Semantic Workflow Adaption in Support of Workflow Diversity

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* — The application of business process execution and guidance to environments with highly dynamic situations and workflow diversity is hindered by rigid predefined workflow models. Software engineering environments constitute an acute example where developers could benefit from automated workflow guidance if the workflows were made sufficiently concrete and conformant to actual situations. A context-aware software engineering environment was developed utilizing semantic processing and situational method engineering to automatically adapt workflows utilizing an adaptive process-aware information system. Workflows are constructed via context knowledge congruent to the current situation. Preliminary results suggest this technique can be beneficial in addressing high workflow diversity while providing useable guidance and reducing workflow modeling effort.**

Keywords- application of semantic processing; domain-oriented semantic applications; automated workflow adaptation; situational method engineering; process-aware information systems; software engineering environments

I. INTRODUCTION

Business process management (BPM) and automated human process guidance have been shown to be beneficial in various industries [10][15]. However, BPMs often prerequisite and rigid model makes its application in highly dynamic and possibly evolving domains with diverse workflows, such as software engineering (SE), difficult. SE has multiform and divergent process models, unique projects, multifarious issues, a creative and intellectual process, and collaborative team interactions, all of which affect workflow models. These challenges have hitherto hindered automated concrete process guidance and often relegated processes to generalized and rather abstract process models (Open Unified Process, VM-XT, etc.) with inanimate documentation for guidance. Manual project-specific process model tailoring is typically done via documentation without investing in automated workflow guidance. Although automated workflows could assist overburdened software engineers by providing orientation and guidance for problems, guidance that does not coincide with the reality of the situation must be ignored and may cause the entire system to be mistrusted. Thus, adaption and pertinence to the dynamic and diverse SE situations is requisite for adoption of automated workflow guidance in SE environments (SEEs).

While classical application techniques may lend themselves to foreseeable common workflows with conformant sequences (*intrinsic workflows*), workflow integration for non-generalized diverse workflows that are external to the process model (*extrinsic*) presents a challenge. Considering SE, guidance is desirable for issues such as specialized refactoring, fixing bugs, etc., yet it is generally not feasible to pre-model workflows for SE issue processing, since SE issue types can vary greatly (tool problems, component versioning, merge problems, documentation inconsistencies, etc.). Either one complex workflow model with many branches is necessary that takes all cases into account, or many workflow variants need to be modeled, adapted, and maintained for such dynamic environments. The associated exorbitant expenditures thus limit workflow usage to well-known common sequences as typically seen with industrial BPM usage.

To briefly illustrate, SE issues that are not modeled in the standard process flow of defined SE processes (such as OpenUP [19] and VM-XT [25]) include bug fixing, refactoring, technology swapping, or infrastructural issues. Since there are so many different kinds of issues with ambiguous and subjective delineation, it is difficult and burdensome to universally and correctly model them in advance for acceptability and practicality. Many tasks may appear in multiple issues but are not necessarily required, bloating different SE issue workflows with many conditional tasks if pre-modeled. Figure 1 shows such a workflow just for bug fixing which is explained in the following.

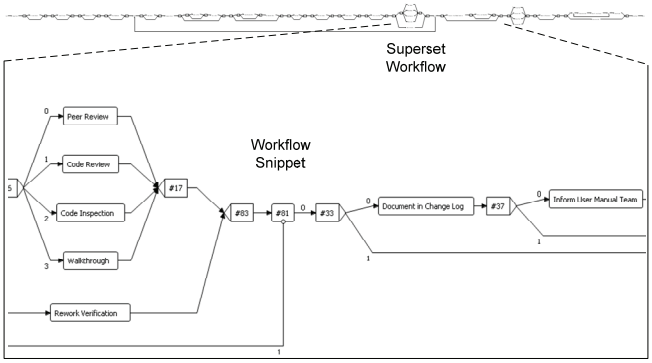

Figure 1. Example of pre-modeled workflow for bug fixing

Copyright (c) IARIA, 2010 ISBN: 978-1-61208-104-5

The above workflow contains over 30 activities, and the snippet shows only different reviewing tasks from which one is chosen due to different project parameters (risk, urgency). Thereafter, it loops back if the post-review code or document rework was insufficient. The subsequent tasks deal with documenting the changes. Again, due to different project parameters the appropriate task has to be chosen, whereby none, one, or both of the tasks may be applicable.

The resulting workflow problems for environments such as SE are first that the exorbitant cost of modeling diverse workflows results in the absence of *extrinsic workflow* models and subsequently automated guidance for these types, yet these unique cases are often the ones where guidance is most desirable. Second, rigid, pre-defined workflow models are limited in their adaptability, thus the workflows become situationally irrelevant and are thus ignored. Third, entwining the complex modeling of situational property influences (as risk or urgency) on workflows within the workflows themselves incorporates an implicit modeling that unduly increases their complexity and makes correct maintenance difficult. The cognitive effort required to create and maintain large process models syntactically can lower the attention towards the incorporated semantic problem-oriented content.

Previous work has described a holistic approach that includes semantic technologies for SE lifecycles [18] and context-awareness [16], while this work focuses on applying context-awareness utilizing semantic processing and situational method engineering [22] for automatically adapting workflows in a process-aware information system. Support is provided for both *intrinsic workflows,* denoting workflows pre-modeled in archetype SE processes, and *extrinsic workflows,* indicating sets of activities not modeled in workflows of those archetype processes. The modeling of contextual property influences is transferred from the workflows themselves to an ontology, simplifying the modeling and making property effects explicit. Dynamic on-the-fly workflow generation and adaptation using contextual knowledge for a large set of diverse workflow variants is thus supported, enabling pertinent workflow guidance for workers in such environments.

The remainder of this paper is organized as follows: the solution approach is described in Section II. In Section III, the realization is portrayed and then evaluated in Section IV. Related work is discussed in Section V, followed by the conclusion.

II. SOLUTION APPROACH

As a background to the solution approach, the incorporated frameworks that affected the environment and influenced the solution will first be discussed.

A. Software Engineering Environment

CoSEEEK (Context-aware Software Engineering Environment Event-driven frameworK) [16] consists of a hybrid semantic computing approach towards improved context-aware SEEs. The conceptual architecture is shown in Figure 2. *Event Extraction* consists of *SE Tool* sensor events (e.g., creation of a certain source code file) that are acquired and then stored in an *XML Tuple Space*, where it may optionally be annotated with relevant contextual information (e.g., link to a requirement for traceability). *Event Processing* detects higher-level events. This may result in workflow adjustment (e.g., according to the type of source code file, an activity Implement Solution or Implement Test may be chosen), and the software engineer is informed of a change in tasks via process management in their IDE (Integrated Development Environment). The *Context Module* includes an ontology and reasoner.

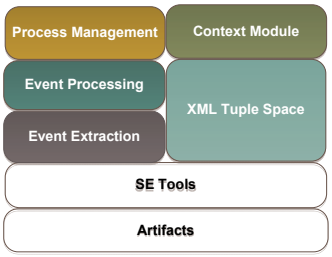

Figure 2. CoSEEEK conceptual architecture

Process Management requires an adaptable process-aware information system due to the dynamic nature of the problem the current approach seeks to address. Therefore the AristaFlow BPM suite (formerly ADEPT2) [3] was chosen for its realization. It allows authorized agents to dynamically adapt and evolve the structure of process models during runtime. Such dynamic process changes do not lead to an unstable system behavior, i.e., none of the guarantees achieved by formal checks at build-time are violated due to the dynamic change at runtime. Correctness is ensured in two stages. First, structural and behavioral soundness of the modified process model is guaranteed independent from whether or not the change is applied at the process instance level. Second, when performing structural schema changes at the process instance level, this must not lead to inconsistent or erroneous process states afterwards. AristaFlow applies well-elaborated correctness principles in this context [23]. Despite its comprehensive support for dynamic process changes, ADEPT2 has not considered automated workflow adaptations.

CoSEEEK provides comprehensive automated process support to address the aforementioned challenges. That implies workflows belonging to SE processes as well as workflows dealing with SE issues that are not modeled in those processes. While the automated support for *intrinsic workflows* is described in [17], the support for *extrinsic workflows* and the approach for their semantic problem-oriented modeling utilizing situational method engineering is the focus of this paper.

B. Application of Situational Method Engineering

Situational method engineering adapts generic methods to the actual situation of a project. This is done based on different influence factors called process properties which capture the impact of the current situation, and product properties which realize the impact of the product currently being processed (for this context the type of component, e.g.,

a GUI or database component). To strike a balance between rigidly pre-modeled workflows and no process guidance, the idea is to have a basic workflow for each case that is then dynamically extended with activities matching the current situation. The construction of the workflows utilizes a case base as well as a method repository. The case base contains a workflow skeleton for each different case. This workflow only contains the absolutely fundamental activities that are always executed for that case. The method repository contains all further activities whose execution is possible according to the case. To be able to choose the appropriate activities for the current artifact and situation, the activities are connected to properties that realize product and process properties of situational method engineering.

Each SE issue, such as refactoring or bug fixing, is mapped to exactly one case relating to exactly one workflow skeleton. To realize a pre-selection of activities (e.g., Create Branch or Code Review) which semantically match an issue, the issue is connected to the activity via an n-m relation. The activities are in turn connected to properties specifying the dependencies among them. The selection of an activity can depend on various process as well as product properties. To model the characteristic of an issue leading to the selection of concrete activities, the issue is connected to various properties. The properties have a computed value indicating the degree in which they apply to the current situation. An example for a property would be 'risk' with a value of 'very high', marking that issue as a very high-risk issue. Utilizing the connection of activity and property, selection rules for activities based on the values of the properties can be specified.

C. Information Gathering

To leverage the automatic support for *extrinsic workflows*, the computation of the values of the properties is a key factor. The approach presented in this paper unifies process and product properties in the concept of the property, which can be influenced by a wide range of factors. The integration of different modules and applications and the unification of various project areas in CoSEEEK enable automatic computation of the values comprising context knowledge. On the one hand, tool integration can provide meaningful information about the artifact that is processed in the current case. For example, if the artifact is a source code file, static code analysis tools such as PMD can be used to execute various measurements on that file, revealing various potential problems. If a high coupling factor was detected, this would raise the product property 'risk' associated to that file. On the other hand, the integration of various project areas like resource planning entails context knowledge about the entire development process. An example would be the raising of the property 'risk' if the person processing the current case is a junior engineer.

Both of these aspects deal with implicit information gathering. Since not all aspects of a case are necessarily covered by implicit information, and not all options for gaining knowledge about the case are always present, the system also utilizes explicit information gathering from the user processing the case. To enable and encourage the user to provide meaningful information, a simple response mechanism is integrated into the CoSEEEK GUI (to be shown in the next section). Via this mechanism, the user can directly influence process as well as product properties. To keep the number of adjustable parameters rather small, the concept of a product category was introduced. The product category unites the product properties in a pre-specified way. An example for this would be a database component versus a GUI component: the database component is likely to have more dependencies, whereas the GUI component presumably has more direct user impact. The influence of the product categories on the different properties is specified in advance and can be adapted to fit various projects. Selected process properties can be set directly. The computation of all other influences on the properties is explained in the following section.

D. Activity selection and sequencing

To be able to dynamically build up the workflow for an SE issue, after completing the computation of the property values activities have to be selected and placed in the correct order. This is done utilizing the connection between properties and activities. An activity can depend on one or more properties. Examples include selection rules such as:

- 'Choose activity code inspection if risk is very high and criticality is high and urgency is low' or
- 'Choose activity code review if risk is high and criticality is high'.

The selection of activities results in an unordered list of activities that have to be correctly sequenced and inserted into the workflow skeleton. To guarantee this, CoSEEEK uses a set of simple semantic constraints. These constraints do not only enforce which activities are permitted in a particular workflow, but also determine their correct ordering. A predefined sequencing of the activities is required since the workflow is built up at the beginning of its execution. That implies that each of the activities must have a binary relation to all other activities so that every possible set of activities can be sequenced. For the time being, the approach presented requires a proper specification of these constraints and only deals with linear workflows. Future work will concentrate on constraints that are more complex and the integration of workflow patterns. Table I enumerates the constraints currently used.

TABLE I. ACTIVITY SEQUENCING CONSTRAINTS

Constraint	Meaning
X before Y	if X and Y are present, X should appear before Y
X after Y	if X and Y are present, X should appear after Y
required after	if Y is present X must also be present, after Y
required before	if Y is present X must also be present, before Y
mutual exclusion	if X is present the presence of Y is prohibited

Structural integrity of the workflows is guaranteed based on the built-in mechanisms of AristaFlow, which imply correctness checks for each change operation applied to the workflow as discussed in [3].

Copyright (c) IARIA, 2010 ISBN: 978-1-61208-104-5

E. Concrete Procedure

The concrete procedure for the handling of an SE issue in the presented approach is as follows. As entry point for the workflow there is an event in the framework indicating that an SE issue is assigned to a user. This event can come from various sources. Examples include the assignment of an SE issue to a person in a bug tracker system or the manual triggering by a user via the GUI. The next step is the determination of a case for that issue like 'Bug fixing' or 'Refactoring'. Depending on the origin of the event, this can be done implicitly or explicitly by the user.

When the case is specified, the workflow starts for the user using the workflow skeleton assigned to that case, as does the contextual information-gathering phase for the properties of the case.

After having determined the properties for the case, the additional activities matching the current situation and product are selected. This set of activities is then checked for integrity and correctly sequenced utilizing semantic constraints. Subsequently, the activities are integrated into the running process instance.

If one or more of the properties change during the execution of the workflow, the prospective activities are deleted (if still possible) and a new sequence of activities is computed for the rest of the workflow.

III. TECHNICAL REALIZATION

This section describes the concrete implementation of the SE issue process explained in the preceding section.

All communication between the modules is performed using an XML implementation of the Tuple Space paradigm [6] on top of the eXist XML database [5] for event storage and Apache CXF for web service communication. To enable CoSEEEK to receive events from external SE tools, the Hackystat framework [8] is used, which provides a rich set of sensors for various applications. In the concrete case, the bug tracker Mantis is used in conjunction with a sensor that generates an event when an SE issue is assigned to a person. That event contains information about the kind of issue for case selection and about the person. In case of a real ad hoc issue that is not recorded in a bug tracker, the event for instantiation of an issue workflow can be triggered from the GUI as well, requiring the user to select a case manually.

The event is then automatically received by the process module which instantiates a skeleton workflow based on the process template relating to the selected case. The activity components of AristaFlow for these workflows are customized to communicate over the Tuple Space and thus enable user interaction during the execution of each task. The first task of each SE issue is 'Analyze Issue' to let the user gain knowledge about the issue and provide information about process and product properties to the system via the GUI. The GUI is a lightweight web interface developed in PHP that can be executed in a web browser as well as preferably directly in the users IDE. Figure 3 shows the GUI enabling the user to directly set process properties and to choose a product category that affects product properties. On the lower part of the GUI, the current task is shown as well as one possible upcoming task from other workflows the user is working in. In that way, task switching is facilitated without subjecting the user to information overload showing all available tasks of all open workflows. Via the dropdown list at the bottom of the GUI, the user can switch between available tasks for the case when the pre-selection is inappropriate.

The Context module has three main responsibilities: it realizes the case base, the method repository, and contains context information about the entire project. This information is stored in an OWL-DL [28] ontology to unify the project knowledge and enable reasoning over it. The use of an ontology reduces portability, flexibility and information sharing problems that are often coupled to relational databases. Additionally, ontologies facilitate extensibility since they are, in contrast to relational databases, based on an open world assumption and thus allow the modeling of incomplete knowledge. To programmatically access the ontology, the Jena API [13] is used within the Context Module.

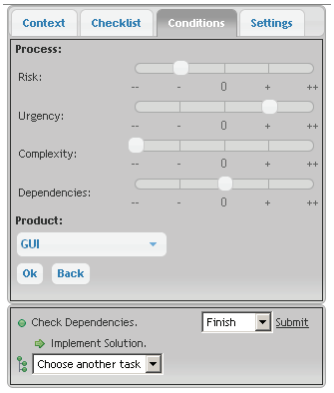

Figure 3. GUI for property acquisition

The adaptation of the running instances works as follows: The skeleton process is instantiated, offering the user the aforementioned 'Analyze Issue' task to provide information. The information from the user is encapsulated in an event that is received by the process module. The process module queries the context module, which provides the set of activities to be inserted in the process instance and performs the adaptation. Thus, the process is already aligned to the current situation and product when the user continues.

A. Context Module

This subsection describes how the context module utilizes the ontology to derive property values and select appropriate activities. To leverage real contextual-awareness, the ontology features various concepts for different areas of a project. These are semantic enhancements to process management utilized for *intrinsic workflows*, quality management, project staffing, or traceability. For process management the concepts of *Activity*, *Workflow*, *Assignment,* and *AtomicTask* are used to enrich processes, activities, and tasks with semantic information. Quality management features the concepts of the *Metric*, *Measure*, *Problem*, *Risk*, *Severity* and *KPI* (key performance indicator) to incorporate

and manage quality aspects in the project context. The concepts of *Person*, *Team*, *Role*, *Effort*, *SkillLevel* and *Tool* are integrated to connect project staffing with other parts of the project. To further integrate all project areas and facilitate a comprehensive end-to-end traceability the concepts of *Tag* and *Event* can be connected and used in conjunction with all other concepts. Due to space limitations, only the concepts directly relevant to the discussion of *extrinsic workflows* are explained. Figure 4 illustrates the relating classes in the ontology.

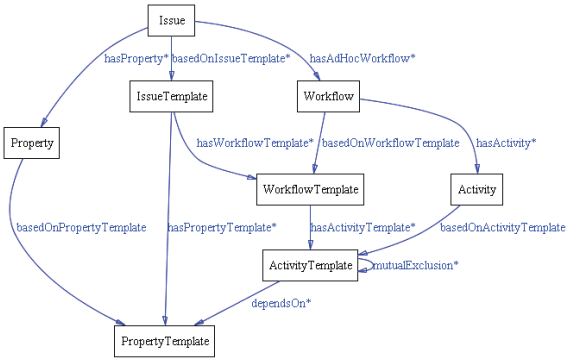

Figure 4. Classes in the Ontology

To predefine the different SE issues, a set of template classes has been defined with their skeleton workflows and activities as well as the properties applying to them. Each *IssueTemplate* is connected to a *WorkflowTemplate* that stores the information about the concrete process template in AristaFlow and is in turn connected to multiple *ActivityTemplates*. These define the set of possible *Activities* that can be inserted in the *Workflow* of that issue. The *IssueTemplate* is also connected to one or more *PropertyTemplates*, yielding the capability to specify not only a unique set of *Activities* for each *Issue*, but also a unique set of *Properties* with a unique relation to the *Activities*.

When a new SE *Issue* is instantiated, it derives the *Workflow* and the *Properties* from its associated *IssueTemplate*. Each *Property* holds a value indicating how much this *Property* applies to the current situation. These values can be influenced by various factors that are also defined by the *PropertyTemplate*. Figure 5 exemplifies three different kinds of influences that are currently used. Future work will include the integration of further concepts of the ontology influencing the *Properties* as well as extending the ontology to further leverage the context knowledge available to CoSEEEK.

The *ProductCategory* specified in the GUI has a direct influence on the product *Properties*. Furthermore, there can be *Problems* relating to the processed *Artifact* indicated by violations of metrics. The *SkillLevel* of the *Person* dealing with the SE *Issue* serves as example for an influence on the process properties here. There are four possible relations between entities affecting the *Properties* and the *Properties* capturing strong and weak negative as well as positive impacts (where Figure 5 only shows the weak ones, 'enhances' and 'deteriorates'). These are all used to compute the values of the *Properties*. The values are initialized with '0 (neutral)' and incremented / decremented by one or two based on the relations to the different influences. The values are limited to a range from '-2 (very low)' to '2 (very high)', thus representing five possible states for the degree to which the property applies to the current situation.

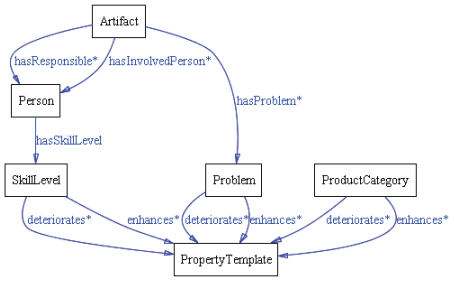

Figure 5. Influences on Properties

To select appropriate *Activities* according to the current properties, six possible connections are utilized. These are 'weaklyDependsOn', 'stronglyDependsOn' and 'dependsOn', meaning the *Activity* is suitable if the value of the *Property* is '1 (high)' or '2 (very high)', or just positive and the other three connections for negative values (for simplicity, Figure 4 only shows 'dependsOn'). Each *Activity* can be connected to multiple *Properties*. Based on an *Issue*, for each attributed *ActivityTemplate* a SPARQL query is dynamically generated which returns the corresponding Activity if the *Properties* of the current situation match. Listing 1 shows such a query for an *Activity* 'act' that is based on an *ActivityTemplate* 'at' and depends on two different *Properties* 'prop1' and 'prop2' which are, in turn, based on *PropertyTemplates* 'pt1' and 'pt2'.

Listing 1 Activity selection SPARQL query

```
PREFIX project:
<http://www.htw-aalen.de/coseeek/context.owl#>
SELECT ?act
WHERE {
   ?act project:basedOnActivityTemplate ?at.
   ?at project:title "AT_CodeReview".
   ?issue project:title "CodeFixRequired".
   ?issue project:hasProperty ?prop.
   ?prop project:basedOnPropertyTemplate ?pt.
   ?at project:weaklyDependsOn ?pt.
   ?prop project:weight "1".
   ?issue project:hasProperty ?prop2.
   ?prop2 project:basedOnPropertyTemplate ?pt2.
   ?at project:stronglyDependsOn ?pt2.
   ?prop2 project:weight "2".}
```

Lastly, the semantic constraints are mapped to connections between *ActivityTemplates* (for simplicity, Figure 4 only shows 'mutualExclusion'). To guarantee semantic correctness, the algorithm first checks if all required activities are in place or if a mutual exclusion constraint is violated. Utilizing the before / after constraints, the sequencing is finally done via a simple sorting of the list of activities. For simplicity, an abstraction from workflow patterns such as loops or decisions is made here.

Copyright (c) IARIA, 2010 ISBN: 978-1-61208-104-5

The significance of this contribution is on the one hand that SE issue workflows that are extrinsic to archetype SE processes are not only explicitly modeled, but also dynamically adapted to the current issue and situation based on various properties derived from the current product, the context, and the user. Thus, it is possible to provide situational and tailored support and guidance for software engineers processing SE issues. On the other hand, the proposed approach shows promise for improvement and simplification of process definition activities for *extrinsic workflows*. The initial effort to define all the activities, issues, properties, and skeleton workflows may not be less than predefining huge workflows for the issues, but the reuse of the different concepts is furthered. Thereafter the creation of new issues is simplified since they only need to be connected to activities they should contain that are later automatically inserted to match the current situation. Yet the main advantage is of a semantic nature: the process of issue creation is much more problem-oriented using the concepts in the ontology versus creating immense process models. The process engineer can concentrate on activities matching the properties of different situations rather than investing cognitive effort in the creation of huge rigid process models matching every possible situation. Likewise, the analysis of issues allows simple queries to the ontology returning problem-oriented knowledge such as 'Which activities apply to which issues' or 'Which activities are applied for high risk time critical situations'.

IV. EVALUATION

This section illustrates the advantages of the proposed approach via a synthetic but concrete practical scenario generated in a lab environment to ascertain scalability and performance of the initial approach using different measurements. It remains difficult to prove the applicability of the approach for the majority of real world SE use cases, thus future work will include practical case studies utilizing CoSEEEK with industrial partners of the research project. Additionally, CoSEEEK is in use by the CoSEEEK development team itself for the development of CoSEEEK.

A. Scenario Solved

The concrete scenario considered shows two possible generated workflows for the bug fix issue presented in Section I. For this scenario, a set of properties has been defined as well as activities and their dependencies on the properties. The first case deals with a fix of a GUI component. That component is assumed to be part of a simple screen not often used by customers. The second case deals with a database component. The fix is assumed to have an impact on multiple tables in the database. Table II depicts the chosen properties for the cases as well as the values that were chosen for them by the developer via the CoSEEEK web GUI. It is assumed that no other influences exist for the properties.The chosen values lead to the selection of different activities for the different workflows. For instance, due to the direct user impact of the GUI component, the activity Document in Patch/Release Change Log has been chosen as illustrated in Figure 6 (the generated workflow has been rearranged for better readability).

TABLE II. EXAMPLE SME PROPERTIES OF CASES

	Component	GUI (Case 1)	DB (Case 2)
Product Properties	criticality	o	+
	user impact	++	o
	dependencies	-	+
	complexity	o	+
	risk	o	+
Process Properties	risk	-	o
	urgency	o	-
	complexity	-	+
	dependencies	o	o

Due to the risk and complexity of the database component and the task relating to it, the creation of a separate branch as well as a code review and the explicit check for dependencies have been prescribed as depicted Figure 7.

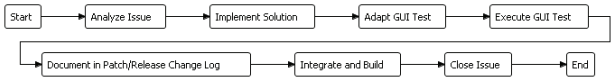

Figure 6. Example Workflow GUI Component

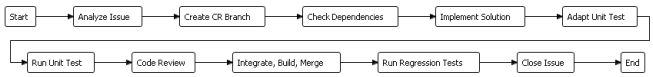

Figure 7. Example Workflow Database Component

Ignoring the abstraction from workflow patterns, they are nevertheless much simpler than the pre-modeled example mentioned in the Problem Scenario section. This automated adaption thus supports workflow diversity, reducing complexity and maintenance compared to all-encompassing models. The scenario illustrates the usefulness of the guidance via the chosen activities by these two considerable different workflows containing tasks matching the situation as well as the processed artifact. Future case studies will be used to further evaluate the usefulness of the workflows and to refine the properties and their relation to the activities.

B. Performance Measurement

Due to space limitations, only the area of the concept that is likely to have the greatest performance impact was selected for measurement. This is the sequencing of the concrete activities based on the constraints in the context module. For performance testing, the test system consisted of an AMD dual core Opteron 2.4 GHz processor, 3.2GB RAM, Windows XP Pro SP3, and Java Runtime Environment 1.5.0_20. All measurements were executed five times consecutively using the average of the last three measurements.

The sequencing of the activities is separated into two parts to yield better runtime performance: when a new SE issue is defined via the issue template or the number of the attributed activity templates changes, an indexing procedure is started. This procedure uses the 'after', 'before', 'requiredAfter', and 'requiredBefore' constraints to generate

Copyright (c) IARIA, 2010 ISBN: 978-1-61208-104-5

a simple index for all activity templates for one issue template. The index is later used for the concretely selected activities of an issue to accelerate the sequencing. For the measurement, it is assumed that half of the activities that are possible for one issue have been selected. Table III depicts the measured total values for indexing and sequencing for different numbers of activities. The values show that after the indexing, which happens usually only once after the definition of an issue type, the sequencing is not resource-intensive.

TABLE III. CONTEXT MODULE LATENCY MEASUREMENTS

Number of activity templates per issue	Indexing latency (ms)	Number of activities per issue	Sequencing latency (ms)
10	5	5	0
50	15	25	0
100	41	50	7
500	890	250	11
1000	3532	500	15

As can be seen from the plain increase of computation times, the results show adequate performance for the CoSEEEK approach.

V. RELATED WORK

The combination of semantic technology and process management technology has been used in various approaches. The concept described in [9] utilizes the combination of Petri Nets and an ontology to achieve machine readable process models for better integration and automation. This is achieved creating direct mappings of Petri Net concepts in the ontology. The main focus of the approach presented in [11] is the facilitation of process models across various model representations and languages. It features multiple levels of semantic annotations as the meta-model annotation, the model content annotation, and the model profile annotation as well as a process template modeling language. The approach described in [12] presents a semantic business process repository to automate the business process lifecycle. Its features include checking in and out as well as locking capabilities and options for simple querying and reasoning that is more complex. Business process analysis is the main focus of COBRA presented in [21]. It develops a core ontology for business process analysis with the aim to provide better easier analysis of processes to comply with standards or laws like the Sarbanes-Oxley act. The approach described in [26] proposes the combination of semantic and agent technology to monitor business processes, yielding an effective method for managing and evaluating business processes. These approaches feature a process-management-centric use of semantic technology, while CoSEEEK not only aims to further integrate process management with semantic technology; it also integrates contextual information on a semantic level producing novel synergies alongside new opportunities for problem-oriented process management.

With regard to automatic workflow support and coordination, several approaches exist. CASDE [7] utilizes activity theory to provide a role-based awareness module managing mutual awareness of different roles in the project. CAISE [2], a collaborative SE framework, enables the integration of SE tools and the development of new SE tools based on collaboration patterns. Caramba [4] features support for ad hoc workflows utilizing connections between different artifacts, resources, and processes to provide coordination of virtual teams. UML activity diagram notation is used for pre-modeled workflows. 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. These approaches primarily focus on the coordination of dependencies between different project members and do not provide unified, context-aware process guidance incorporating *intrinsic* as well as *extrinsic workflows*.

The problem of rigid processes unaligned to the actual situation is addressed in different ways by approaches like Worklets [1], DECLARE [20], Agentwork [13], or Pockets of Flexibility (PoF) [24]. Worklets feature the capability of binding sub-process fragments or services to activities at runtime, thus not enforcing concrete binding at design time. DECLARE provides a constraint-based model that enables any sequencing of activities at runtime as long as no constraint is violated. A combination of predefined process models and constraint-based declarative modeling has been proposed in [24], wherein at certain points in the defined process model (called Pockets of Flexibility) it is not exactly defined at design time which activities should be executed in which sequence. For such a PoF, a set of possible activities and a set of constraints are defined enabling some runtime flexibility. However, the focus of DECLARE as well as PoF is on the constraint-based composition and execution of workflows by end users, and less on automatic workflow adaptations. Agentwork features automatic process adaptations utilizing predefined but flexible process models, building upon ADEPT1 technology. The adaptations are realized via agent technology and used to cope with exceptions in the process at runtime. As opposed to the CoSEEEK approach, these approaches do not utilize semantic processing and do not incorporate a holistic project-context unifying knowledge from various project areas. For a complete discussion of flexibility issues in the process lifecycle, we refer to [27].

VI. CONCLUSION

The SE domain epitomizes the challenge that automated adaptive workflow systems face. Since SE is a relatively young discipline, automated process enactment in real projects is often not mature. One of the issues herein is the gap between the top-down abstract archetype SE process models that lack automated support and guidance for real enactment, and exactly the actual execution with its bottom-up nature. An important factor affecting this problem are activities belonging to specialized issues such as bug fixing or refactoring. These are on the one hand not covered by archetype SE processes and are on the other hand often so variegated that pre-modeling them is not feasible or currently cost-effective.

Copyright (c) IARIA, 2010 ISBN: 978-1-61208-104-5

The synergistic CoSEEEK approach automatically adapts workflows in a process-aware information system by combining semantic-based SEE context knowledge with situational method engineering and automated process instance adaptations. SE issue processing is decomposed into various activities influenced by different process and product properties dependent on the actual situation, the project context knowledge, and the product that is the subject of the current SE issue. Based on these properties, an issue workflow is constructed automatically, dynamically, and uniquely for every SE issue. By combining a case base with a method repository, all activities that are requisite for an issue are automatically included, avoiding the necessity of building the current workflow from scratch.

The broader application of this approach would benefit domains similar to SE that exhibit dynamics and high workflow diversity with adaptable workflows for uncommon workflows, providing useable context-relevant guidance while reducing workflow modeling effort and maintenance by modeling influences outside of the workflows themselves.

Future work will consider issue learning for automated tailoring of process templates and to reduce external user information needs, continuous adaptation of product properties, automated case-learning, and process analysis of executed workflow instances.

ACKNOWLEDGMENTS

The authors wish to thank Stefan Lorenz for his assistance with implementation details. 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] Adams, M., ter Hofstede, A., Edmond, D., and van der Aalst, W., 'Worklets: A service-oriented implementation of dynamic flexibility in workflows,' In: Proc. Coopis'06, LNCS 4275, pp 291-308 (2006)

[2] Cook, C., Churcher, N., and Irwin, W., 'Towards Synchronous Collaborative Software Engineering,' in Proceedings of the Eleventh Asia-Pacific Software Engineering Conference. Busan, Korea, pp. 230-239 (2004)

[3] Dadam, P. and Reichert, M., 'The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support - Challenges and Achievements,' Springer, Computer Science - Research and Development, 23(2), pp. 81-97 (2009)

[4] Dustdar, S., 'Caramba—A Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Teams,' in Distributed and Parallel Databases Vol. 15, Issue 1, Kluwer Academic Publishers Hingham, MA, USA (2004)

[5] Meier, W., 'eXist: An Open Source Native XML Database,' in Web, Web-Services, and Database Systems, Springer, LNCS vol. 2593/2009, pp. 169-183 (2009)

[6] Gelernter, D., 'Generative communication in Linda,' ACM Transactions on Programming Languages and Systems, 7(1):80-112, January 1985 (1985)

[7] Jiang, T., Ying, J., and Wu, M., 'CASDE: An Environment for Collaborative Software Development,' in Computer Supported Cooperative Work in Design III, Springer, pp. 367-376 (2007)

[8] Johnson, P.M., 'Requirement and Design Trade-offs in Hackystat: An In-Process Software Engineering Measurement and Analysis System,' in Proc. of the 1st Int. Symp. on Empirical Software Engineering and Measurement, IEEE Comp. Soc., pp. 81-90 (2007)

[9] Koschmider , A. and Oberweis, A., 'Ontology based Business Process Description,' in Proceedings of the CAiSE´05 workshops, 2005

[10] Lenz, R. and Reichert, M., 'IT Support for Healthcare Processes - Premises, Challenges, Perspectives,' Data and Knowledge Engineering, 61(1): pp. 39-58 (2007)

[11] Lin, Y. and Strasunskas, D., 'Ontology-based Semantic Annotation of Process Templates for Reuse,' in Proceedings of the 10th International Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'05), 2005

[12] Ma, Z., Wetzstein, B., Anicic, D., and Heymans, S., and Leymann, F.,. 'Semantic Business Process Repository' In *Proc. of the Workshop on Semantic Business Process and Product Lifecycle Management*, 2007

[13] McBride, B., 'Jena: a semantic web toolkit,' Internet Computing, Dec. 2002

[14] Müller, R., Greiner, U., and Rahm, E., 'AGENTWORK: A Workflow-System Supporting Rule-Based Workflow Adaptation,' Data Knowl. Eng. 51(2), pp. 223-256 (2004)

[15] Müller, D., Herbst, J., Hammori, M., and Reichert, M., 'IT Support for Release Management Processes in the Automotive Industry,' Proc. 4th Int'l Conf. on Business Process Management (BPM'06), Vienna, LNCS 4102, pp. 368-377 (2006)

[16] Oberhauser, R., 'Leveraging Semantic Web Computing for Context-Aware Software Engineering Environments,' In G. Wu (ed.) Semantic Web, In-Tech, Vienna, Austria, 2010, pp. 157-179 (2010)

[17] Oberhauser, R., 'Towards Automated Test Practice Detection and Governance,' Int. Conf. on Advances in System Testing and Validation Lifecycle, pp. 19-24 (2009)

[18] Oberhauser, R. and Schmidt, R., 'Towards a Holistic Integration of Software Lifecycle Processes using the Semantic Web,' In: Proc. 2nd Int. Conf. on Software and Data Technologies (ICSOFT'07), Vol. 3, 2007, pp. 137-144 (2007)

[19] OpenUP, http://epf.eclipse.org/wikis/openup/ [June 2010]

[20] Pesic, M., Schonenberg, H., and van der Aalst, W.M.P., 'DECLARE: Full Support for Loosely-Structured Processes,' Proc. 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), pp. 287-298. Annapolis (2007)

[21] Pedrinaci, C., Domingue, J., and Alves de Medeiros, A.: 'A Core Ontology for Business Process Analysis' (2008), pp. 49-64

[22] Ralyté; J., Brinkkemper, S. and Henderson-Sellers, B. (Eds.), 'Situational Method Engineering: Fundamentals and Experiences,' Proc. IFIP WG 8.1 Working Conference, Sep. 2007, Geneva (2007)

[23] Rinderle-Ma, S., Reichert, M., and Weber, B., 'Relaxed Compliance Notions in Adaptive Process Management Systems,' In: Proc. 27th Int'l Conf. on Conceptual Modeling (ER'08), October 2008, Barcelona, LNCS 5231, pp. 232-247 (2008)

[24] Sadiq, S., Sadiq, W., and Orlowska, M., 'A framework for constraint specification and validation in flexible workflows,' Information Systems 30(5):349 – 378 (2005)

[25] Rausch, A., Bartelt, C., Ternité, T., and Kuhrmann, M., 'The V-Modell XT Applied - Model-Driven and Document-Centric Development,' in 3rd World Congress for Software Quality, Vol. III, Online Supplement, pp. 131—138 (2005)

[26] Thomas, M., Redmond, R., Yoon, V., and Singh, R., 'A Semantic Approach to Monitor Business Process Performance,' Communications of the ACM, pp. 55-59, 2005

[27] Weber, B.; Sadiq, S., and Reichert, M., 'Beyond Rigidity - Dynamic Process Lifecycle Support: A Survey on Dynamic Changes in Process-aware Information Systems,' Computer Science - Research and Development, 23(2): 47-65 (2009)

[28] World Wide Web Consortium, 'OWL Web Ontology Language Semantics and Abstract Syntax,' (2004) [June 2010]

Copyright (c) IARIA, 2010 ISBN: 978-1-61208-104-5