Supporting Software Development Teams with a Semantic Process- and Artifact-oriented Collaboration Environment

Sebastian Weber[1], Andreas Emrich[2], Jörg Broschart[1], Eric Ras[1], Özgür Ünalan[1]

[1]Fraunhofer Institute for Experimental Software Engineering (IESE)
Fraunhofer-Platz 1
67663 Kaiserslautern, Germany
{weber,broschart,ras,uenalan}@iese.fhg.de

[2]Institute for Information Systems (IWi) at the
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, Campus D3.2
66123 Saarbrücken, Germany
andreas.emrich@dfki.de

Abstract: The focus of this paper is on how to support small software teams in tailoring and following organization-specific process models by using a lightweight and flexible approach to reduce the visible complexity of software projects. We introduce the SPACE (Semantic Process- and Artifact-oriented Collaboration Environment) concept, which describes working processes and an associated approach. These models are integrated semantically, thereby enabling various kinds of analytic techniques, and thus making it easier to cope with the complexity of processes. Pre-defined templates can be configured to actual working processes and artifacts exchanged in such processes. In this paper, we adapt SPACE to the software engineering domain by using the domain-specific Software Organization Platform (SOP). In this context, the templates contain process and artifact descriptions of software process models, such as V-Model, RUP, or agile development.

1 Introduction

Nowadays, numerous projects still fail despite enhancements in software engineering (SE) and project management techniques [CH07] [GL05]. One of the main reasons is complexity, which results, e.g., from having to coordinate tasks in a distributed development setting or from the increasing number of different project stakeholders. As in any other process, the software development process consists of different activities. Feldman et al. distinguish two classes of roles in software development, namely, technical roles developing the software (e.g., requirements engineer or coder) and management roles for planning and managing project executions (e.g., product manager, project planner, or project manager) [FE00]. The technical roles perform the core activities, i.e., the creation of the actual product, whereas the management roles perform the context activities, such as communication among the stakeholders, change management, etc.

Especially in small and medium-sized enterprises (SME), the staff is not acquainted with such context activities. The lack of process- and technique-specific knowledge (e.g., how to conduct interviews for requirements elicitation) leads to longer development cycles. Such activities are often skipped, especially in time-critical situations [SP01].

As such, context activities are critical for the success of a software engineering process [WE08]. So the question is: How do we get SMEs to follow certain process models (especially if these models require process-specific knowledge)? And how can they handle the overall complexity that arises from software development?

Software engineering research has proposed various process models for software development (e.g., V-Model) over the years, which should help development teams to overcome such problems. Although these models are intended to reduce the risk of project failure, practice shows that SMEs often assume the effort for modeling or tailoring an organization-specific process model to be higher than the benefit in terms of project quality. In consequence, such organizations often follow their own "chaotic" development process (often not even documented), resulting in a negative impact on the project's execution and final outcome [DNW05].

In this paper, we introduce a domain-independent meta-model infrastructure called SPACE (Semantic Process- and Artifact-oriented Collaboration Environment), which supports flexible process and artifact models in order to enable SMEs to create, manage, and apply meta models. In this context, artifacts are working resources of a process. E.g., from a project point of view, an artifact is a project element that is used as input or output of project activities. These models describe overall processes from different detail perspectives. We apply this approach to the support of software development teams with our SE concept SOP (Software Organization Platform) [WE08], which aims at supporting the collaboration of software developing teams and comprises the following cornerstones:

Lifecycle artifact and process management deals with the creation of software artifacts. An SOP should support the definitions of software artifact models and process models that define which sequence artifacts have to be created. These models are enriched with semantic information and define attributes of artifacts used for to generating concrete end-user templates and provide traceability on the meta and instance levels.

Knowledge management aims at storing personal, project-specific, and organizational experience and knowledge. An SOP should leverage this information by supporting its users with pro-active recommendations, which should not only support the application of artifact and process models (i.e., the creation of artifact and process instances), but also their creation.

Stakeholder collaboration is geared towards supporting collaborative development of artifacts by promoting and simplifying communication between stakeholders. This can be reached indirectly by supporting the establishment of social networks between the stakeholders or directly through communication features (e.g., commenting or annotating artifacts) within the platform. As with *knowledge management*, support should not only aim at the instance level (i.e., applying the models) but also on the meta model (i.e., supporting the creation of the models).

We already started to implement such an SOP as a lightweight, semantically enabled, and user-driven wiki-based platform called SOP 2.0 (s. Figure 1). This rich internet application (RIA) facilitates the flexible and collaborative creation of processes and artifacts through its visualization capabilities, such as visual templates and editors.

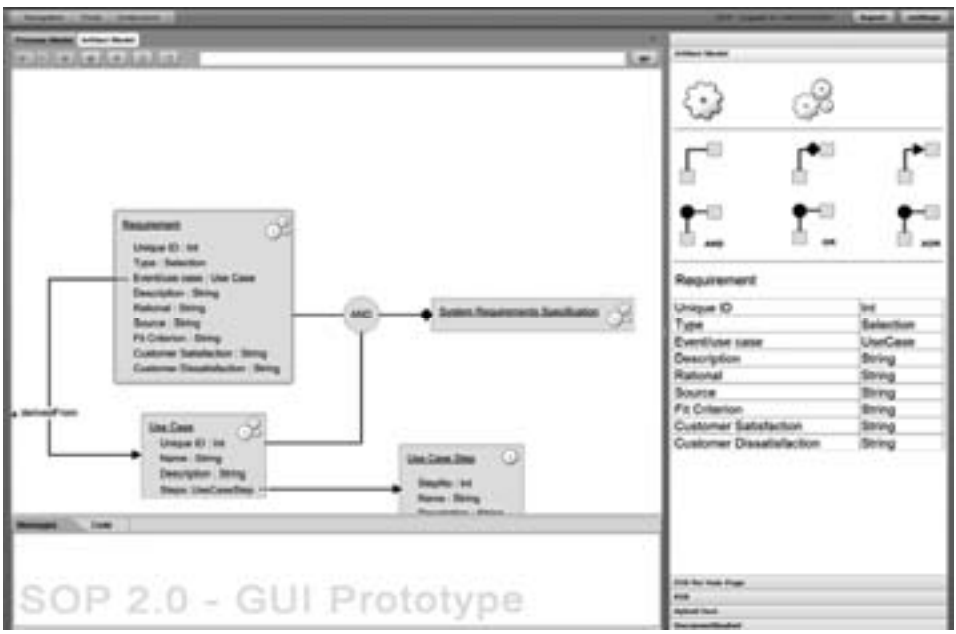

Figure 1: Mockup of SOP 2.0 (work in progress)

The "2.0" in SOP 2.0 stands for the Web 2.0 spirit where people collaboratively work on the production of software artifacts and thereby create a collective intelligence (i.e., knowledge and experience). Figure 2 shows the connection between SPACE, SOP, and SOP 2.0. Throughout this paper, we describe the SPACE concept and illustrate it with appropriate use cases from SOP in the domain of SE.

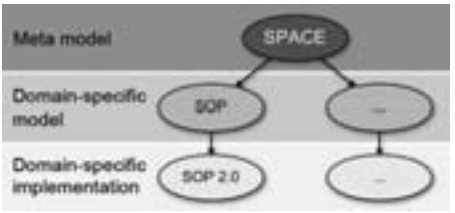

Figure 2: Genealogy of SOP and SPACE

2 Related Work

In terms of supporting stakeholder collaboration and context work in software development projects, processes are crucial for successful software projects. Besides the effort required for defining a custom-tailored model, the problem shared by many tools is that they do not provide any functionality for applying and executing the model (i.e., they do not pursue the "prosumer" idea).

Processes performed by human beings are in the focus of current research activities and industrial developments. With *BPEL4People* [WS07], a standard has been established that enables the integration of human tasks into BPEL workflow engines. Companies like IBM, Oracle, Microsoft, and Intalio have integrated this standard into their workflow engines. However, these tasks are mostly only connected to the immediate resources of the workflow engine; i.e., the environment – the engine itself – defines the relationships to other aspects such as change management or knowledge management.

Processes are also considered by some *semantic wikis* such as Ontobrowse [HS08], which provides technical documentation for services or processes (in the SOA scope). Though it provides templates and semantic descriptions for processes, these do not enable role-specific views on the process or visual perspectives. Moreover, the model is not as generic as SPACE, as it focuses on the documentation of SOA aspects.

ARIS (Architecture of Integrated Information Systems) [SC92] is a concept for modeling information systems with different views for data, resources, organization, control, and services. These views show what kind of models could be considered within SPACE, as SPACE currently only addresses two views. SPACE can be the basis for a platform that enables the modeling and execution of integrated models as specified through ARIS.

The problem with many tools is that they are based on many different, proprietary process modeling approaches, which causes high coordination efforts with the special knowledge and preferences of working staff or institutions. The OMG tries to tackle the complexity resulting from the diversity of modeling languages with the Software Process Engineering Metamodel (SPEM) which is defined as a UML Profile. The modeling language of SOP 2.0 is based on SPEM and thus is extensible and easy to access by other software thanks to a standardized language.

As an example of the Software Engineering domain, the *V-Modell® XT Projektassistent* [VM08] deals only with one model, which is often inappropriate for SMEs. In addition, such a tool is too static for agile scenarios because the process is predefined and can only be tailored in terms of discarding activities. The generated templates are only isolated documents (i.e., Word documents) and not linked. As a result, small organizations are often deterred and, therefore, avoid development process models or, instead, proceed in an ad-hoc and unstructured manner (s. Section 1).

Similar to SPACE, *IRIS Process Author* is a visual process management system that enables collaborative authoring and tailoring of process assets (i.e., artifacts) [IR09]. However, in contrast to SPACE, IRIS lacks semantic support, which is the basis for sophisticated traceability. Futher, IRIS only addresses SE processes, whereas SPACE implementations are not limited to particular domains. Thus, IRIS is comparable with SOP 2.0, which is a system based on SPACE aiming at software process support.

The difference between many related SE tools to SPACE is that they mainly aim at the implementation phase in the development cycle and not at the whole lifecycle. For example, the *NetBeans collaboration project* [NB08] is a collaboration framework within NetBeans that supports programmers regarding collaboration (e.g., distributed code reviews) and communication (e.g., via chats). The *Jazz* [IBM08] [FR07] project pursues a similar way of tool support within the Eclipse IDE. It goes far beyond the NetBeans approach, as it also integrates a wide range of existing tools (i.e., products from the Rational product portfolio) addressing the complete software lifecycle (e.g., requirements engineering or project management). However, this approach is closely related to the Rational product family. Furthermore, there exist no (transparent) semantic connections between artifacts between the tools.

Similar to SOP 2.0, a German research project called *Teamserver* [GFT08] aims at supporting small organizations and small software projects. However, the main focus is on the integration of typical open-source tools, such as a bug tracking tool. The main emphasis lies on code generation and testing.

3 Process Model

A software development process can be described by models such as the waterfall model, the V-Model, etc. Often, it is difficult to gain a thorough understanding of the respective model, which is required to apply it to the development practice in the company (s. Section 1). In this section, we describe the meta-concept SPACE and how it addresses process modeling and execution. The SPACE process model can guide the way for an SE-specific SOP to support software engineering processes.

In order to tackle these problems, a software company should be supported by means of pre-defined process models, which can be tailored to the specific needs of the respective stakeholders. Moreover, it should have a flexible structure that does not force users to follow every single step as intended by the model, but enables them to choose their own course of action. In SPACE, tailoring is also not a task that has to be performed before applying the chosen process; rather it can also be changed, even throughout process execution. Especially for SMEs, which often lack process- or method-specific knowledge, this approach makes it a lot easier to advance complex process models.

First, a concept for the process models themselves is needed. As SPACE is intended to be a basis for both a modeling and an execution platform, it must incorporate both process models and process instances (i.e., concrete model instantiations). The model defines a default sequence flow of activities. Certain information models are associated with these activities (s. Section 4 on the "Artifact Model"). As this approach seeks to be flexible at the instance level, the user is not forced to follow the process.

The platform proposes appropriate courses of action to the user according to the process model (*soft processes*). Consistency checks control the state of the artifacts that are associated with a certain activity. Also, a recursive check over all previous process activities and their respective artifacts is performed. This information is used to tell the user whether the process he performs conforms to the modeled process or not. For example, a consistency check validates whether a requirements specification is complete or not. The concept of soft processes gives the users freedom of choice for their everyday working tasks and thereby empowers them with new means of flexibility. They can decide spontaneously which path to follow and can delay tasks that require further information.

When following a modeled process within SPACE, the user has *personalized views* on the specific process, i.e., he can zoom in or out at specific process segments. The level of detail can be pre-configured by the role a specific user has. An architect might have a different view on the process than a programmer, requirements engineer, etc. Views are meant to be tailored to the respective needs of the stakeholder roles in order to reduce information overload. In fact, those views do not prevent access to detailed information if desired, but emphasize the expected relevant information for the respective role. As an example, a test engineer might not be interested in details of the requirements engineering phase, such as interview documents, but he may require insights into relevant information within use cases, such as the flow of events. Other information contained in the use case, such as actors, might not be relevant for him. As real-world processes can become very complex due to various process variants [GS05], this approach ensures that the user has a minimal but sufficient view on the process.

There may also exist relationships between the different process models. In addition, it may make sense to define certain *sub-processes* in separate process models to reduce complexity in order to have more stability against changes, increase reusability, and improve modularity. This enables easier management of the process models.

Semantic annotations can be used on both the model and instance levels. Whenever possible, the semantic annotations should be incorporated into the templates for certain process models. This takes away complexity from the user's point of view and ensures that obvious relationships are modeled without extra effort. However, when semantic annotations only apply to specific instances, they have to be captured by the user. The user wants ease of use and thus not the complexity of modeling ontologies or similar semantic descriptions. Thus, a platform based on SPACE supports the user in this task with appropriate decision support, which generates the semantic annotations in the background. Nevertheless, the process model must hold the information that describes how to configure such decision support facilities. Further, semantic information is also the basis for the aforementioned consistency checks and the impact analysis (see below).

The semantic annotations allow for *traceability* of various kinds: Not only from artifact to artifact (as in the common understanding of traceability in software engineering), but also from process to process, process to artifact, user to process, role to process, etc. This is the basis for various kinds of analytic techniques, e.g., an impact analysis. It could show which processes are concerned when a process segment changes, which requirements are affected when another requirement changes, etc. This traceability support provides valuable insights to the effects of certain changes. Especially for employees who are not familiar with certain quality assurance techniques, this can facilitate their working experience tremendously.

In the context of SE, a domain-specific platform following the SPACE approach is an SOP where the processes are SE processes, such as project management, requirements engineering, or coding, and the stakeholders are project managers, requirements engineers, programmers, etc.

4 Artifact Model

Besides focusing on the processes of a software project (s. Section 3), the project can also be viewed from an output-oriented perspective. Here, the following questions are central: Which artifacts are created during the project? What relationships exist between the different artifact types? What are the interrelationships with the process model?

The artifact model is associated with the process model, as it defines the different artifact types that are being transformed throughout process execution. In the scope of SOP, artifacts can be, e.g., requirements specifications that are associated with the process activity "Requirements Analysis".

As in the process models, *personalized views* should provide different role-focused levels of abstractions (i.e., blinding out irrelevant details for particular stakeholders to reduce information overload) regarding the presentation of the artifact structure. The interrelationships between all model and instance elements can be viewed through different *perspectives*. A perspective is an aspect-specific focus on a certain process. In contrast to the aforementioned views, these perspectives express the complete structure of artifacts and processes.

In an artifact model, there are *relationships* between artifacts, which constitute the overall process from an artifact-oriented perspective. An instantiation of the model causes the generation of instances of the defined artifacts. In addition, for some cases, the artifact instances can be automatically placed into relationships according to the model. In other cases, the user has to define the relationship manually but is assisted by the underlying artifact model. Furthermore, the *kind of relationship* between the artifacts may be different. The user can choose between simple relationships that define orders or aggregation relationships. The cardinality of artifact types describes how many instances of an artifact type can participate in a relationship with another artifact instance. Moreover, more complex situations are possible, where an artifact consists of exactly one of two different artifact types. The artifact relationships can also span more than one phase of the process model. To model such artifact relationships, semantic relations have to be used that support cardinalities, generalization, and logical operators.

Artifacts also have an *internal structure* that comprises attributes representing data or describing relationships to other artifacts. For example, for the Volére Requirement Shell, a conflict with another requirements document would be modeled by a semantic relationship in the internal structure of the artifact.

Concrete *templates* are generated that actively support the user regarding the creation of artifacts using the attributes and relationships of the templates (s. Figure 3). Based on the semantic information stored in the templates, relationships to other artifact instances are known. The system can support the user by providing advanced templates that recommend a list of relevant artifact instances for defining relationships between artifact instances. To give an example: Starting from a requirements artifact, the user could link existing use cases from a list recommended by the template. The benefit of this mechanism is that it facilitates traceabilty of changes and impact or consistency analysis. Consequently, the relationships defined in the artifact model can be realized quickly and easily on the instance level.

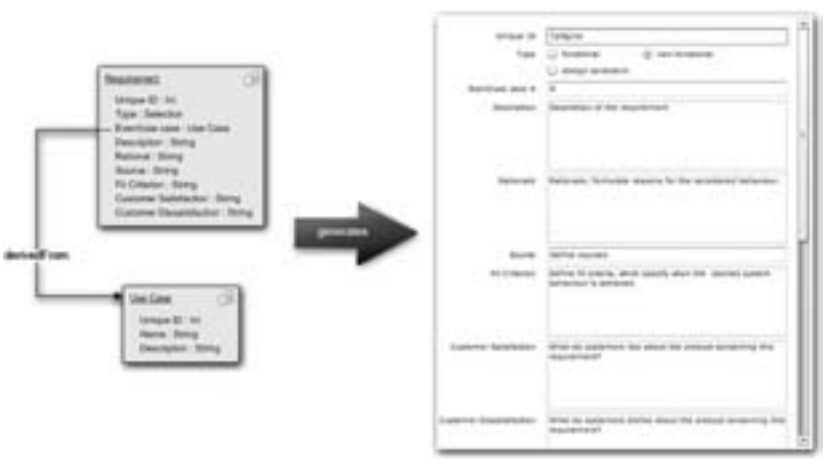

Figure 3: Example of a transformation of artifact descriptions and relationships on the meta-level into end-user templates. Completing the template form causes the creation of an artifact instance.

5 Example Scenario – Test Management

This section shows how the concepts of process and artifact models can be applied to a concrete scenario from the software engineering domain. Figure 4 shows a simplified extract from an integration testing phase.

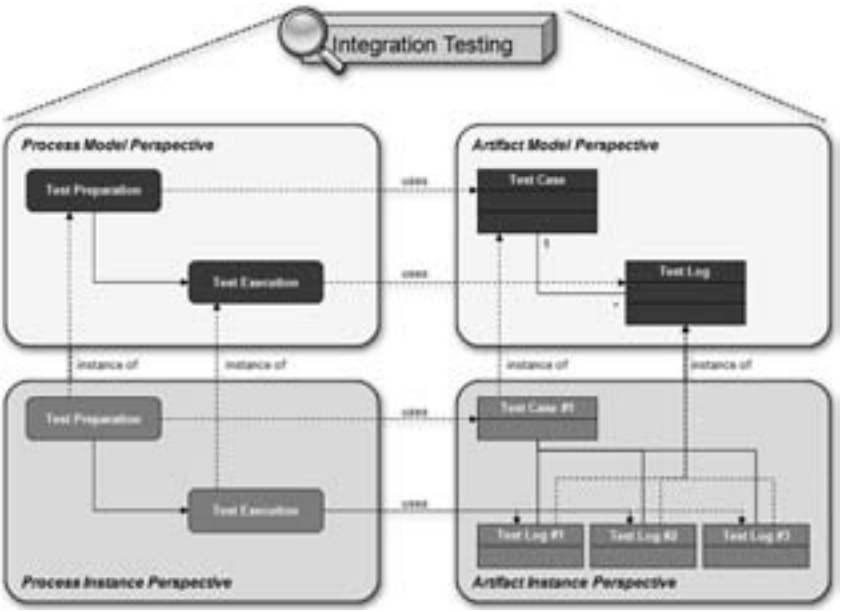

Figure 4: Perspectives on a test management scenario

The example shows how different perspectives describe the phase “integration testing”: The model perspectives show how the process is modeled and which artifacts are associated with respective process activities. The cardinality can also be defined in these relationships; e.g., during test preparation, at least one test case has to be specified, etc.

The artifacts in the instance perspectives are concrete instances of the artifacts in the model perspectives. In this scenario, each modeled activity is instantiated by exactly one activity in the process instance perspective.

For the relationship between the artifacts “Test Case” and “Test Log” in the artifact model perspective, the cardinality states that several test logs may refer to one test case. In the example, only one test case is instantiated (“Test Case #1”) but several test logs refer to “Test Case #1”.

From the user’s view, he would perform a concrete working process on the basis of the process model creating and using artifacts according to the artifact model. These artifact types provide appropriate templates, whereas the process model provides courses of action for the user’s current activity. The complexity of the development process model is hidden from the user, so SPACE provides comprehensive assistance to the user.

6 Implementation of the Prototype (SOP 2.0)

The previous sections dealt with the underlying SPACE concept and how it is adapted to the software engineering domain. This section briefly introduces the current work of the implementation of SOP 2.0, which constitutes a prototype implementing the SOP concept (s. Figure 1 of Section 1). Current work in progress is the development of the artifact model support.

SOP 2.0 is based on the wiki platform MediaWiki [MW08] [BA08], which is also the base of the world's largest wiki – Wikipedia. Semantic MediaWiki (SMW) [SMW08] forms the semantic foundation of SOP 2.0. Because MediaWiki is mainly usable for text-based services, it is not perfectly suitable for visualizing complex issues. From the usability point of view, it lacks a lot of characteristics (e.g., drag & drop, desktop-like user guidance, etc.) that users expect from a Web-based application nowadays. Thus, we decided to build a framework where SMW forms the foundation and a Flex layer on top enables arbitrary sophisticated extensions, especially for visualizing data sets.

With this framework, it is possible to extend SMW with advanced editors that constitute an abstraction from the underlying wiki pages by enabling the wiki user to create artifacts in a visual way guided by the process described by the process and artifact models. Semantic annotations through SMW attributes and typed links enable the creation of meta-models (e.g., an artifact model or a process model). The user visually assembles models by dragging and dropping elements. These elements represent concrete artifacts that can be linked across different perspectives and abstraction layers.

As already addressed in Section 1, one of the strengths of our approach is the idea of collaboration. This means that process and artifact models are created by different stakeholders in different roles concurrently. The models grow with the lifecycle (i.e., on-the-fly-tailoring) of a software project by adapting and refining. As an example, at the beginning of a project, a project manager defines the initial coarse-grained project plan with a few central artifacts and processes. Then, different specialists refine different aspects of the process or artifact models on different abstraction layers (perspectives, views). Consequently, this approach is flexible in such a way that SMEs are able to develop their simple organization-specific models on a high level of abstraction, whereas other organizations might implement a complex process model, such as the V-Model.

7 Conclusion & Outlook

SPACE enables stakeholders to collaboratively develop artifacts in a visual and process-oriented manner. We distinguish meta-models, e.g., an artifact model for the requirements phase, from instances (e.g., concrete requirements created via generated templates). In an initial phase, stakeholders collaboratively choose, customize, or create process and artifact models. With the help of perspectives dealing with different aspects of the system and different levels of abstraction, process and artifact models can be created and elements can be linked arbitrarily.

Based on the models, the platform generates semantically enriched templates and enables traceability between artifacts. These templates provide pre-configured sets of processes and artifact models that can be easily reused or tailored to the specific needs of the stakeholders. The wiki-based approach enables customizing the models on-the-fly, i.e., a process can also be changed during its execution.

For future work, we plan to leverage this semantic information for further analytic techniques, such as impact analysis, cost estimations, etc. In addition, we are currently working on a PID (Proactive Information Delivery) [HO06] feature, where intelligent assistance supports stakeholders by providing context-based and personalized information.

In addition to the aforementioned default use case, where the platform can be used for creating and connecting artifacts and process models, the platform can be utilized for several other scenarios. As an example, the platform can be used for improving documentation in software projects. Nowadays, documenting is often neglected because of time pressure and inappropriate tools [GR02]. Existing tools are often generic, are not integrated into the tool chain, and are not semantically enriched. Our tool can help to support documentation by generating templates from artifact models that also define the relationships between the artifacts (i.e., document tempates). The tool enables software teams to perform automatic consistency and completeness checks.

Although this paper focuses on supporting software engineers, the platform is not restricted to this domain (s. Figure 1). The concept of process and artifact models is domain-independent and can also be transferred to other scenarios. E-Learning processes could be modeled in a similar manner and could be enhanced by a proactive information delivery feature. SPACE could also be used as the basis for a business collaboration platform, where different partners could negotiate common processes that describe interactions in their partnerships. Extensions could monitor the execution of the process and could provide reporting mechanisms to make it possible to control the process. This could be used for instance, to keep track of service level agreements.

By and large, SPACE can be the basis for a ubiquitous collaboration platform that can be applied to many different domains. The pre-defined templates foster easier adoption of complex processes or techniques and the traceability features help to handle the complexity arising from changes. Overall, SPACE uses the semantics to perform tedious context tasks in the background, and finally lets process owners focus on their main purpose: their working process. Along with the SOP concept and the SOP 2.0 implementation, SPACE can provide comprehensive assistance for software development teams. It can take away complexity from the user and make it easier to keep track of the complex relationships between the artifacts in a software development project.

References

[Ba08] Barrett, V.D.: MediaWiki, O'Reilly Media, 2008.

[CH07] The Standish Group International: CHAOS Report 2007: The Laws of CHAOS, 2007.
[DNW05] Durissini, M.; Nett, B.; Wulf, V.: Kompetenzentwicklung in kleinen Unternehmen der Softwarebranche. Zur Praxisorientierung im Software Engineering. Mensch und Computer 2005. Kunst und Wissenschaften. Grenzüberschreitungen der interaktiven ART. Munich, Germany, S. 91-100., 2005.
[Fe00] Feldmann, R. L. et al.: Applying Roles in Reuse Repositories, Chicago, USA, 2000.
[Fr07] Frost, R.: Jazz and the Eclipse Way of Collaboration. IEEE Software vol. 24, 2007.
[GFT08] Beneken, G.; Feindor, R.; Thurmayr, J.: IntegrierteWerkzeugunterstützung füur kleine Projekte: Der Rosenheimer Team-Server. GI-Jahrestagung 2008, S. 317-319, 2008.
[Gl05] Glass, R.: IT Failure Rates--70% or 10-15%? IEEE Software vol. 22, 2005.
[Gr02] Gruhn, V.: Process-Centered Software Engineering Environments, A Brief History and Future Challenges. Kluwer Academic Publishers, vol. 14, S. 363-382, 2002.
[GS05] Grief, J.; Seidlmeier, H.: Modellierung von Flexibilität mit Ereignisgesteuerten Prozessketten (EPK). Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK), 4. Workshop der Gesellschaft für Informatik e.V. (GI) und Treffen ihres Arbeitskreises, 2005.
[Ho06] H. Holz et al.: Task-based process know-how reuse and proactive information delivery in TaskNavigator. Proceedings of the 15th ACM international conference on Information and knowledge management, Arlington, Virginia, USA, S. 522-531, 2006.
[HS08] Happel, H.-J.; Seedorf, S.: Documenting Service-Oriented Architectures with Ontobrowse Semantic Wiki. Munich, Germany, 2008.
[IBM08] IBM - Jazz overview. Accessed on December 19 2008: http://www-01.ibm.com/software/rational/jazz/
[IR09] IRIS Process Author. Accessed on February 12 2008: http://www.osellus.com/IRIS-PA
[MW08] MediaWiki Project Website. Accessed on December 19 2008: http://www.mediawiki.org/wiki/MediaWiki.
[NB08] The NetBeans Collaboration Project. Accessed on December 19 2008: http://collab.netbeans.org/.
[Sc92] Scheer, A.-W.: Architektur integrierter Informationssysteme, Berlin, Heidelberg, New York: Springer, 1992.
[SMW08] Semantic MediaWiki Project Website. Accessed on December 19 2008: http://www.semantic-mediawiki.org/wiki/Semantic_MediaWiki.
[SP01] Sommerville, V.I.; Philippsborn, H.E.: Software Engineering. Addison-Wesley, 2001.
[VM08] Das V-Modell XT Werkzeuge. Accessed on December 19 2008: http://v-modell.iabg.de/index.php?option=com_content&task=view&id=27&Itemid=51.
[We08] Weber, S. et al.: A Software Organization Platform (SOP). Learning Software Organizations (LSO) 2008, Rom, Italy, 2008.
[WS07] WS-BPEL Extension for People Specification (BPEL4People), Version 1.0, 2007. Accessed on December 19 2008: http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/