Automated Software Engineering Process Assessment: Supporting Diverse Models using an Ontology

Gregor Grambow and Roy Oberhauser
Computer Science Dept.
Aalen University
Aalen, Germany
{gregor.grambow, roy.oberhauser}@htw-aalen.de

Manfred Reichert
Institute for Databases and Information Systems
Ulm University
Ulm, Germany
manfred.reichert@uni-ulm.de

***Abstract*—Current software engineering process assessment reference models rely primarily on manual acquisition of evidence of practices. This manually collected data is then correlated with expected model attributes to assess compliance. Such manual data acquisition is inefficient and error-prone, and any assessment feedback is temporally detached from the original context by months or years. Yet in order to automate the process data acquisition and assessment, one is confronted with various challenges that such diverse project-specific software engineering environments involve. This paper presents an ontology-based approach for enhancing the degree of automation in current process assessment while simultaneously supporting diverse process assessment reference models (CMMI, ISO/IEC 15504, ISO 9001). It also provides an in-the-loop automated process assessment capability that can help software engineers receive immediate feedback on process issues. The evaluation showed the approach's technical feasibility, model diversifiability across various process assessment models (CMMI, ISO/IEC 15504, ISO 9001), and suitable performance and scalability. The approach can reduce the effort required to determine process compliance, maturity, or improvement, and can provide more timely and precise feedback compared to current manual process assessment methods and tools.**

Keywords-software engineering process assessment tooling; semantic technology; Capability Maturity Model Integration; ISO/IEC 15504; ISO 9000

I. INTRODUCTION

This article extends our previous work in [1]. Processes - be they technical, managerial, or quality processes, are an inherent part of software engineering (SE), and subsequently so is process assessment and process improvement [2]. Software process improvement typically involves some assessment, and common reference model assessment standards utilize external audits (CMMI [3], ISO 15504 [4], and ISO 9001 [5]) that are performed manually to gather compliance evidence. Often the maturity of software organizations is assessed based primarily on their process-orientation and correlation of processes to a reference model.

If SE processes were supported or enacted by process-aware information systems (PAIS), then the efficiency of data acquisition and analysis for process assessment could also be improved. One prerequisite - the adoption and use of automated process enactment support is relatively rare in SE projects. This can be attributed to a number of factors: (1) software development projects face a high degree of new and changing technological dependencies (typically impacting project tool environments, knowledge management, process integration, and process data acquisition); (2) significant process modeling effort is necessary and PAIS usage has been somewhat restrictive [6]; (3) SE processes are knowledge processes [7], so that the exact operational determination and sequencing of tasks and activities is not readily foreknown; and (4) most current process models are too inflexible to mirror such detailed operational dynamics.

We developed the Context-aware Software Engineering Environment Event-driven frameworK (CoSEEEK) [8] to improve SE process support and guidance in an automated fashion. That way, enhanced support features are possible, such as automatically gathering information from the environment and users, uniting it with information from a knowledge base, and utilizing this information for on-the-fly process optimization (Section IIIC provides more information on CoSEEEK). Given such a context-aware event-driven automated process guidance system, we investigated the feasibility of enabling in-the-loop automated process assessment support. Our ontology-based approach semantically enhances a PAIS for SE operational process enactment and assessment support.

The paper is organized as follows: Section II describes the attributes of three common SE process reference models used in later sections. Section III describes general requirements and the solution approach for automated process assessment. An evaluation of this approach is described in Section IV. Section V positions related work relative to the solution approach. Section VII concludes the paper.

II. PROCESS ASSESSMENT MODELS

Three of the most mature and prevalent process assessment approaches used in software projects (CMMI, ISO/IEC 15504 / SPICE, and ISO 9001) are described in order to later show how automation was achieved. Despite the differences, with ISO 9000 being more of a requirement model and CMMI and SPICE meta-process models, they are similarly used for assessing process compliance or maturity. All three models have several basic concepts in common:

They define basic activities to be executed in a project such as 'Identify Configuration Items' for configuration management. (These will be mapped by a concept called *base practice* in our approach.) These activities are grouped together (e.g., 'Establish Baselines' in the configuration management example, with these groupings being mapped by a concept called *process* in our approach.) In turn, the latter are further grouped (e.g., 'Configuration Management') to allow further structuring. (This will be mapped by a concept called *process category* in our approach.) To be able to rate these practices and processes, the assessment models feature a *performance scale* to quantify the assessment. Finally, most models use the quantified assessments to assign *capability levels* to processes.

A. CMMI

CMMI (Capability Maturity Model Integration) [3] is one of the most widely used assessment models. It exists in different constellations, from which CMMI-DEV (CMMI for Development) is utilized in our context. The CMMI staged representation model comprises five *maturity levels* (1-'Initial', 2-'Managed', 3-'Defined', 4-'Quantitatively Managed', 5-'Optimizing'). The levels indicate 'Degree of process improvement across a predefined set of process areas, in which all goals within the set are attained' (cf. [3]). To implement this, each of the levels has subordinate activities that are organized as follows: A maturity level (e.g., '2') has *process categories* (e.g., 'Support') that have *process areas* (e.g., 'Configuration Management') that have *specific goals* (e.g., 'Establish Baselines') that finally have specific practices (e.g., 'Identify Configuration Items'). To illustrate the CMMI, the maturity levels, categories, and areas are shown in the following table:

To quantify the assessment, CMMI has a performance scale (1-'unrated', 2-'not applicable', 3-'unsatisfied', 4-'satisfied'). Using these concepts, process assessment is applied as follows:

- Rate each generic and specific goal of a process area using the introduced performance scale.
- A maturity level is achieved if all process areas within the level and within each lower level are either 2 or 4 (cf. the performance scale introduced).

In addition to these concrete activities and maturity levels, CMMI features *generic goals* (e.g., 'Institutionalize a Managed Process') with *generic practices* (e.g., 'Control Work Products'). These are subordinate to capability levels (0-'Incomplete', 1-'Performed', 2-'Managed', 3-'Defined', 4-'Quantitatively Managed', 5-'Optimizing'). The latter indicate 'Achievement of process improvement within an individual process area' (cf. [3]). SCAMPI (Standard CMMI Appraisal Method for Process Improvement) [9] is the official CMMI appraisal method. It collects and characterizes findings in a Practice Implementation Indicator Database. According to SCAMPI, there is a direct relationship between specific and generic goals (SG and GG), which are required model components, and the specific and generic practices (SP and GP), which are expected model components. Satisfaction of the goals is determined by a detailed investigation, and alternative practices could be implemented that are equally effective in achieving the intent of the goals.

TABLE I. CMMI

Mat. Level	Category	Process Area
2	Support	Configuration Management (CM/SCM)
2	Support	Measurement and Analysis (MA)
2	Project Man.	Project Monitoring and Control (PMC)
2	Project Man.	Project Planning (PP)
2	Support	Process and Product Quality Assurance (PPQA)
2	Project Man.	Requirements Management (REQM)
2	Project Man.	Supplier Agreement Management (SAM)
3	Support	Decision Analysis and Resolution (DAR)
3	Project Man.	Integrated Project Management (IPM)
3	Process Man.	Organizational Process Definition (OPD)
3	Process Man.	Organizational Process Focus (OPF)
3	Process Man.	Organizational Training (OT)
3	Engineering	Product Integration (PI)
3	Engineering	Requirements Development (RD)
3	Project Man.	Risk Management (RSKM)
3	Engineering	Technical Solution (TS)
3	Engineering	Validation (VAL)
3	Engineering	Verification (VER)
4	Process Man.	Organizational Process Performance (OPP)
4	Project Man.	Quantitative Project Management (QPM)
5	Support	Causal Analysis and Resolution (CAR)
5	Process Man.	Organizational Performance Management (OPM)

B. ISO/IEC 15504 (SPICE)

The SPICE (Software Process Improvement and Capability Determination) [4][10] model is an international standard for measuring process performance. It originated from the process lifecycle standard ISO/IEC 12207 [11] and maturity models such as CMM (the predecessor of CMMI). SPICE comprises six *capability levels* (0-'Incomplete process', 1-'Performed process', 2-'Managed process', 3-'Established process', 4-'Predictable process', 5-'Optimizing process'). Each of the latter has one or multiple *process attributes* (e.g., '2.1 Performance Management'). In the following table, the capability levels and process attributes are shown:

A process reference model was included in the initial version of the standard. This was later removed to support different process models (or the ISO/IEC 12207). Thus, mappings to various process models are possible. In this paper, the examples use the initial process model specifications for illustration. These comprised *process categories* (e.g., 'Organization') with *processes* (e.g., 'Improve the process') that contained *base practices* (e.g., 'Identify reusable components'). SPICEs measurement model applies the following performance scale for

assessment: 1-'not achieved' (0-15%), 2-'partially achieved' (16% - 50%), 3-'largely achieved' (51% - 85%), and 4-'fully achieved' (86% - 100%).

TABLE II. SPICE

Cap. Level	Name	Process Attribute
0	Incomplete process	-
1	Performed process	Process Performance
2	Managed process	Performance Management Work Product Management
3	Established process	Process Definition Process Deployment
4	Predictable process	Process Measurement Process Control
5	Optimizing process	Process Innovation Process Optimization.

SPICE does not use assessments of practices to directly determine whether an overall capability level is achieved, but uses them to assign to each process one or more capability levels and to use them to recursively calculate assessments for projects and organizations. The assessment comprises the following steps:

- Assess every base practice with respect to each of the process attributes.
- Determine the percentage of base practices of one process that have the same performance scale with respect to one process attribute.
- Assessment of the processes: Assign the capability level for process attributes where all base practices of the process have performance scale 3 or 4 and for all lower capability levels, the same applies with performance scale 4.
- Assessment of a project is done by using the mathematical mean of the ratings of all of its processes.
- Assessment of an organization is done by using the mathematical mean of the ratings of all of its projects.

C. ISO 9001

ISO 9000 comprises a family of standards relating to quality management systems. ISO 9001 [5] deals with the requirements organizations must fulfill to meet the standard. Formal ISO 9001 certifications have gained great importance for organizations worldwide. The ISO 9001 assessment model uses no capability scale; it only determines whether a certain practice is in place. Therefore, a simple performance scale suffices: 0-'not satisfied', 1-'satisfied'. The assessed practices are structured by *process sub-systems* (e.g., 'Organization Management') that contain *main topic areas* (e.g., 'Management responsibility'). In turn, the latter contain *management issues* (e.g., 'Define organization structure'). Based on these concepts, a recursive assessment can be applied rating an organization by its *process sub-systems* and the contained *management issues* with a pass threshold of 100%. Our approach is targeted at creating more quality awareness in companies, not at replacing or conducting formal reviews. Therefore, the standard ISO 19001:2011 (Guidelines for auditing management systems) [12] is not taken into account here.

D. Summary

As shown by these three assessment models, the approaches to process assessment differ significantly. This applies for the concepts utilized as well as for the applied procedures: For example, CMMI knows two different types of levels that have subordinate activities. For ISO/IEC 15504, the levels have certain attributes that serve to assess all existing practices. As opposed to the two other models, ISO 9001 does not apply levels or different performance scales. These differences hamper convergence to a unified model or approach and present the primary technical challenge.

III. AUTOMATED PROCESS ASSESSMENT

This section describes the approach taken to provide automated process assessment including the requirements elicited for such an approach, application concept, conceptual framework, and procedure applied. The approach extends and annotates process management concepts, enhancing them with additional information required for assessment. The aim of our approach is not to replace manual ratings of processes conducted by humans or to be used in formal process audits. It shall rather contribute to the quality awareness of a company and provide information on the current state of the process as it is executed. Therefore, our approach, despite adding automated rating facilities, still integrates and relies on manual ratings or confirmations for ratings.

A. Requirements

This sub-section briefly elicits basic requirements for providing automated integration of process assessment and process execution facilities into SE project environments. These requirements are:

- R:Proc: If a system aims at automatically integrating assessments with the executed process, the first basic requirements is that the system must be aware of the process and be able to govern the latter.
- R:Cntx: To be able to not only be aware of the planned process, but also integrate with the real operational process as it is enacted by SE project members, facilities must be in place the provide usable context information to the system.
- R:MultModel: To be able to provide flexible support for diverse projects and organizations, the assessment approach should not be tied to a single assessment model. It should support implementation and customization of various models.
- R:Integrate: An automated assessment approach should not produce much additional effort or disturb users in their normal work. Therefore, the assessment facilities should integrate seamlessly with normal process execution.

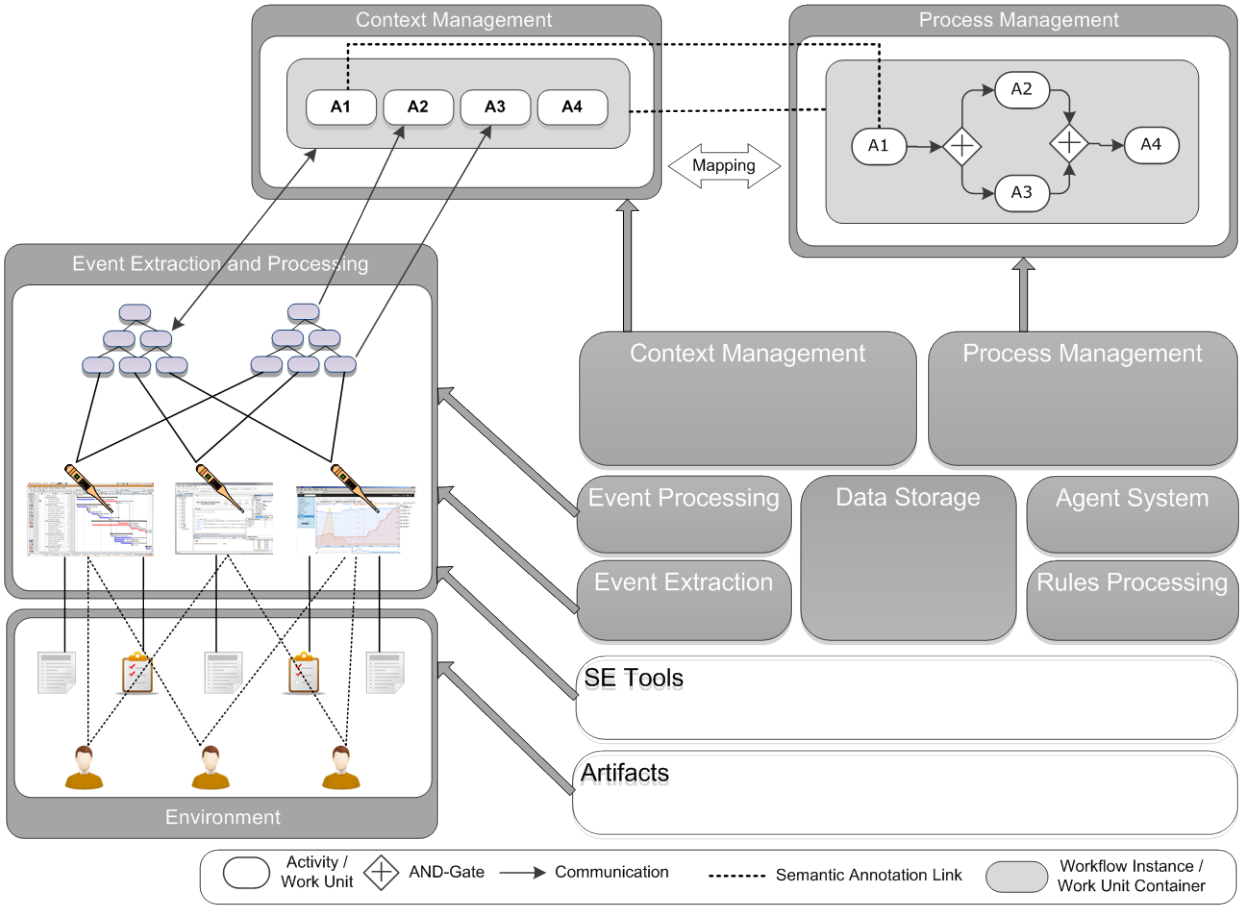

Figure 1. Application concept for automated process assessment.

- R:Auto: To avoid unnecessarily burdening users, an automated assessment approach should enable automate ratings to the degree feasible . However, it must also incorporate facilities for humans to interfere or override automated ratings.

B. *Concept for Application*

As aforementioned, to be able to integrate process assessment tightly into the software developments process and everyday work in SE projects, our approach is realized within the CoSEEEK framework [8]. The latter provides holistic process and project support by integrating various technologies for context awareness and management as well as dynamic and adaptive process management. The different components of the framework and the contextual integration capabilities are illustrated in Figure 1.

The different components of the framework are loosely coupled and feature reactive event-based communication via the central *Data Storage* component. As the framework shall be context-aware, a way of acquiring context data is necessary. In an SE project, context consists mostly of different actors that use SE tools to manipulate a variety of SE artifacts. To gain an awareness of these, the following approach is taken: The *Event Extraction* component of the framework features a set of sensors that are integrated into various SE tools. These sensors generate events as users change the states of various artifacts. As these events are of rather atomic nature, the *Event Processing* component aggregates them to derive higher-level events that contain more semantic value.

To be able to utilize contextual knowledge directly for process guidance, the *Process Management* and *Context Management* components work tightly together: The former enables the dynamic implementation and execution of processes (cf. requirement R:Proc) while the latter stores, processes and evaluates the context information (cf. requirement R:Cntx) using an ontology and reasoning. Furthermore, it encapsulates the *Process Management* from the other components. Thus, all process communication is routed over the *Context Management* component, which enhances it with context information and utilizes it to adjust the process execution.

To enable a reasonable level of dynamicity and automation, CoSEEEK also features to further components: The *Rules Processing* component enables the flexible and simple definition and execution of certain automatisms as rules. The *Agent System* component enables CoSEEEK to react to different dynamic situations in which different conflicting goals have to be evaluated and decisions made.

Based on these components, CoSEEEK provides a variety of different functionalities that support different aspects of automated process and project support for SE projects:

- Quality Management: CoSEEEK enhances the automated detection of quality problems in the source code by facilitating the automated assignment of software quality measures to counteract these problems. The measures are seamlessly integrated into users' workflows to minimize user disturbance and maximize efficiency. For further reading on this topic, see [13].
- Knowledge Management: CoSEEEK enables the collection and management of knowledge in SE projects. This information is semantically enhanced, and thus CoSEEEK can automatically provide the appropriate knowledge to the users at the appropriate point in the process. For further reading on this topic, see [14].
- Exception Handling: CoSEEEK enables a flexible and generic exception handling approach that is capable of detecting various exceptions related to activities as well as artifacts. Furthermore, the appropriate exception handling, the responsible person for applying that handling, and the appropriate point in the process to apply it can be automatically determined. For further reading on this topic, see [15].
- Task Coordination: CoSEEEK features the ability to automatically coordinate activities of different areas of a SE project. This comprises the automatic notification of users in case of certain changes to artifacts or activities as well as the automatic issuing of follow-up actions required by other actions or changes. For further reading on this topic, see [16].
- Extended process support: CoSEEEK incorporates facilities to implement a greater coverage of activities carried out in SE projects as SE process models. Many dynamic activities and workflows that are not covered by the models can be modeled and executed, featuring a suitable simple modeling and transformation facility. For further reading on this topic, see [17].

C. Conceptual Framework

To achieve extended assessment functionality, process management concepts were enhanced. These are defined in the *Context Management* component and are associated with a *Process Management* component that manages process execution. Thus, assessment concepts can be tightly and seamlessly integrated with process execution (cf. requirement R:Integrate). Figure 2 shows a simple workflow in the *Process Management* component: This workflow is defined by 'Workflow Template 1' that contains four activity templates. Both of these concepts are mirrored in the *Context Management* component by the *Work Unit Container Template* that contains *Work Unit Templates*. When the workflow is to be executed, it is instantiated in the *Process Management* component and then represented by a workflow instance ('Workflow Instance 1') containing the activities to be processed. These two concepts are again mirrored in the *Context Management* component by the *Work Unit Container* that contains *Work Units*. These have explicitly defined states that are automatically synchronized with the states in the *Process Management* component. That way, the *Context Management* component is aware of the current execution state of workflows and activities.

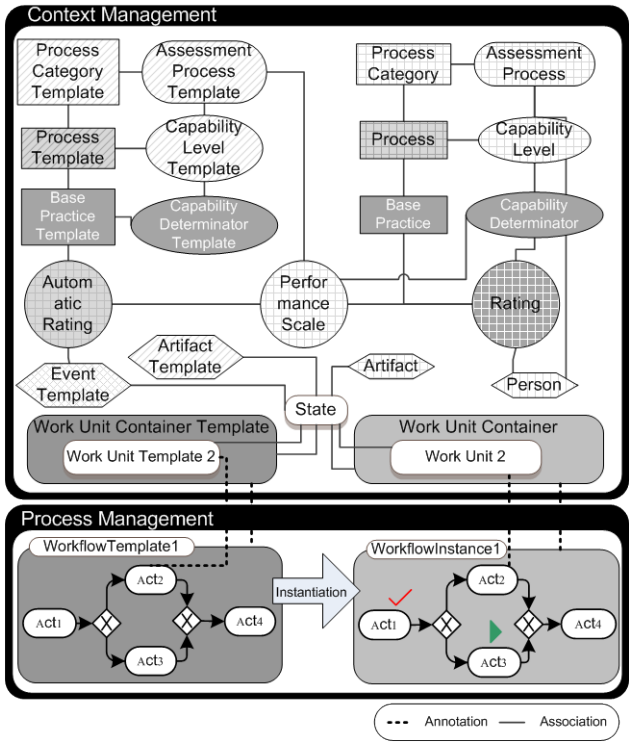

Figure 2. Conceptual framework for automating process assessment.

Similar to the *Work Unit Containers* and their templates, the concepts for process assessment are separated into template concepts for definition and individual concepts holding the actual values of one execution. These concepts are abstract and generic to be applicable to various models (cf. requirement R:MultModel). The *Assessment Process Template* defines one process assessment model. In alignment to the aforementioned assessment approaches, it features templates for *Process Categories*, *Processes*, and *Base Practices* as well as *Capability Levels*. The latter are general level concepts used to model various capability or maturity levels that can be calculated for other concepts such as *Base Practices* or *Assessment Processes*. To explicitly configure how the capability level achievement will be determined, *Capability Determinator Templates* are used. The *Assessment Process Template* also defines a number of *Performance Scales* that are used for the assessment later. For all these concepts, there are individual counterparts used for each concrete assessment that are based on the template concepts. Table 1 depicts their relevant properties including a short description.

TABLE III. CONCEPTS PROPERTIES

Property	Description
Assessment Process Template	
capabilityLevels	all defined capability levels templates
procCatTempls	all defined process category templates
Capability Level Template	
calcFor	concept, for which the level is calculated
capDet	attached capability determinator templates
perfScale	required performance scale for achievement
scaleRatio	ratio of capability determinators that must meet required performance scale
subCL	subordinate capability level template
subCLPerfScale	required performance scale of subordinate level
Level	number indicating the level
Capability Determinator Template	
Source	base practice to be assessed
Target	capability level, for which this determinator is used

For flexibility in the assessment calculation, the *Capability Level Templates* have a property 'calcFor' that is used to attach them to the target concept to be calculated (e.g., the whole assessment process when calculated for a project of a single process). As proposed by the three introduced models, level achievement calculation can rely on the assessment of the designated practices (be they required or expected) or subordinate levels. Therefore, the achievement of a capability level is determined by the following properties: 'perfScale' defines which *Performance Scale* the attached *Capability Determinators* has, and via 'scaleRatio' a ratio of *Capability Determinators* can be defined as required for the *Performance Scale*. Additionally, as the *Capability Levels* are connected to other subordinate levels, the *Performance Scale* of their determinators can also be used (cf. SPICE, required by the 'subCLPerfScale' property).

The assessment of the concrete individual concepts is then applied via the explicit *Rating* concept, which connects a *Performance Scale* with a *Base Practice* and a *Capability Determinator*. It can also be connected to a concrete *Person* who will then be asked to do the assessment. To support automation in the assessment procedure and unburden the users, it is also possible to automate ratings with *Automated Rating*. It can be connected to an *Event Template* concept that, in turn, is connected to the *States* of *Artifacts* or *Work Unit Containers*. That way, it can be configured so that when the *Concept Management* component receives certain status change events, a certain *Performance Scale* is assigned to a certain rating. Examples of such a definition include: 'Assign *Performance Scale* 1 if workflow x is present (created)' or 'Assign *Performance Scale* 2 if workflow x is completed' or 'Assign *Performance Scale* 3 if *Artifact* y is in state z'.

D. Assessment Procedure

The concrete assessment procedure applied to rate process performance is shown in Listing 1. The following algorithm describes how a concrete *Assessment Process* is created from its template, how the ratings are applied to the different *Base Practices* contained in the process, and how achievement of maturity/capability levels is determined.

Listing 1. The Rate Process Performance algorithm in pseudocode.

```
Require: Project P, AssessmentProcessTemplate APT,
Person Pers
01: AssessmentProcess AP ← createConcept(APT)
02: linkConcepts(P, AP)
03: for all APT.processCategoryTemplates PCT do
04:  ProcessCategory PC ← createConcept(PCT)
05:  linkConcepts(AP, PC)
06:  for all PCT.processTemplates PT do
07:    Process PR ← createConcept(PRT)
08:    linkConcepts(PC, PR)
09:    for all PRT.basePracticesTemplates BPT do
10:      BasePractice BP ← createConcept(BPT)
11:      linkConcepts(PR, BP)
12:    end for
13:  end for
14: end for
15: for all APT.capabilityLevelTemplates CLT do
16:  CapabilityLevel CL ← createConcept(CLT)
17:  linkConcepts(AP, CL)
18:  linkConcepts(CL, CLT.calculatedFor)
19:  for all CLT.capabilityDeterminatorTemplates
          CDT do
20:    CapabilityDeterminator CD ←
              createConcept(CDT)
21:    linkConcepts(CL, CD)
22:    List relatedBPs ← getRelatedBasePracts(CD,
                          AP)
23:    for all relatedBPs BP do
24:      new rating(CD, BP,
           AP.getStandardPerformanceScale,Pers)
25:    end for
26:  end for
27: end for
28: automatedRating(AP)
29: manualRating(AP)
30: for all AP.capabilityLevels CL do
31:  checkAchievement(CL)
32: end for
```

The algorithm requires a concrete project and an *Assessment Process Template* to be used for that project. The first part of the algorithm (lines 01-14) then creates a structure comprising *Process Categories*, *Processes,* and *Base Practices* for the new *Assessment Process*. For this paper, the following two functions are used: 'createConcept' creates an individual concept from a given template and 'linkConcepts' links two individual concepts together.

The second part of the algorithm (line 15-27) creates the *Capability Level* structure. Therefore, the *Capability Levels* and their attached *Determinators* are created first. Thereafter the *Determinators* are linked to the *Base Practices* they use for determining capability. This is done using the function 'getRelatedBasePractices' that gets all *Base Practices* in the current *Assessment Process* that are configured to be connected to a certain *Capability Determinator* via their templates. For each of these *Base Practices*, a new *Rating* is created linking them to the *Capability Determinator*. To this *Rating*, a standard *Performance Scale* (usually the one equal to 'not achieved') and a responsible person are attached.

The third part of the algorithm (lines 28-32) deals with the concrete assessment. During the whole project, an

automated rating is applied whenever a matching event or status change happens. At the end of a project (or anytime an assessment is desired), the manual rating is applied, distributing the rating information to the responsible person (cf. requirement R:Auto). The latter can then check the automated rating, rate practices that have not yet been rated, or distribute certain parts of the assessment to others who can provide the missing information needed to rate the practices. The final action applied is to check the achievement for each *Capability Level* of an *Assessment Process*.

E. Technical Realization

This section gives insights on the technical realization of our process assessment concept and the CoSEEEK framework. The technical implementation of each of CoSEEEK's components is shown in Figure 3.

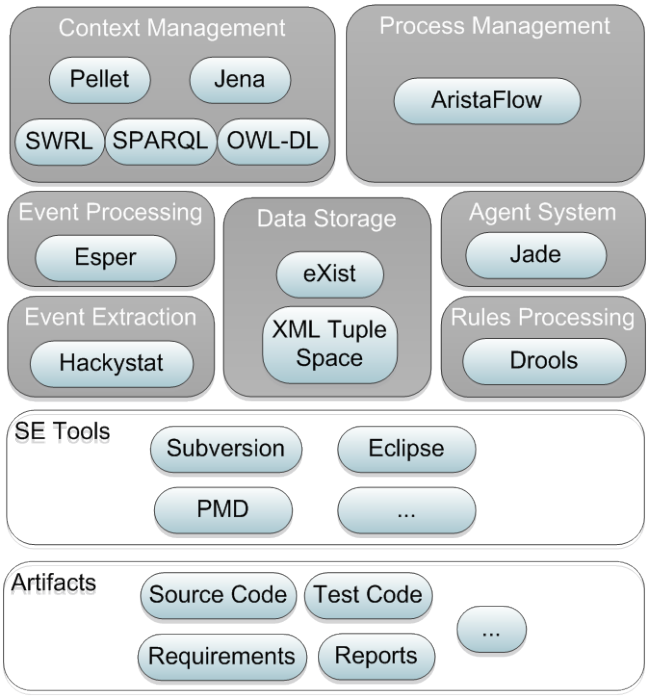

Figure 3. Technical Realization.

As mentioned before, CoSEEEK's context awareness builds primarily on sensors that are integrated into various applications. Applications with sensor support include, as shown in the figure, the version control management system Subversion, the integrated development environment Eclipse, or the quality measurement tool PMD. Artifacts whose state can be monitored this way include source or test code, requirements, or various reports.

The *Event Extraction* component is implemented with the Hackystat [18] framework. The latter provides a rich set of sensors for the aforementioned applications. Furthermore, it features an open architecture for the implementation of new sensors. The aggregation of these events is done via complex event processing (CEP) [19] enabled by the tool esper [20]. The latter provides easy facilities to define and execute CEP patterns. This, together with the sensors, enables the recording of various activities people really execute using SE tools like IDEs (Integrated Development Environments). Thus, the execution of several activities relating to the assessment of the process can be automatically detected and their achievement level can be adjusted.

The communication of the different components is realized via the tuple space paradigm [21]. The latter, in turn, is implemented via a tuple space that has been built on top of the XML database eXist [22]. The *Agent System* component is implemented via the FIPA [23] compliant JADE framework [24] and the *Rules Processing* component with JBoss Drools [25].

The *Process Management* component is implemented with AristaFlow [26]. The latter was chosen due to its capabilities concerning correctness and flexibility. It enables the correct adaptation even of running workflows. In particular, during run-time, selected workflow instances can be dynamically and individually adapted in a correct and secure way; e.g., to deal with exceptional situations or evolving business needs. Examples of workflow instance changes supported by AristaFlow include the dynamic insertion, deletion, or movement of single workflow activities or entire workflow fragments respectively.

The *Context Management* component applies semantic web technology. This comprises an OWL-DL [27] ontology for knowledge organization and Pellet [28] as reasoner for inferences and logical classifications. The usage of ontologies reduces portability, flexibility, and information sharing problems that are often coupled to technologies like relational databases. Furthermore, ontologies support extensibility since they are, in contrast to relational databases, based on an open world assumption and thus allow the modeling of incomplete knowledge.

IV. EVALUATION

This section evaluates our approach by applying it to the three different process assessment models introduced in Section II, and further elucidates technical realization details. A selection of the applied concepts is shown in Figure 4 for all of the three models.

A. CMMI

An excerpt of the implementation of the CMMI model is shown in Figure 4(a). On the upper half, the templates for defining the CMMI concepts are shown: The assessment of the process is carried out in a slightly different way than the reference model of the CMMI, since our concept does not feature explicit goal concepts. Moreover, the assessment for the maturity levels is done directly with the specific and generic practices and not by using the latter for the goals and these, in turn, for the maturity levels.

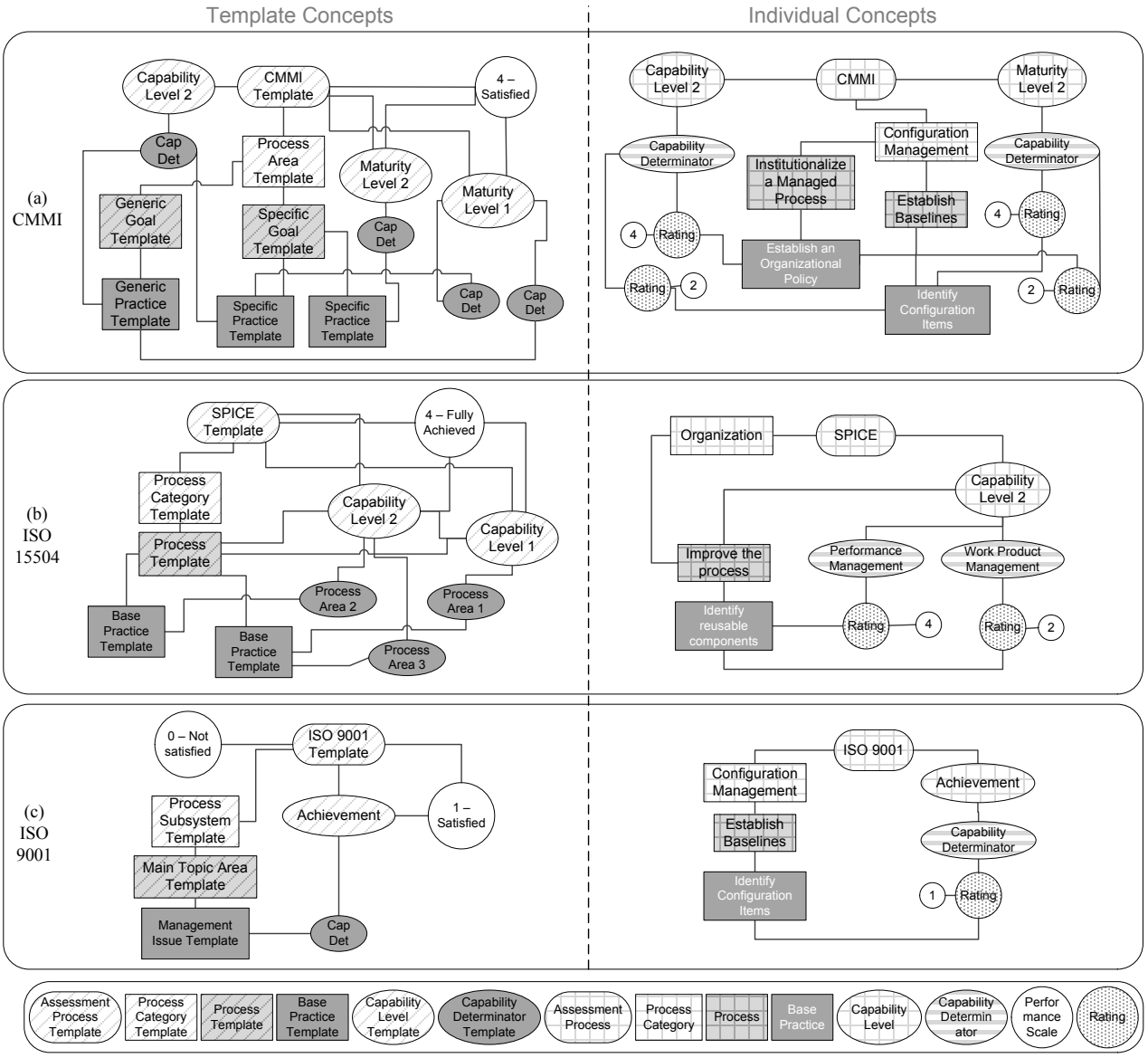

Figure 4. Realization for specific reference models: (a) CMMI (b) ISO 15504 (c) ISO 9001.

The structure of the process is built by the *Process Category Template* (used for the process areas CMMI), the *Process Template* (used for the specific goals CMMI), and the *Base Practice Template* (used for the specific practice of CMMI). Connected to the 'CMMI Template' (implemented by the *Assessment Process Template*) are also the 'Maturity Levels' (implemented by the *Capability Level Template* concept). In addition to this structure with the specific goals and maturity levels, the applied concepts can also be used to implement the generic goals of CMMI with their generic practices and the relating capability levels as illustrated. For the *Assessment Process Template*, the maturity levels are connected to the *Capability Determinators* of all specific practices that belong to the relating maturity level. The *Capability Determinators* also realize connections to *Base Practices* that implement CMMIs generic practices applied to the respective process area (implemented by a connection from the Base Practice, the Process, and the Process Category, cf. 'Establish an Organizational Policy', 'Institutionalize a Managed Process', and 'Configuration Management' in Figure 4). Similar connections can be established for the capability levels, so that the staged or the continuous representation of CMMI to assess respectively the maturity of a whole organization or its capabilities concerning the different process areas. For the capability determination, the *Assessment Process Template* is also connected to the *Performance Scales* that will be used for it. The figure shows one example of them (4 – Satisfied).

On the lower part of Figure 4(a), the individual concepts for the assessment of one concrete project with CMMI are illustrated. It shows one exemplary maturity level and one process area with one specific goal with one specific practice. The *Capability Determinators* of the maturity level are connected to the specific practices that shall be rated via the *Rating* that has an assigned *Performance Scale*. A similar excerpt of the structure is shown for the capability levels and generic goals in the figure.

The achievement calculation for the maturity levels is done with the 'perfScale' and 'scaleRatio' properties of the *Capability Level Template*: That way it can be defined that 100% of the *Capability Determinators* must have the Performance Scale '4' or '2' as defined in the CMMI model. If calculations for all of the projects of an organization were in place, maturity indicators for the entire organization could use the lowest maturity level achieved by all projects.

B. ISO/IEC 15504 (SPICE)

An excerpt of the implementation of the SPICE model is shown in Figure 4(b). In this case the names of the concepts match with the names used in SPICE (e.g., for capability levels or base practices). The *Performance Scales* are defined for the *Assessment Process Template* similar to the CMMI implementation, e.g., 4 – Fully Achieved (86%-100%) as shown in the figure. The process areas that are subordinate to the capability levels in SPICE are implemented using the *Capability Determinator Templates*. Each of the latter is connected to all *Base Practice Templates* to enable their rating concerning all process attributes as required by SPICE.

The lower part of Figure 4(b) again shows an excerpt of the individual concepts used for the assessment of a concrete project. It comprises an exemplary capability level with its two process attributes and an exemplary process category with one process and one base practice.

The SPICE assessment works as follows: All base practices are rated according to all process attributes, and capability levels are determined for the processes. A level is achieved if all its related process areas have only ratings with *Performance Scales* '3' or '4', and the process areas of the subordinate levels all have *Performance Scale* '4'. The assessment of the project is the mathematical mean of the assessments of the processes, and can thus be easily computed without explicit modeling. The same applies to the assessment of a whole organization.

C. ISO 9001

As ISO 9001 is a requirement and not a process model, it must be mapped to the organization's process. This can be applied by connecting *automated ratings* to *events* occurring in the execution of *work unit containers* representing the real execution of a related workflow or be applied manually by a person doing a manual rating. An excerpt of the implementation of the ISO 9001 assessment model is shown in Figure 4(c). In this case, the upper part of the figure again shows the template concepts for defining the model. Compared to the other two models, ISO 9001 is simpler: It knows no capability levels and only two performance scales (as shown in the figure). Therefore, there is only one *Capability Level Template* defined that is used to determine achievement for the whole ISO 9001 assessment. That template has one *Capability Determinator Template* for each management issue.

The lower part of Figure 4(c) again shows the individual concepts used for a concrete assessment using a concrete example for a process subsystem, a main topic area, and a management issue. The assessment is applied by the 'perfScale' and 'scaleRatio' properties of the single *Capability Level*, specifying that all *Capability Determinators* must have the *Performance Scale* '1'. As ISO 9001 knows no project level, this can be added by using a separate Assessment Process for each project, and cumulating the assessment over the whole organization (if all projects have achieved, the whole organization has achieved).

D. Performance and Scalability

Process assessment approaches often comprise dozens or even hundreds of concepts (e.g., SPICE has over 200 base practices), which implies the creation of an even higher number of concepts in the ontology to enable automated assessment. Therefore, the utilization of a separate ontology for process assessment is considered to keep the operational ontology of the CoSEEEK framework clean. Furthermore, to support stability and performance, the CoSEEEK ontologies are not managed as plain files but stored in a database (using Protégé functionality). The test configuration consisted of a PC with an AMD Dual Core Opteron 2.4 GHz processor and 3.2GB RAM with Windows XP Pro (SP3) and the Java Runtime Environment 1.5.0_20, on which CoSEEEK was running networked via Gigabit Ethernet to a virtual machine (cluster with VMware ESX server 4.0, 2 GB RAM allocated to the VM, dynamic CPU power allocation) where the AristaFlow process server is installed.

The approach supports model diversity, and thus the ontology size can vary based on various reference models. Scalability of the approach was assessed, since a large number of concepts can be required with complicated models such as SPICE - which has over 200 *Base Practices* that require linking to all process areas and calculation of all *Capability Levels* for the *Processes*. The most resource intensive point is when the entire *Assessment Process* for a project is created, thus performance and scalability tests were conducted for the automatic creation of linked ontology concepts, scaling the number of concepts to account for smaller to larger models.

The results obtained were: 1.7 seconds for the creation and linking of 100 concepts, 14.2 seconds for the creation and linking of 1000 concepts, and 131.4 seconds for the creation and linking of 10000 concepts. The results show that the computation time is acceptable with approximately linear scaling. The slight reduction in average creation time for a single concept is perhaps explainable by reduced initialization percentages and caching effects. At this stage, the performance of the Rate Process Performance algorithm (Listing 1) was not assessed since it is fragmented across a project timescale (at the beginning the concepts are created

and later the ratings are applied), it is dependent on human responses (manual ratings), and live project data has not as yet been collected.

V. Related Work

This section reviews different areas of related work: At first, approaches covering the basic requirements for implementing automated process assessment support are reviewed. This includes automated process and project support as well as the contextual integration of process management. After that, approaches enabling semantic extensions to process management concepts are examined. Finally, approaches aiming at directly supporting automated assessments are discussed.

A. Automated Process Support

To integrate automated assessments with operational process execution, holistic process support should be enabled by that system. In related work, many approaches target that topic. However, many of them focus strongly on governing activities and their dependencies. One of them is the framework CASDE [29]. It utilizes activity theory to provide a role-based awareness module managing mutual awareness of different roles in the project. The collaborative SE framework CAISE [30] enables the integration of SE tools and the development of new SE tools based on collaboration patterns. Caramba [31] provides coordination capabilities for virtual teams. It features support for ad-hoc workflows utilizing connections between different artifacts, resources, and processes. For pre-modeled workflows, UML activity diagram notation is used. For ad-hoc workflows not matching a template, an empty process is instantiated. In that case, work between different project members is coordinated via so-called Organizational Objects. The process-centered SE environment EPOS [32] applies planning techniques to automatically adapt a process instance if certain goals are violated. All of these approaches have capabilities for supporting and automating process execution in SE projects, yet none enacted an entire SE process model and thus failed to provide holistic project support.

B. Contextual Process Integration

As discussed in the requirements section, to be integrated with the real operational process of SE projects, a system providing process support must also take into account contextual data. In related work, numerous approaches for context modeling exist, including frameworks like Context Management [33], CASS [34], SOCAM [35], and CORTEX [36]. These provide support for gathering, storing, and processing context data, but leave the reaction to context changes to the application, or use rule-based approaches that are hard to maintain. There are only few approaches combining context-awareness with workflows. One of these is inContext [37] that makes heavy use of context knowledge for supporting teamwork. However, inContext does not offer the necessary capabilities to implement whole SE process models.

C. Semantic Process Extensions

As aforementioned, the assessment concepts elaborated in this work are implemented as semantic extensions of process management concepts. This enables tight integration with process execution. In related work, there are various approaches implementing such extensions to process management for different purposes: COBRA [38] focuses business process analysis and, for that purpose, presents a core ontology. With the latter, it supports better and easier analysis of processes to comply with standards or laws like the Sarbanes-Oxley act. [39] presents a semantic business process repository that fosters automation of the business process lifecycle and offers capabilities for checking in and out, as well as locking and options for simple querying and complex reasoning. The approach presented in [40] features multiple levels of semantic annotations: a meta-model annotation, a model content annotation, and a model profile annotation as well as a process template modeling language. With these annotations, it aims at facilitating process models across various model representations and languages. A concept for machine-readable process models is presented by [41]. It targets achieving better integration and automation and utilizes a combination of Petri Nets and an ontology, whereby direct mappings of Petri Net concepts in the ontology are established. [42] describes an approach that proposes an effective method for managing and evaluating business processes via the combination of semantic and agent technology to monitor business processes. None of these approaches provides a general semantic extension that allows direct interaction with and management of process execution as well as extensibility to achieve an integration between the latter and process assessment approaches.

D. Automated Process Assessment Support

The core area of related work for this paper is automated process assessment. In this area, a number of approaches exist. One of these constitutes a multi-agent system approach that is presented in [43], to enable automatic measurements for the SW-CMM (Software Capability Maturity Model). The latter is combined with the GQM (Goal-Question-Metric) [44] method, where Goals of the SW-CMM are used as a first step for GQM.

An OWL ontology and reasoner approach for CMMI-SW (CMMI for Software) is presented in [45]. In contrast to our approach, the size of the ontology caused issues for the reasoner. A software process ontology in [46] enables the capturing of software processes on a conceptual level. An extension includes specific models such as SPICE or CMMI. Ontological modeling of both CMMI and ISO 9001 as well as certain process interoperability features is shown in [47]. The authors identify issues in consistently implementing both models simultaneously. This problem was addressed in our approach by including concepts abstracted from a single model. In [48], a Process-Centered Software Engineering Environment supports process implementation focused on CMMI and a Brazilian process improvement model. For CMMI-specific appraisals, multiple supportive tools are available such as the Appraisal Assistant [49]. However, these focus only on CMMI / SCAMPI support.

We provide a more general and flexible approach, since the applied concepts are abstracted from a single model. In contrast to above related work that focused on one or two specific models, ours is capable of assessment model diversity as shown in Section IV. Furthermore, it integrates automated SE process enactment support and supports a combination of automated and manual ratings. That way, the assessment is tightly and automatically integrated with SE process execution support, providing the option of automatic on-the-fly assessments while preserving the ability for humans to manually rate practices and processes. This can support quality awareness.

VI. Conclusion and Future Work

This paper has described an ontology-based multi-model holistic approach for automating the assessment of software engineering processes. Extending our prior work in [1], richer technical details were presented and related work was expanded. Also general requirements for such an approach were described, those being R:Proc, R:Cntx, R:MultModel, R:Integrate, and R:Auto. Then the differences between three common SE process reference models were elucidated. Thereafter, our conceptual framework with semantic extensions to a process-aware information system was presented. It was shown how process reference models such as CMMI, ISO 15504, and ISO 9001 were unified in the ontology and the algorithm that performs the assessment was described. The evaluation demonstrated the technical feasibility, model diversity, and that performance with current technology for expected application scenarios is sufficient.

Our approach is not meant to replace manual ratings or formal appraisals. In our opinion, this is not possible in an automated fashion due to the many factors influencing such ratings in real world process execution. However, our approach can support automatic data collection, supplement manual ratings of practices or processes, contribute to the quality awareness of an organization, and (automatically) highlight areas for process optimization. Furthermore, it can help prepare an organization for a formal appraisal.

Future work involves empirical studies to evaluate the effectiveness of the approach in industrial settings with a variety of software organizations, with various SE process lifecycle models in various projects, at various process capability levels and utilizing different process assessment standards simultaneously.

Acknowledgment

This work was sponsored by the BMBF (Federal Ministry of Education and Research) of the Federal Republic of Germany under Contract No. 17N4809.

References

[1] G. Grambow, R. Oberhauser, and M. Reichert, "Towards Automated Process Assessment in Software Engineering," 7th Int'l Conf. on Software Engineering Advances, 2012, pp. 289-295.

[2] P. Bourque and R. Dupuis, (ed.), "Guide to the Software Engineering Body of Knowledge", IEEE Computer Society, 2004.

[3] CMMI Product Team, "CMMI for Development, Version 1.3," Software Engineering Institute, Carnegie Mellon University, 2010.

[4] ISO, "ISO/IEC 15504-2 -- Part 2: Performing an assessment," 2003.

[5] R. Bamford, and W. J. Deibler, "ISO 9001: 2000 for software and systems providers: an engineering approach," CRC-Press, 2004.

[6] M. Reichert and B. Weber, "Enabling Flexibility in Process-aware Information Systems – Challenges, Methods, Technologies," Springer, 2012.

[7] G. Grambow, R. Oberhauser, and M. Reichert, "Towards Dynamic Knowledge Support in Software Engineering Processes," 6th Int'l Workshop Applications of Semantic Technologies, 2011, pp. 149.

[8] R. Oberhauser and R. Schmidt, "Towards a Holistic Integration of Software Lifecycle Processes using the Semantic Web," Proc. 2nd Int. Conf. on Software and Data Technologies, 3, 2007, pp. 137-144.

[9] SCAMPI Upgrade Team, "Standard CMMI Appraisal Method for Process Improvement (SCAMPI) A, v. 1.3," Software Engineering Institute, 2011.

[10] ISO, "ISO/IEC 15504-5:2012 -- Part 5: An exemplar software life cycle process assessment model," 2012.

[11] ISO, "ISO/IEC 12207:2008 -- Software life cycle processes," 2008.

[12] ISO, "ISO 19011 - Guidelines for auditing management systems," 2011.

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

[14] G. Grambow, R. Oberhauser, and M. Reichert, "Knowledge Provisioning: A Context-Sensitive Process-Oriented Approach Applied to Software Engineering Environments," Proc. 7th Int'l Conf. on Software and Data Technologies, 2012.

[15] G. Grambow, R. Oberhauser, and M. Reichert, "Event-driven Exception Handling for Software Engineering Processes," 5th Int'l Workshop on event-driven Business Process Management, LNBIP 99, 2011, pp. 414-426.

[16] G. Grambow, R. Oberhauser, and M. Reichert, "Enabling Automatic Process-aware Collaboration Support in Software Engineering Projects," Selected Papers of the ICSOFT'11 Conference. Communications in Computer and Information Science (CCIS) 303, pp. 73-89, 2012.

[17] G. Grambow, R. Oberhauser, and M. Reichert, "Contextual Generation of Declarative Workflows and their Application to Software Engineering Processes," *Int'l Journal On Advances in Intelligent Systems,* vol. 4, no. 3 & 4, pp. 158-179, 2012.

[18] P.M. Johnson, "Requirement and design trade-offs in Hackystat: An in-process software engineering measurement and analysis system," Proc. 1st Int. Symp. on Empirical Software Engineering and Measurement, 2007, pp. 81-90.

[19] D.C. Luckham, "The power of events: an introduction to complex event processing in distributed enterprise systems," Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2001

[20] Esper: http://esper.codehaus.org/ [January 2013]

[21] D. Gelernter, "Generative communication in Linda," ACM Transactions on Programming Languages and Systems (TOPLAS), 7(1), 1985, pp. 80-112

[22] W. Meier, "eXist: An open source native XML database," Web, Web-Services, and Database Systems, LNCS, 2593, 2009, pp. 169-183

[23] P.D. O'Brien and R.C. Nicol, "FIPA — Towards a Standard for Software Agents," BT Technology Journal, 16 (3):51-59, 1998.

[24] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE - A FIPA-compliant Agent Framework," Proc. 4th Int'l Conf. and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents. London, 1999.

[25] P. Browne, "JBoss Drools Business Rules," Packt Publishing, 2009.

[26] P. Dadam and M. Reichert, "The ADEPT project: a decade of research and development for robust and flexible process support," Computer Science-Research & Development, 23(2), 2009, pp. 81-97.

[27] World Wide Web Consortium, "OWL Web Ontology Language Semantics and Abstract Syntax," 2004.

[28] E. Sirin, , B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," Web Semantics: Science, Services and Agents on the World Wide Web, 5(2), 2007, pp. 51-53.

[29] T. Jiang, J. Ying, and M, Wu, "CASDE: An Environment for Collaborative Software Development," Computer Supported Cooperative Work in Design III, LNCS, 4402, 2007, pp. 367-376

[30] C. Cook, N. Churcher, and W. Irwin, "Towards synchronous collaborative software engineering," Proc. 11th Asia-Pacific Software Engineering Conference, 2004, pp. 230-239

[31] S. Dustdar, "Caramba—a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams," Distributed and parallel databases, 15(1), 2004, pp. 45-66

[32] R. Conradi, C. Liu, and M. Hagaseth, "Planning support for cooperating transactions in EPOS," Information Systems, 20(4), 1995, pp. 317-336

[33] P. Korpipipää, J. Mantyjarvi, J. Kela, H. Keranen, and E.J. Malm, "Managing context information in mobile devices," IEEE Pervasive Computing 2(3), pp.42-51, 2003

[34] P. Fahy and S. Clarke, "CASS – a middleware for mobile context-aware applications," Proc. Workshop on Context-awareness (held in connection with MobiSys'04), 2004.

[35] T. Gu., H.K. Pung, and D.Q. Zhang, "A middleware for building context-aware mobile services," Proc. IEEE Vehicular Technology Conference (VTC), Milan, Italy, pp. 2656 – 2660, 2004.

[36] G. Biegel. and V. Cahill, "A framework for developing mobile, context-aware applications," Proc. 2nd IEEE Conference on Pervasive Computing and Communication, pp. 361 - 365 , 2004

[37] C. Dorn, S. Dustdar, "Sharing Hierarchical Context for Mobile Web services," Distributed and Parallel Databases 21(1), pp. 85-111, 2007.

[38] C. Pedrinaci, J. Domingue, and A. Alves de Medeiros, "A Core Ontology for Business Process Analysis," LNCS 5021, pp. 49-64, 2008.

[39] Z. Ma, B. Wetzstein, D. Anicic, S. Heymans, and F. Leymann, "Semantic Business Process Repository," Proc. Workshop on Semantic Business Process and Product Lifecycle Management, pp. 92–100, 2007

[40] Y. Lin and D. Strasunskas, "Ontology-based Semantic Annotation of Process Templates for Reuse," Proc.10th Int'l Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'05), 2005.

[41] A. Koschmider and A. Oberweis, "Ontology based Business Process Description," Proc. CAiSE´05 Workshops, pp. 321-333, 2005.

[42] M. Thomas, R. Redmond, V. Yoon, and R. Singh, "A Semantic Approach to Monitor Business Process Performance," Communications of the ACM 48(12), pp. 55-59, 2005

[43] M.A. Seyyedi, M. Teshnehlab, and F. Shams, "Measuring software processes performance based on the fuzzy multi agent measurements," Proc. Intl Conf. on Information Technology: Coding and Computing (ITCC'05) – Vol. II, IEEE CS, 2005, pp. 410-415.

[44] V.R. Basili, V.R.B.G. Caldiera, and H.D. Rombach, "The goal question metric approach," Encycl. of SW Eng., 2, 1994, pp. 528-532.

[45] G.H. Soydan and M. Kokar, "An OWL ontology for representing the CMMI-SW model," Proc. 2nd Int'l Workshop on Semantic Web Enabled Software Engineering, 2006, pp. 1-14.

[46] L. Liao, Y. Qu, and H. Leung, "A software process ontology and its application," Proc. ISWC2005 Workshop on Semantic Web Enabled Software Engineering, 2005, pp. 6–10.

[47] A. Ferchichi, M. Bigand, and H. Lefebvre, "An ontology for quality standards integration in software collaborative projects," Proc. 1st Int'l Workshop on Model Driven Interoperability for Sustainable Information Systems, 2008, pp. 17-30.

[48] M. Montoni et al., "Taba workstation: Supporting software process deployment based on CMMI and MR-MPS," Proc. 7th Int'l Conf. on Product-Focused Software Process Improvement, 2006, pp. 249-262.

[49] Appraisal Assistant, http://www.sqi.gu.edu.au/AppraisalAssistant/about.html [June 2013]