Chapter 8
Context-Aware and Process-Centric Knowledge Provisioning: An Example from the Software Development Domain

Gregor Grambow, Roy Oberhauser and Manfred Reichert

Abstract With the increasing availability of information and knowledge, effective knowledge utilization is becoming a growing and key competency within organizations in various knowledge-intensive fields. One current challenge in process-oriented work, such as that exhibited in new product development projects, is the provisioning of contextually-relevant knowledge to the knowledge workers at the appropriate point in their process. This chapter provides background on technical challenges, referring to the software engineering domain to exemplify these. Thereafter, a practical solution approach based on the Context-aware Software Engineering Environment Event-driven framework (CoSEEEK) is presented. Subsequently, it is shown how automated knowledge provisioning within processes, contextual adaptation of processes, and collaborative process support can be realized.

Keywords Context awareness · Process awareness · Automatic knowledge provisioning · Knowledge management · Semantic processing

8.1 Introduction

In various domains, process-orientation and explicit process management are beneficial [1–3], fostering both project efficiency [4] and product quality [5–7]. However, the quality of process-oriented work in various knowledge-intensive

G. Grambow (✉) · M. Reichert
Institute for Databases and Information Systems, Ulm University, Ulm, Germany
e-mail: gregor.grambow@uni-ulm.de

M. Reichert
e-mail: manfred.reichert@uni-ulm.de

R. Oberhauser
Computer Science Department, Aalen University, Aalen, Germany
e-mail: roy.oberhauser@htw-aalen.de

L. Razmerita et al. (eds.), *Innovations in Knowledge Management*,
Intelligent Systems Reference Library 95, DOI 10.1007/978-3-662-47827-1_8

domains depends on the proper utilization of available knowledge[1] by knowledge workers [8–10]. Respective domains include healthcare, software, and automotive; especially new product development is a knowledge-intensive task [11, 12]. From a knowledge perspective, organizations develop their own local organization-specific knowledge systems [13]. In turn, these may overlap with other knowledge systems (e.g., discipline-specific, product-specific, market-specific, etc.). To a limited degree, such human-based knowledge systems may be represented within IT-based knowledge management systems (KMS) [14].

Drucker [15] has argued that knowledge-worker productivity will be the biggest managerial challenge of the 21st century. When considering current IT-based KMS solutions, knowledge utilization and effectiveness remains an issue [16]. While a KMS can store and retrieve knowledge, it does not really solve the real problem: providing the required knowledge to the right person at the right time for dealing with the right situation. For instance, retrieval and dissemination of the stored knowledge can become problematic when knowledge is highly dependent on the process and context of the participating persons. Typically, knowledge workers are responsible and tasked to retrieve and utilize knowledge on their own (active, free-access retrieval). However, this can be problematic and inefficient in certain situations. For example, not all workers may be aware of the knowledge they should attempt to retrieve (e.g., new knowledge or changes to the knowledge store) at different points in time or while working with new processes. Additionally, humans are prone to forgetfulness, especially in stressful situations, and therefore, even manual retrieval can become problematic.

Thus, the automatic contextual filtering and provisioning of structured knowledge, as well as the automated realignment of processes to changing knowledge, will become increasingly important KMS capabilities, especially in light of the increasing proliferation of information and knowledge. In order to cope with these issues, systems must be aware of context, processes, and knowledge to have the following capabilities:

- Provision knowledge to workers that is aligned with the task at hand.
  Knowledge is typically relevant only to specific situations. Knowledge redundancy (e.g., providing knowledge the human is already well aware of) or overload (e.g., too much knowledge at once) may be detrimental, in that the KMS may be ignored or rejected.
- Adapt users' processes to knowledge and context changes.
  Processes in knowledge-intensive fields may need to adapt the sequencing of activities to align themselves to the knowledge or contextual situation.
- Use knowledge to support collaborative processes.
  This includes automatically inferring impacts of any process activity and notifying or including appropriate collaborators in the processes.

[1]Since knowledge can be transformed into information when articulated, and information can be turned by a mind into knowledge, this chapter uses these terms interchangeably.

This chapter provides insights into how an automated information system can support the above capabilities. In particular, it addresses the following questions:

- How should information be stored to enable automatic information processing and dissemination?
- How can information be automatically distributed to those need it?
- How can the relevant information be injected at the right point into the users' operational process?
- How can a process be automatically realigned based on changes to knowledge?
- How can collaborative work be supported with knowledge?

Our knowledge management approach is illustrated with examples from the software development domain. Within the field of software engineering (SE), software development projects are collaborative, knowledge-intensive, and process-centric [17]. They exhibit the aforementioned issues and represent a knowledge management (KM) environment in which the three capabilities enumerated above can be exemplified. Developers and testers may participate collaboratively in multiple projects dealing with different products simultaneously and on teams that may be globally distributed. Due to resource and schedule constraints, developers should be able to enter and leave projects quickly and efficiently, which can be daunting considering that complex tasks require specific knowledge. Processes that should govern such tasks are usually manually implemented without automated guidance—presenting a further challenge for process-awareness, and these knowledge-intensive processes need to adapt to the dynamic knowledge situation. With regard to context, since the involved artifacts, tool chain, and actors are solution-oriented, the environment can be heterogeneous with dynamic contexts playing a significant role. Effective KM remains a crucial factor for successful software projects [17]. This chapter gives a comprehensive overview about the different knowledge management capabilities of our approach and system. Further reading to the discussed features can be found in our prior publications and the upcoming doctoral thesis of Grambow [18–25].

This chapter is organized as follows: the next section provides an overview of current approaches. Section 8.3 describes issues in knowledge-intensive projects, including problems and general requirements. Section 8.4 presents a solution approach, including a concept and an implementation framework for the SE domain. Then, Sect. 8.5 illustrates automated knowledge provisioning within processes, while Sect. 8.6 focuses on the knowledge-based contextual adaptation of processes, and Sect. 8.7 shows how knowledge-based collaborative processes are supported. Finally, Sect. 8.8 summarizes the chapter and designates future challenges. A glossary and references are provided at the end followed by a section with additional resources for the reader.

8.2 Overview of Approaches in the Software Engineering Domain

This section discusses various approaches, focusing on the example domain of SE. KM in complex and knowledge-intensive projects requires more than only storing and retrieving knowledge. A tool or system that aims to comprehensively support knowledge workers must provide holistic support for the entire project and for the collaborating knowledge workers. Therefore, approaches beyond the classical KM category are discussed that provide project and collaboration support for SE knowledge workers. Another factor especially important in SE is knowledge about the produced product and its quality. Therefore, approaches supporting software quality management (QM) are mentioned.

SE Knowledge Management Bjørnson and Dingsøyr [26] provides a systematic review of studies on the application of KM in SE, categorizing the studies according to the various KM schools: systems, cartographic, engineering, commercial, organizational, spatial, and strategic. Kurniawati and Jeffery [27] presents a study about the usage of a process-oriented KM tool in a small-to-medium-sized software development company. In particular, this tool allows for web-based documentation and support for the SE process model. The study showed good acceptance of the tool and that it really does support the developers. The approach presented in [28] focuses on KM, considering various risks in SE projects. The approach incorporates the modeling of risk archetypes and scenarios to model risk impact and resolution strategies as well as to provide reusable project management knowledge. Basili et al. [29] presents the knowledge dust and pearls approach, which aims to facilitate the application of an experience base containing information that has been analyzed and organized into experience packages. Looking beyond the SE domain, [30] presents a study of various KMS classified in different areas: knowledge-based systems, data mining systems, information and communication technology, database technology, modeling, and expert systems providing decision support. The presented approaches narrowly focus on management, storage and retrieval of information.

SE Quality Management Support The quality of the produced product and related knowledge involved are crucial success factors for a project. In order to be able to provide automated support for QM, continuous awareness about the quality state is crucial. Source code metrics are one means in SE of assessing quality. In [31], a report is provided about the application of such a metric program at Motorola. It describes a set of different views on metrics to support their successful application and reports success in several areas by using software metrics. Offen and Jeffery [32] describes a formal meta-model enabling measurement in SE. It puts strong focus on storing, interpreting and analyzing gathered data. Further, a practical framework is also developed supporting the creation of models for software measurement, connection of these to measurement tools, and storage of the results.

A comprehensive industry survey about the success of metric programs is presented in [33].

However, these approaches only deal only with the use of metrics, but not with tool-supported automated QM quality management. In the following, therefore, a selection of approaches concerning automated measurement tools is discussed. PR-Miner [34] enables automated analysis of source code and efficient and automated extraction of implicit, undocumented programming rules from it. Further, it automatically detects violations to these rules. Another tool is the Empirical Project Monitor (EPM) [35], which aims to support effective software process management by providing quantitative data. It collects and measures data from different repositories within software development support systems and presents that data graphically to the users in order to generate an awareness of the project progress. The collection and aggregation of data about users' programming behavior is offered by the modular framework ElectroCodeoGram (ECG) [36]. It comprises a set of sensors as well as modules for integrating the data gathered by the sensors. ElectroCodeoGram provides micro-process data to support researchers in understanding how programming is carried out on a fine-grained level. A similar approach shown in [37] is called SUMS (Standard User Monitoring Suite). SUMS features acquisition facilities for different programming languages, applying neural networks and Bayesian analysis to achieve automated learning features. While the mentioned tools offer advanced data acquisition, aggregation, and interpretation facilities for different kinds of data in SE projects, they address a relatively narrow quality area.

SE Collaboration Support Knowledge-intensive projects typically require communication and collaboration among knowledge workers in order to work on complex tasks. Existing approaches support such collaboration with related knowledge. For example, CASDE [38] and CoolDev [39] make use of activity theory. CASDE supports mutual awareness between different actors and their activities via a role-based awareness module. In turn, CoolDev manages activities performed by a single person in the context of global cooperative activities. It is realized as an integrated development environment (IDE) plugin capable of monitoring activities carried out with other plugins. Another approach is taken by CAISE [40], a framework that enables the integration of other SE tools and supports the development of new SE tools based on collaboration patterns. Other frameworks like Syde [41], SPACE [42], and ADAMS [43], take an artifact-centric approach. Syde is based on an extended view on source control management. It can automatically inform every developer about any changes another developer makes, even if the changes have not yet been synchronized to the common code repository. It enables synchronous development. SPACE (Semantic Process- and Artifact-oriented Collaboration Environment) takes another approach by managing two types of interconnected models for processes and artifacts. That way it enables a set of supportive features, e.g., personalized user views or comprehensive artifact traceability. ADAMS (ADvanced Artefact Management System) is even more

artifact-centric: it models the whole project in terms of its artifacts. Thus it features sophisticated versioning and locking approaches, fine-grained traceability of the artifacts, and an event module capable of informing users about any relevant event. The above mentioned tools focus on the collaboration perspective of humans and activities and neglect other aspects of comprehensive KM.

SE Project Support Numerous approaches exist that aim at providing some kind of SE project support based on knowledge. Respective approaches mostly target a distinct area. For example, [44] describes knowledge support approaches during process execution, consisting of the domain-oriented software development environments (DOSDE) as well as the enterprise-oriented software development environments (EOSDE). Another category of approaches for SE project support puts it focus on a model-driven approach. Representatives of this category include the Transforms Environment [45] and the model-driven approach described in [46]. Being situated on the M2 level of the OMG model layers, the Transforms Environment uses parts of the SPEM process meta-model and tailors it for MDA processes. The model-driven approach suggested in [46] also applies a model-driven procedure, in this case in order to support deployment and variability of software processes. While all these approaches provide a certain amount of project and knowledge assistance, they lack a comprehensive approach to optimally support project participants in their context, knowledge, and with their workflows.

8.3 Current Issues

This section describes problem areas and requirements that a solution must address for knowledge provisioning in process-oriented and knowledge-intensive projects.

8.3.1 Problem Areas

Concerning holistic knowledge support for contemporary projects in various domains, the problem areas can be classified in two categories: direct knowledge-related problems about processed artifacts, human collaboration, or used tools; and process-oriented problems. Because these areas are intertwined, if these two problem areas are not addressed properly, effective IT-based KM is impeded. Figure 8.1 illustrates these problems in the context of the SE domain. The project is separated into three process domains illustrating the relation of concrete KM problems and related process problems. Domains such as these have been mentioned several times in the literature (e.g., [47] or [48]). Both of these are conceptually analogous, and we herewith use Dowson's. It distinguishes three domains.

- *Process modeling*: processes are modeled and process models, including actors, tools, or artifacts, are situated.

- *Process enactment*: the modeled processes are implemented by means of workflow management technology [49].
- *Process performance*: the real-world-process takes place, including humans, the concrete artifacts, or concrete software tools used by the humans.

Knowledge Management (Fig. 8.1 (1)). The first problem area concerns classical KM. This comprises, for example, knowledge about the correct use of tools and technologies in an organization, its organizational structure, or other concrete approaches like how to apply source control management for the artifacts produced.

Quality Management (Fig. 8.1 (2)). The second area deals with the assets produced in the organization: artifacts in the SE domain. In particular, knowledge about the quality of these artifacts, occurring problems, reported bugs, or approaches to bug fixing are of primary importance. Organizations are often not aware about the state of their products' artifacts. Problems often remain undiscovered and reveal themselves either near the end of a project or later during use by the customer. Proactive QM is often not implemented. When software quality measures are applied under high time pressure, they often disrupt the development process or do not match the applying person's situation or abilities, and are thus less effective and efficient.

Information Coordination (Fig. 8.1 (3)). Knowledge workers collaborate, and thus efficient and effective coordination is crucial. For knowledge workers, information about the tasks and artifacts processed by co-workers is vital.

As mentioned, these problem areas directly relate to various kinds of knowledge and are not the only ones for contemporary projects. The problem areas are all situated in the *process performance domain*, where users interact with real tools and with each other. Because such projects are often complex, their processes have to be planned, modeled, and explicitly managed. In particular, their implementation and use is crucial for effective KM. Therefore, in the following, three more generic problem areas relating to processes are discussed (as also illustrated in Fig. 8.1).

Process Automation (Fig. 8.1 (A)). Processes are modeled in the *process modeling domain* using specific process modeling tools and notations. In many organizations, explicit support for processes remains at that modeling domain level. Process implementation is considered as the activity of releasing a process model document to all process participants. When no PAIS are in place to govern or support the actual *process enactment*, the real-world process can often and easily deviate from the modeled process as it is executed in the process performance domain.

Context Integration (Fig. 8.1 (B)). While Process-Aware Information Systems (PAIS) can provide organizations with IT-based process support and help govern process execution, only a limited amount of the work actually done in knowledge-intensive projects is even captured in process models. The PAIS are often unaware about the tools used, the variety of (partly unexpected) events that happen in everyday work, or the great number of potentially interrelated artifacts.

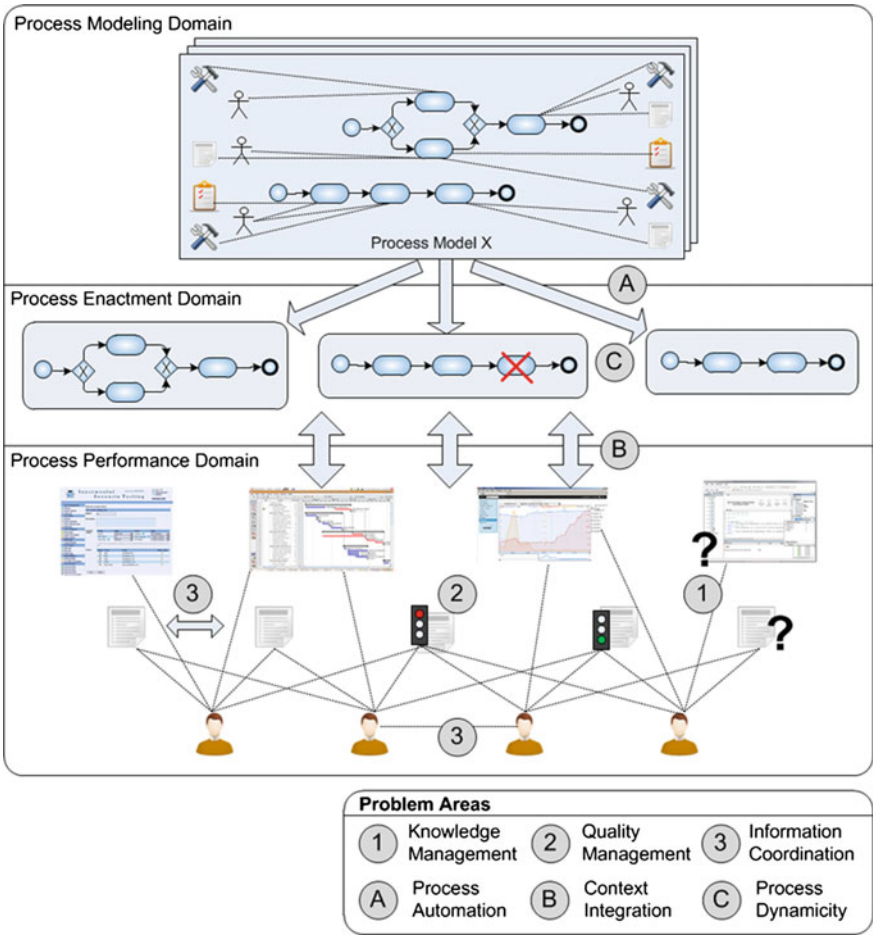

Fig. 8.1 Problem areas mapped to process domains

Thus the process, as it is really executed, differs from the one executed in a PAIS, and the latter becomes (at least partly) irrelevant.

Process Dynamicity (Fig. 8.1 (C)). Another problem with process implementation relates to the dynamicity of the executed process. If an organization has a system in place that governs and supports the process, the support provided by that tool can be beneficial in keeping the real world process aligned. But this mostly only applies as long as nothing requires a change in the operationally running process [50]. For example in SE, this can be a received bug report from an important customer that requires one or more developers to deviate from their standard development schedule.

8.3.2 Basic Requirements

For a system to cope with the above problem areas, it must fulfill certain requirements. These requirements are organized around the basic problem areas (RA relates to a requirement concerning problem area A from Fig. 8.1). The more advanced problems will be covered in dedicated sections: KM will be covered in Sect. 8.5, QM in Sect. 8.6, and information coordination in Sect. 8.7. Please note that fundamental system abilities such as distributing tasks to its users or correctness of process execution are presumed. Although the requirements are tailored toward the SE example domain to make them concrete, they can easily be adapted for other domains.

A system aiming for holistic process and knowledge support should incorporate the following features:

- *Additional Process Information* (RA.1): incorporate various types of supplementary information contained in process models (e.g., artifact hierarchies or supportive information like checklists). These should be integrated into the execution semantics of the executing PAIS to facilitate consistency between modeled and enacted processes;
- *Abstract and Operational Processes* (RA.2): model abstract processes (like the lifecycle of a whole project) and also operational concrete processes (like concrete development tasks). Both types of process areas (abstract and concrete) should be seamlessly integrated;
- *Seamless Integration* (RA.3): integrate seamlessly into everyday work. Usage should not be cumbersome and specific process or knowledge support should not distract users from their work;
- *Context-data Acquisition* (RB.1): automatically acquire context data from its environment, classifying the current situation;
- *Context-data Processing* (RB.2): automatically process acquired context data to react to changing contextual conditions;
- *Context /Process Integration* (RB.3): integrate acquired context data with its process model and the associated data to be able to align the enacted process with the actually performed process;
- *Dynamic Workflow Changes* (RC.1): enable changes to running process instances; and
- *Automated Workflow Changes* (RC.2): automate instance changes of running workflows to be able to autonomically react to changing situations.

8.4 Automated Knowledge Provisioning Approach

This section gives details on the basic solution approach comprising the abstract concepts as well as the implementation architecture of a system that enables comprehensive and holistic knowledge support for contemporary projects.

8.4.1 Abstract Knowledge Provisioning Concept

This section gives insights on the basic principles of the system we have developed that amalgamates a knowledge-based system (KBS) with an adaptable process-aware information system (PAIS) and a contextually-aware system. Figure 8.2 shows the major components in this concept.

Contemporary PAIS only offer a limited number of concepts like activities, workflows, data elements, users, and roles. To be able to execute processes in line with the actual project work, a system should have additional modeling capabilities. Our concept enables the integration of various interconnected entities that enable the explicit modeling of complex artifact hierarchies with diverse properties for each artifact (*Context Management* and *Knowledge Management* in Fig. 8.2). Further, it enables the relation of such artifacts to a similarly complex and flexible hierarchy of interconnected activities of different types (*Adaptable Process Management* in Fig. 8.2). Besides these, various other concepts are also implemented to enable a comprehensive modeling of complete process models for execution [20, 25].

Another limitation of contemporary PAIS is the fact that they mostly apply rigid and pre-defined workflows. In our opinion, rigid workflows applied in automated systems are an important cause for their dissonance in practice. Therefore, our concept not only comprises facilities to provide dynamic adaptation of running workflows for users (*Adaptable Process Management* in Fig. 8.2), but also to let the system perform automated process adaptations in alignment with context data representing the current project situation (*Context Management* in Fig. 8.2).

Context data is also crucial for a system that seeks to provide holistic project and process support. Therefore, our system integrates facilities to automatically gather context data from various sources (*Environment Sensors* and *Event Extraction* in Fig. 8.2). Further, aggregation and processing of the data is automated (*Event Processing* in Fig. 8.2), i.e., data can be delivered to the components that use it in a reasonable granularity and with more semantic value.

Providing automation in knowledge-intensive projects is challenging. A system aiming at comprehensive project support must be able to automate a large number of different types of tasks while still being flexible and transparent to the user. To enable this, our system combines different technologies for supporting different tasks: semantic web technology enables automatic classification capabilities, rule engine technology automates simple recurring tasks (*Rule Processing* in Fig. 8.2), and an agent system adds more autonomic capabilities (*Agent System* in Fig. 8.2).

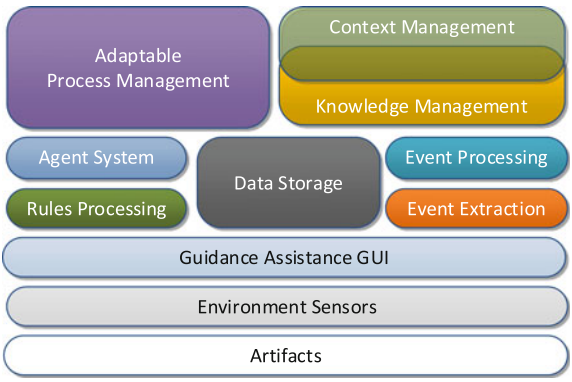

Fig. 8.2 Automated knowledge provisioning conceptual architecture (domain independent)

In order to enable a system to provide knowledge assistance in a holistic and automated way for entire projects, a more comprehensive approach to KM must be taken into account. Our system comprises an active KM component managing the user relevant knowledge in alignment with context data (*Knowledge Management* in Fig. 8.2). Furthermore, it not only stores and manages that knowledge, but also explicitly manages internal knowledge that enables the system to react to various situations in a project in an appropriate way (*Data Storage* in Fig. 8.2).

Finally, system providing comprehensive project support and tackling different areas necessarily implies a certain amount of complexity. Such a system involves a fair number of different components and modules and has to process various kinds of dynamic data. Enabling efficient communication of the different components with various kinds of data while preserving extensibility can be a serious issue. Therefore, all framework communication is event-based and loosely-coupled in order to be able to easily integrate new components as well as new kinds of data.

8.4.2 Knowledge Provisioning Framework

The concept above was then implemented for the SE domain and named CoSEEEK (Context-aware Software Engineering Environment Event-driven frameworK). It unites adaptive process management with semantic web technology and a sensor framework to provide holistic support for SE projects. Users can store and annotate knowledge in a semantic wiki and thus make it machine-accessible and -readable. To be able to not only transfer this knowledge automatically back to the users, but also to maximize the suitability and effect of that knowledge, CoSEEEK tailors it to the current situation of each and every individual participating in the project. This becomes possible on one hand by guiding the users with dynamic workflows; on the other by having a multitude of active sensors in various SE tools connected to the framework. These sensors provide accurate information on the various artifacts users manipulate in a project and also on tasks they execute even if they are external

to their planned workflows. This enables CoSEEEK to match meta information in the knowledge base to various properties of the situations the users are in, and automatically inject the knowledge into the users workflows. That way, users can be provided automatically tailored knowledge that matches their current needs. Figure 8.3 details the technical architecture of CoSEEEK followed by an explanation of the different components and their interaction.

The different parts of the concept previously discussed are realized by the different components shown in Fig. 8.3. To enable communication between the different components that facilitates extensibility and exchangeability, all communication is event-based using a *Data Storage* component for event storage. The integration of CoSEEEK with its environment is realized via an *Event Extraction* and an *Event Processing* component that enable the automatic acquisition and processing of events from other SE tools using sensors. Context data is then centrally managed by a *Context Management* component. To integrate the data with process execution and extend this with additional knowledge, the *Context Management* component is tightly integrated with a *Process Management* component that is in charge of workflow execution. The latter component also manages dynamic adaptations to workflows to conform to changing situations. To enable comprehensive knowledge support for entire projects, a separate component centrally manages knowledge. That *Knowledge Management* component is also tightly integrated with the *Context Management* component to facilitate context-based knowledge provisioning. Finally, an agent system and a rule engine offer tight integration of configurable automatisms into the framework to support users in their complex tasks.

In the following, the technical realization of the different components is briefly discussed. The event-based communication and storage within the framework is implemented via a specialized tuple space [51] that uses the XML database eXist [52]. Each module and the applied sensors can write in that tuple space and register to be automatically notified about events relating to a specific topic. The sensors are realized via the Hackystat framework [53], which offers a rich set of sensors that can be integrated into various applications like source control management systems or IDEs [cf. requirement RB.1 (Context-data Acquisition)]. The sensors automatically create events for various real events like the change of an artifact. Such events can be of rather atomic nature and with low semantic value. Therefore, to produce events with more semantic value and not burden the event system with numerous micro events, the complex event processing (CEP) [54] tool Esper [55] is applied to create higher-level events out of various low-level events [cf. requirement RB.2 (Context-data Processing)].

To enable CoSEEEK to apply various kinds of automatisms and act autonomously in various situations, the multi-agent system JADE [56] and the rule engine JBoss Drools [57] are integrated. An example for such automatisms is automatically determining an appropriate software quality measure to apply to counteract a detected quality problem in the source code, and then automatically assigning the measure to the appropriate user based on various factors. This will be further described in Sect. 8.6.

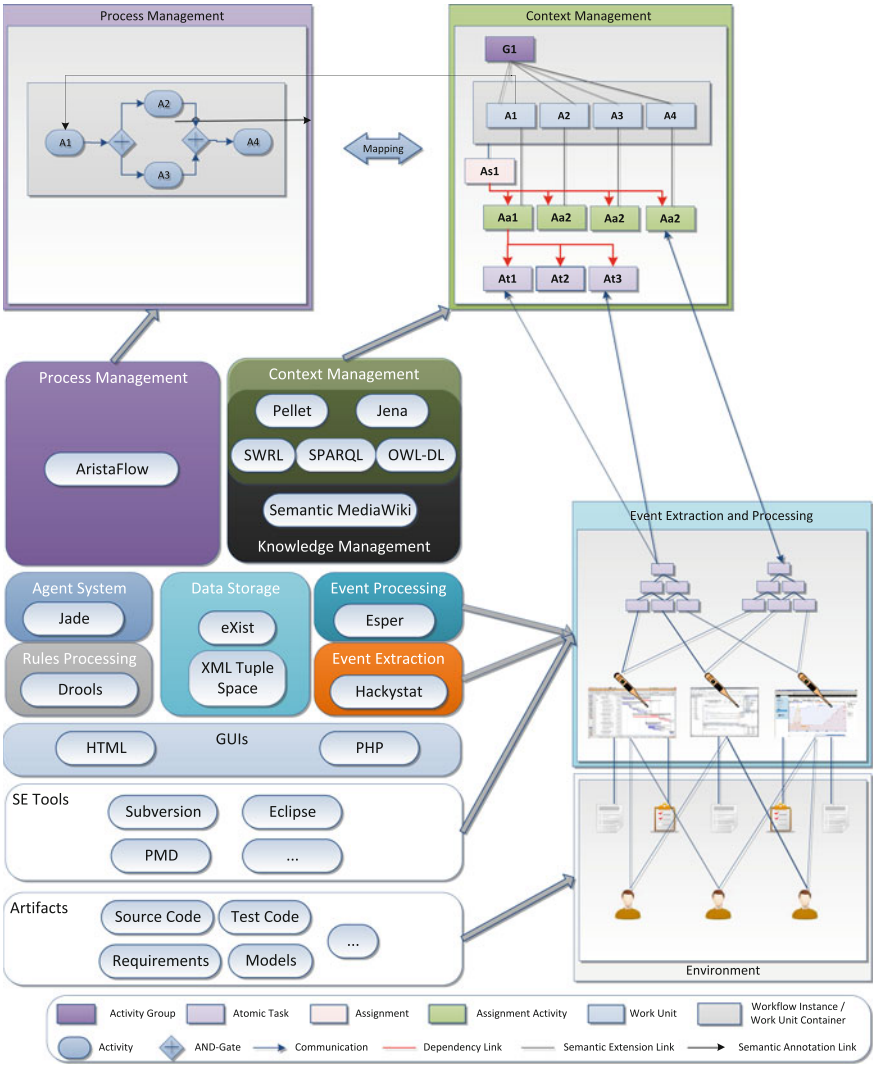

Fig. 8.3 CoSEEEK framework

For management of the workflows in CoSEEEK, the AristaFlow [58, 59] PAIS is integrated. It offers numerous advantages for the correct and dynamic enactment of workflows, featuring a correctness-by-construction principle that only allows the user to create correct workflows. This correctness is continuously enforced during the entire execution lifecycle. In addition, it enables dynamic changes even to running workflow instances [cf. requirement RC.1 (Dynamic Workflow Changes)] and guarantees the correctness of the workflows before and after the adaptations.

The *Context Management* as well as *Knowledge Management* components rely on semantic web technology. For user-related knowledge, the *Knowledge Management* component integrates the Semantic MediaWiki [60]. That way, the users can enter knowledge like in a common wiki, but can also semantically tag their entries, enabling automated usage of that knowledge by CoSEEEK. This will be further detailed in Sect. 8.5. Internal knowledge that the system utilizes with both components is stored within an OWL-DL ontology [61]. To exploit the full potential of the semantic web technology, the reasoner Pellet [62] is used together with the Jena framework [63] for programmatic access to the concepts. In addition to that, rules can be applied via SWRL [64] within the ontology, and queries can be posed via SPARQL [65].

The ontology is not only used to model contextual data, it is also tightly coupled with the *Process Management* component in order to realize useful extensions to the workflows and model complete process models [cf. requirement RA.1 (Additional Process Information)]. That way, it is also possible to enrich operational workflows with various granularities of activities and additional user-related information. It abstracts from the internal workflow logic (cf. [18]) to make workflow use less cumbersome for humans [cf. requirement RA.3 (Seamless Integration)], while still being able to automatically govern the abstract processes to which the operational workflows belong [cf. requirement RA.2 (Abstract and Operational Processes)]. Furthermore, by the close integration of process-related information in the ontology with the contextual data, a seamless integration of both can be applied [cf. requirement RB.3 (Context/Process Integration)]. This tight integration of the Context Management and Process Management components makes it possible to automatically utilize context data to apply automated adaptations for aligning the process with reality [cf. requirement RC.2 (Automated Workflow Changes)].

The environment of CoSEEEK, which primarily consists of artifacts, humans, and tools within a project, is integrated in two ways: the entities are modeled in the *Context Management* component and, via sensors, their state can be kept up to date with the real world entities. For providing the supporting and governing functionalities, CoSEEEK offers a set of simple web-based GUIs. To enable seamless integration into everyday work [cf. requirement RA.3 (Seamless Integration)] and not disturb the software developers, the main GUI was also realized as a plugin for common software IDEs like Microsoft Visual Studio and Eclipse.

8.5 Automated Knowledge Provisioning in Processes

As stated, knowledge worker projects as well as the knowledge management can be challenging. In particular, this applies to SE as it involves new product development, which is a knowledge-intensive task [11]. Further, software processes can be mostly considered as knowledge processes [66]. It has been shown that an automated system supporting KM can be beneficial [67]. In SE projects, nowadays,

wikis are often used for such tasks as they enable distributed access to knowledge. However, the retrieval of respective knowledge is often problematic as the knowledge organization in a wiki used by dozens or even hundreds of people can be challenging [68]. For example, if one developer encounters a best practice for a recurring situation, e.g., the application of a design, he might enter it in such a wiki. The retrieval of that information is problematic. On the one hand, the information is only passively stored and another developer might not even be aware of its existence when encountering a problem. On the other, even when using the wiki, the information might not be found because one might search quite differently than the one who stored the information had in mind. This section gives insights on the knowledge provisioning concept we have created. For further reading on that topic see [21, 22]. Section 8.5.1 discusses specific requirements, while Sect. 8.5.2 shows the different components involved. Section 8.5.3 discusses the specific concepts, and the last sub-section gives a concrete example for automatic knowledge provisioning.

8.5.1 Knowledge Provisioning Requirements

To overcome the aforementioned problems, a system aiming for holistic process and knowledge provisioning should incorporate the following features:

- *Knowledge storage* (R1.1): store user-relevant knowledge in an appropriate way;
- *External knowledge integration* (R1.2): integrate knowledge from external sources;
- *Automatic knowledge access* (R1.3): automatically access, use, and distribute knowledge stored in the system;
- *Context-data utilization* (R1.4): utilize contextual information to select appropriate knowledge for different situations and persons;
- *Knowledge injection* (R1.5): automatically inject knowledge into process enactment and performance; and
- *Knowledge provisioning configuration* (R1.6): enable users to configure knowledge provision.

8.5.2 Knowledge Provisioning Components

To meet the above requirements, we developed a system that comprises tightly integrated active components relating to process, context, event, and knowledge management. These components and their interaction are illustrated in Fig. 8.4.

Recalling the requirements, effective knowledge management and provisioning necessitates that information suitable to the user's situation be seamlessly integrated

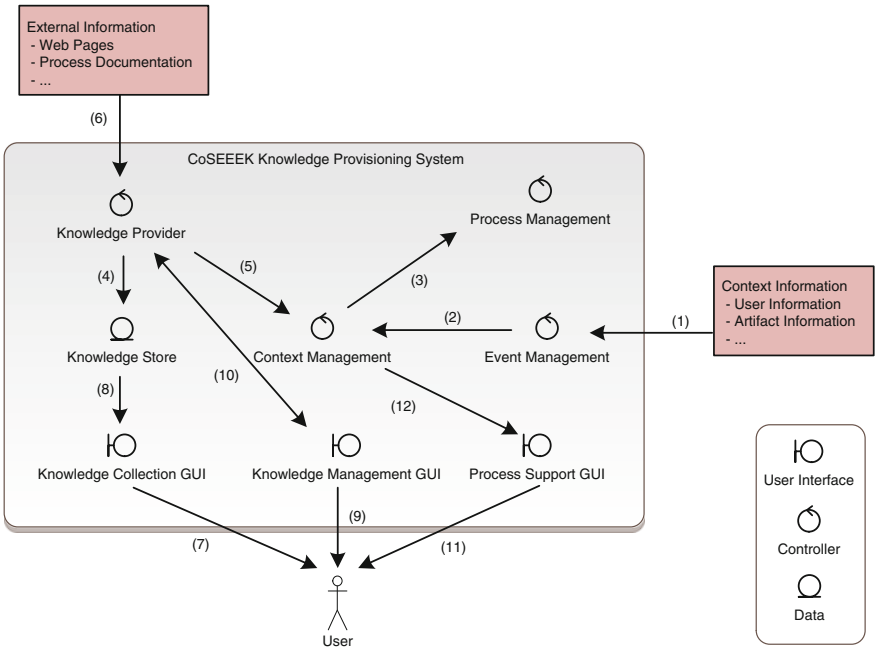

Fig. 8.4 Knowledge management components (using symbols from robustness diagrams)

into his or her current process. This is achieved by the integration of multiple components as described in the following. The *Context Management* component, a central component of the system, stores information about users, artifacts, tools, and various other project entities. The *Event Management* component, in turn, automatically collects information from the environment by the aforementioned sensors (1) and delivers it to the *Context Management* component (2) [cf. requirement R1.4 (Context-data utilization)]. The *Process* and *Context Management* components are tightly integrated and together realize the enactment of entire process models. The *Knowledge Provider* that is in charge of managing the provision of knowledge to users directly communicates with the *Context Management* component (5), and thus has direct access to context information [cf. requirement R1.4 (Context-data utilization)] and to process information [cf. requirement R1.5 (Knowledge injection)].

As also mentioned in the requirements, automatic knowledge provisioning relies on effective acquisition and storage of the knowledge and the ability of the provisioning system to access and utilize that knowledge. The storage is realized by a separate component called the *Knowledge Store* [cf. requirement R1.1 (Knowledge Storage)]. The latter allows the *Knowledge Provider* semantic access (4) [cf. requirement R1.3 (Automatic knowledge access)] to the stored knowledge that is obtained from a special *Knowledge Collection GUI* (8) that allows users to enter and tag their knowledge (7).

However, even if a system contains useful knowledge for users, it would still be marginalized by users if it is unable to deliver it in a way fitting to their current tasks and workflows. Therefore, the knowledge chosen by the system is passed from the *Knowledge Provider* to the *Context Management* component (5). That component, in turn, utilizes its tight connection to the *Process Management* component (3) to determine the time point to inject the knowledge in the process [cf. requirement R1.5 (Knowledge Injection)] and then deliver that knowledge to the *Process Support GUI* (12) that makes it visible to the user (11).

Finally, even if a knowledge provisioning system is effective, it will never comprise all possible matching knowledge. Therefore, the integration of external knowledge sources is managed by the *Knowledge Provider* (6), so that these can be easily provided to users [cf. requirement R1.2 (External knowledge integration)]. The configuration of external knowledge and the entire knowledge provisioning process can be managed by users by utilizing the *Knowledge Management GUI* (9) [cf. requirement R1.6 (Knowledge provision configuration)], which communicates with the *Knowledge Provider* (10).

8.5.3 Knowledge Provisioning Process

This section discusses how knowledge is managed within the system. To be able to explicitly reference and provide each unit of information, a separate concept has been introduced in the ontology that is called a *Guidance Item* (GI). It is used by the *Knowledge Provider* to access and classify the knowledge integrated into the system. The GI has a set of properties enabling information management. The relevant ones are shown in Table 8.1.

The properties of the GI comprise information about the knowledge represented by the GI as well as information relevant to contextual knowledge provisioning. However, the knowledge must be injected into the user's process in a defined way to make it effective. This is governed by four distinct properties, managing when

Table 8.1 GI properties

Type	Knowledge can occur in various types that are distinguished by this property, like checklist, information, best practice, notice, or tutorial
Origin	This property denotes if the GI is stored within the system or coming from an external source
Compilation	This property denotes if the GI is static or if the system dynamically compiles it. In the latter case, the system matches entered tags users add to the knowledge in the Knowledge Store to process and context information and thus creates specifically tailored knowledge support for the users' situation
Tags	This property contains tags used to dynamically compile knowledge for users with dynamic GIs
Link	This property stores a direct link to the knowledge represented by this GI if the GI is static

Table 8.2 Knowledge injection properties

GI alignment	This property governs how the knowledge is shown to the user in relation to the activity it relates to. 'Pre' means that the GI is shown at the beginning of an activity and 'Post' means it will be shown at the end of an activity
GI alignment	This property indicates if the lifecycle of the GI is tied to the lifecycle of the relating activity. If so, the GI will only be available as long as the activity is active
GI usage	This property distinguishes between the values 'Required' and 'Optional'. Required GIs must be reviewed by the user and can even block activity termination if they are tied to an activity
Item Compilation	This property relates to the GI's 'Compilation' property and manages how the system uses runtime context information to dynamically compile GIs matching the current situation. One example would be a database development checklist for junior engineers

and how to apply knowledge support to different kinds of activities as shown in Table 8.2.

Not all combinations of these properties are allowed, for more information see [22].

8.5.4 *Knowledge Provisioning Example*

Recalling the introductory example from this section, this subsection gives a brief concrete example for our knowledge provisioning concept that is illustrated by the following figure and explained afterwards.

During the course of a project, different steps are performed to enable automated knowledge provisioning as illustrated in Fig. 8.5.

1. Utilizing the Knowledge Collection GUI, users can collect knowledge while working in a project. They can tag this knowledge to support later discovery by humans or any automated system. Examples of tags on that information include

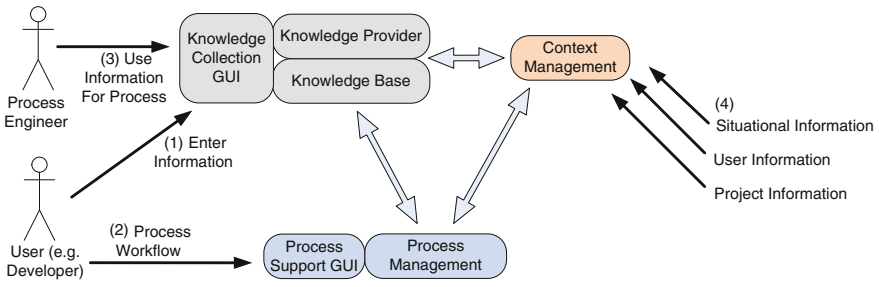

Fig. 8.5 Knowledge provisioning example

'junior' to indicate applicability for junior engineers, or 'backend' or 'frontend' to relate them to a specific implementation area. As a concrete example for this, Fig. 8.6 shows such a knowledge collection GUI concretely depicting different items of knowledge (guidance) a user has created.

2. The process of the project is managed and governed automatically by the system, including various operational workflows belonging to the process. Activities to be processed by humans are automatically delivered to them. Examples of activities governed that way include 'Implement Solution', where new source code is developed, or 'Run Developer Test', where source code is tested by the developer.
3. The governed workflows can be annotated by process engineers to make use of GIs and thus automatically deliver knowledge to the other users. Examples for such GIs include implementation or testing checklists, or specific notes as, e.g., hints about a relevant design pattern.
4. Applying a multitude of sensors in various applications, the system continuously detects new facts about the current situation. This makes it possible to tailor the knowledge provision to the user's current situation. For example, a junior engineer working at the frontend of an application could be provided a pre-GI containing the aforementioned item concerning a GUI-related design pattern when starting his 'Implement Solution' activity.

8.6 Knowledge-Based Contextual Adaptation of Processes

For manufacturers, the state and quality of their produced product is of primary importance. Therefore, knowledge about the product, its quality, relating problems, and quality measures to overcome the problems are crucial. Quality and quality

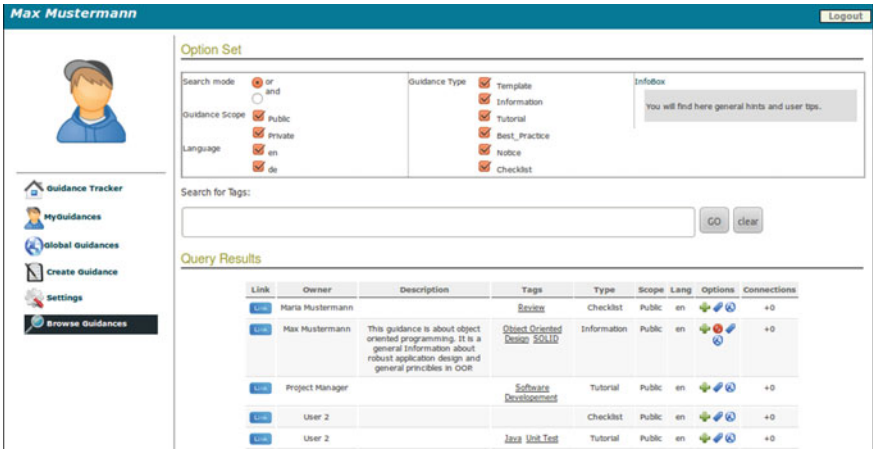

Fig. 8.6 Knowledge collection GUI screenshot

issues should typically be viewed holistically. For SE, software is intangible, and acquiring and relating quality issues to source code artifacts can be problematic. Furthermore, the effective and efficient application of software quality measures to proactively improve the product's quality as well as reactively correct discovered quality issues is even more challenging. One way to address quality issues systematically is to utilize knowledge to adapt processes in alignment with the users' context. For further reading on that topic see [19, 20]. This section is organized as follows: Sect. 8.6.1 introduces the knowledge-based adaptation concept, Sect. 8.6.2 elicits advanced requirements for such an approach, and Sect. 8.6.3 extends the presented approach to satisfy these requirements.

8.6.1 Concept for Knowledge-Based Contextual Adaptation of Processes

As a concrete scenario to illustrate this concept in the SE domain, we will use the automated integration of quality measures into processes. To support this critical area, we have integrated facilities into CoSEEEK that enable the automated integration of software quality measures into the development process via dynamic workflow adaptations. This section will introduce the basics regarding this facility by a simple example. It deals with proactive quality measures that users have identified as being useful, and have been entered into the knowledge base to be easily reused. Figure 8.7 illustrates how our system can facilitate such knowledge reuse actively.

As aforementioned, the user (e.g., a developer) enters a proactive software quality measure (an advice to analyze the modularity of the source code to proactively aid maintainability) into the *Knowledge Store* via the *Knowledge Collection GUI* and tags it in a way such that the system can identify it as such. It is thus available for other users when they are processing tasks relating to software development. In Fig. 8.7, such a workflow is shown: it is the 'Develop Solution Increment Workflow' that deals with the development of new software from the OpenUP [69] process. CoSEEEK governs that workflow within its *Process Management* component and manages related additional information and entities, like the processed artifacts or checklists in the *Context Management* component. That way, CoSEEEK's *Knowledge Provider* is aware of the activities and artifacts of the user's process and can thus provide matching information. In this example, CoSEEEK can automatically integrate a new activity relating to the proactive software quality measure right after the 'Implement Solution' activity, since it was detected that this quality measure would match the artifacts processed by that activity. With this approach, a seamless integration of QM with normal process execution is achieved.

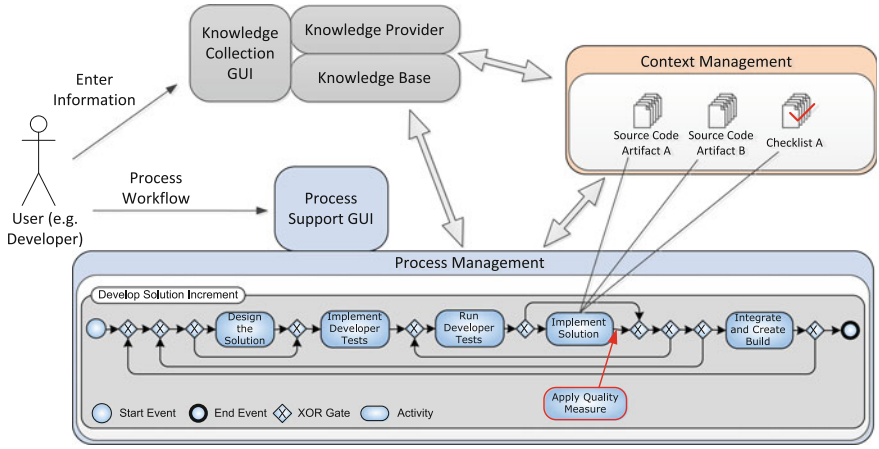

Fig. 8.7 Knowledge integration example

8.6.2 *Requirements for Knowledge-Based Contextual Adaptation of Processes*

In reality, a multitude of different factors influence quality measure provisioning. If they are not considered, the latter cannot be executed in an effective and efficient way. Section 8.6.3 will introduce a more complex extended approach to quality measure provisioning. A system aiming for holistic knowledge-based contextual adaptation of processes should incorporate the following features:

- *Problem awareness* (R2.1): be aware of problems in the assets produced within the organization (e.g., source code for SE);
- *Opportunity awareness* (R2.2): be aware of opportunities when users could apply actions (e.g., quality measures) to improve the situation (e.g., the quality of an artifact) without significantly delaying the process;
- *Strategic action alignment* (R2.3): strategically align possible actions (e.g., software quality measures) with goals of the current project (e.g., quality goals);
- *Proactive actions* (R2.4): Include not only reactive actions (e.g., reactive quality measures) dealing with existing problems, but also proactive actions (e.g., proactive quality measures) to prevent problems;
- *Context-sensitive actions* (R2.5): Enable context-sensitive tailoring of the actions (e.g., quality measures) so that they fit to the current situation and person;
- *Context monitoring* (R2.6): continuously monitor the context (e.g., quality of artifacts) and also identify the impact of actions (e.g., quality measures) on artifacts;
- *Seamless integration* (R2.7): enable seamless integration of the provided actions (e.g., quality measures) with the standard process to not delay the latter or disturb the participants.

8.6.3 Extended Concept for Knowledge-Based Contextual Adaptation of Processes

Recalling the problems and requirements we already elicited, there are many factors that play a role for successful automated quality support. On one hand, the system must be aware of the problems in artifacts [cf. requirement R2.1 (Problem awareness)]. On the other, it must be aware of the users' activities and the process to not hamper the process with inappropriate actions such as quality measures [cf. requirement R2.2 (Opportunity awareness)]. Furthermore, the measures must be in line with the goals of the project [cf. requirement R2.3 (Strategic action alignment)] and the current situation of the person applying them [cf. requirement R2.5 (Context-sensitive actions)]. In order to be able to exploit the usefulness of such measures, the system should manage proactive as well as reactive measures [cf. requirement R2.4 (Proactive actions)] and the applied measures should be assessed for their impact and utility [cf. requirement R2.6 (Context monitoring)]. Finally, the system should enable seamless integration of the measures into the standard development process to not disturb the users [cf. requirement R2.7 (Seamless integration)]. To be able to conform to this set of different and complex factors, we have defined a multi-step approach to automated QM that uses a second internal knowledge system within the *Context Management* component. This approach is illustrated in Fig. 8.8 and explained in the following.

The approach presented in Fig. 8.8 is separated into three phases. The detection phase is applied to generate an awareness of the systems environment. This includes source code artifacts [cf. requirement R2.1 (Problem awareness)] and user activities [cf. requirement R2.2 (Opportunity awareness)]. In the processing phase, a quality trend analysis of the source code takes place and, based on that, a quality measure prioritization including proactive and reactive software quality measures [cf. requirement R2.4 (Proactive actions)] in line with projects goals [cf. requirement R2.3 (Strategic action alignment)]. The proposed measures are then tailored to the users' situations [cf. requirement R2.5 (Context-sensitive actions)] and seamlessly integrated into their running workflows [cf. requirement R2.7 (Seamless

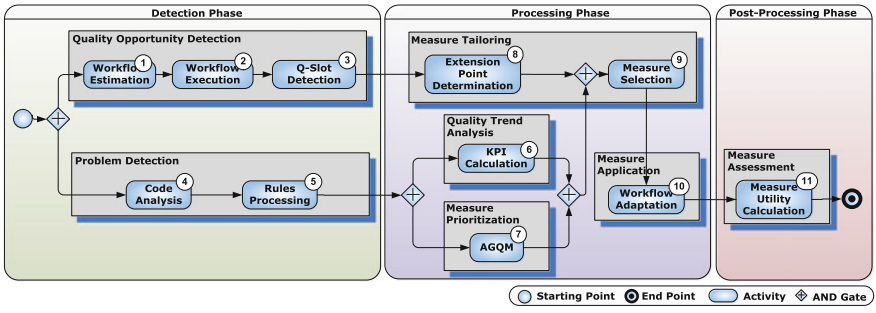

Fig. 8.8 Quality management approach

integration)]. To evolve the knowledge system, in the post-processing phase there is also a measure utility assessment [cf. requirement R2.6 (Context monitoring)] that reveals what measures were effective and ineffective.

The different steps of this approach are briefly explained in the following. They can be separated into three procedures: problem processing, opportunity processing, and measure assessment. The first one, problem processing, comprises the following steps. During the course of the project, the quality of the artifacts is continuously monitored, e.g., by static code analysis tools [Code Analysis (4)]. In turn, via the *Event Management* component, these tools are also monitored and the creation of a code analysis report is recognized by the system. These reports are then automatically transformed into a unified format. On such unified reports, pre-defined rules are executed that assess if any metric exceeds a given threshold, categorizes these cases as problems, and then automatically assigns an appropriate software quality measure to each problem [Rules processing (5)]. To obtain more meaningful values representing the global state of the artifacts, the metrics from the unified reports are aggregated to KPIs afterwards [KPI Calculation (6)]. As the number of assigned measures usually exceeds the capacities of a project, the assigned measures are later prioritized by an agent-based automated goal-question-metric [70] to align them to the quality goals of the project [AGQM (7)].

The second procedure deals with the quality opportunities in the users' workflows. This relates to users' tasks that are part of the process and opportunities to apply actions (i.e., quality measures) without delaying such tasks. Therefore, the different user tasks have to be estimated concerning time consumption by humans at the beginning [Workflow Estimation (1)]. These tasks are then automatically imported into the system and, for each of them, a dedicated workflow is started. After that, the workflows are executed within the system by the users [Workflow Execution (2)]. The system can, based on the estimated times and the actual times, carry out a so-called Q-Slot detection (3). This means that the system determines if a person has time left for the application of an action (i.e., software quality measure) without delaying the planned tasks. When the system has recognized a person with time left for a quality measure, the concrete point in one of his workflows where the measure application shall be integrated is determined [Extension Point Determination (8)]. This is done via semantic enhancements to the workflows in the *Context Management* component (cf. [20]). To make the applied measures as effective as possible, context-based measure selection is carried out by the system incorporating multiple properties of the situation and the intended user [Measure Selection (9)]. When the appropriate person, measure and extension point have been determined, the system automatically and seamlessly integrates the measure into the potentially running workflow of the person via the dynamic adaptation capabilities of AristaFlow [Workflow Adaptation (10)].

The third procedure deals with the assessment of measures that have been applied by the users. Therefore, the calculation of the KPIs representing the state of the source code is continuously executed [KPI Calculation (6)]. Therefore, it can serve as an indicator for the effectiveness of applied quality measures by comparing values before and after their application. At user-configured points in the process,

the effectiveness and usefulness of the applied measures (measure utility) will be automatically calculated by the system utilizing the KPIs [Measure Utility Calculation (11)]. The values obtained by this calculation will then be used in future measure proposals to improve the effectiveness of the applied measures.

Via the described approach, it becomes possible to effectively and systematically manage and provision knowledge regarding the quality of the artifacts an organization produces. Furthermore, that knowledge is actively used by the system to support and improve the situation (e.g., quality) by automatically distributing appropriate actions (i.e., matching quality measures) that fit a user's context and will adapt their process accordingly.

8.7 Knowledge-Based Collaborative Process Support

In knowledge-intensive projects, the essential collaboration between the knowledge workers involves concurrent or cooperative work on various complex artifacts. In some cases, one might depend on the work of others on a certain artifact, in other cases changes might interfere with each other or might entail additional work for someone. In particular, artifacts often relate to and can impact each other, e.g., the requirements specification may change, entailing changes to source code artifacts, while the implementation is already operational. For further reading on that topic see [23, 24]. Section 8.7.1 introduces specific requirements and Sect. 8.7.2 presents the collaboration concept.

8.7.1 Advanced Collaboration Requirements

To provide effective support for such projects, an automated aiming for holistic process and knowledge support should incorporate the following features:

- *Notification delivery* (R3.1): deliver notifications of interest to applicable users in case an artifact or the state of a task of a colleague changes;
- *Impact identification* (R3.2): identify the impact of the execution of a certain activity on certain artifacts;
- *Automatic activity initiation* (R3.3): automatically initiate certain follow-up activities to enable users to react to changes certain activities have caused. For example, if one of two associated artifacts is changed in an incompatible way, another activity could be initiated to also change the associated artifact;
- *Applicable actor identification* (R3.4): Be able to automatically identify the responsible person for a follow-up activity;
- *Configurability* (R3.5): Enable users to flexibly configure the way follow-up activities are initiated.

8.7.2 Collaboration Support Concept

The first requirement deals with passive coordination, where the system delivers information but does not actively affect the process. To enable such information distribution, the system relies on its event management and sensor infrastructure. When activities are executed by humans and artifacts are manipulated, both are usually done using some designated tool and can thus be detected by CoSEEEK. To exploit this for configurable notifications, an explicit notification concept is introduced. The properties of this concept are shown in Table 8.3. Utilizing this notification concept, both generic and personal notifications become possible that will be automatically delivered to the target person by CoSEEEK.

Requirements R3.2–R3.5 deal with active coordination, where the system affects the executed activities. This is a far more complex collaboration situation, in particular when it concerns associated artifacts that are part of different areas of a project, such as requirements management, implementation, or test management. Therefore, a set of prerequisites have to be satisfied to enable automated support: First, the project is split into hierarchically different components, such as areas or modules. These modules are then connected to each other, for example, to model the fact that a specific part of a requirements specification relates to a specific source code package or project (similar to traceability). Second, information is provided to indicate under which circumstances one area affects the other. Finally, different components are classified, for example if one source code package realizes the interface of a component.

With these facts modeled in CoSEEEK's *Context Management* component, a five-step procedure supports the configurable issuing of follow-up activities based on the occurrence of certain events. The first step of this procedure is applied to determine areas that might be affected by an activity. This step is configurable by the users and can take various contextual factors into account. Applied to the aforementioned example, for a requirements change such a configuration could be 'Search for affected areas in case of technical issues if an activity implies a change to a requirement'. Such a configuration would require the system to have access to the requirements. This can be established if the requirements are managed within a requirement management tool for which a sensor can be applied. After that, in a second step, the concrete target for a follow-up activity can be determined. For this

Table 8.3 Notification properties

Source	This denotes the entity to be monitored. Possible sources include various types of artifacts or different granularities of activities
Trigger	This denotes the event happening in context of the source entity that will be the trigger for the notification to be delivered. This can be the completion of an activity or the state change of an artifact
Target	This denotes the target, to which the notification will be delivered. This can be concrete persons or, to enable generic pre-configured notifications, also roles in a project

example, this would be a source code package that relates to the changed requirement. In a third step, a matching responsible person is identified for the follow-up activity. For this example this would be the developer responsible for the identified source code package. If none is defined, the system searches super-components of the package in the hierarchy and if no responsible can be found, the activity would be issued for the development team leader, who could then distribute it to the most appropriate developer. After that, the concrete activity to be issued has to be determined. It can take into account various contextual properties regarding involved artifacts, areas, sections and the activity that was the trigger. In the final step of the procedure, the follow-up activity must be integrated into the running process. This can be done either by starting a separate workflow for it or, if it matches properties of a running workflow, by integrating it into one of these. The adaptation of running workflow is applied in the same manner as described in Sect. 8.6.

By integrating contextual data and the combination of active and passive coordination capabilities, our concept can overcome various problems and support collaboration in knowledge-intensive projects. Active information distribution can be used to proactively counteract emerging problems, while passive information distribution can keep project participants updated and aware without obstructing the current process.

8.8 Summary and Conclusion

To summarize, with the growing volume of knowledge and the need for knowledge workers to efficiently utilize knowledge collaboratively, it is important that organizations have options that go beyond passive knowledge management techniques and that they also pursue the systematic active provisioning of knowledge. For such provisioning not to disrupt ongoing knowledge work, the system must possess contextual awareness and adapt to changes in both context and knowledge, integrating the provisioning of knowledge in such a way that is aligned to their current process (i.e., worker-goal awareness), and utilize knowledge to actively support worker collaboration.

The software development domain was used to exhibit these knowledge challenges, beginning with an overview of related current approaches in the software engineering (SE) domain. This was followed by a discussion of the problems and issues and the resulting requirements. We then described our holistic knowledge provisioning approach, first in an abstracted conceptual form followed then by a technical implementation for the SE domain called the CoSEEEK framework. To exemplify how it addresses the challenges using concrete scenarios, the chapter then illustrated automated knowledge provisioning within processes, knowledge-based contextual adaptation of processes, and support for knowledge-based collaborative processes.

Future challenges include the integration and utilization of distributed extra-organizational knowledge bases, cross-granular process and contextual dependencies, and automated semantic annotation techniques.

Glossary

The terms below are defined practically for the purpose of understanding this chapter, and not intended to be definitive or comprehensive.

Context-awareness. Perception of a system's surroundings via information that can be used to characterize the situation. This information can consist of various things like other systems, humans, actions, events, or related artifacts.

Information. Facts and data organized to describe a particular situation or condition. Knowledge communicated or received concerning a particular fact or circumstance.

Knowledge. Familiarity, acquaintance, experience with, understanding, or perception of some subject, involving facts, truths, principles, beliefs, perspectives, concepts, judgments, expectations, methodologies, or know-how. Within organizations, it frequently becomes embedded in documents or repositories, as well as in organizational routines, processes, practices, and norms [71]. It is a "justified belief that increases an entity's capacity for taking effective action" [72]. Information can be converted into knowledge once cognitively processed, and knowledge can be transformed into information if codified or articulated in symbolic forms.

Knowledge base (KB). A repository of knowledge, typically utilizing some form of storage.

Knowledge management (KM). A systematic and organizational process for retaining, organizing, sharing, and updating (collective) knowledge critical to individual performance and organizational competitiveness [73].

Knowledge systems. Organizations as social collectives can be viewed as knowledge systems, representing the cognitive and social nature of organizational knowledge and its embodiment in the individuals' mind and practices as well as the practices and culture of the organization [72].

Knowledge management systems (KMS). To support human knowledge systems, IT-based knowledge management systems support the codification and sharing of knowledge, the creation and maintenance of knowledge repositories, and knowledge networking [72] or collaboration.

Knowledge-based system (KBS). A system that uses knowledge, either in an open or closed form, to adjust its own behavior.

Process-aware information systems (PAIS). Information systems that enable the automated implementation of processes comprising their whole lifecycle, including modeling, enactment, and monitoring.

References

1. Lenz, R., Reichert, M.: IT support for healthcare processes-premises, challenges, perspectives. Data Knowl. Eng. **61**(1), 39–58 (2007)
2. Müller, D., Herbst, J., Hammori, M., Reichert, M.: IT support for release management processes in the automotive industry. In: Proceedings of 4th International Conference on Business Process Management, pp. 368–377 (2006)
3. Mutschler, B., Reichert, M., Bumiller, J.: Unleashing the effectiveness of process-oriented information systems: Problem analysis, critical success factors, and implications. Syst. Man Cybern. Part C Appl. Rev. IEEE Trans. **38**(3), 280–291 (2008)
4. Gibson, D.L., Goldenson, D.R., Kost, K.: Performance results of CMMI-based process improvement. Technical Report, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh (2006)
5. Heravizadeh, M.: Quality-aware business process management. PhD Thesis, Queensland University of Technology (2009)
6. Lohrmann, M., Reichert, M.: Efficacy-aware business process modeling. In: Proceedings of 20th International Conference on Cooperative Information Systems, pp. 38–55 (2012)
7. Lohrmann, M., Reichert, M.: Understanding business process quality. In: Business Process Management, pp. 41–73. Springer, Berlin (2013)
8. Gloet, M., Terziovski, M.: Exploring the relationship between knowledge management practices and innovation performance. J. Manuf. Technol. Manage. **15**(5), 402–409 (2004)
9. Künzle, V., Weber, B., Reichert, M.: Object-aware business processes: Fundamental requirements and their support in existing approaches. Int. J. Inf. Syst. Model. Des. (IJISMD) **2**(2), 19–46 (2011)
10. Mundbrod, N., Kolb, J., Reichert, M.: Towards a system support of collaborative knowledge work. In: Proceedings of Business Process Management Workshops, pp. 31–42 (2013)
11. Ramesh, B., Tiwana, A.: Supporting collaborative process knowledge management in new product development teams. Decis. Support Syst. **27**, 213–235 (1999)
12. Müller, D., Reichert, M., Herbst, J.: A new paradigm for the enactment and dynamic adaptation of data-driven process structures. In: Proceedings 20th International Conference on Advanced Information Systems Engineering, pp. 48–63 (2008)
13. Bonifacio, M., Bouquet, P., Cuel, R.: Knowledge nodes: the building blocks of a distributed approach to knowledge management. J. Univ. Comput. Sci. **8**(6), 652–661 (2002)
14. Maier, R.: Knowledge Management Systems: Information and Communication Technologies for Knowledge Management. Springer, New York (2002)
15. Drucker, P. F.: Knowledge-worker productivity: the biggest challenge. Knowl. Manage. Yearbook 2000–2001 (1999)
16. Davenport, T. H.: Rethinking knowledge work: a strategic approach. McKinsey Q. **1**(11), 88–99 (2011)
17. Lindvall, M., Rus, I.: Knowledge management in software engineering. IEEE Softw. **19**(3), 26–38 (2002)
18. Grambow, G., Oberhauser, R., Reichert, M.: User-centric abstraction of workflow logic applied to software engineering processes. In: Proceedings of 1st Workshop on Human-Centric Process-Aware Information Systems, LNBIP112, pp. 307–321 (2012)
19. Grambow, G., Oberhauser, R.: Towards automated context-aware selection of software quality measures. In: Proceedings of 5th International Conference on Software Engineering Advances, pp. 347–352 (2010)
20. Grambow, G., Oberhauser, R., Reichert, M.: Contextual injection of quality measures into software engineering processes. Int. J. Adv. Softw. **4**(1–2), 76–99 (2011)
21. Grambow, G., Oberhauser, R., Reichert, M.: Towards dynamic knowledge support in software engineering processes In: Proceedings of 6th International Workshop on Applications of Semantic Technologies (AST'11), held in conjunction with INFORMATIK'11, LNI 192, p. 149 (2011)

22. Grambow, G., Oberhauser, R., Reichert, M.: Knowledge provisioning: a context-sensitive process-oriented approach applied to software engineering environments. In: Proceedings of 7th International Conference on Software and Data Technologies, pp. 506–515 (2012)
23. Grambow, G., Oberhauser, R., Reichert, M.: Towards automatic process-aware coordination in collaborative software engineering. In: Proceedings of 6th International Conference on Software and Data Technologies, pp. 5–14 (2011)
24. Grambow, G., Oberhauser, R., Reichert, M.: Enabling automatic process-aware collaboration support in software engineering projects. In: Selected Papers of the ICSOFT'11 Conference. Communications in Computer and Information Science (CCIS) 303, pp. 73–89 (2012)
25. Grambow, G.: Context-aware Process Management for the Software Engineering Domain. Doctoral Thesis, Ulm University (2015). (to appear)
26. Bjørnson, F.O., Dingsøyr, T.: Knowledge management in software engineering: a systematic review of studied concepts, findings and research methods used. Inf. Softw. Technol. **50**(11), 1055–1068 (2008)
27. Kurniawati, F., Jeffery, R.: The long-term effects of an EPG/ER in a small software organisation. In: Proceedings of Australian Software Engineering Conference, pp. 128–136 (2004)
28. Barros, M.O., Werner, C.M.L., Travassos, G.H.: Supporting risks in software project management. J. Syst. Softw. **70**(1–2), 21–35 (2004)
29. Basili, V., Costa, P., Lindvall, M., Mendonca, M., Seaman, C., Tesoriero, R., Zelkowitz, M.: An experience management system for a software engineering research organization. In: Proceedings of 26th Annual NASA Software Engineering Workshop, pp. 29–35 (2001)
30. Liao, S.: Knowledge management technologies and applications—literature review from 1995 to 2002. Expert Syst. Appl. **25**(2), 155–164 (2003)
31. Daskalantonakis, M.K.: A practical view of software measurement and implementation experiences within Motorola. Softw. Eng. IEEE Trans. **18**(11), 998–1010 (1992)
32. Offen, R.J., Jeffery, R.: Establishing software measurement programs. Softw. IEEE **14**(2), 45–53 (1997)
33. Gopal, A., Krishnan, M.S., Mukhopadhyay, T., Goldenson, D.R.: Measurement programs in software development: determinants of success. Softw. Eng. IEEE Trans. **28**(9), 863–875 (2002)
34. Li, Z., Zhou, Y.: PR-Miner: automatically extracting implicit programming rules and detecting violations in large software code. In: ACM SIGSOFT Software Engineering Notes, vol. 30, pp. 306–315 (2005)
35. Ohira, M., Yokomori, R., Sakai, M., Matsumoto, K., Inoue, K., Torii, K.: Empirical project monitor: a tool for mining multiple project data. In: Proceedings of International Workshop on Mining Software Repositories (2004)
36. Schlesinger, F., Jekutsch, S.: ElectroCodeoGram: an environment for studying programming. TeamEthno-online, vol. 2, pp. 30–31 (2006)
37. Nystrom, N.A., Urbanic, J., Savinell, C.: Understanding productivity through non-intrusive instrumentation and statistical learning. . In: Proceedings of 2nd Workshop on Productivity and Performance in High-End Computing (2005)
38. Jiang, T., Ying, J., Wu, M.: CASDE: An environment for collaborative software development. In: Computer Supported Cooperative Work in Design III, LNCS, 4402, pp. 367–376 (2007)
39. Lewandowski, A., Bourguin, G.: Enhancing support for collaboration in software development environments. In: Computer Supported Cooperative Work in Design III, LNCS, 4402, pp. 160–169 (2007)
40. Cook, C., Churcher, N., Irwin, W.: Towards synchronous collaborative software engineering. In: Proceedings of 11th Asia-Pacific Software Engineering Conference, pp. 230–239 (2004)
41. Hattori, L., Lanza, M.: Syde: a tool for collaborative software development. In: Proceedings of 32nd International Conference on Software Engineering, pp. 235–238 (2010)
42. Weber, S., Emrich, A., Broschart, J., Ras, E., Ünalan, Ö.: Supporting software development teams with a semantic process-and artifactoriented collaboration environment. In: Proceedings of Software Engineering (Workshops), pp. 243–254 (2009)

43. de Lucia, A., Fasano, F., Oliveto, R., Tortora, G.: Fine-grained management of software artefacts: the ADAMS system. Softw. Pract. Experience **40**(11), 1007–1034 (2010)
44. de Oliveira, K.M., Zlot, F., Rocha, A.R., Travassos, G.H., Galotta, C., de Menezes, C.S.: Domain-oriented software development environment. J. Syst. Softw. **72**(2), 145–161 (2004)
45. Maciel, R.S.P., da Silva, B.C., Magalhães, P.F., Rosa, N.S.: An integrated approach for model driven process modeling and enactment. In: Proceedings of Software Engineering, 2009. SBES'09. XXIII Brazilian Symposium on, pp. 104–114 (2009)
46. Aleixo, F.A., Freire, M.A., dos Santos, W.C., Kulesza, U.: Automating the variability management, customization and deployment of software processes: a model-driven approach. In: Enterprise Information Systems, pp. 372–387. Springer, Berlin (2011)
47. Dowson, M.: Consistency maintenance in process sensitive environments. In: Proceedings of Process Sensitive Software Engineering Environments Architectures Workshop (1992)
48. Conradi, R., Fernström, C., Fuggetta, A., Snowdon, R.: Towards a reference framework for process concepts. In: Software Process Technology, pp. 1–17. Springer, Berlin (1992)
49. Reichert, M., Weber, B.: Enabling Flexibility in Process-aware Information Systems—Challenges, Methods, Technologies. Springer, Berlin (2012)
50. Reichert, M., Rinderle-Ma, S., Dadam, P.: Flexibility in process-aware information systems. In: Transactions on Petri Nets and Other Models of Concurrency II, pp. 115–135 (2009)
51. Gelernter, D.: Generative communication in Linda. ACM Trans. Program. Lang. Syst. (TOPLAS) **7**(1), 80–112 (1985)
52. Meier, W.: eXist: an open source native XML database. In: Web, Web-Services, and Database Systems, LNCS, 2593, pp. 169–183 (2009)
53. Johnson, P.M.: Requirement and design trade-offs in Hackystat: an in-process software engineering measurement and analysis system. In: Proceedings of 1st International Symposium on Empirical Software Engineering and Measurement, pp. 81–90 (2007)
54. Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Longman Publishing Co., Inc., Boston (2001)
55. Esper. Website: http://esper.codehaus.org. Visited: September (2013)
56. Bellifemine, F., Poggi, A., Rimassa, G.: JADE–A FIPA-compliant agent framework. In: Proceedings of 4th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents, pp. 97–108 (1999)
57. Browne, P.: JBoss Drools Business Rules. Packt Publishing, Birmingham (2009)
58. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. Comput. Sci. Res. Develop. **23**(2), 81–97 (2009)
59. Lanz, A., Reichert, M., Dadam, P.: Robust and flexible error handling in the AristaFlow BPM Suite. In: Proceedings of CAiSE'10 Forum, Information Systems Evolution, pp. 174–189 (2011)
60. Krötzsch, M., Vrandecic, D., Völkel, M.: Semantic mediawiki. In: Proceedings of International Semantic Web Conference, pp. 935–942 (2006)
61. World Wide Web Consortium: OWL Web Ontology Language Semantics and Abstract Syntax (2004)
62. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a practical owl-dl reasoner. Web Semant. Sci. Serv. Agents World Wide Web **5**(2), 51–53 (2007)
63. McBride, B.: Jena: a semantic web toolkit. Internet Comput. IEEE **6**(6), 55–59 (2002)
64. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: a semantic web rule language combining OWL and RuleML. W3C Member Submission **21**, 79 (2004)
65. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C WD 4 (2006)
66. Kess, P., Haapasalo, H.: Knowledge creation through a project review process in software production. Int. J. Prod. Econ. **80**(1), 49–55 (2002)
67. Teigland, R., Fey, C.F., Birkinshaw, J.: Knowledge dissemination in global R&D operations: an empirical study of multinationals in the high technology electronics industry. In: MIR: Management International Review, pp. 49–77 (2000)

68. Schaffert, S., Bry, F., Baumeister, J., Kiesel, M.: Semantic wikis. IEEE Softw. **25**(4), 8–11 (2008)
69. Kroll, P., MacIsaac, B.: Agility and Discipline Made Easy: Practices from OpenUP and RUP. Pearson Education, New York (2006)
70. Basili, V.R., Caldiera, V.R.B.G., Rombach, H.D.: The goal question metric approach. Encycl. Softw. Eng. **2**, 528–532 (1994)
71. Davenport, T. H., Pruzak, L.: Working Knowledge: How Organizations Manage What They Know. Harvard Business Press, Boston (2000)
72. Alavi, M., Leidner, D. E.: Review: knowledge management and knowledge management systems: conceptual foundations and research issues. MIS Q. 107–136 (2001)
73. Davenport, T.H., David, W., Beers, M.C.: Successful knowledge management projects. Sloan Manage. Rev. **39**(2), 43–57 (1998)

Additional Resources on Related Topics: Books

74. Tiwana, A.: The Knowledge Management Toolkit: Practical Techniques for Building a Knowledge Management System. Prentice Hall PTR, New Jersey (2000)
75. Davenport, T.H., Probst, G.J.: Knowledge Management Case Book: Siemens Best Practices. Wiley, New York (2002)
76. Liebowitz, J.: Knowledge Management: Handbook. CRC Press, Boca Raton (1999)
77. Dalkir, K.: Knowledge Management in Theory and Practice. Routledge, London (2013)
78. Ruggles, R.: Knowledge Management Tools. Routledge, London (2012)

Articles

79. Davenport, T.H., David, W., Beers, M.C.: Successful knowledge management projects. Sloan Manage. Rev. **39**(2), 43–57 (1998)
80. Alavi, M., Leidner, D.E.: Review: knowledge management and knowledge management systems: conceptual foundations and research issues. MIS Q. 107–136 (2001)
81. IEEE Transactions on Knowledge and Data Engineering

Conferences and Workshops

82. IEEE International Conference on Information Reuse and Integration (IRI)
83. International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)
84. International Conference on Intelligent Systems and Knowledge Engineering (ISKE)
85. International Workshop on Knowledge Acquisition, Reuse and Evaluation (KARE)
86. Workshop on Knowledge Engineering and Software Engineering (KESE)