SITUATIONAL METHOD ENGINEERING APPLIED FOR THE ENACTMENT OF DEVELOPMENT PROCESSES
An Agent based Approach

Holger Seemueller, Holger Voos
Mobile Robotics and Mechatronics Lab, University of Applied Sciences Ravensburg-Weingarten D-88241 Weingarten, Germany

Benjamin Honke, Bernhard Bauer
Programming Distributed Systems Lab, Institute of Computer Science, University of Augsburg, D-86135 Augsburg, Germany

Keywords: Multiagent systems, Product line engineering, Model-driven development, Method engineering, Development processes.

Abstract: Interdisciplinary product development is faced with the collaboration of diverse roles and a multitude of interrelated artifacts. Traditional and sequential process models cannot deal with the long-lasting and dynamic behavior of the development processes of today. Moreover, development processes have to be tailored to the needs of the projects, which are usually distributed today. Thus, keeping these projects on track from a methodology point of view is difficult. In order to deal with these challenges, this paper will present a novel method engineering and enactment approach. It combines the ideas of workflow technologies and product line engineering for method engineering as well as agent technology for the development process enactment.

1 INTRODUCTION

Nowadays, innovative, industrial products are the result of different disciplines, such as mechanics and computer science. They are increasingly characterized by the influence of several domains working together to develop e. g. new products, add new functionalities, or improve the performance done in a geographically distributed and international context. Currently, these interdisciplinary systems are getting more and more complex, whilst the integration efforts grow sometimes exponential. With the increasing complexity of the product, the development process is faced with new challenges concerning e. g. integration, collaboration, and coordination.
In the context of such a development process many different roles with different background and domain specific knowledge can be encountered. These roles are working on different levels of abstraction with diverse techniques, vocabularies, and models producing an intensive amount of work products and artifacts. Based on existing artifacts, new artifacts evolve either by model transformations using a model-driven engineering (MDE) approach (Kent, 2002), such as OMG's MDA (OMG, 2003), or by hand resulting in dependencies and interactions between them. Often, interdisciplinary dependencies are the origin for new functionality by combining e. g. mechanical with software components, which hence leads to the need for an intensive focus on them. In addition, also the evolution of these interrelated artifacts has to be managed between different levels of abstraction during process execution.

Classical development processes, such as V-model XT and RUP (Broy and Rausch, 2005; Kruchten, 2000), are mainly ”paperware” and do not support elaborated tailoring of the project or organisation, whereas newer approaches such as agile development processes lack integration in complex development processes.

Indeed, these approaches provide good support for certain engineering domains, but a variable process execution framework, guiding multiple disciplines around different abstraction levels and domain specific information, is still missing. As processes are long-lasting and faced with a huge degree of dynamics and uncertainty, they cannot be handled by traditional process models with fixed sequences of activities.

To overcome these shortcomings, we propose an MDE approach, in which the development process is described with methods of software product line engineering and method engineering as well as agent technology is applied for enactment.

Thus the goals of our approach are:

- Ease of method engineering, in particular tailoring of development processes using MDE;
- Make development processes alive, i. e. executable;
- Support for distributed system development and modeling
- Pro-active support of the developers;
- Monitoring and evaluation for optimizing development processes;
- Comprehensible development processes and steps for safety critical application.

The rest of the paper is structured as follows. Section 2 sketches current issues of today's development processes and summarizes the techniques applied in our approach. Section 3 focuses on the design and enactment of a process, while Section 4 describes the runtime support using agents. After presenting related work in Section 5, a conclusion is given in Section 6.

2 BASICS

This section presents the necessary background on technologies used in the context of our approach, namely methodologies, situational method engineering, product lines, as well as agent technology.

To confront the complexity of multidisciplinary engineering activities a lot of standards and guidelines were developed during the last years. The VDI guideline 2206 (VDI, 2004) describes methodologies for the development of mechatronic systems in general. The basic idea is the usage of the V-Model being adapted to the specific needs of mechatronics and therefore multidisciplinary systems.

Not only in the field of automotive the V-Model XT combined with process maturity models such as CMMI (Ahern et al., 2008) or SPICE (Dorling, 1993) offers well documented process references. Additionally, there are cross-domain de facto standards, such as SysML (OMG, 2008) or EAST-ADL2 (ATESST, 2008) serving as system models on a higher level of abstraction integrating several domain specific information. Moreover, domain specific standards such as AUTOSAR (AUTOSAR, 2010) aim on simplification of knowledge exchange, collaboration, and integration. Other standards such as XMI (OMG, 2007) or STEP (ISO, 2002) support tool interoperability by common data exchange formats.

Situational Method Engineering (SME) (Saeki, 1994; Harmsen, 1997; Brinkkemper et al., 1998; Mirbel and Ralyte, 2005; Sunyaev et al., 2009) is a discipline exactly facing our goals by providing strategies and techniques for building methods and processes tailored for the situation at hand, i. e. respecting special requirements on individual products, domain specific processes, disciplines, and other available resources.

Product line engineering (PLE) techniques (Clements and Northorp, 2001; Pohl et al., 2005; CAFE, 2004; Families, 2005) are a way to customize the software to be delivered using e. g. feature models (Kang et al., 1990) to specify the features a product should support or not. By adapting PLE to process line engineering, where a process line of similar processes uses a common factory that assembles and configures parts (i. e. features) designed to be reused across the varying development processes in the process line, highly tailored processes can be generated. These development processes can be modeled using standard business process and/or workflow tools and further refined by product and process specific information via semantic or rule-based annotations to provide extended traceability, activity-based validation, and best practice capabilities.

To achieve an agile and flexible behaviour of the process enactment, agent technology is applied. In general, an agent can be characterized as a component, which acts autonomously in some environment to meet a specific objective (Wooldridge, 2009). According to this, a multiagent system consists of a number of single agents, each communicating and negotiating with each other to reach an overall goal of the system. Thus, multiagent systems are a good starting point to support dynamic, large-scale, globally distributed processes and pro-active process coordination through their autonomous decision making.

3 AGENT BASED APPROACH - PROCESS DESIGN

Our approach combines aforementioned technologies to provide a flexible framework for design and execution of development processes across different domains and locations. This section describes the MDA-based method engineering approach for tailoring project specific processes as basis for agents' decision making (see later in Section 4).

3.1 Software Process Line Approach for Method Engineering

To take variable aspects between different projects or process instantiations into consideration, our method engineering is build around a method repository, which stores reusable domain specific method assets, such as activities, artifacts, roles, or other supporting process guidance information. Assets within the method repository are independent of concrete process instances, but serve as building blocks for the situation at hand. According to software product lines, where “a common, managed set of features satisfy the specific needs of particular market segment or mission are developed from a common set of core assets in a prescribed way” (Clements and Northorp, 2001, page 522), here a common set of method fragments is used to specify the particular, situational mission of software product engineering.

To enable situational process configuration, a description for these method fragments is needed. Such a description must be specialized enough to provide domain specific information about situations and the context in which they can be applied. This means, a fragment should provide information about concerned engineering aspects, such as structure, behaviour, timing, or safety as well as information about the domain specific process phase (e. g. system design, software design, mechatronic design) it is optimized for. Product specific specializations, such as activities specialized for diesel engine calibration or windshield wiper development, are conceivable, too.

3.2 Model-driven Method Engineering

Typical SME starts with analyzing the process requirements, e. g. through goals, and refines them to high-level methods, which are then refined to more detailed processes. We adapt this approach and follow an MDE approach namely OMGs MDA with computational independent model (CIM), platform independent model (PIM), and platform specific model (PSM). Especially, as model-based techniques have become more and more mature during the last years, it is convenient for our goals.
It starts on an abstract level for describing common method fragments without any implementation details, like within reference process houses. That non-technical or business-oriented process on CIM level not only serves as method base for further refinement steps into process execution or agent details, but also for general business management activities. Based on this CIM-view different model transformations, e. g. to MS Project, are conceivable. However, they do only provide support for process communication and documentation purposes and do not provide clear execution semantics.

In our approach, we focus on a transformation from that non-technical development process model to a model which details specific execution information more than conventional process frameworks. According to our process line approach, described in Section 3.1, a goal-driven conversion transforms method base information on CIM level into a workflow skeleton (PIM), which afterwards can be refined such that the process can be executed on an agent-based system.

After building that technical model and adding specific semantic information (see Section 3.3), further transformation steps can be applied to come up with a PSM, which controls the concrete run-time behaviour of our method. While annotated process models could be transformed into workflow code optimized e. g. for some process engine, our approach proposes transformation into a model, which pre-configures the autonomous behaviour of agents at run-time as described above similar e. g. to the SHAPE project (Hahn et al., 2009).

By the means of this model-driven approach, creating an evolutionary method base with additional execution support is enabled. Thereby new fragments or best practices can simply be integrated.

This MDA-based approach not only provides method agents with project specific workflows on which agents can make decisions, but enables annotation of situational semantic information or guidelines regarding the behaviour of process and/or product parts. As described above, the method repository enables an iterative incremental way for building a method base, which can be used for generating the workflow skeleton on implementation level. At project start, a project manager describes situational project requirements. Thereby he identifies necessary process phases as well as domain aspects, i. e. a set of process goals, such as special safety or timing requirements, which have to be taken into account during the product development. Afterwards, he specifies the product as the goal of the planned process whereupon the method repository can be queried for method fragments coming into consideration for the situation at hand. Based on fitting fragments and available input/output relationships a workflow skeleton can be assembled.

3.3 Annotations

To provide additional semantics with activities, artifacts, processes, etc. for method agents' configuration, the PIM enables annotating these fragments with a couple of information, whereas the most relevant ones are sketched as follows:

Annotation can be a mixing of syntactical, semantic and rule based annotations. MDA artifacts as input or output of an activity can be annotated with syntactic meta model information in order to focus large meta models to activities at hand. Additionally, method fragments can be described in a semantic domain to provide artifacts with domain specific semantic instance data about the component which has to be modeled, e. g. the semantics of diesel engines or windshield wipers, as long as the editing activity is also specialized for these artifact instances. By the means of semantic annotations, it is possible to make different models of computation, such as CAD models and UML models or other different artifact instantiations, comparable. Finally, artifacts can be annotated with validation rules, such as OCL (OMG, 2006) or RuleML (Boley et al., 2005), to prescribe necessary conditions for indicating the validity of artifacts as some kind of pre-conditions and/or post-conditions.

On the other side, activities can also be annotated with additional execution semantics on PIM level. By relating input with output information by the means of semantic relationships or rules, an activity can prescribe general guidelines or experienced best practices, such as necessary transformations, dependencies in between, or designated/forbidden actions during activities.

Afterwards, a so designed model is used for different scenarios: First of all, the annotated process model can be analyzed in front of the project even more than with conventional techniques. Already on model level, specialized analyses, like the dataflow analysis from Saad and Bauer (2010) , can validate process behaviour properties before its execution. Furthermore, these annotations and transitive relationships between workflow activities and their input/output relationships can be used by agents for process enactment and guidance. An additional value also comes along with flexible model-driven reconfiguration possibilities. Thereby, changes on abstracted workflows can be analyzed and validated before they are re-deployed on the agent-based run-time environment and changes affect process execution directly.

4 AGENT BASED APPROACH - ENACTMENT

Agent technology is a promising technique to enhance long-lasting and flexible process enactment as shown in several research projects (SHAPE, 2010; Burmeister et al., 2008). For the enactment of our development processes, basically, five different kinds of agents can be identified, namely User Agents (UA), Tool Agents (TA), Method Agents (MA), Repository Agent (RA) and Directory Agent (DA).

4.1 Repository Agent

The Repository Agent is responsible on the one side for storing pre-defined process fragments used during the method engineering phase to develop the tailored method and on the other side to support the MA with the necessary process fragments, which have to be executed by different agents. Moreover, architectural descriptions e. g. in East-ADL or SysML, are stored for the MA.

4.2 User Agents

User Agents are the interface between the agent-based execution mechanism and a human user. From the engineer's point of view, an UA deals as a ”personal assistant”, offering him needed information about the next process step and the work he has to achieve. To identify himself, the agent provides a login mechanism. Depending on the role of an engineer within the development process, the agent might offer different functionality.
From the MA's point of view, the UA serves as a representation of the engineer. All the communication and negotiation with the human engineer as a target is handled via this agent.

4.3 Directory Agent

UAs register at the Directory Agent with information such as supported skills and roles. During run-time of the system other agents can look e. g. for several RA or UA, to achieve flexible work distribution depending on the skills and availability of e. g. engineers.

4.4 Method Agents

Method Agents represent an important part for process execution. A domain neutral architecture description language such as SysML or EAST-ADL2 as well as the development process fragments serve as a basis for the agents. Thus, they own a comprehensive knowledge about the entire system such as the structure, specific components or interrelations between them. This knowledge together with the concretized process skeleton according to the software process line approach in touch with the idea of method engineering acts as the foundation for decision making and coordination of needed activities for successful product development.

MAs are strongly interrelated with UAs, as work is distributed to an appropriate engineer via its UA. Additionally, the progress, conditions, and constraints may be checked in cooperation with TAs respectively their encapsulated artifacts or model data. According to this data, next process steps are chosen.

For realisation, a hierarchical organisation of the agents is intended. This means, that a main coordinator manages and coordinates other subordinated MAs.

The usage of multiagent technology implies advantages in contrast to traditional centralised solutions: Due to its autonomous and pro-active behaviour and proven suitability in distributed and complex systems. In our opinion, the application of a multiagent system is convenient for our approach. Additional characteristics such as the loose coupling of the agents can furthermore be used for process variability.

4.5 Tool Agents

Tool Agents serve as wrappers around tools needed during the development process such as IDEs, modeling or analysis tools, CAD etc. Tool Agents communicate with MAs to check specific constraints on a domain specific model, e. g. constraints on the weight, cost, timing aspects etc. , or to check, if specific output artifacts of a process step has been achieved. As a human engineer is working with this tool, also communication between TA and UA is intended. Additionally, semantic annotations as described in section 3.3 can be used to apply domain specific semantic instance data to the artifact, which is currently edited by the tool.
In order to access the necessary internal model data, tool specific adapters have to be developed. The access can be realized e. g. with plugins, tool specific APIs or exchange data formats such as STEP or XMI.

5 RELATED WORK

Previous work already exists about the support of flexible and dynamic process execution.

In EDONA (Ougier and Terrier, 2008), the objective is the construction of an open platform facilitating the realisation of chains of development by providing an interoperability and interchange architecture for automotive development processes and tools. EDONA's idea behind the integration platform is to provide access to a common storage space accessible by any tool chain. Therefore, its goal is the provision of a common meta model to define the data exchange and integration between the partners, a common technical architecture based on the Eclipse Equinox platform, and a set of more generic tools and tool interoperation bridges. Whereas our approach focuses on method engineering and enactment, it can be combined with EDONA's idea to obtain interoperability.

In Aldazabal et al. (2008) the authors suggest a service oriented middleware, called ModelBus, connecting model-based development tools and the services they offer. Thereby, process enactment and process orchestration tools can be used to create/orchestrate/monitor composite services by combining the different services from the different tools into a workflow described in a language such as BPMN (BPMN, 2009). Again the focus is on model exchange and not on method engineering and enactment.

The SHAPE project (SHAPE, 2010) investigates the development and realisation of enterprise systems with ideas of MDE. As proposed by the MDA concept, it separates the modeling into the three abstraction levels CIM, PIM and PSM and tries to fill the gap between them with model transformation. From this approach we borrow the idea of process enactment using agent technology.

Burmeister, Arnold, Copaciu and Rimassa (2008) follow an approach of applying multi-agent based technologies, namely BDI-agents, for business processes modeling and execution. Mainly, the usage of agent technology with its ability of flexibility and proactivity provides agile behaviour of the entire business process management system whereas the ”process plan” is described in terms of project goals and subgoals and associated plans, which achieves the respective goals. We adapt the notion of business process modeling as well as using agent technology, however, in the area of method engineering and enactment.

6 CONCLUSIONS AND FUTURE RESEARCH

In the introduction we stated several goals to achieve with our approach. This is done in the following way:

- The ease of method engineering is reached by adopting MDE and the usage of a method repository. Thus, we can start with a high-level process and have the possibility to tailor it to the requirements of the product under development.

- Development processes are enacted, i. e. can be executed in a distributed system development and modeling through an agent-based realisation.
- Distributed system development is supported through agents' proven suitability in such systems.
- The pro-active support of the developers is achieved through pro-active agents.
- The aspect of monitoring and evaluation for optimizing development processes was not outlined in detail in this paper, but can be performed analogous to business processes.
- Comprehensible development processes and steps for safety critical applications are given, as process activities are clearly defined and documented as well as the agent-based implementation can document these aspects.

In a first prototype, we implemented our approach with the usage of the Eclipse Process Framework (EPF) (Eclipse Foundation, 2010a) on CIM level and reached a transformation into a Java Workflow Tooling (JWT) (Eclipse Foundation, 2010b) model on PIM level by configuring the EPF model with features and domain specific information. The next steps are the transformation of the PIM model into the agent system Jadex (University of Hamburg, 2010) to achieve the method enactment.

REFERENCES

Ahern, D. M., Clouse, A., and Turner, R. (2008). *CMMI Distilled: A Practical Introduction to Integrated Process Improvement (3rd Edition)*. Addison-Wesley Professional.

Aldazabal, A., Baily, T., Nanclares, F., Sadovykh, A., Hein, C., Esser, M., and Ritter, T. (2008). Automated model driven development processes. In *Proceedings of the ECMDA workshop on Model Driven Tool and Process Integration*. Fraunhofer IRB Verlag.

ATESST (2008). Advancing traffic efficiency and safety through software technology: (www.atesst.org).

AUTOSAR (2010). Automotive open system architecture (www.autosar.org).

Boley, H., Grosof, B., and Tabe, S. (2005). *RuleML Tutorial*. The RuleML Initiative.

BPMN (2009). Business Process Model and Notation, Version 2.0 - Beta 1. http://www.omg.org/spec/BPMN/2.0/.

Brinkkemper, S., Saeki, M., and Harmsen, F. (1998). Assembly techniques for method engineering. *Advanced Information Systems Engineering*, page 381.

Broy, M. and Rausch, A. (2005). Das neue V-Modell XT. *Informatik-Spektrum*, 28:220–229.

Burmeister, B., Arnold, M., Copaciu, F., and Rimassa, G. (2008). BDI-agents for agile goal-oriented business processes. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 37–44, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

CAFE (2004). Cafe - from concepts to application in system-family engineering, (http://www.esi.es/cafe/). ITEA project.

Clements, P. C. and Northorp, L. M. (2001). *Software Product Lines: Practices and Patterns*. Addison Wesely.

Dorling, A. (1993). Spice: Software process improvement and capability determination. *Software Quality Journal*, 2:209–224.

Eclipse Foundation (2010a). Eclipse Process Framework Project (EPF). http://www.eclipse.org/epf/.

Eclipse Foundation (2010b). Java Workflow Tooling (JWT). http://www.eclipse.org/jwt/.

Families (2005). Fact-based maturity through institutionalisation lessons-learned and involved exploration of system-family engineering, (http://www.esi.es/families/). ITEA project.

Hahn, C., Shafiq, O., Benguria, G., Kmper, S., and Berre, A. J. (2009). Model transformation and deployment architecture description. SHAPE Deliverable 5.1.

Harmsen, A. F. (1997). *Situational Method Engineering*. PhD thesis, University of Twente.

ISO (2002). Standard for the Exchange of Product model data. ISO standard 10303.

Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, A. (1990). Feature-oriented domain analysis (foda) feasibility study. Technical report, CMU/SEI-90-TR-021, SEI, Carnegie Mellon University.

Kent, S. (2002). Model driven engineering. *Integrated Formal Methods*, pages 286–298.

Kruchten, P. (2000). *The Rational Unified Process: An Introduction, Second Edition*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Mirbel, I. and Ralyte, J. (2005). *Situational method engineering: combining assembly-based and roadmap-driven approaches*, volume 11. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

OMG (2003). MDA Guide Version 1.0.1. http://www.omg.org/mda/.

OMG (2006). Object Constraint Language, Version 2.0. http://www.omg.org/spec/OCL/2.0/.

OMG (2007). MOF 2.0 / XMI Mapping Specification, v2.1.1. http://www.omg.org/technology/documents/formal/xmi.htm.

OMG (2008). Systems Modeling Language (OMG SysML^{TM}). http://www.omgsysml.org.

Ougier, F. and Terrier, F. (2008). EDONA: an Open Integration Platform for Automotive Systems Development Tools. *ERTS 2008 - Toulouse*.

Pohl, K., Boeckle, G., and van der Linden, F. (2005). *Software Product Line Engineering - Foundations, Principles, and Techniques*. Springer Verlag, Berlin Heidelberg.

Saad, C. and Bauer, B. (2010). Data-flow based model analysis. *Second NASA Formal Methods Symposium (accepted)*.

Saeki, M. (1994). Software specification & design methods and method engineering.

SHAPE (2010). Shape Project. http://www.shape-project.eu.

Sunyaev, A., Hansen, M., and Krcmar, H. (2009). Method Engineering: A Formal Description. *Information Systems Development*, pages 645–654.

University of Hamburg (2010). Jadex - BDI Agent System. http://jadex.informatik.uni-hamburg.de/.

VDI (2004). *Entwicklungsmethodik für mechatronische Systeme. VDI Richtlinie 2206*. Beuth Verlag.

Wooldridge, M. J. (2009). *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2nd edition.