Leveraging Semantic Web Computing for Context-Aware Software Engineering Environments

Roy Oberhauser
Aalen University
Germany

1. Introduction

The increasing and ongoing need for and reliance on software in almost all industries, coupled with increasing code volume and complexity and changing technologies, results in increasing productivity pressure on software engineers to deliver greater software functionality within stringent cost, time, and quality constraints. Moreover, the software maintenance phase is affected by these pressures, and new approaches are also needed for improving the efficiency and effectiveness of corrective and progressive maintenance activities. The nature of many software engineering (SE) projects, especially in small and medium enterprises (SMEs), often undergo rapid technology, tool, and process changes. Although coupled with ever shorter delivery cycles, this area has, however, not yet lent itself to the type of optimization that business process management (BPM) has succeeded in achieving in other industries and disciplines. It is somewhat ironic that software and IT technology has played a significant role in achieving results for BPM, yet the application to SE processes has not succeeded. Among the various challenges, software engineering environments (SEEs) typically consist of heterogeneous tool environments not optimally tuned to leveraging the semantic value of data that become available in these SEEs during a SE project.

Computer-Aided Software Engineering (CASE) tools are software applications that support software engineering activities within a software development process. Due to the incessant lack of standardization in data formats, interfaces, protocols, and agreed upon (common) data models, tools have typically been created with their own internal interaction and data model paradigms. Yet the focus of industry on BPM systems (BPMS) via Enterprise Application Integration (EAI) with Service-Oriented Architecture (SOA) has fueled a wide adoption of web service (WS) toolkits, and has recently had a ripple effect in SE with an increasing provision of RESTful and SOAP-based WS access to tool functionality and data. However, the access to this functionality and data has not been exploited via Semantic Web (SemWeb) technologies, and herein lies potential. SemWeb adds machine-processable semantics to data (Berners-Lee et al., 2001). SemWeb computing (SWC) allows for greater and improved automation and integration of information due to its formal structuring of

information, clearly defined meanings and properties of domain concepts, and standardized exchange formats. The application of SWC within a confined heterogeneous SEE setting where external interchange of data is not a primary concern can provide certain advantages, for example for context-aware computing (CAC).
This chapter explores CoSEEEK (Context-aware Software Engineering Environment Event-driven frameworK), a hybrid semantic web computing approach towards improved context-aware SEEs. The approach is based on an event-based computing paradigm, utilizing multi-agent computing for active SE processing components. Space-based computing is used as a common data repository, decoupling the interaction and data of tools and agents and supporting heterogeneous and flexible configurations. A process-aware information system (PAIS) is utilized to support adaptable SE processes, giving SE engineers process support while supporting the degree of adaptability appropriate for the organization. A conjunction of paradigms enables CAC to be applied in heterogeneous SEE settings and exhibit proactive and reactive behaviors that can improve the effectiveness and efficiency of SEEs. The hybrid SemWeb focus avoids the perhaps unjustifiable time and resource investments that a comprehensive integration would require for the tool, artifact, person, process, project, measure, practice, event, and data models in an SEE along with the inevitable continual changes, while leveraging the noticeable benefits for software engineers in responsiveness of the SEE. The experimental results validate the current feasibility and potential benefits that such a distributed, decentralized, event-based, hybrid semantic web approach can bring to SEEs. The chapter is organized as follows: section 2 reviews the current literature; section 3 discusses the requirements and constraints; section 4 describes the solution approach while section 5 details a current realization; section 6 then discusses the results which are followed by a conclusion in section 7.

2. Literature Review

With regard to SE tool interoperability in SEEs, one attempt at standardization was the Portable Common Tool Environment (H-PCTE) ISO/IEC 13719:1998, "a distributed object management system (OMS) standardized by ISO/IEC and ECMA which is intended to be used as a basis of distributed software development environments (SDE), or, more generally, distributed document editing systems." It specifies various services such as data management, schema management, access and concurrency controls, notifications, and bindings for C, Ada, and IDL (CORBA). At this point it is not relevant to the industry, as no commonly used SE tools today utilize or promote this or alternative standards.
(Arbaoui et al., 2002) and (Gruhn, 2002) provide an overview of Process-Centered Software Engineering Environments (PCSEEs). (Adams et al., 2006) describes worklets, a SOA-based extensible repertoire of self-contained sub-processes aligned to each task, from which a dynamic runtime selection is made depending on the context of the particular work instance. An example of a metamodel-based PCSEE framework is OPEN (Object-oriented Process, Environment and Notation), which addressed business, quality, model, and reuse issues (Henderson-Sellers, 2002) and is based on CORBA and not on WS. It has not been active since 2006. Another example is DiME, which is a proprietary, integrated, collaborative environment for managing product definition, development and delivery processes and information (Koenig, 2003).

As to integration of SemWeb technologies in the SE lifecycle, (Oberhauser & Schmidt, 2007) discuss a holistic approach. (Calero et al., 2006) includes ontology work on SWEBOK (Software Engineering Body of Knowledge), software maintenance, software measurement, and other related SE ontologies. (Happel & Seedorf, 2006) describe the application of ontologies in SE and a framework for classifying them. With regard to software artifacts, (Bontcheva & Sabou, 2006) present an ontology learning approach that exploits a range of information sources associated with software projects. Work utilizing the Semantic Web for automated software engineering purposes includes (Dinger et al., 2006).
While it appears that relatively little work on context-aware SEEs has been done, much work regarding context-awareness has been done in the area of ubiquitous and pervasive computing, e.g., (Ferscha et al., 2004). Examples of frameworks for building context-aware applications include the Java Context-Awareness Framework (JCAF) (Bardram, 2005) and the ContextToolkit (Dey & Abowd, 2000). Considering the combination of SWC with CAC, (Adi et al., 2000) describe a semantic data modeling approach for situation detection that defines and describes events and their relationships to other events, objects, and tasks. (Christopoulou et al., 2005) describe Context Meta-Model (CMM), an ontology-based three layer metamodel for context with a formal mapping to OWL DL.

3. Requirements and Constraints

For creating a solution approach for context-aware SEEs, specific requirements and constraints must be considered. As shown in the context diagram of Fig. 1, for typical SE projects, artifacts are retained in a configuration management (CM) tool-based repository, e.g., CVS, Subversion, etc., and thus changes to artifacts can be readily detected and SE events can be generated. People involved in SE activities interact with the SEE primarily via the use of SE tools (shown as SE Tool Services), and to support context-awareness these interactions should generate SE events transparently. SE actions may also be directed to appropriate SE tools or tool services. Other project-specific inputs into the SEE are tailored SE processes, workflows, and best practices, as well as general and domain-specific SE knowledge and quality assurance methods.

Fig. 1. SEE context diagram

While numerous and various requirements for a context-aware SEE may be considered due to the diversity in SEEs, the following general requirements were important to shaping the solution approach (the square brackets below indicate the abbreviated form):

- Automatic selection of proposed quality measures should be based on contextual problems or risks [Req:AutoQM]
- Quality measures should be adjusted based on new events and states across the project [Req:QMAdj]
- Automated project task assignment for a software engineer should be based on task difficulty (which maps to skill level), employee availability, and roles [Req:AutoTask]
- Tasks to be performed by a software engineer shall be adjusted based on context within the realm of constraints allowed in the process workflow [Req:TaskAdj]
- The black-box view of solution use cases should demonstrate context-awareness not otherwise currently easily provided [Req:BlackBox]
- Support for heterogeneous operating systems and SE tool implementations [Req:Heterog]
- Avoid internal access to SE tools [Req:Encap]
- Support for distributed SEEs [Req:Dist]

4. CoSEEEK Solution Approach

To achieve improved and more holistic solutions for SEEs, the CoSEEEK approach is a synthesis of various areas of computing shown in (Fig. 2), specifically semantic web computing (SWC), service-oriented computing (SOC), space-based computing (SBC), multi-agent computing (MAC), event-based computing (EBC), complex-event processing (CEP), context-aware computing (CAC), rule-based computing (RBC), and process-aware information systems (PAIS). These will be discussed below.

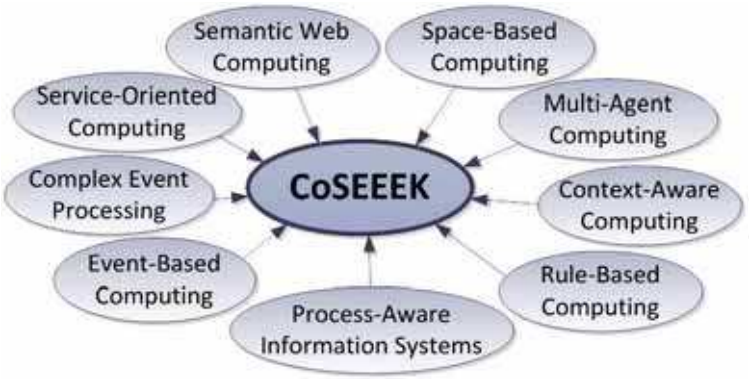

Fig. 2. The CoSEEEK synergistic solution approach to SEE

Semantic web computing, with its formal structuring of information and machine-processable semantics, has the potential to improve SE automation and information integration. One of the issues facing SWC is the creation and adoption of standardized ontologies in OWL (Web

Ontology Language) (Horrocks et al., 2004) for the various industry domains to precisely define the semantic meaning of the domain-specific concepts. The additional modeling effort incurred by ontologies must result in savings elsewhere (Oberle, 2006). Coupled with customized SEEs needs and the rapidly changing and specialized nature of many SE tools, a transitional hybrid stage is proposed. CoSEEEK utilizes the advantages of the distributed and heterogeneous support OWL provides, and relies on defining the semantic meaning of a common subset of the key concepts necessary to adjust quality measures, project task assignments, or workflows based on events within a shared agent-accessible context.

Service-oriented computing, with its reliance on Web Services (WS), provides platform-neutral integration for arbitrary applications (Alonso et al., 2003). The advantages of WS for SE are discussed in (Dinger et al., 2006). Some SE tools already support WS via SOAP or REST, and this access to data and functionality produced or consumed by the SEE tools can be leveraged, e.g., by agents, for enhanced collaboration and distributed data exchange across heterogeneous services in a loosely-coupled fashion. All CoSEEEK inter-process communication is via WS in support of [Req:Dist] and [Req:Heterog].

Space-based computing is a powerful paradigm for coordinating autonomous processes by accessing tuples (an ordered set of typed fields) in a distributed shared memory (called a tuple space) via messaging (Gelernter, 1985), thereby exhibiting linear scalability properties by minimizing shared resources. Work on semantic enhancement of tuple spaces includes sTuples (Khushraj et al., 2004), which extends the object-oriented JavaSpace implementation (Freeman et al., 1999) with an object field of type DAML-OIL Individual. (Tolksdorf et al., 2005) and (Tolksdorf et al., 2005a) describe work on Semantic Tuple Spaces. The Triple Space Computing (TSC) project[1] aims to develop a communication and coordination framework for the WSMX Semantic Web Service platform (Bussler et al., 2005) (Simperl, 2007). CoSEEEK leverages a non-SemWeb XML-based SBC to support a common shared context accessible by loosely-coupled agents. This also supports [Req:Dist] [Req:Heterog] as well as the hybrid SemWeb approach.

Multi-agent computing or Multi-Agent Systems (MAS) have been researched extensively[2] [3]. Agent-based event management approaches includes Sense, Think & Act (ST&A), which exhibits function-driven, goal-driven (local goals), and collaborative goal-driven (global goals) behaviors. Tool-specific agents are used to invoke tool functionality, retrieve data, or provide event sources. In CoSEEEK, the agents are employed in the style of the blackboard architecture pattern (Hayes-Roth, 1985); thus agents do not interact directly, resulting in loose-coupling and functional flexibility. SBC is utilized and events are placed in spaces where subscribing agents are notified of changes. This supports [Req:Encap] and [Req:Dist].

Event-based computing allows the flow of the software functionality to be determined by events, supporting context-awareness with temporal data and allowing reactive and proactive behaviors. Proactive in this sense is behavior that is preventative in regard to SE problems, and may still be a response to some event and not necessarily self-triggered.

Complex event processing (Luckham, 2002) or event stream processing (ESP) is a concept to deal with meaningful event detection and processing using pattern detection, event

[1] http://tsc.deri.at

[2] *The Journal of Autonomous Agents and Multiagent Systems*, Publisher: Springer Science+Business Media B.V.

[3] *Whitestein Series in Software Agent Technologies and Autonomic Computing*, published by Springer Science+Business Media Group

correlation, and other techniques to detect complex events from simpler events. With CoSEEEK, once complex events are detected, workflow adjustments can be made in a PAIS, and developers are informed about changes via Task Management.

Context-aware computing is concerned with the acquisition of context (e.g., using sensors to perceive a situation), the abstraction and understanding of context (e.g., matching a perceived sensory stimulus to a context), and application behavior based on the recognized context (e.g., triggering actions based on context) (Schmidt, 2003). Event Condition Actions (ECA) is enabled via a semantic reasoner. CoSEEEK utilizes CAC to support the requirements [Req:AutoQM][Req:QMAdj][Req:AutoTask][Req:BlackBox].

In *rule-based computing* a collection of rules is applied to a collection of facts via pattern matching using efficient algorithms such as Rete (Forgy, 1982). It may be advantageous with regard to the transitional hybrid SemWeb support that non-context-specific rules (e.g., artifact quality rules) be maintained separately. CoSEEEK sees advantages to utilizing RBC for such purposes, for example triggering quality events at certain thresholds. This also reduces the ontology complexity.

Process-aware information systems separate process logic from application code while avoiding data- or function- centricity. Workflow management systems (van der Aalst & van Hee, 2002) can be viewed as an enabling PAIS technology, whereas a key feature of PAIS is to support process change (Reichert & Dadam, 1997; Müller et al., 2004; Pesic et al., 2007). Since SE in a project setting is in view for this chapter, the uniqueness of each project and the complexity will likely cause unforeseen changes to become necessary in a subset of SE processes. The CoSEEEK approach utilizes PAIS to support the requirement for adaptable SE processes [Req:TaskAdj].

The combination of these various computing paradigms enhances the ability of the CoSEEEK approach to deal with various difficulties that arise in supporting context-aware SEEs while fulfilling the requirements. The following discussion describes considerations regarding the key aspects of event processing, the conceptual architecture, and the context model.

4.1 Event Processing

Due to the very heterogeneous nature of SE tooling, today's available tools are typically built with an information island mentality, and at best integration into a widely-used IDE (Integrated Development Environment) is considered, e.g., Eclipse[4]. Given the lack of standards and support for sourcing tool SE events, various techniques such as proxies, tool agents, plugins, or wrappers may be used to generate such SE events.

As illustrated in Fig. 3, events from SE tooling are acquired and then stored in the common space, where it may optionally be annotated with any relevant contextual information by any agent after this point. Event processing mainly includes CEP to detect higher level events. Agents with subscriptions to the space are notified if appropriate, and proactive and reactive behaviors are supported. This may result in workflow adjustment, and the software engineer is informed of a change in tasks or measures via task management. The responses and actions by software engineers to task management via SE tools cause further event acquisition, and so on.

[4] http://eclipse.org

Fig. 3. Event flow

4.2 Solution Architecture

The conceptual view of the CoSEEEK solution architecture is shown in Fig. 4. *Artifacts* is a placeholder for the artifacts that are produced or used in a software project. These are accessed usually via tools directly or indirectly on a file system. *SE Tools* is a placeholder for independent development and testing tools that are tied into CoSEEEK. *Agents* provides behavior agents as well as management agents for each SE tool and CoSEEEK process for application control and integration in the architecture. The *Event Extraction* consists primarily of event sensors and data collection for SE tools. *Data Storage* provides event and data storage in a loosely-coupled fashion via an XML Space implementation. This allows CoSEEEK (e.g., the agents) to be reactive to event or data changes and still be loosely-coupled, thus enabling integration without dependencies. *Event Processing* applies CEP and any contextual annotation to events. *Process Management* is aware of and responsible for SE process conformance of activities and supports adaptive task management. *Context Management* contains a semantic reasoner that tracks and adapts the context as needed and generates appropriate events to initiate behavior. The *SemWeb Integration* module is OWL-aware and is responsible for loading, storing, and synchronizing OWL between the space and the Context Management module, e.g., using Jena or Protégé generated Data Access Objects (DAOs).

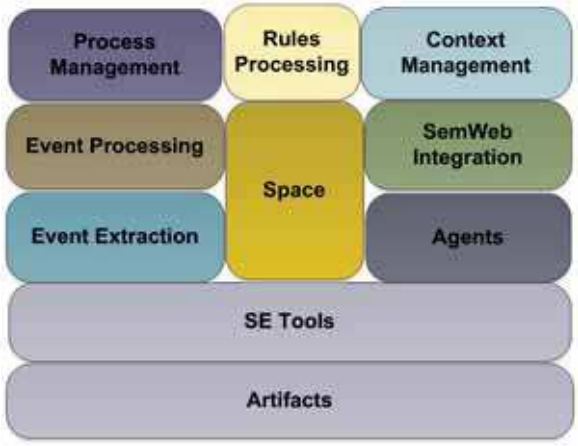

Fig. 4. Conceptual view of CoSEEEK architecture

4.3 Context Model

The context model developed for CoSEEEK will be described in conformance with the context model analysis framework presented in (Bolchini et al., 2007) and is summarized in Table 1.

Modeled aspects are the set of context dimensions managed by the model:

- *Space*: no location-specific aspects were as yet necessary, but could easily be added to handle geographically distributed projects.
- *Time*: temporal aspects are managed, e.g., the timeframes allowed or available for certain tasks or quality measures.
- *Absolute/relative space and time*: Both absolute and relative time aspects are needed. The absolute time is needed for logging and archival purposes as well as supporting traceability. Relative time is used for determining the time available for tasks or measures.
- *Context history*: The context history is used as experience to adjust and improve current and future relations between problems, risks, and preemptive and reactive measures
- *Subject*: the point of view of the context is both the user and the application.
- *User profile (Role or Features based):* The profile of the user, e.g., their experience, is considered explicitly in the context.

Representation features are the general characteristics of the model itself:

- *Type of formalism*: Ontology-based
- *Formality level*: OWL-Lite compatibility was achieved and results in the best performance and computability characteristics.
- *Flexibility*: the context is bound to the SE domain. However, an adaptation to new SE concepts is supported
- *Variable context granularity*: aspects deal with different abstraction levels, e.g., artifacts, activities, persons, and the project.
- *Valid context constraints*: the number of admissible contexts is constrained, e.g., a person executes an activity within a project

Context management and usage refers to the way the context is built, managed and exploited:

- *Context construction:* the context description is built centrally at design-time, rather than dynamic run-time agreement among partners.
- *Context reasoning*: reasoning on context data is enabled by the model to infer properties. Current usage is however rule-based, but inference of new facts is foreseen for future usage.
- *Context quality monitoring*: the quality of the retrieved context information is not considered or managed.
- *Ambiguity/Incompleteness management*: ambiguous, incoherent or incomplete context information is not interpolated or mediated. Only valid data is accepted.
- *Automatic learning features*: the model was designed to support this aspect, and although it does not yet exhibit automatic learning features in the currently supported use cases, this is desirable and will be developed in the future
- *Multi-context model*: All contexts are represented in a single central model instance, with the advantage that the reasoning has access to all the possible data.

In summary, the essence of the CoSEEEK approach is the conjointment of the computing paradigms of Fig. 2. The event processing flow, the space-centric solution architecture, and

the SEE-specific context model elaborate on how these can be combined in order to fulfill the requirements for SEEs described in section 3.

Category	**Context**
Space	Future
Time	Supported
Space/Time coordinates (Relative or Absolute)	A,R
Context history	Supported
Subject (User or Application)	Projects, Activities, Artifacts Employees, Risks, Problems, Measures
User profile (Role or Features based)	Supported
Variable context granularity	Supported
Valid context constraints	Supported
Type of formalism	Ontology via OWL
Formality level	OWL-Lite or OWL-DL
Flexibility	Only within the domain
Context construction (Distributed or Centralized)	Centralized
Context reasoning	Future
Context quality monitoring	-
Ambiguity/Incompleteness mgmt.	-
Automatic learning features	Future
Multi-context model	-

Table 1. CoSEEEK context model support

5. Solution Realization

The primary focus of the initial solution realization was sufficient technical validation of the CoSEEEK approach.

5.1 Realization Requirements

For a realization of CoSEEEK, further requirements were elaborated and an extract thereof is listed in simplified form:

- Utilization of OpenUP[5] for SE processes in the PAIS. OpenUP is a lean Unified Process that applies iterative and incremental approaches within a structured SE lifecycle.
- Generate dynamic checklist items based, e.g., on code complexity and test coverage
- Assignment of quality measures:
    - Effort (Low/Med/Hi) can be different conceptual instances with any granularity (e.g., person hours, Low/Med/Hi, or based on a worker formula) and can be made equivalent for a query.
    - Assign an artifact review if quality rules triggered a quality event and the risk is high. If time allows within the iteration, assign an inspection.
    - Assign refactoring as needed
    - Quality measures for the future (list of open measures)
- Remember problems and unfinished measures.
- Report "Top 10" problems periodically.
- Considers person availability
- Suggests proactive measures (for risks) and reactive measures (for problems)
- Developer receives notifications via emforge and mylyn, including checklists
- Web Service-based XML Space implementation with different collections for events, context, and the ontology

Following are some key functional scenarios that incorporate a subset of the above context-aware requirements. The Automatic Assignment Scenario Fig. 5 assigns a qualified worker to a task based on their role, skill level, and current availability. SOC is supported for direct assignment retrieval by a tool.
In the automatic quality measure scenario of Fig. 6, a new problem event is generated and placed in the Space, e.g., by the Rules Processing Agent, whereby the Context Management is notified due to its Space subscription and retrieves and processes the event. SemWeb Integration is used to instantiate a new problem in the ontology. The semantic reasoner is then invoked which has a rule that is triggered when a Problem exists with the status new in the ontology. A countermeasure is chosen based on various criteria and the ontology in the Space is synchronized via SemWeb Integration. Other subscribed agents are notified of an ontology change and may then respond to the new measures, e.g., the rule agent.

[5] http://epf.eclipse.org/wikis/openup/

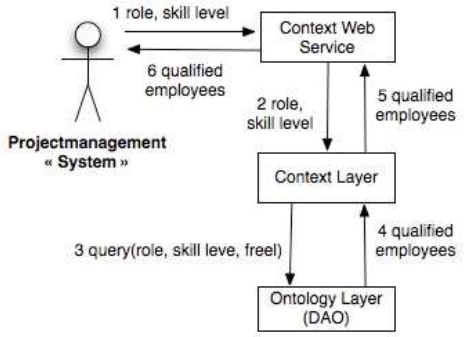

Fig. 5. Context-aware automatic assignment scenario [Req:AutoTask]

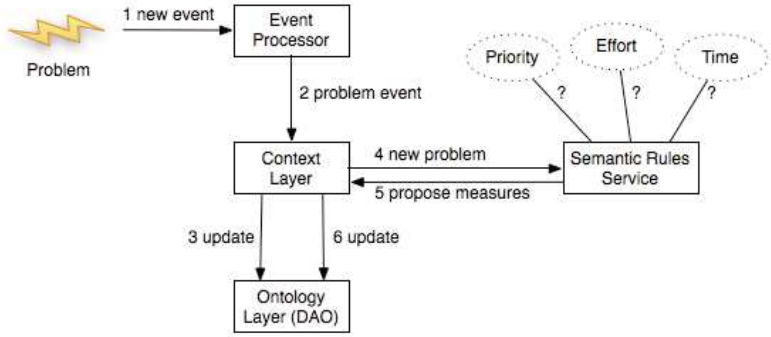

Fig. 6. Context-aware automatic quality measure scenario [Req:AutoQM]

In the scenario of Fig. 7, a list of the top 10 problems may be retrieved via SOC, for display by a tool.

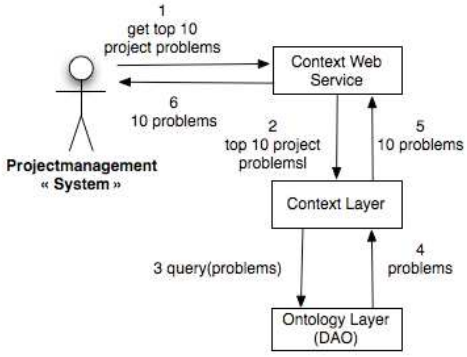

Fig. 7. Context-aware top 10 problems [Req:Top10]

5.2 Implementation Architecture

The CoSEEEK Implementation Architecture is shown in Fig. 8. The *Artifacts* consist of source code and test code. The *SE Tools* consisted of Eclipse as a representative for Integrated Development Environments (IDEs), JUnit[6] to represent SE test tools, Subversion[7] to represent version control systems for artifacts, PMD[8] and Metrics[9] for static analysis tools, and EmForge[10] and Mylyn[11] for task management tools. *Event Extraction* utilized Hackystat sensors (Johnson, 2007) for event extraction, an agent forwarding the events to the XML Space implementation which uses eXist[12] as a storage backend. *Event Processing* utilized Esper[13] for CEP. As an agent platform, WADE (Workflows and Agents Development Environment) was used (Caire et al., 2008). With regard to *Process Management*, ADEPT2 (Dadam et al., 2007) was utilized as PAIS technology. *Rules Processing* was performed by Drools[14]. For *Context Management* and semantic reasoning, the Rete-based inference engine Bossam (Jang & Sohn, 2004) was employed. It supports reasoning over OWL and SWRL ontologies and RuleML rules. *SemWeb Integration* for loading, storing, and synchronizing OWL between the space and the Context Management module was achieved using Java-based Data Access Objects (DAOs) generated using the Protégé ontology editor.

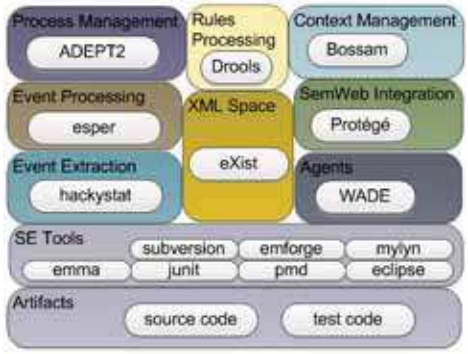

Fig. 8. CoSEEEK implementation architecture

No suitable XML Space implementation was found, thus an XML Space was realized in keeping with SOC using Apache CXF[15] for SOAP-based WS and eXist as a backend. Since in CoSEEEK SBC is used for storage, retrieval, and change notification of key shared data such as events, context, and ontologies, its performance was evaluated in section 6.

[6] http://junit.org
[7] http://subversion.tigris.org
[8] http://pmd.sourceforge.net
[9] http://metrics.sourceforge.net/
[10] http://www.emforge.org
[11] http://www.eclipse.org/mylyn/
[12] http://exist.sourceforge.net/
[13] http://esper.codehaus.org
[14] http://www.jboss.org/drools/
[15] http://cxf.apache.org/

5.3 Leveraging CAC and SWC

To achieve context-awareness, an analysis of common SE concepts was performed and then incorporated into the ontology shown in Fig. 9. The modeling focused on high-value and reusable SE concepts such as activities, problems, risks, and quality measures and practices. Relations to artifacts are used, but artifact details are maintained outside of the ontology. Tools are minimally modeled in their relation to events.

The concept of a Template was introduced for denoting prescribed relationships and properties, e.g., by predefined processes such as OpenUP. A Template contains generic metadata such as preconditions, postconditions, required artifacts, produced artifacts, responsible roles, etc.

For example, an instance of the ActivityTemplate would have this specific metadata for "Design the Solution" activity. Once this activity is actually started, an Activity class is instantiated that is based on this ActivityTemplate (and remains after completion for historical context). This allows a comparison of the actual activity state vs. the prescribed state and allows common problems and risks associated with the activity to be tied to the ActivityTemplate, while real problems are tied to the Activity instance.

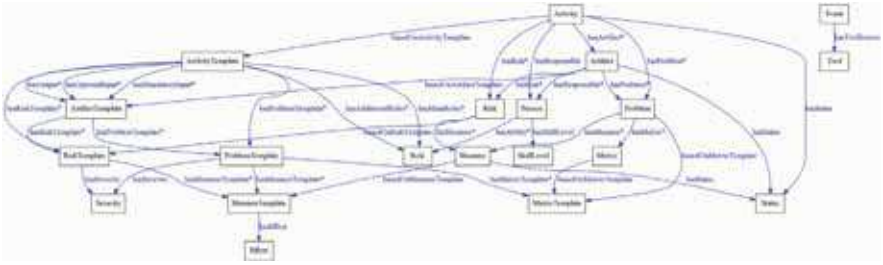

Fig. 9. CoSEEEK implemented ontology

The SemWeb Integration and Context Management design components are shown in Fig. 10, while Fig. 11 shows the dynamic interactions for an activity event with ontology synchronization and context processing.

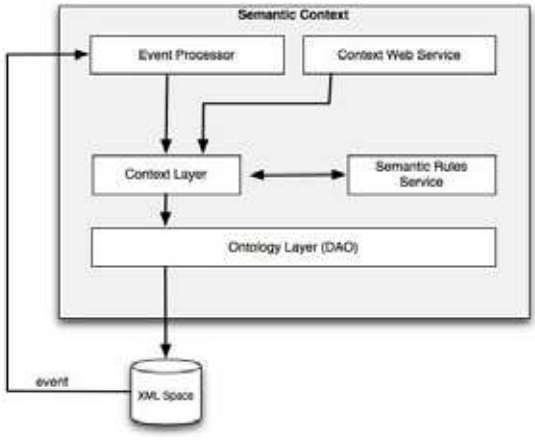

Fig. 10. SemWeb Integration and Context Management

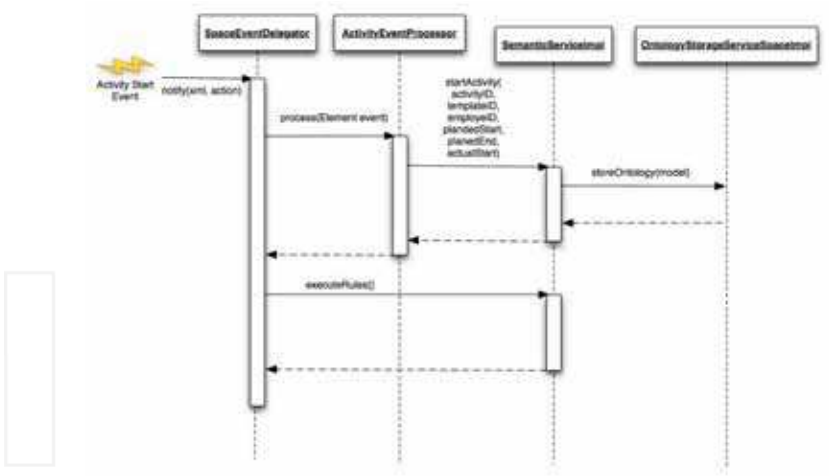

Fig. 11. Activity event ontology update and context processing

Events necessary to support the prescribed level of context-awareness had the following contextual relevance:

Activity Begin and End events:
- Logging
- Relevant for task assignment
- Provides the current context for a person and the project

Artifact Begin and End events:
- Logging
- Relevant for task assignment
- Provides the current context for a person and the project

New Problem event:
- Logging
- Triggers the automatic suggestion of quality measures
- Relevant for the Top 10 problems

Completed Measure event:
- Logging
- Relevant for the Top 10 problems

An example of a semantic reasoner rule in Bossam is shown in Listing 1. This rule ensures that a quality measure is assigned when a new problem is detected.

```
reasoner.tell("rule ruleMeasuresForNewProblems is"
+ " if Problem(?p) and sem:class(?o)"
+ " and hasProblemStatus(?p, ?status) and
[?status=problemStatus_New]"
+ " and hasProblem(?thing, ?p)"
+ " then sem:assignMeasureForNewProblem(?o, ?p, ?thing)");
```

Listing 1. Example context-based quality measure assignment rule

5.4 Leveraging rule-based computing

The rules engine Drools was used to address areas where semantic-agnostic rules excel, e.g., to assess the quality of artifacts or to generate dynamic checklists. The rules engine does not have direct access to Context Management, but rather to non-SWC context that is available to all agents in the XML Space. This separation of responsibilities allows context-centric processing not to be burdened with tool-specific and lower-level quality processing functionality, and thus permit the focus on ontology-centric higher-level intricate context and relation-centric support via SWC. This is the essence of the hybrid SWC approach in CoSEEEK, leveraging SWC technologies for common and high-value SE areas as seen in the ontology of Fig. 9. Benefits include lower maintenance and training costs compared to comprehensive SWC.

For the realization, events generated by Hackystat sensors due to invocation of the PMD and Metrics tools causes the rule agent to take the output of the tools in XML form and transform them via XSLT into an intermediate simplified XML form for parsing by the rule engine. Any negative results cause the artifact context in the space to be adjusted with the actual quality problems, see Fig. 13 for an example.

```
- <context category="Artifact" id="artifact_3" type="Code">
  - <recipients>
      <recipient email="foo.bar@my.com" language="en"/>
      <recipient email="blubb.bar@my.com" language="de"/>
    </recipients>
  - <q-evaluation>
      <risk level="medium"/>
    - <q-measures>
        <q-measure measure="measureTemplate_Review" status="proposed"/>
      </q-measures>
    - <q-checklist>
        <q-checklist-item name="EFGH"/>
        <q-checklist-item name="CCUI"/>
        <q-checklist-item name="CCMN"/>
      </q-checklist>
      <testingNeeded level="Low"/>
      <size>358</size>
    </q-evaluation>
</context>
```

Fig. 13. Artifact context

The rules primarily determine quality problems, and during checklist generation a map of quality problems to checklist items is referenced with a set of locale-specific checklist items serving as a basis for the natural language checklist items. Quality measures assigned by Context Management are also incorporated when these are not already addressed by task assignment, e.g., assigning refactoring and testing tasks. A dynamic customized checklist is generated in XHTML and a link sent to the software engineer as a task via emforge in order to address the quality issues, see Fig. 14. When the engineer processes the checklist, the XML is archived in a collection in the XML Space.

Fig. 14. Dynamically generated XHTML checklist screenshot

The choice of checklist items is dynamically generated, temporally context-dependent and containing items deemed applicable. The software engineer is thus not bothered by irrelevant checklist issues, and as such perceives the SEE as context-aware in regard to quality management behavior [Req:AutoQM][Req:BlackBox].

5.5 Leveraging Process-Awareness

For SEE process support, OpenUP processes were mapped to ADEPT2 process templates as illustrated in the figure below.

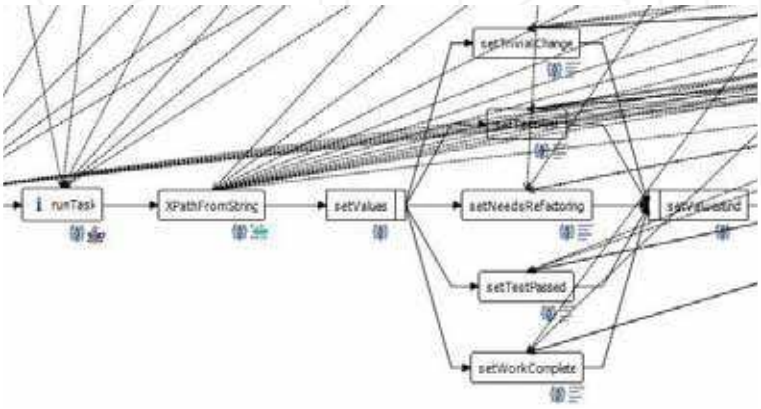

Fig. 15. Start Task Template

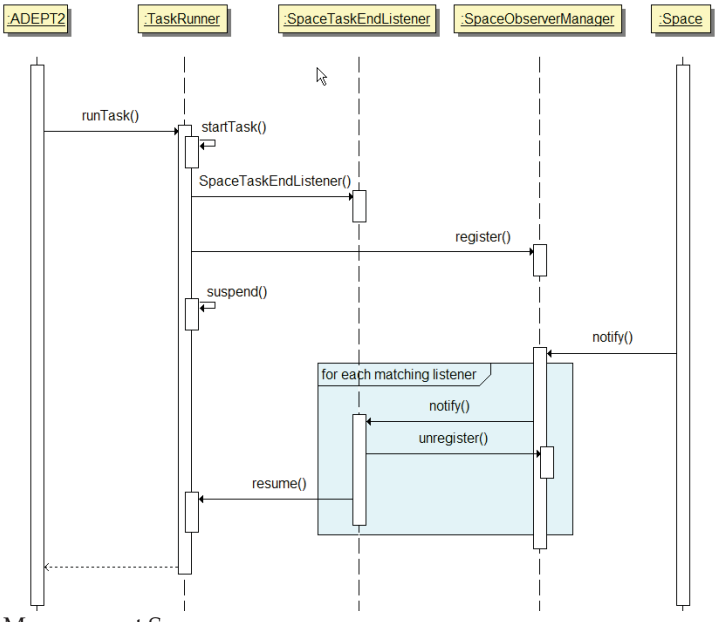

Fig. 16. Task Management Sequence

The ADEPT2 process templates required an integration mechanism to place new tasks events in the Space for retrieval and processing by the EmForge task management agent, and then wait for their notification of task completion. A component TaskRunner was developed that provides two methods: runTask which starts a new task and waits for the end event in the Space, and the method stopTask which removes an open task. A sequence diagram for runTask is shown in the figure below.
The software engineer perceives changes in the IDE, in this case Mylyn displays the emforge task list as shown below. Any changes to the tasks or usage of tools by the software engineer generate events that adjust the context and possibly process and may result in task adjustments. Since CoSEEEK is aware of all open tasks, and to allow for assignment flexibility and maintain focus, the software engineer typically sees only the current task and optionally next one (to allow for mental preparation and reduce surprises).

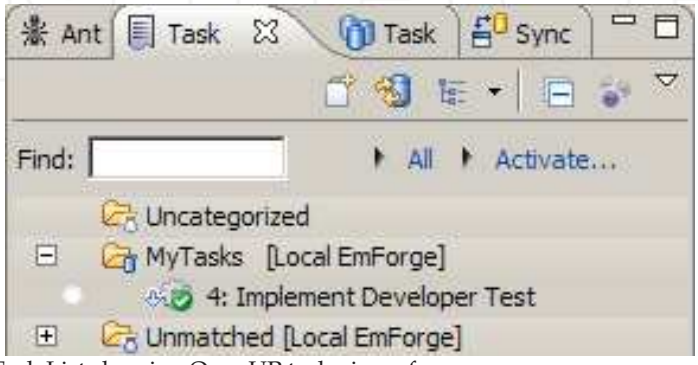

Fig. 17. Mylyn Task List showing OpenUP task via emforge

6. Results

The preliminary results for this research focused on the viability of the CoSEEEK approach, with the realization meeting its functional scenario goals, general requirements, and detailed realization requirements. As to any performance and scalability limitations, the performance evaluation principally has SWC and CAC limitations in view. Due to the centricity and dependency on the new SBC implementation, sufficient performance in this area was also verified.

6.1 SBC Performance

For measuring the SBC performance of the CoSEEEK XML Space implementation, the configuration consisted of 10 AMD Opteron 180 Dual Core 2,4 GHz 3GB PCs running Windows XP Pro SP2, Java JDK 1.6.0_06, and Apache CXF 02.01.03 on a 100MBit/s Ethernet network. 1 PC was configured as the Space server with eXist 1.2.5-rev8668 as a backend. For the read or write operations, up to 8 PCs (each with a client) were used as space clients all performing the same operation. For notifications, 1 PC wrote 1000 events in the space and up to 8 clients received notifications. The measurements were repeated 3 times and the averages presented in the following tables.

Clients	Write time (ms)	% increase	Notification time (ms)	% increase
1	14.0	-	14.7	-
2	14.1	0%	14.4	-2%
4	15.0	7%	15.0	4%
8	17.0	13%	17.0	13%

Table 2. Average notification performance for 1000 events

Clients	Write time (ms)	% increase
1	13.9	-
2	16.3	18%
4	23.0	41%
8	61.7	168%

Table 3. Average write performance for 1000 events

Clients	Read time (ms)	% increase
1	10.7	-
2	9.5	-11%
4	12.4	31%
8	41.8	237%

Table 4. Average read performance for 1000 events

Table 2 shows that notification performance was not significantly affected by an increase in peers, and the scalability is sufficient for SEE purposes. The results for writing into the space in Table 3 show the bottleneck effect of the chosen data consistency limitation of allowing only sequential writes. Since the space is used primarily as a blackboard coordination mechanism, such heavy parallel writing by multiple agents is not expected. Table 4 shows the read performance, but again due to the chosen data consistency mechanism, the reads

are synchronized to wait until the write completes, and vice versa. If this becomes a serious bottleneck in industrial settings, various optimizations are possible.

6.2 SWC and CAC Performance

For measuring the semantic reasoner performance and scalability, the test configuration consisted of an Intel Core 2 Duo 2,0 GHz PC running Mac OS X 10.5.6, 2 GB RAM, Java JRE 1.5.0, Bossam 0.9b45, and Protégé 3.3.1. The averages from 3 measurements are shown in the following tables.

```
resultPossibleEmployee = reasoner.ask("query q is Person(?employee)"
+ " and available(?employee, 1)"
+ " and hasAbility(?employee, " + role + ")"
+ " and hasSkillLevel(?employee," + skillLevel + ");");
```

Listing 2. Example context-based task assignment rule

Person instances	Time (ms)	Factor increase in time
10	60	-
100	83	1.4
1000	274	3.3
10000	7397	27.0

Table 5. Query performance vs. person instances

```
resultArtifact=reasoner.ask("query q is Artifact(?artifact)"
+ " and basedOnArtifactTemplate(?artifact, ?template);");
```

Listing 3. Artifact rule

Artifact instances	Time (ms)	Factor increase in time	Factor increase in artifacts
10	55	-	
100	62	1.1	10
1000	137	2.2	10
10000	1578	11.5	10
20000	3936	2.5	2
40000	10793	2.7	2

Table 6. Query performance vs. artifact instances

In the person query, the effect of the multiple conditions shows a larger impact to the performance than for the artifact query for the cases of 100 and 1000 instances. Since overall time was measured, in both cases the jump from 1000 to 10000 instances caused the time to jump from the millisecond range to seconds, at which point operating system multitasking and scheduling may play a more significant factor and account for the anomaly in the factor

increases in time. The 20000 and 40000 artifact instance measurements show that this remains consistent thereafter for larger sets.
Since the current usage of the reasoner and context-awareness is supportive and requires no hard real-time user or system response latencies, these results show that current usage of such an approach for typical SE project sizes with typical SE IT hardware is feasible.

7. Conclusion

To deal with the current and coming challenges facing SE, new approaches for SEEs are needed that coalesce the heterogeneous and distributed tool data and events that occur, contextualize them, and then proact or react to provide software engineers with a cohesive and improved SEEs. While semantic web computing is an obvious candidate, due to the uniqueness of each project context, the advantages of must be weighed against some of the difficulties and the investments required, and a pragmatic hybrid approach may be reasonable in the interim.
The CoSEEEK approach, with its synthesis of various computing paradigms, provides advantages for addressing this situation. The semantic meanings of a common subset of the key concepts are used to adjust quality measures, project task assignments, or workflows based on events within a shared agent-accessible context. Combined with a semantic reasoner, context-aware proactive and reactive behaviors that can improve the effectiveness and efficiency of SEEs are exhibited. The reduced ontology focus avoids the perhaps unjustifiable time and resource investments in SWC that a comprehensive integration would require for the tool, artifact, and data models in an SEE along with their continuous changes, while leveraging the visible benefits in responsiveness of the SEE. Context-aware behavior is perceived by humans via task assignments and dynamically generated checklists.
The event processing flow coupled with the solution architecture provides flexibility and loose-coupling in the processing of events. The current CoSEEEK ontology and context model supported the SE scenarios, and additional ontologies can be incorporated for expanded reasoning if desired. As a side note, by utilizing EDA, RBC, and PAIS, no sequential process logic was needed to support the scenarios, which furthers flexibility and potential reuse and is a key for the adaptability of the infrastructure in ever-changing SEEs. The results validated the current technical feasibility and potential benefits that the CoSEEEK approach can bring to SEEs. Future work includes planned empirical studies of CoSEEEK in SE industrial settings.

8. Acknowledgements

The author thanks Tobias Gaisbauer, Michael Vögeli, and Daniel Sebel for their assistance with the experiments, implementation, and figures.

9. References

Adams, M.; ter Hofstede, A.H. M.; Edmond, D. & van der Aalst, W. M. P. (2006). Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows, In: *Lecture Notes in Computer Science*, Vol. 4275, Springer Berlin / Heidelberg, ISSN 0302-9743

Adi, A.; Borzer, D. & Etzion, O. (2000). Semantic Event Model and its Implication on Situation Detection. In *Proceedings of the Eighth European Conference on Information Systems* (Hansen HR, Bichler M, Mahrer H eds.), Vienna, pp. 320-325

Alonso, G.; Casati, F.; Kuno, H. & Machiraju, V. (2003). *Web Services – Concepts, Architectures and Applications*, Springer Verlag, Berlin

Arbaoui, S.; Derniame, J.; Oquendo, F. & Verjus, H (2002). A Comparative Review of Process-Centered Software Engineering Environments, In: *Annals of Software Engineering*, Vol. 14, Issue 1-4 (December 2002), J. C. Baltzer AG, Science Publishers Red Bank, NJ, USA, ISSN:1022-7091

Bardram, J. (2005). The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications, in *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, Vol. 3468/2005, ISBN 978-3-540-26008-0

Berners-Lee, T.; Hendler, J. & Lassila, O. (2001). The Semantic Web, *Scientific American*, May 2001, pp. 28-37

Bolchini, C.; Curino, C. A.; Quintarelli, E.; Schreiber, F. A. & Tanca, L. (2007). A data-oriented survey of context models. SIGMOD Rec. 36, 4 (Dec. 2007), pp. 19-26

Bontcheva, K. & Sabou, M. (2006). Learning Ontologies from Software Artifacts: Exploring and Combining Multiple Sources, In: *Proceedings of 2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006)*

Bussler, C.; Kilgarriff, E.; Krummenacher, R.; Martin-Recuerda, F.; Toma, I. & and Sapkota, B. (2005). WSMX Triple-Space Computing, http://wsmo.org/TR/d21/v0.1, June 2005, D21 v0.1

Caire, G.; Gotta, D. & Banzi, M. (2008). WADE: a software platform to develop mission critical applications exploiting agents and workflows. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track*, pp. 29-36.

Calero, C.; Ruiz, F. & Piattini, M. (Eds.) (2006). *Ontologies for Software Engineering and Software Technology*, ISBN: 978-3-540-34517-6

Christopoulou, E.; Goumopoulos, C. & Kameas, A. (2005). An ontology-based context management and reasoning process for UbiComp applications, In: *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient intelligence: innovative Context-Aware Services: Usages and Technologies* (Grenoble, France, October 12 - 14, 2005). sOc-EUSAI '05, vol. 121. ACM, New York, NY, pp. 265-270

Dadam, P.; Reichert, M.; Rinderle, S.; Jurisch, M.; Acker, H.; Göser, K.; Kreher, U. & Lauer, M. (2007). ADEPT2 - Next Generation Process Management Technology. *Heidelberger Innovationsforum*, Heidelberg, April 2007

Dey, A. & Abowd, G. (2000). The Context Toolkit: Aiding the Development of Context-Aware Applications, in *Proceedings of the Workshop on Software Engineering for Wearable and Pervasive Computing*, Limerick, Ireland

Dinger, U.; Oberhauser, R. & Reichel, C. (2006). SWS-ASE: Leveraging Web Service-based Software Engineering, In: *Proceedings of the International Conference on Software Engineering Advances (ICSEA'06)*, IEEE Computer Society Press.

Ferscha, A.; Hechinger, M.; Mayrhofer, R.; dos Santos Rocha, M.; Franz, M.; and Oberhauser, R. (2004). Digital Aura, In: *Advances in Pervasive Computing*, Vol. 176, Austrian Computer Society (OCG), pp. 405-410

Forgy, C. (1982). "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", *Artificial Intelligence*, 19, pp 17–37, 1982.

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

Gruhn, V. (2002). Process-Centered Software Engineering Environments, A Brief History and Future Challenges, In: *Annals of Software Engineering*, Springer Netherlands, Vol. 14, Numbers 1-4 / (December 2002), J. C. Baltzer AG, Science Publishers Red Bank, NJ, USA. ISSN:1022-7091, pp. 363-382

Happel, H. & Seedorf, S. (2006). Applications of Ontologies in Software Engineering, In: *Proceedings of the Workshop on Semantic Web Enabled Software Engineering (SWESE)* at the 5th International Semantic Web Conference (ISWC 2006)

Hayes-Roth, B. (1985). A blackboard architecture for control, *Artificial Intelligence*, Volume 26, Issue 3 (July 1985), pp. 251-321.

Henderson-Sellers, B. (2002). Process Metamodelling and Process Construction: Examples Using the OPEN Process Framework (OPF), In: *Annals of Software Engineering*, Vol. 14, Issue 1-4, (December 2002), ISSN:1022-7091, pp. 341–362

Horrocks, I.; Patel-Schneider, P.; McGuinness, D. & Welty, C. (2007). OWL: a Description Logic Based Ontology Language for the Semantic Web. In: *The Description Logic Handbook: Theory, Implementation, and Applications (2nd Edition)*, chapter 14. Cambridge University Press

Jang, M. & Sohn, J. (2004). Bossam: an extended rule engine for OWL Inferencing, *Proceedings of RuleML 2004* (LNCS Vol. 3323), Nov. 8, 2004

Johnson, P. (2007). Requirement and Design Trade-offs in Hackystat: An in-process software engineering measurement and analysis system, *Proceedings of the 2007 International Symposium on Empirical Software Engineering and Measurement*, Madrid, Spain, September, 2007.

Khushraj, D.; Lassila, O. & Finin, T. (2004). sTuples: Semantic Tuple Spaces, In: Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), pp. 268-277

Koenig, S. (2003). Integrated Process and Knowledge Management for Product Definition, Development and Delivery, In: *Proceedings of the IEEE International Conference on Software-Science, Technology & Engineering (SWSTE)*, pp.133

Koenig, Shai (2003). "Integrated Process and Knowledge Management for Product Definition, Development and Delivery," swste, pp. 133, *IEEE International Conference on Software-Science, Technology & Engineering*

Luckham, D. (2002). *The Power of Events - An Introduction to Complex Event Processing in Distributed Enterprise Systems*, Addison-Wesley, ISBN 0-201-72789-7

Müller, R.; Greiner, U. & Rahm, E. (2004). AgentWork: A workflow system supporting rule-based workflow adaptation. *Data and Knowledge Engineering*, 51 (2), pp. 223-256

Oberhauser, R. & Schmidt, R. (2007). Towards a Holistic Integration of Software Lifecycle Processes using the Semantic Web, In: *Proceedings of the 2nd International Conference on Software and Data Technologies (ICSOFT 2007)*, Vol. 3 - Information Systems and Data Management, 2007, pp. 137-144

Oberle, D. (2006). *Semantic Management of Middleware*, The Semantic Web and Beyond, Vol. 1, Springer, New York, ISBN 0387276300

Pesic, M.; Schonenberg, M.; Sidorova, N. & van der Aalst, W. (2007). Constraint-based workflow models: change made easy. In: *Proceedings of the 15th Int'l Conf. on Cooperative Information Systems (CoopIS'07)*, Vilamoura, Algarve, Portugal, LNCS 4803, pp. 77-94

Reichert, M. & Dadam, P. (1997). A framework for dynamic changes in workflow management systems. In: *Proc. 8th Int'l Workshop on Database and Expert Systems Applications*, Toulouse, pp. 42–48.

Schmidt, A. (2003). Ubiquitous Computing - Computing in Context, PhD dissertation, Lancaster University, U.K.

Simperl, E.; Krummenacher, R. & Nixon, L. (2007). A coordination model for triplespace computing, *Proceedings of the 9th International Conference on Coordination Models and Languages (Coordination)*, Springer Verlag, June 2007

Tolksdorf, R.; Bontas, E. P. & Nixon, L. J. (2005). Towards a Tuplespace-Based Middleware for the Semantic Web, *Proceedings of the 2005 IEEE/WIC/ACM international Conference on Web intelligence*, (September 19 - 22, 2005). Web Intelligence. IEEE Computer Society, Washington, DC, pp. 338-344

Tolksdorf, R.; Nixon, L.; Bontas, E. P.; Nguyen, D. M. & Liebsch, F. (2005a). Enabling real world Semantic Web applications through a coordination middleware, *Proceedings of the 2nd European Semantic Web Conf.* ESWC'05, 2005

Van der Aalst, W. & van Hee, K. (2002). Workflow management: models, methods, and systems, MIT Press.

INTECH

Semantic Web
Edited by Gang Wu

ISBN 978-953-7619-54-1
Hard cover, 310 pages
Publisher InTech
Published online 01, January, 2010
Published in print edition January, 2010

Having lived with the World Wide Web for twenty years, surfing the Web becomes a way of our life that cannot be separated. From latest news, photo sharing, social activities, to research collaborations and even commercial activities and government affairs, almost all kinds of information are available and processible via the Web. While people are appreciating the great invention, the father of the Web, Sir Tim Berners-Lee, has started the plan for the next generation of the Web, the Semantic Web. Unlike the Web that was originally designed for reading, the Semantic Web aims at a more intelligent Web severing machines as well as people. The idea behind it is simple: machines can automatically process or “understand” the information, if explicit meanings are given to it. In this way, it facilitates sharing and reuse of data across applications, enterprises, and communities. According to the organisation of the book, the intended readers may come from two groups, i.e. those whose interests include Semantic Web and want to catch on the state-of-the-art research progress in this field; and those who urgently need or just intend to seek help from the Semantic Web. In this sense, readers are not limited to the computer science. Everyone is welcome to find their possible intersection of the Semantic Web.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Roy Oberhauser (2010). Leveraging Semantic Web Computing for Context-Aware Software Engineering Environments, Semantic Web, Gang Wu (Ed.), ISBN: 978-953-7619-54-1, InTech, Available from: http://www.intechopen.com/books/semantic-web/leveraging-semantic-web-computing-for-context-aware-software-engineering-environments

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821