which impact workflow models (Cugola et al., 1995)(Dellen & Maurer, 1996). These challenges have hitherto hindered automated concrete SE process guidance and often relegated SE processes to generalized and rather abstract process models (e.g., VM-XT (Rausch et al., 2005)) that are operationally vague, consisting of inanimate documentation-centric process guidance.

We have previously developed an automated context-sensitive workflow adaptation approach in support of quality assurance (Grambow et al., 2011b). Yet comprehensive quality assurance requires more than automated context-aware process support. Any new product development is a knowledge-intensive task (Ramesh & Tiwana, 1999) and software processes are mostly knowledge processes (Kess & Haapasalo, 2002). *Knowledge management* and operational *knowledge provisioning* are essential for distributing knowledge among actors and retaining and exploiting experience. Automated systems can support this by capturing, maintaining, reusing, and transferring

knowledge (Teigland, Fey, & Birkinshaw, 1998). Because of their ability to easily create and access information, Wikis are frequently used for knowledge management in SE, yet the retrieval of contextually relevant information remains difficult (Schaffert et al, 2008). Information is captured and stored, but its reuse remains problematic. If knowledge use were connected with the operational SE process execution in a context-sensitive manner, effective and efficient knowledge utilization could be automated and better supported while improving overall product and process quality.

SEEs provide a uniquely challenging domain for a comprehensive and viable solution that automates the integration of *knowledge* and *process management*. In this regard, this paper extends our previous work (Grambow et al., 2011a), contributing a practical approach that elucidates the realization of context-sensitive knowledge provisioning in conjunction with a process-aware information system (PAIS), while supporting additional knowledge types and guidance configurability. This paper abstracts and generalizes the definition of the knowledge the system shall provide. Furthermore, a new component is introduced that enables more flexible and generalized knowledge management and automates the provisioning of that knowledge.

The remainder of this paper is structured as follows: Section 2 discusses the requirements, Section 3 describes the solution concept, and Section 4 uses an application scenario to demonstrate the solution approach. Section 5 details the technical realization. Related work is discussed in Section 6, and Section 7 draws conclusions.

2 REQUIREMENTS

This section elicits detailed requirements for a system seeking to provision context-sensitive and process-aware knowledge. Note that for the scope of this paper, our knowledge-based system considers knowledge to be represented by information that can be utilized dynamically by the system. We do not differentiate between knowledge and information. Furthermore, a user workflow is a defined sequence of operational activities a user performs to reach a certain goal (e.g., coding to develop new functionality).

A system that actively and automatically supports knowledge provisioning should fulfill the following requirements:

R:KnowStor. To be able to use and disseminate knowledge in a project, the presence of some facility to store knowledge is required. The storage and management of knowledge for users shall not be cumbersome. Additionally, to enable an automated system to access and use this information, machine-readable semantic annotation shall be supported.

R:CntxProc. SE projects are highly dynamic with a multitude of events (relating to processes, artifacts, users) occurring at run time. A system that aims at providing situational information support must have facilities to acquire and process context information and to use it for information provision.

R:ExtKnow. Since not all the information users require is stored locally in the system, a facility to integrate external information sources is required. Examples of such information include process documentation or external web pages.

R:ProcMgmt. To achieve automated knowledge support in a user process, a means of automatically governing user process models is required. That way, concrete workflows can be executed for the users that govern their operational activities in conjunction with the abstract process. Otherwise, these activities could quickly diverge from the plan, and a system to provide automated information support could fail, due to its lack of awareness of the current process activities of the user.

R:AutoKnowProv. To realize automated knowledge provisioning for processes, both automated retrieval of stored information and automated process support are essential. To support their convergence, a facility that enables the context-sensitive provisioning of information at the appropriate points in a process is required.

R:CfgKnowDist. Certain process information may be specialized and not generally applicable. Furthermore, the amount of information provided must not exceed certain thresholds, so that users are not subjected to information overload and thus overlook important information. Therefore, the information provision should be configurable, e.g., based on the needs or preferences of projects, processes, or users.

3 KNOWLEDGE PROVISIONING INTEGRATION CONCEPT

This section presents the concept behind our solution approach. The first subsection provides an overview of how the component collaborations address different problem areas to achieve the desired capabilities, with succeeding subsections further elaborating various aspects of the approach. The last

subsection concretely describes how information is used within the process to support users.

3.1 Component Collaboration

Our solution approach aims to provide the user with a consistent knowledge provisioning system (KPS) that realizes holistic information support for the SE process. The Context-aware Software Engineering Environment Event-driven frameworK (CoSEEEK) (Oberhauser, 2010) was used as a basis. Various components in the CoSEEEK KPS collaborate. This is illustrated in Figure 1 and explained in the following.

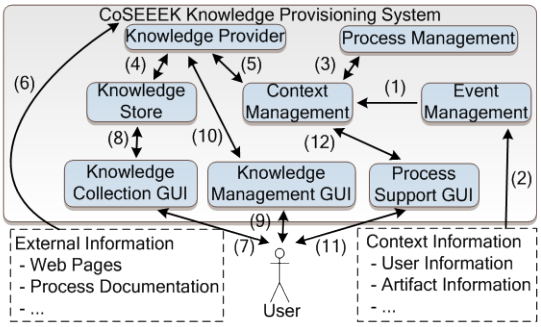


Figure 1: The CoSEEEK System.

The central component of the KPS is the *Context Management* component, which stores, aggregates, and processes all high-level information relevant to the KPS. It incorporates context information about users, artifacts, and various events as well as information from process execution. It receives context information from the *Event Management* component (1), whose responsibility is the acquisition and aggregation of events from the SE environment (2) (cf. *R:CntxProc*). This is realized by a set of sensors integrated in external tools, such as IDEs (Integrated Development Environments) or source control systems used within a SE project.

The *Process Management* component enables a SE process model implementation as well as operational process support. This is done by means of automated workflows actively governed by that component. The *Context* and *Process Management* components work together to enable the usage of context information in the process and to better align process execution with reality (cf. *R:ProcMgmt*).

Knowledge management is realized by the *Knowledge Store* and *Knowledge Provider*. The former is utilized to store user-relevant information and make it available to the KPS via machine-readable semantics. The *Knowledge Provider* coordinates that information (4) and provides it to the *Context Management* component (5) to be injected into the users' workflows (3) in the appropriate context (cf. *R:AutoKnowProv*). The *Knowledge Provider* is also responsible for the abstract definition of user relevant information within the KPS (called *Guidance Items*), which is referenced by the *Context Management* component, as well as for the integration of external information resources (6) (cf. *R:ExtKnow*).

The user can enter relevant project or SE information (e.g., best practices) using the *Knowledge Collection GUI* (7). That information is stored in the *Knowledge Store* (8) (cf. *R:KnowStor*). The *Knowledge Management GUI* (9) allows users to integrate external information (e.g., external process documentation) or configure the way the information is provided (e.g., 'This guidance is applicable to this role at that point in the process') (cf. *R:CfgKnowDist*). The configuration for information provisioning is stored directly in the *Knowledge Provider* (10). All guidance is then distributed to the user by the *Process Support GUI* (11). That GUI receives its information from the *Context Management* component (12), which unites information on the activity from the *Process Management* component (3) with additional SE information from the *Knowledge Provider* (5).

3.2 Context and Process

The *Context* and *Process Management* components collaborate. While the latter is responsible for operational workflow guidance, the former requires direct access to all concepts related to process management, since it unites these with context data from *Event Management* and *Knowledge Provider*. To enable *Context Management* to be process-aware and directly intervene in workflow execution, process concepts are modeled in the *Context Management* component as well, and are connected with their equivalents in the *Process Management* component. This is illustrated in Figure 2 and explained in the following.

Figure 2 shows the *Context* and *Process Management* components containing various concepts: In *Process Management,* there is a workflow instance containing the four activities 'A1' – 'A4'. In *Context Management,* there are mappings and extensions of these concepts. A *Work Unit Container* maps the workflow and *Work Units* map the activities. Via these mappings, the *Context Management* component controls workflow execution, is in charge of controlling activity termination, enabling various factors to be

incorporated in the activity termination procedure. This can be used for injecting information into the users' workflow. For example, a checklist can be incorporated that is automatically provided to the user and that must be processed. In that case, the KPS can require both the completion of the current user activity and the completion of the related checklist.

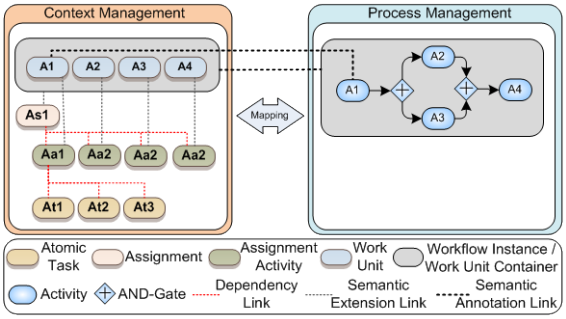


Figure 2: Context and process management.

There are also user and context-related extensions shown in Figure 2: The *Assignment* represents the activity the user performs utilizing the *Work Unit Container* to which it is associated. An example of this distinction would be an *Assignment* called 'Development of new Feature xy' that would use the *Work Unit Container* called 'Standard development workflow'. That way the KPS governs not only how something should be done, but also what will be done. A concrete example of the application of this to an SE workflow will be shown when all concepts are applied in a scenario in Section 4. The same kind of extension is utilized for the activities of a workflow. The *Work Unit*, which maps the activity, is used to create an awareness of workflow execution in the *Context Management* component. It is used only for workflow governance, while the user task associated to that activity is explicitly represented by the *Assignment Activity*.

The final extension to the activities is *Atomic Tasks*. They represent basic low-level tasks like checking-in source code or executing unit tests. To support the automatic detection of the actual execution of these operational tasks, *Atomic Tasks* are associated to a tool, like a version control system, which can be monitored by *Event Management* sensors. That way, a direct connection to the operational actions of the user is established. This is further described in the following section.

3.3 Event Management

The *Event Management* component provides the *Context Management* component with information about the environment including tools, users, or artifacts. All data processed in the *Event Management* component is organized as events. These events are automatically acquired by sensors. These are integrated into various tools used in a SE project like IDEs or bug trackers. That way, they automatically produce events and send them to the KPS. Typical examples of such events are saving a file in an editor, creating an analysis report with a static code analysis tool, or creating a bug in a defect database. Based on these atomic events, the *Event Management* component applies further steps: Events are aggregated to higher-level events like 'User X is applying a Bug Fix on Artifact Y'.

3.4 Information and Knowledge

The management of user-related information is realized using the *Knowledge Provider* and *Knowledge Store* and the two knowledge management GUIs. Information can be collected and stored within the KPS (internal information) and can be integrated from other sources (external information). Internal information is collected via the *Knowledge Collection GUI* that enables users to annotate that information with tags. Examples for tags are 'junior engineer', 'front end development', or 'high risk' that can then be used to automatically select the information to support the users in their context. This is accomplished by the *Knowledge Provider* that also manages the integration of external information. For the organization of information in the KPS, the concept of a *Guidance Item (GI)* is utilized. All guidance the KPS can distribute to users is defined by the GIs created with the *Knowledge Management GUI* and stored within the *Knowledge Provider*. To enable contextually relevant information, the GI has several parameters defined as follows:

Definition 1. A ***guidance item*** is tuple *GI* = (id, type, origin, compilation, tagSet, frequency, rank, active, link, language) where

a) id ∈ Identifiers, Identifiers denotes all valid identifiers over a given alphabet.

b) type ∈ {*Checklist*, *Information*, *Best Practice*, *Notice*, *Tutorial*} denotes the type of the information defined by the GI.

c) origin ∈ {*internal*, *external*} denotes the origin of the information defined by the GI.

d) compilation ∈ {static, dynamic} denotes the way the information defined by the GI is compiled.

e) tagSet is a finite set of tags with tag ∈ STRING used to classify the information defined by the GI.

f) interval ∈ {hour, day, week, month, year, assignment, activity, task} denotes the maximum frequency with which GI is to be shown to a user.

g) rank ∈ {1, …, 10} denotes the quality of GI collected by users to whom it was shown.

h) active ∈ BOOLEAN indicates if GI is to be used by the system (activated).

i) link ∈ STRING ∪ NULL denotes the location of the information defined by the GI if it is static.

j) language ∈ STRING denotes the language of the information defined by the GI.

Each GI has a set of tags used to describe the information for the KPS as well as for the users entering and managing it. Tags can be any type of identifying property indicating to what the GI applies, like 'Activity', 'Junior Engineer', or 'High Risk Artifact'. A GI is an abstract unit of guidance information that may contain an arbitrary number of positions or sub-items. The information defined by it can be static or dynamically compiled by the *Knowledge Provider*. The latter is only possible for internal information. If a GI is internal and dynamic, the *Knowledge Provider* will use the tags of the GI to query the *Knowledge Store* for items that are tagged in the same way, creating the GI out of these. Based on its type, a GI will be treated differently by the KPS. For example, a checklist will have a check mark for each sub-item, while plain information will just be shown to the user. To avoid bothering the users repeatedly with the same information, an interval can be defined for which the GI will be shown maximally once. Finally, to improve the quality of the shown information, users can rate it, which impacts the ranking of the GIs.

3.5 Process-centered Information Support

Automatic knowledge support must be aligned to a person's context; otherwise, it is likely to be irrelevant. Therefore, the activities performed by the users and governed by the *Context Management* component are the initiators for GI provisioning. For these activities, three properties decide how the *Context Management* component presents GIs to the user: These properties are called *GI alignment*, *target obligation*, and *GI usage*. *GI alignment* governs when a GI is shown to the user in relation to the activity that is the target of the GI. There are three options: 'Pre' GIs are shown at the beginning of the activity. 'Post' GIs are shown at the end of an activity. 'Pre/Post' GIs incorporate both of the aforementioned, allowing, e.g., a checklist to be updated with additional items.

Target obligation associates the connection to the target activity. Some GIs like checklists may be directly tied to a target activity. These are called 'Synchronous' and their lifecycle depends directly on the target activity. Other GIs may be shown based on certain events (including, activity termination). These are called 'Asynchronous' and can have a pre-defined lifetime.

GI usage distinguishes between 'Required' and 'Optional' GIs. Using 'Required' GIs, the target activity will not be marked complete without also acknowledging the GI.

Table 1 shows how these three properties can be combined for the introduced activity concepts (*Assignment, Assignment Activity,* and *Atomic Task*).

Table 1: Activity Guidance Item properties.

GI Target	Target Properties						
	GI Alignment			Target Obligation		GI Usage	
	pre	post	pre/ post	sync	async	req	opt
Assignment							
Pre-GI	X				X		X
Post-GI		X		X		X	
Combined-GI			X	X			X
Assignment Activity							
Pre-GI	X			X		X	
Post-GI		X			X		X
Combined-GI			X	X		X	
Atomic Task							
Pre-GI	X				X		X
Post-GI		X			X		X

As the activity concepts have different properties, the types of GI provision for them slightly differ. For example, an *Atomic Task* that is executed by a user can be detected by the KPS, but it is not governed by a workflow. For such a task, GIs cannot be required, as there is no means to prevent the user from simply switching to another task. There is also no event indicating completion of an *Atomic Task* to the KPS. Post-GIs are nevertheless possible, as they can be shown to the user even when he switches to another task. The *Context Management* component is in charge of tailoring the size of the GIs. It can decide how many GIs will be shown to the users at a certain point in the process and how many sub-items are allowed. This

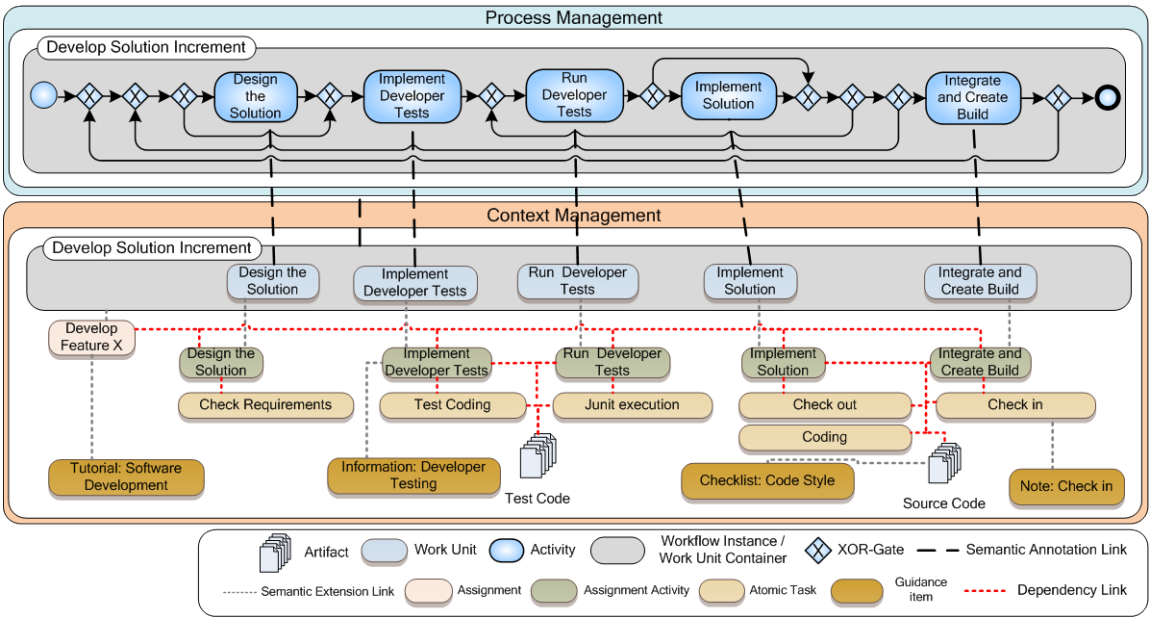


Figure 3: GI Application Example

information is stored as part of the process information in the *Context Management* component and can thus exploit other context information: For example, a development activity conducted in a situation with high schedule pressure could have a checklist with fewer items than it would have if more time were available.

The concrete procedure of GI provisioning is as follows: The *Knowledge Provider* receives an event from the *Context Management* component indicating some point in the process where guidance could be feasible (e.g., user X has started activity Y). If needed, the *Knowledge Provider* requests additional context information from the *Context Management* component (e.g., what artifact the user is currently working on). After that, the *Knowledge Provider* queries its database for GIs with tags matching the context. For external GIs, links are passed back to the *Context Management* component. For internal GIs, the *Knowledge Store* is queried. After having received the GIs, the *Context Management* component sends the appropriate number of GIs to the *Process Support GUI* to be displayed to the user.

4 APPLICATION SCENARIO

This section presents an application scenario that applies the approach to a concrete software development workflow. It is based on the Open UP process, a derivate of the Unified Process (Scott, 2002) created by the Eclipse foundation. The workflow is called 'Develop Solution Increment' and is targeted at supporting the creation of new software. Figure 3 illustrates the implementation of the workflow. It incorporates user-related extensions: the *Assignment Activities* and two exemplary artifacts (source code and test code). The example presents a sample selection of possible guidance for such a workflow.

As mentioned before, the workflow deals with the creation of software functionality. That goal of the workflow is represented by the *Assignment* ('Develop Feature X'). The workflow contains five activities that deal with designing, implementing, integrating, and creating and executing developer tests. Each of these activities is mapped in the *Context Management* component and extended by the user-related *Assignment Activities*. The latter, in turn, are extended by the *Atomic Tasks* representing concrete actions executed using a specific SE tool and manipulating (a) specific artifact(s) (e.g., checking-in source code).

When a user executes this workflow, the *Knowledge Provider* is automatically notified when the execution reaches a point for which a GI is defined. This is done via a GI concept in the *Context Management* component indicating what type of GI could be shown at that point and how it should be shown. A GI could be defined directly for one of the activity concepts or indirectly if a GI is defined for an artifact that is processed by one of them. Based on that information, the *Knowledge Provider* can

search for matching GIs. This is realized by the tags of the GIs, using the information from the *Context Management* component as tags (e.g., that a GI applies to a certain type of activity or artifact). The *Knowledge Provider* can also request additional information to find better matching GIs for the current situation (e.g., the role of the applying person). After having looked up the information, the *Knowledge Provider* sends it to the *Context Management* component that presents it to the user as configured in the GI concept.

The first of four GIs applied to this workflow is a tutorial associated to the *Assignment*, which contains introductory information on the specific properties of software development in that organization. This could be tagged as applicable to junior engineers that recently joined the organization. The second GI is a link to external information from the Open UP website containing a guideline for developer testing. The third GI is associated to source code artifacts and contains a specific checklist on coding style. The last of the four GIs is associated to the *Atomic Task* of checking in. In that case, it is a note from the configuration manager containing important information about the current development branch.

5 TECHNICAL REALIZATION

This section provides details on the technical realization of the approach and its user interface.

5.1 Technologies

The approach enhances CoSEEEK and the realization is shown in Figure 4 and described in the following. Event-based communication was chosen due to the context-aware nature of the framework, which gathers and processes events from its environment and internal components. A loose-coupling of components, which also facilitates the addition or removal of functionality, is achieved using a tuple space (Gelernter, 1985) on top of a native XML database (Meier, 2009).

For the *Event Management* component, event acquisition is achieved via Hackystat and its tool-based sensors (Johnson, 2007). Event aggregation utilizes the complex event processing framework Esper (Luckham, 2001) to detect and process higher-level activity events.

Both the *Knowledge Provider* and the *Context Management* components are realized via semantic web technology. That technology, in particular ontologies, is advantageous (Gasevic et al, 2006): it provides a taxonomy for the modeled entities and their relations, as well as a vocabulary including logical statements about these entities. Well-structured ontologies enable automated consistency checking and enhance interoperability between different applications and agents, supporting knowledge sharing and reuse. Because of its maturity, OWL-DL (McGuinnes and Van Hamelen, 2004) is used as an ontology language. The semantic reasoner Pellet (Sirin et al, 2007) is also used for processing and SPARQL (Prud'hommeaux and Seaborne, 2006) is integrated for querying the ontology. Programmatic ontology access uses the Jena framework (McBride, 2002). Two separate ontologies were used to facilitate scalability and manageability and to enable a separation of concerns: an ontology for knowledge management (*Knowledge Provider*) and an ontology for SE process execution (*Context Management*).

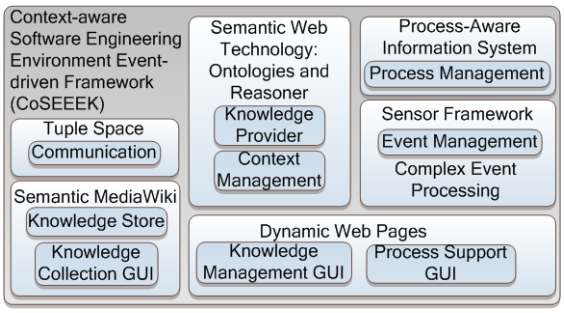
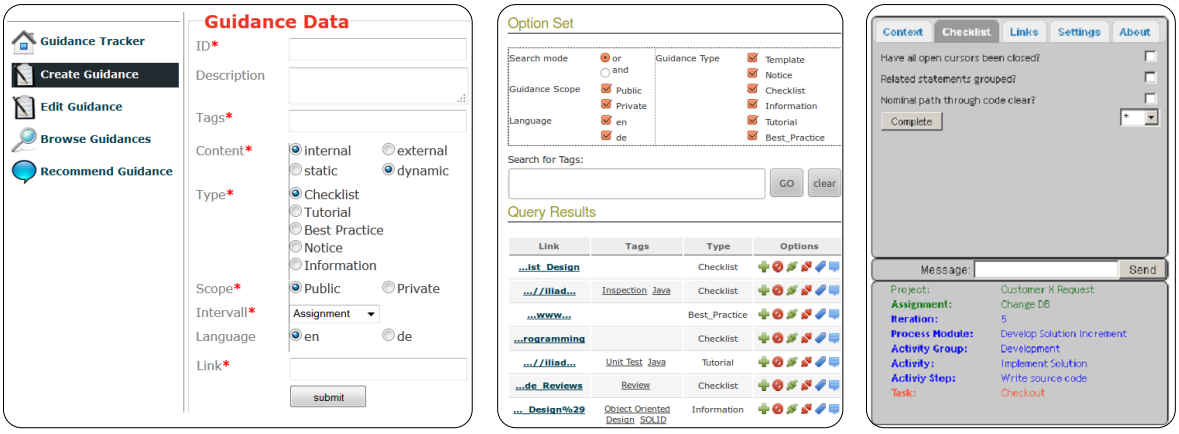


Figure 4: Technologies utilized.

The *Process Management* component is based on the dynamic Process-Aware Information System (PAIS) AristaFlow (Dadam and Reichert, 2009). AristaFlow supports dynamic runtime adaptation of workflows while also providing correctness guarantees. This supports the *Context Management* component's ability to context-sensitively adapt the executing process. The *Knowledge Store* and the *Knowledge Collection GUI* are realized using the Semantic MediaWiki (Krötzsch et al, 2006). The latter provides facilities for easy collection of knowledge from users and for tagging this information with machine-readable semantics. To provide the *Knowledge Provider* with easy access to that information, a SPARQL endpoint was added to the wiki using the SPARQL server Joseki, a part of the Jena framework (McBride, 2002). Finally, via dynamic web pages, the *Knowledge Management GUI* supports GI configuration while the *Process Support* GUI can be accessed in an IDE (like Eclipse or Visual Studio). These two GUIs provide process support for software developers, and are briefly described in the succeeding section.



(a) GI Creation GUI (b) GI Browsing GUI (c) Process Support GUI

Figure 5: GUI Screenshots

5.2 Implementation Prototype

This section briefly illustrates the prototype implemented to evaluate our approach. Therefore, Figure 6 shows the developed GUIs used to interact with the KPS. Since, for our realization, the *Knowledge Collection GUI* is the Semantic MediaWiki (Krötzsch et al, 2006) interface, it is not pictured. The *Knowledge Management GUI* is shown in Figure 5a and 5b; 5a shows on the left side a menu providing access to basic GI functionality (creating, editing, and browsing), and to its right a frame used to create a new GI with all necessary options. There is an option for creating a private GI, enabling users to add personal GIs that are only shown to them. Figure 5b shows the GI browsing functionality, where users can apply various options and tags to locate and modify GIs. A guidance tracker can show users their last GI-related actions and the related actions of other users. Users can recommend their private GIs to other users, e.g., a new best practice. Figure 5c shows the *Process Support GUI*, which is directly integrated into the IDE (both Eclipse and Visual Studio 2010 as a browser-based plugin). Note that due to screen space limitations and partner requirements, a different GUI style evolved. The lower section of that GUI gives comprehensive information about the activity that is currently performed, ranging from the project in which it is executed, over the *Assignment Activity*, to the *Atomic Task* currently being performed. In the upper section, an active GI of the type checklist is shown. It has three items that have to be checked and can be completed using the given 'Complete' button. Additionally, the GI can be rated, influencing its rank for future prioritization and use.

6 RELATED WORK

There are many approaches targeting knowledge management support for SE. (Bjornson and Dingsoyr, 2008) present a literature study about knowledge management in SE. Therein they describe two different kinds of approaches to knowledge management: technocratic schools rely largely on information or management technology, while behavioral schools focus more on organizational or strategic aspects of the implementing company. In the following, approaches that can be categorized belonging to the technocratic schools are discussed.

(Kurniawati and Jeffery, 2004) present a study of the usage of a process-oriented knowledge management tool in a small-to-medium-sized software development company. That tool allows for web-based documentation and support for the SE process model. The study showed that the tool was accepted by users and really supported them. However, compared to our approach the tool has no advanced automation features and lacks dynamic tailored support for users.

(Barros and Werner, 2004) describe an approach to develop, retrieve, and reuse management knowledge and experience concerned with software development risks. The approach incorporates the modeling of so-called risk archetypes and scenarios to model risk impact and resolution strategies and to provide reusable project management knowledge.

The knowledge dust and pearls approach (Basili et al, 2001) aims at facilitating the application of a so-called experience base. The latter shall contain

information about experiences that has been analyzed and organized in packages. The approach combines both short-term and long-term oriented features in knowledge creation and sharing. It shall provide low-barrier access to knowledge and support the initial creation of it.

Outside of the SE domain there have also been various efforts seeking to support knowledge management. (Liao, 2003) presents a study reviewing a multitude of approaches to knowledge management systems. These are classified in different areas: knowledge-based systems, data mining systems, information and communication technology, database technology, modeling, and expert systems providing decision support. One example coming from the data mining area is presented by (Nemati et al, 2002): By the extension of a data warehouse, a so-called knowledge warehouse is built that shall facilitate capturing as well as retrieving and sharing knowledge.

As opposed to the above approaches, our novel approach does not solely focus on the acquisition, storage, and organization of knowledge. In fact, it provides a holistic solution that automates the provisioning aspect in the information lifecycle, strongly focusing on the context-sensitive and process-oriented provisioning of knowledge to the users.

7 CONCLUSIONS

By enabling context-sensitive knowledge provisioning and contextualizing knowledge management, more effective utilization of knowledge in the actual human activities becomes possible. Thus, it is our view that knowledge management and automated knowledge provisioning will play an increasingly important role in automated process-aware information systems. The application of this to the SE domain is especially challenging in this regard due to its intensely knowledge-oriented, collaborative, communicative, and highly dynamic development processes.

We have presented a practical approach for utilizing context management to provide relevant knowledge, at appropriate points for the user in their operational process and abiding by given constraints. In this paper, we have introduced an active knowledge management component (the *Knowledge Provider*) and added an explicit abstract definition for that knowledge to be used as automatic guidance (the *Guidance Item*). By blending process awareness, the CoSEEEK Knowledge Provisioning System is not only aware of the current context, but can also situationally align the knowledge to the past (post-guidance items) and likely future activities (pre-guidance items). Thus, current passive process documentation and guidance can become active operational knowledge that is contextually provisioned. Our approach avoids requiring rigid processes that have inhibited automated workflow guidance for dynamic process areas such as SE.

By enhancing a PAIS with semantic technology, our approach offers greater degrees of flexibility with regard to understanding the contextual implications of the workflow model and managing the process. The technical realization showed the concept's feasibility and practicality. By supporting the users with relatively simple GUIs, the complexity of semantic technology could be hidden and usability increased. Our approach does not require that the entire information contents be semantically annotated; only categorization as metadata via tagging is required. Thus, currently passively available knowledge and guidance forms can be utilized and contextually activated for a user.

Future work includes an empirical investigation in conjunction with industrial project partners in live software projects. Planned features include the integration of tool-generated dynamic knowledge.

To support contextually-sensitive active knowledge management, ongoing challenges remain. Further work and wider adoption is needed towards practical techniques and standards for context modeling and knowledge packaging that allows (process-oriented) knowledge and documentation to be easily generated, appropriately annotated for semantics and contexts, automatically integrated in information systems, and automatically and contextually provisioned in operational situations.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Alexander Grünwald for his help with the technical realization. This work was sponsored by the BMBF (Federal Ministry of Education and Research) of the Federal Republic of Germany under Contract No. 17N4809.

REFERENCES

Basili, V., et al., 2001. An experience management system for a software engineering research organization. Proc. *26th Annual NASA SW Engineering Workshop*, 29-35.

Barros, M.O., Werner, C.M.L., Travassos, G.H., 2004. Supporting risks in software project management. Journal of Systems and Software, 70(1-2), 21-35.

Bjornson, F. O., Dingsoyr, T., 2008. Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. *Information and Software Technology*, 50(11), 1055-1068.

Brooks, F.P.: No Silver Bullet: Essence and Accidents of Software Engineering, *Information Processing*, 1986.

Cugola, G. et al, 1995. How to deal with deviations during process model enactment. Proc. *17th Int'l Conf. on Software engineering*, 265-273.

Dadam, P., Reichert, M., 2009. The ADEPT project: a decade of research and development for robust and flexible process support - challenges and achievements. *Computer Science - Research and Development*, 23(2), 81-97.

Dellen, B., Maurer, F., 1996. Integrating planning and execution in software development processes. Proc. *5th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 170-176.

Gasevic, D., Djuric, D., Devedzic, V.: Model driven architecture and ontology development. Springer-Verlag, 2006.

Gibson, D.L., Goldenson, D.R., Kost, K., 2006. *Performance results of CMMI-based process improvement*. Technical Report, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh.

Gelernter, D., 1985. Generative communication in Linda, *ACM Transactions on Programming Languages and Systems*, 7(1), 80-112.

Grambow, G., Oberhauser, R., 2010. Towards Automated Context-Aware Selection of Software Quality Measures. In Proc. of the *Fifth Intl. Conf. on Software Engineering Advances*, 347-352.

Grambow, G., Oberhauser, R., Reichert, M., 2010a. Semantic Workflow Adaption in Support of Workflow Diversity. In Proc. *4th Int'l Conf. on Advances in Semantic Processing*, 158-165.

Grambow, G., Oberhauser, R., Reichert, M., 2011a. Towards Dynamic Knowledge Support in Software Engineering Processes. In Proceedings of the *6th International Workshop on Applications of Semantic Technologies*.

Grambow, G., Oberhauser, R., Reichert, M., 2011b. Contextual Injection of Quality Measures into Software Engineering Processes. *Int'l Journal on Advances in Software*, 4(1 & 2), 76-99.

Heravizadeh, M., 2009. *Quality-aware business process management*. PhD Thesis, Queensland University of Technology.

Johnson, P.M., 2007. Requirement and Design Trade-offs in Hackystat: An In-Process Software Engineering Measurement and Analysis System. In Proc. of *1st Int. Symposium on Empirical Software Engineering and Measurement*, 81-90.

Jones C., 2010. Get Software Quality Right. *Dr Dobb's Journal*, June 28.

Kess, P., Haapasalo, H., 2002. Knowledge creation through a project review process in software production. *Int'l Journal of Production Economics*, 80(1), 49-55.

Krötzsch, M., Vrandecic, D., Völkel, M., 2006. Semantic mediawiki. Proc. *Int'l Semantic Web Conference*, 935-942.

Kurniawati, F., Jeffery, R., 2004. The long-term effects of an EPG/ER in a small software organisation. Proc. *Australian Software Engineering Conf.*, 128-136.

Lenz, R., Reichert, M., 2007. 'IT support for healthcare processes-premises, challenges, perspectives', *Data & Knowledge Engineering*, 61(1), 39-58.

Liao, S., 2003. Knowledge management technologies and applications--literature review from 1995 to 2002. *Expert systems with applications*, 25, 155-164.

Luckham, D.C., 2001. *The power of events: an introduction to complex event processing in distributed enterprise systems*. Addison-Wesley Longman Publishing Co.

McBride, B., 2002. Jena: a semantic web toolkit, *Internet Computing*, Nov 2002, 55-59.

McGuinness, D.L., Van Harmelen, F., 2004. OWL web ontology language overview. W3C recommendation.

Meier, W., 2009. eXist: An open source native XML database. *Web, Web-Services, and Database Systems*, LNCS, 2593, 169-183.

Müller, D., Herbst, J., Hammori, M., and Reichert, M., 2006. 'IT support for release management processes in the automotive industry'. Proc. *4th Int'l Conf. on Business Process Management*, 368-377.

Nemati, H. R., Steiger, D. M., Iyer, L. S., & Herschel, R. T., 2002. Knowledge warehouse: an architectural integration of knowledge management, decision support, artificial intelligence and data warehousing. *Decision Support Systems*, 33, 143–161.

Oberhauser, R., 2010. Leveraging Semantic Web Computing for Context-Aware Software Engineering Environments. In *Semantic Web*, Gang Wu (Ed.), 157-179.

Prud'hommeaux, E., Seaborne, A., 2006. SPARQL Query Language for RDF, W3C WD 4.

Ramesh, B., Tiwana, A., 1999. Supporting collaborative process knowledge management in new product development teams. *Decision support systems,* 27, 213-235.

Rausch, A., et al., 2005. The V-Modell XT Applied–Model-Driven and Document-Centric Development. Proc. *3rd World Congress for Software Quality*, Vol III, 131-138

Scott, K., 2002. *The unified process explained*. Addison-Wesley Professional.

Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y., 2007. Pellet: A practical OWL-DL Reasoner. *Journal of Web Semantics, 5(2), 51-52.*

Schaffert, S., et al., 2008. Semantic wikis. *IEEE Software*, 25(4), 8-11.

Teigland, R.E., Fey, C., Birkinshaw, C., 1998. Knowledge Dissemination. Global R&D Operations: Case Studies in Three Multinationals in the High Technology Electronics Industry, *MIR: Management International Review*, 49-77.

Yourdon E., 2004. *Death March*, 2nd Ed. Prentice Hall, Upper Saddle River, NJ.