Newcastle University

COMPUTING

SCIENCE

Towards Cloud-Based Software Process Modelling and Enactment

Sami Alajrami, Alexander Romanovsky, Paul Watson and Andreas Roth

Towards Cloud-Based Software Process Modelling and Enactment

S. Alajrami, A. Romanovsky, P. Watson and A. Roth

Abstract

Model Driven Engineering (MDE) considers models as a key artifact in software processes, and focus on the creation of models and transformations between them in order to (semi) automatically generate code. In this paper, we step back and consider the software process model itself as a key artifact that can be enacted and semi automated. We support our vision by proposing an architecture for a cloud-based software processes modelling and enactment environment which integrates software development tools and maintains repositories of modelling artifacts and the history of development.

Bibliographical details

ALAJRAMI, S., ROMANOVSKY, A., WATSON, P., ROTH, A.

Towards Cloud-Based Software Process Modelling and Enactment
[By] S. Alajrami, A. Romanovsky, P. Watson, and A. Roth
Newcastle upon Tyne: Newcastle University: Computing Science, 2014.

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1432

Abstract

Model Driven Engineering (MDE) considers models as a key artifact in software processes, and focus on the creation of models and transformations between them in order to (semi) automatically generate code. In this paper, we step back and consider the software process model itself as a key artifact that can be enacted and semi automated. We support our vision by proposing an architecture for a cloud-based software processes modelling and enactment environment which integrates software development tools and maintains repositories of modelling artifacts and the history of development.

About the authors

Sami Alajrami is a PhD student at the school of Computing Science at Newcastle University and his work focuses on novel ways of delivering software engineering in the cloud. He received his BSc. degree in Computer Systems Engineering from Al-Azhar University - Gaza, Palestine in 2010, and MSc. in Internet Technologies and Enterprise Computing from Newcastle University in 2012. Sami has worked at the Cloud & Security Lab in Hewlett Packard Labs, Bristol as a software intern where he has been involved in the Security Intelligence as a Service (SILAS) and the Situational Awareness as a Service (SAaaS) projects.

Alexander (Sascha) Romanovsky is a Professor in the Centre for Software and Reliability, Newcastle University. His main research interests are system dependability, fault tolerance, software architectures, exception handling, error recovery, system structuring and verification of fault tolerance. He received a PhD degree in Computer Science from St. Petersburg State Technical University and has worked as a visiting researcher at ABB Ltd Computer Architecture Lab Research Center, Switzerland and at Istituto di Elaborazione della Informazione, CNR, Pisa, Italy. In 1993 he became a postdoctoral fellow in Newcastle University, and worked on the ESPRIT projects on Predictable Dependable Computing Systems (PDCS), Design for Validation (DeVa) and on UK-funded projects on the Diversity, both in Safety Critical Software using Off-the-Shelf components. He was a member of the executive board of EU Dependable Systems of Systems (DSoS) Project, and between 2004 and 2012 headed projects on the development of a Rigorous Open Development Environment for Complex Systems (RODIN), and latterly was coordinator of the major FP7 Integrated Project on Industrial Deployment of System Engineering Methods Providing High Dependability and Productivity (DEPLOY). He now leads work on fault tolerance in Systems of Systems within the COMPASS project and is Principal Investigator of Newcastle's Platform Grant on Trustworthy Ambient Systems.

Paul Watson is Professor of Computer Science and Director of the Digital Institute. He also directs the £12M RCUK-funded Digital Economy Hub on Social Inclusion through the Digital Economy. He graduated in 1983 with a BSc in Computer Engineering from Manchester University, followed by a PhD on parallel graph reduction in 1986. In the 80s, as a Lecturer at Manchester University, he was a designer of the Alvey Flagship and Esprit EDS systems. From 1990-5 he worked for ICL as a system designer of the Goldrush MegaServer parallel database server, which was released as a product in 1994. In August 1995 he moved to Newcastle University, where he has been an investigator on research projects worth over £30M. His research interest is in scalable information management with a current focus on Cloud Computing. Professor Watson is a Chartered Engineer, a Fellow of the British Computer Society, and a member of the UK Computing Research Committee.

Dr Andreas Roth has been with SAP Research since 2006. SAP is the market leader in enterprise application software and the largest software developer in Europe. SAP Research is the global technology research and innovation unit of SAP conducting collaborative research with academic partners and co-innovation projects with industry partners and customers. Dr Roth is a Research Expert and Theme Lead for Business Application Engineering Tools within the Software Engineering research unit. In this role he is

responsible for leading technical innovations of the research theme and collaborating with SAP's product groups in research transfer projects.

Suggested keywords

SOFTWARE PROCESS MODELLING
PROCESS ENACTMENT
SOFTWARE ENGINEERING
SOFTWARE WORKFLOWS
CLOUD COMPUTING.

Towards Cloud-Based Software Process Modelling and Enactment

Sami Alajrami[1], Alexander Romanovsky[1], Paul Watson[1], and Andreas Roth[2]

[1] School of Computing Science, Newcastle University, Newcastle upon Tyne, UK
{s.h.alajrami,alexander.romanovsky,paul.watson}@ncl.ac.uk
[2] SAP SE, Karlsruhe, Germany andreas.roth@sap.com

Abstract. Model Driven Engineering (MDE) considers models as a key artifact in software processes, and focus on the creation of models and transformations between them in order to (semi) automatically generate code. In this paper, we step back and consider the software process model itself as a key artifact that can be enacted and semi automated. We support our vision by proposing an architecture for a cloud-based software processes modelling and enactment environment which integrates software development tools and maintains repositories of modelling artifacts and the history of development.

Keywords: Software Process Modelling, Process Enactment, Software Engineering, Software Workflows, Cloud computing.

1 Introduction

As software systems are becoming more pervasive, the complexity of these systems has been increasing and the notion of systems of systems has been adopted. This complexity makes producing software systems a difficult task due to the increasing gap between the problem and the software implementation domains [13]. Model-Driven Engineering (MDE) was introduced as an approach to bridge this gap. MDE is driven by models which are used along with model transformation techniques to (semi) automate the code generation.

Models are used in most engineering domains to provide abstraction from the real world. In software systems, models are used for different purposes such as: documentation, testing, static analysis, and code generation. The use of models helps in representing the problem in a systematic way and displays the right amount of details for different perspectives and at different stages of development.

At the same time, the cloud computing paradigm has evolved to simplify organizational IT management and maintenance and cut both operational and expenditure costs. Cloud offers computing resources on demand using different service models (infrastructures, platforms, and software) and different deployment models (public, private, community, and hybrid) [19].

As the cloud is being widely adopted by both research and industry, researchers have started investigating the potential of using it for some software development phases (especially the computing intensive ones e.g. testing) [21], [5], [22]. In general, there are two perspectives to realize the potential collaboration between cloud and software engineering; (a) the use of cloud to support the software development process, (b) advancing software development methodologies to suite developing software for the cloud. The work presented in this paper fits in the first perspective.

MDE is centred around the creation of models and their relevant transformation techniques in order to automatically generate parts of models or code from other models. Typically, the focus is on modelling of individual phases of the software development process. In [18], authors developed an enactment environment for MDA processes, while authors in [4] investigated the potential of combining MDE and cloud, and proposed the notion of Modelling as a Service (MaaS). In this paper, we focus on the software development process models as a key software artifact, these models can be enacted and the non-interactive or repetitive tasks can be automated. We propose an architecture for software workflows enactment environment in the cloud.

The rest of the paper is structured as follows: a background discussing the use of cloud for software development and software process modelling is established in section 2. Section 3 describes the general architecture for our cloud-based software development platform. The paper concludes with a brief summary of future work.

2 Motivation and Background

2.1 Software Engineering in the Cloud

Today, Global Software Development (GSD) has become a popular development model where teams are distributed (sometimes across continents) and use different sets of tools to support and manage the development process. Developers have their own computers and need to have the tools that they need installed and configured. In addition, each team needs access to shared repositories and collaboration tools. The distribution in GSD brings multiple challenges to software development processes such as: restricted communication, less shared project awareness, and inconsistent builds [7]. Provisioning of software development environments in the cloud should proof beneficial as moving the development process to the cloud not only can reduce the amount of resources (time, money, and manpower) spent on the set up and configuration for each software project, but also can address some of the GSD challenges as shown in [14].

Cloud's accessibility facilitates distributed development by providing a shared development environment (artifacts and tools). Furthermore, cloud can bridge the gap between development and deployment environments. Having a virtually unlimited pool of resources in the cloud helps in allocating sufficient resources to certain heavy computing software development tasks (e.g. model checking or

testing). Eventually, using the cloud to support software development processes will help software teams to focus their efforts on the core problem rather than on setting up and maintaining development environments. There are some commercial cloud-based tools that support different phases of the software development process such as: IDEs (e.g. codenvy [3]), testing (e.g. BlazeMeter[4]), issue tracking (e.g. JIRA [5]). However, these tools are dedicated to support one or more phases of the software development process but not the entire process.

2.2 Software Process Modelling

Despite the current trend of embedding high level abstractions in programming languages to avoid code generation bottlenecks, Several approaches to MDE have been introduced: Model-Driven Architecture (MDA) [6], Model-Driven Software Development (MDSD) [25], and Domain Specific Modelling (DSM) [17]. However, the focus has always been on modelling individual phases of the software development process rather than the process itself. Modelling software processes has been investigated since late 80s. There are many motivations which led these investigations including: a) improving the understanding for different perspectives, by visualizing the relevant components for each perspective. b) facilitating communication among team members, and c) supporting project management through reasoning in order to improve the process. Furthermore, the models can be partially automated (e.g. repetitive and non-interactive tasks). Several approaches for software process modelling have been introduced over time, they are categorized into four categories [3]:

1. Rules based (e.g. MARVEL [16])
2. Petri net based (e.g. SPADE [1])
3. Programming languages based (e.g. SPELL [9])
4. UML based (e.g. SPEM [7])

The first three did not receive industrial take up due to their complexity and inflexibility [15]. The UML approach was based on utilizing the wide adoption and acceptance of Unified Modelling Language (UML) for modelling software processes. Several implementations of this approach have been proposed each with different strengths and weaknesses. Authors in [3], compared six UML-based modelling approaches based on a set of software process modelling requirements. The authors also admit that executability and formality are major weaknesses of UML in the context of software process modelling.

Among the previous approaches, SPEM (Software Process Engineering Meta-model) has became an OMG standard for software process modelling. SPEM is based on the concept of interaction between *Roles* that perform *Activities* which

---

[3] https://www.codenvy.com/

[4] http://www.blazemeter.com/

[5] https://www.atlassian.com/software/jira

[6] http://www.omg.org/mda/

[7] http://www.omg.org/spec/SPEM/2.0/

consume (and produce) *Work Products* [8]. However, a major criticism of SPEM in literature is its lack of support for process enactment. As a result, several researchers have proposed different approaches and extensions to support process enactment in SPEM. In [12], authors propose mapping rules to map SPEM models into XML Process Description Language (XPDL) which then can be enacted. In [23], authors propose xSPIDER_ML (a software process enactment language based on SPEM 2.0 concepts). Although xSPIDER_ML is supported with modelling tool and enactment environment, the notion of enactment is limited to process monitoring since developers are supposed to perform their tasks off-line and report their progress to the enactment environment. Authors in [10] introduce eSPEM which is a SPEM extension to allow describing fine-grained behaviour models that facilitate process enactment. They implement a distributed process execution environment [11] based on the FUML standard with emphasis on supporting the ability to share process state on different nodes, suspend and resume process execution, interact with humans, and adapt to different organizations. However, the notion of process enactment in that execution environment also assumes that developers carry out their tasks outside the execution environment and return control back to it once they finish.

Following an enforced formal process modelling can be useful in some cases (e.g. for certifying safety-critical systems). However, in practice, it can be restrictive for the creativity of team members. Organizations have been moving to agile methods to gain more dynamicity and to increase productivity. Therefore, we propose in the next section a less formal, more flexible and adhoc modelling notation than the previously mentioned approaches, with emphasis on the enactment of process models with support of an integrated tool set in the cloud.

3   Proposed Architecture for Cloud-Based Software Process Enactment

As mentioned in the previous section, SPEM lacks support for process enactment. In addition, neither the extensions that are offered by researchers for enactment support are standardized nor widely adopted outside academia. These approaches do not have a proper tool support and do not consider integrating software development tools within the enactment environment. The understanding of software process enactment in most of these approaches seemed to be limited to the concept of process simulation/monitoring. Although this notion of enactment can be useful for project management and monitoring, we think of enactment in a much broader way. Hence, we describe software process enactment as the process of performing software development activities by different actors within an environment that provides enactment support through the integration of development tools and automatic passing of control and data between activities. This means that unlike the approaches mentioned in the previous section, the entire development process execution takes place within one environment where tools and artifacts are available.

3.1 Software Engineering Workflows

Software engineering process can be described as a sequence of operations (*activities*) performed by development team members including customers and managers (*actors*) where activities produce *artifacts* which are used as inputs for other activities. This complies with The Workflow Management Coalition (WfMC) definition of workflow [24] as "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules". Therefore software process can naturally be seen as a workflow. The idea of using workflow technology for software processes is not new, several researchers have investigated it [2], [20], [6].

Based on our description of enactment and software workflows, we propose an architecture to support software process enactment in the cloud. The benefits that software development can gain from the cloud has been discussed in section 2.1. In addition, provisioning of software development environments in the cloud with elastic resources will direct organization's resources towards solving the actual problem. Allowing third parties to integrate their tools makes it possible to try different tool vendors and different versions interchangeably without a huge adjustment effort. The workflow enactment environment passes the execution control between activities as prescribed in the process model. Non-interactive and repetitive tasks can be automated and benefit from the elasticity of the cloud (e.g. run a distributed model checker on two nodes initially and add more nodes if necessary). The artifacts generated from each activity can be accessed by team members (as it is stored on a repository in the cloud) and will be passed to the next relevant activity as an input. Often, software processes can be reused which adds another advantage for software development firms.

The three logical layers of the proposed architecture are illustrated in Figure 1. The top layer is for process modelling where a project manager or a software developer can create/edit models for either higher level abstract processes (e.g project plan) or for daily tasks processes (e.g. implementation). The workflow management layer is where the enactment of the processes takes place while the cloud management layer handles the underlying cloud infrastructure issues (e.g. QoS and multiple cloud providers/models).

The process model contains a description of the activities involved in the process and the data and control flow information which guides the process enactment. Activities are performed by human actors and they are categorized in two types: concrete and abstract. Concrete activities can be either local or external. The local activities can be either a self-contained executable code or interactive (to input decisions or data), while the external activities are web services maintained by third parties. High level abstraction can be provided by abstract activities which are non executable activities and by default will be representing a sub-process. Activities are available in activities pool and can be either created by third party or by the development team. In general, each activity has zero or more input ports and zero or more output ports. The type of input and output artifacts that a port can accept/generate is defined at the
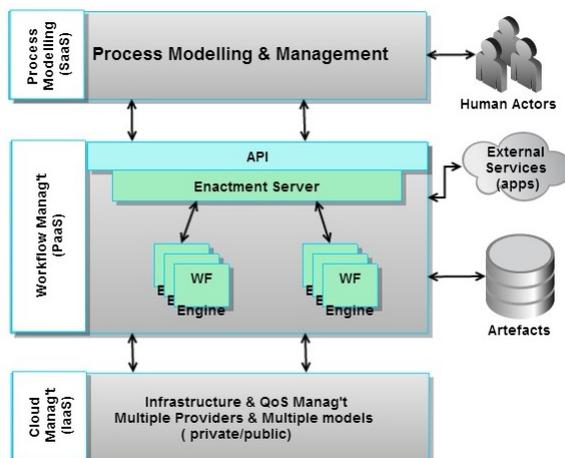
Fig. 1. Proposed architecture to support SW process enactment in the cloud

time of creation of the activity. This guarantees that activities can only be linked to each other when their input/output artifacts are compliant. Activities are executed independently provided that the input needed for them to execute is available. This decoupling allows distributing the execution of the process across several workflow engines (deployed potentially on different virtual machines on the cloud). Activities have configurable parameters to control how the activity will be deployed and executed on the cloud.

3.2 Process Modelling and Definition

Software processes are dynamic and unpredictable. In addition, organizations tailor process models such as waterfall or spiral differently to meet their needs. Hence, a flexible modelling notation is required. This notation needs to be (a) expressive (to express the process and its cloud execution settings), (b) executable, and (c) understandable and easy to use. The notation needs to support a combination of on the fly creation/modification of activities for the purposes of capturing the short term/everyday development and of the longer term activities at the organizational level. Based on that we defined the basic constructs for the software development process model. The process will be represented using a simple graphical notation that can be translated to XML which will then be used to enact the process. The graphical notation can be useful for understanding the process and training new team members. An XML schema has been defined to map the semantics of the graphical notation. These constructs are described in table 1.

Each of the activities can be configured to specify how it will be executed; parameters include (but not limited to): the specified tool support, the cloud

Element Name	Description	Graphical Notation
Abstract Activity	An abstract activity does not execute anything itself, but represents a high level abstraction of one or more activities. Often, it will represent a sub-process.	
Local Activity	A local activity can be an executable code block or a tool that is deployed within the enactment service.	
External Activity (web service)	A tool or service that is deployed and maintained outside the enactment service.	
Interactive Activity	An activity that involves an interaction point where the human actor is asked to provide some input data (e.g. configuration parameters).	
Decision Point	An interaction point where the human actor is asked to decide what to do next.	
Data flow Dependency	A link between two activities A and B, where B cannot start before A provides an input to it.	
Control flow Dependency	A link between two activities A and B, where none of the two is depending on the other. The link here just represents the order of occurrence.	

Table 1. Basic Software Process Modelling Elements

execution requirement (e.g. on public/private cloud), and accepted and generated artifacts. For the sake of simplicity, the notation does not explicitly support modelling of actors at this stage.

Process Examples: Agile methods are widely adopted in industry as they increase the throughput. SCRUM is one of the agile methods which defines a project management framework. This framework defines a set of roles and a set of meetings with different purposes, attendees, and frequencies. Figure 2 illustrates the high level representation of a scrum sprint. This abstraction can be useful from a management perspective. However, it does not specify any details of how developers are going to implement the process. In reality, most software developers use an IDE, an issue tracking tool (e.g. JIRA), a continuous integration framework (e.g. JENKINS), and a version control system (e.g. GIT). These tools are used on daily basis to write, test, store, and integrate code. A model of the daily development process (representing the implementation sub-process) using the notation defined in this subsection is illustrated in Figure 3. The control flow dependency between the "edit issue tracking" and "edit code" activities sets the order of execution, however, since no data dependency is included here, it allows us to perform any of the activities independently. The decision point allows to create a loop based on the decision of the software developer whether to commit his code or to change it or even to edit the issue tracking. Another example is the parallel model checking process (Figure 4) where the model checking activity can be deployed on multiple nodes to utilize the cloud elasticity in order to improve the model checking performance.

3.3 Workflow (Process) Enactment Service

Once the process is defined using the graphical notation described in the previous subsection (which is translated to XML), it will be validated against the process definition schema to make sure the XML file is valid. Next the enactment service should interpret the XML representation of the process and schedule the
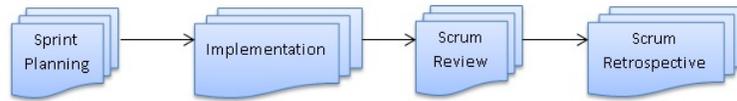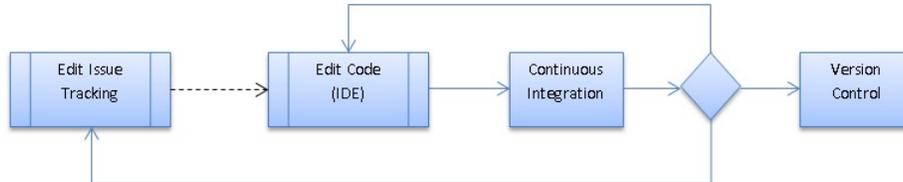
Fig. 2. Scrum high level abstraction

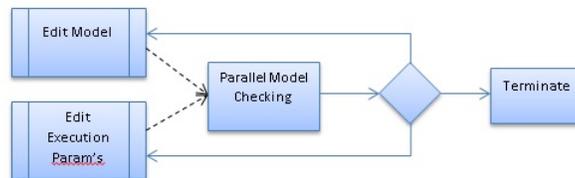**Fig. 3.** Daily technical task by a scrum developer

Fig. 4. Parallel model checking process

execution on as many distributed workflow engines as required. The enactment service consists of an enactment server and multiple workflow engines. The service itself is provided as a web service which accepts requests from any type of clients (desktop, web, mobile/tablet).The enactment service consists of:

- Enactment Server: responsible for managing workflow engines and provide smart scheduling algorithms to optimize cost and performance when the workflow execution is distributed across multiple clouds (if necessary). It is also responsible for monitoring the workflow execution and handling exceptions.
- Workflow Engines: responsible for loading the needed tools and artifacts for executing an activity. It also reports back the execution progress to the enactment server.
- Enactment Service API: provides a standard access to the enactment service.

Using the cloud for executing workflows requires addressing certain issues, such as: portability and QoS of the cloud resources. Authors in [7] identified seven needed quality attributes for a cloud infrastructure to provide tools as a service. These attributes will be used as a guidance for the enactment service implementation.

4 Conclusion and Future Work

In this paper, we proposed a cloud-based software process modelling and enactment environment which harnesses both of cloud and workflows benefits. We considered software process model as a main artifact in the software development process. The process model can be (partially) automated and supported by development tools which are integrated within the enactment environment. We proposed the core of a simple modelling notation for modelling different parts of the software process. Currently, a prototype of the enactment service is being implemented to run on a single cloud initially which will be extended to run on different clouds later. The future work includes: assessing the modelling notation after applying it to more case studies, applying provenance to provide reasoning about the software process, and investigating possible support for interoperability between different tools.

References

1. Bandinelli, S., Fuggetta, A., Ghezzi, C.: Software process model evolution in the spade environment. Software Engineering, IEEE Transactions on 19(12), 1128–1144 (1993)
2. Barnes, A., Gray, J.: Cots, workflow, and software process management: an exploration of software engineering tool development. In: Software Engineering Conference, 2000. Proceedings. 2000 Australian. pp. 221–232 (2000)
3. Bendraou, R., Jezequel, J., Gervais, M.P., Blanc, X.: A comparison of six uml-based languages for software process modeling. Software Engineering, IEEE Transactions on 36(5), 662–675 (Sept 2010)
4. Brunelière, H., Cabot, J., Jouault, F.: Combining Model-Driven Engineering and Cloud Computing. In: Modeling, Design, and Analysis for the Service Cloud - MDA4ServiceCloud'10: Workshop's 4th edition (co-located with the 6th European Conference on Modelling Foundations and Applications - ECMFA 2010) (2010)
5. Bucur, S., Ureche, V., Zamfir, C., Candea, G.: Parallel symbolic execution for automated real-world software testing. In: Proceedings of the Sixth Conference on Computer Systems. pp. 183–198. EuroSys '11, ACM (2011)
6. Chan, D., Leung, K.: Software development as a workflow process. In: Software Engineering Conference, 1997. Asia Pacific ... and International Computer Science Conference 1997. APSEC '97 and ICSC '97. Proceedings. pp. 282–291 (1997)
7. Chauhan, M.A., Babar, M.A.: Cloud infrastructure for providing tools as a service: Quality attributes and potential solutions. In: Proceedings of the WICSA/ECSA 2012 Companion Volume. pp. 5–13. WICSA/ECSA '12 (2012)
8. Combemale, B., Crgut, X., Caplain, A., Coulette, B.: Towards a rigorous process modeling with spem. In: ICEIS (3). pp. 530–533 (2006)
9. Conradi, R., Jaccheri, M.L., Mazzi, C., Nguyen, M.N., Aarsten, A.: Design, use and implementation of spell, a language for software process modelling and evolution. In: Proceedings of the Second European Workshop on Software Process Technology. pp. 167–177. EWSPT '92 (1992)
10. Ellner, R., Al-Hilank, S., Drexler, J., Jung, M., Kips, D., Philippsen, M.: espem a spem extension for enactable behavior modeling. In: Modelling Foundations and Applications, Lecture Notes in Computer Science, vol. 6138, pp. 116–131 (2010)

11. Ellner, R., Al-Hilank, S., Drexler, J., Jung, M., Kips, D., Philippsen, M.: A fuml-based distributed execution machine for enacting software process models. In: Modelling Foundations and Applications, Lecture Notes in Computer Science, vol. 6698, pp. 19–34. Springer Berlin Heidelberg (2011)
12. Feng, Y., Mingshu, L., Zhigang, W.: Spem2xpdl: Towards spem model enactment
13. France, R., Rumpe, B.: Model-driven development of complex software: A research roadmap. In: 2007 Future of Software Engineering. pp. 37–54. FOSE '07, IEEE Computer Society, Washington, DC, USA (2007)
14. Hashmi, S., Clerc, V., Razavian, M., Manteli, C., Tamburri, D., Lago, P., Di Nitto, E., Richardson, I.: Using the cloud to facilitate global software development challenges. In: Global Software Engineering Workshop (ICGSEW), 2011 Sixth IEEE International Conference on. pp. 70–77 (2011)
15. Henderson-Sellers, B., Gonzalez-Perez, C.: A comparison of four process metamodels and the creation of a new generic standard. Information and Software Technology 47(1), 49 – 65 (2005)
16. Kaiser, G., Barghouti, N., Sokolsky, M.: Preliminary experience with process modeling in the marvel software development environment kernel. In: System Sciences, 1990., Proceedings of the Twenty-Third Annual Hawaii International Conference on. vol. ii, pp. 131–140 vol.2 (1990)
17. Kelly, S., Tolvanen, J.P.: Domain-specific modeling: enabling full code generation. John Wiley & Sons (2008)
18. Maciel, R., da Silva, B., Magalhaes, P., Rosa, N.: An integrated approach for model driven process modeling and enactment. In: Software Engineering, 2009. SBES '09. XXIII Brazilian Symposium on. pp. 104–114 (Oct 2009)
19. Mell, P., Grance, T.: The nist definition of cloud computing. National Institute of Standards and Technology 53(6), 50 (2009)
20. Oberweis, A.: Workflow management in software engineering projects. In: Proceedings of the 2nd International Conference on Concurrent Engineering and Electronic Design Automation. pp. 55–60 (1994)
21. Oriol, M., Ullah, F.: Yeti on the cloud. In: Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on. pp. 434–437
22. Pakhira, A., Andras, P.: Leveraging the Cloud for Large-Scale Software Testing A Case Study: Google Chrome on Amazon, chap. Hershey, PA, USA, pp. 252–279. IGI Global (2013)
23. Portela, C., Vasconcelos, A., Silva, A., Silva, E., Gomes, M., Ronny, M., Lira, W., Oliveira, S.: xspider_ml: Proposal of a software processes enactment language compliant with spem 2.0. Journal of Software Engineering and Applications 5(6), 375 – 384 (2012)
24. Specification, W.M.C.: Workflow Management Coalition, Terminology & Glossary (Document No. WFMC-TC-1011). Workflow Management Coalition Specification (1999), http://www.wfmc.org/Download-document/WFMC-TC-1011-Ver-3-Terminology-and-Glossary-English.html
25. Voelter, M., Groher, I.: Product line implementation using aspect-oriented and model-driven software development. In: Software Product Line Conference, 2007. SPLC 2007. 11th International. pp. 233–242. IEEE (2007)