A framework for software process deployment and evaluation

Iván Ruiz-Rube*, Juan Manuel Dodero*

Superior School of Engineering, University of Cádiz, C/ Chile 1, 11002 Cádiz, Spain

Abstract

Context: Software Process Engineering promotes the systematic production of software by following a set of well-defined technical and management processes. A comprehensive management of these processes involves the accomplishment of a number of activities such as model design, verification, validation, deployment and evaluation. However, the deployment and evaluation activities need more research efforts in order to achieve greater automation.

Objective: With the aim of minimizing the required time to adapt the tools at the beginning of each new project and reducing the complexity of the construction of mechanisms for automated evaluation, the Software Process Deployment & Evaluation Framework (SPDEF) has been elaborated and is described in this paper.

Method: The proposed framework is based on the application of well-known techniques in Software Engineering, such as Model Driven Engineering and Information Integration through Linked Open Data. It comprises a systematic method for the deployment and evaluation, a number of models and relationships between models, and some software tools.

Results: Automated deployment of the OpenUP methodology is tested through the application of the SPDEF framework and support tools to enable the automated quality assessment of software development or maintenance projects.

Conclusions: Making use of the method and the software components developed in the context of the proposed framework, the alignment between the definition of the processes and the supporting tools is improved, while the existing complexity is reduced when it comes to automating the quality evaluation of software processes.

Keywords: Software Quality, Software Process Engineering, Model-driven Engineering, Information Integration, Linked Open Data

1. Introduction

Software Process Engineering is the area of software engineering that promotes the systematic production of software by following a series of well-defined technical and management processes, in order to maximize their quality [1]. Thus, organizations need to have methods, techniques and tools to implement a comprehensive strategy of continual quality improvement of their software development and maintenance processes. A comprehensive software processes management requires tools for designing, verifying, validating, deploying and evaluating processes. However, the supporting tools often do not provide mechanisms to include explicit definitions of the processes, which causes a significant lack of consistency between process models and the actual deployment of the tools.

Moreover, experimentation in Software Engineering is a relatively new discipline, which aims to find quantitative answers to specific questions [2]. In order to pose formal experiments, evidence about the procedures, tools and resources used to perform the software life cycle activities is needed. From the data and the evidence generated in different software tools for supporting the software process, the strengths and weaknesses of the elements involved in the software lifecycle can be analyzed. However, designing and constructing automatic evaluation methods of such elements is a highly complex task.

This paper presents a framework for the deployment and evaluation of software processes. Software process deployment refers to the implementation of the definitions of the processes on the operational environment. The deployment of processes covers from organizational aspects, such as the implementation of procedures and the acceptance of commitments; technological aspects, such as the configuration and adaptation of tools; and social aspects, such as teamwork and training, among others [3]. In this paper, we will focus on the technological aspect of the deployment, more specifically, on the adaptation of support tools.

The evaluation of software processes refers to the set of activities needed for measuring the quality of the processes and their suitability to the execution environment...
after be deployed. Evaluation is also a broad issue because it involves the evaluation of the activities developed during the execution of the projects, people, tools, etc. This framework supports some of the previous evaluations, although, a detailed scenario of our proposal for conducting technical reviews on certain projects will be presented.

The foundations of this framework lie in the following research hypothesis: (RH1) inconsistencies between process definitions and execution of the projects could be minimized, in part, by customizing and adapting the support tools and by creating specific templates for them; and (RH2) achieving a global and complete view of the information managed by the support tools would enable automating the quality evaluation in software processes.

This framework is based on the application of two well-known approaches in Software Engineering. First, Model-Driven Engineering (MDE) techniques for automating the deployment of process definitions into support tools, and second, Linked Open Data (LOD) for developing information integration solutions intended for evaluating software processes, using the data managed in those tools. Both techniques have proved to provide significant benefits to the community of software engineering. MDE focuses on the design and transformation of models for improving productivity in software development, whereas LOD simplifies the design of the data integration processes required for publishing and consuming data on the Web.

The rest of this paper is organized as follows: Section 2 focuses on different aspects of the systematic management of software process, and the problems that arise in the elaboration of the proposed framework. In Section 3, the elements that are part of the framework are presented, whereas Section 4 describes how its assessment has been carried out. Other research related to our proposal is described in Section 5. Finally, some conclusions and outlines of future research are collected in Section 6.


A business process is defined as a set of activities that are performed in coordinating an organizational and technical environment to achieve a given business objective [4]. Business Process Management (BPM) follow a cycle of continual improvement, starting with its design using some standard modeling language (e.g. BPMN), its verification and validation, its configuration, the enactment of models for its execution in a BPM engine, and its evaluation. Software processes, which are a particular type of business process, consist of a coherent set of policies, organizational structures, technologies, procedures, and the artifacts needed for designing, developing, installing and maintaining a software product [5].

The explicit definition of processes plays a key role in the major initiatives for software process improvement. For example, process modeling is included in the Organizational Process Definition process area of Capability Matu-

rity Model Integration (CMMI) [6], whereas in ISO/IEC 12207, it is included in the Process Improvement group [7]. Modeling languages enable building process descriptions in an homogeneous way, usually by using a graphical notation. These languages help improve the appropriate understanding of the processes by all stakeholders. Software Process Modeling Language (SPML) share common elements, such as activities, resources, work products, actors and rules.

The Object Management Group (OMG) consortium published in 2002 the Software and Systems Process Engineering Metamodel Specification (SPEM), a language designed for modeling software engineering processes. It is definitely the software process definition language most commonly used [8]. The potential benefits of this language seem quite promising, such as reusing methods and processes across organizations, and laying the foundations for the automation of processes, among others [9]. However, this language has not achieved a sufficient level of acceptance in the industry, but they are only used in academic and research fields [10, 11].

Unlike business processes management, controlled by the BPM systems, there are no complete suites for the definition, configuration, implementation and evaluation of software processes. However, in recent years, thanks in part to the rise of the open source movement, numerous support tools for software management and production [12] are appearing. Along with the development of these types of tools, various platforms to promote and foster cooperation between work team members in software projects and to provide support to end users have also emerged. These platforms, called software forges, are evolving towards the concept of a Application Lifecycle Management (ALM) platform. The latter are designed to integrate and coordinate a number of engineering and management software tools, with the aim of covering all or most of the activities of the software life cycle.

Such support tools require, at the beginning of each new project, considerable efforts for their adaption to the specific requirements of the project and the corporate methodology. In addition, due to the slow acceptance of the SPML, the support tools, forges or ALM platforms do not often incorporate capabilities for linking with the explicit definitions or models of the processes, which causes a significant lack of consistency between the process definition and the further execution of the projects. Also, the absence of mechanisms for automation makes the modeling software process, for instance with the SPEM language, unable to offer a sufficient return on investment to be considered interesting to most companies.

In the continuous quality improvement cycle, the evaluation of processes is essential. Therefore, in order to be able to apply improvement mechanisms, it is necessary to measure and analyze the errors, deficiencies or deviations in the actual process execution. Establishing an automated measurement plan by using BPM systems is usually a relatively simple task, typically using real-time
metric monitoring tools and post-mortem analysis engines. In contrast, in the software engineering processes, collecting data from real projects is a complex activity because the ALM integrated platforms are not wide spread and the software process-aware support tools are in their early stages.

In addition to the analysis of metrics and indicators, technical reviews are another important set of control activities in Software Engineering. These activities are usually quite repetitive and require a significant allocation of human resources, as they are often manual activities. Reviews are usually completed at certain checkpoints throughout the software lifecycle, such as at the end of certain phases, milestones, activities, or iterations (in incremental life cycles) or just before delivery to the client [14].

Despite the fact that the analysis of metrics and software revisions are essential activities to improve software quality, it is common that organizations cannot allocate sufficient effort and human resources to make this work. Therefore, research on novel mechanisms to automate technical reviews is needed.

Software management or production support tools host a large amount of information that can be used for the purpose of evaluating software processes. However, data analysis from projects deployed into software process support systems is still an emerging area in Software Engineering. One of the major problems of the integration of the information generated during the projects is the discrepancy with respect to the data models used in the different tools. Therefore, publishing data with a shared information model is essential to facilitate subsequent processes of mapping the information contained in the different tools.

3. SPDEF

The Software Deployment Process and Evaluation Framework (SPDEF) is our proposal for deploying and evaluating software processes on support tools. This framework is aimed at facilitating the adaptation of support software environments for software production and management, so that the consistency between the definitions of the processes and their actual instantiation will be improved. This framework is also intended to simplify the construction of mechanisms to automatically evaluate software processes.

This framework provides a systematic approach towards the deployment and evaluation of processes, a set of models and relations between these models, and a number of support tools. This method consists of four activities, which involve two different roles, by using five types of tools.

3.1. Activities

Below, each of the activities that are part of the proposed method is described in detail.
proprietary, if the data are exposed in a machine processable way or not, and whether free or proprietary formats are used. Nowadays there is much variability with respect to these factors. In the best case, if the support tool offers a LOD-based Application Programming Interface (API), then it would be ready for use in the context of the process evaluation. In the worst case, we may have tools that do not allow users to freely access their data. Figure 2 shows the different mechanisms for turning the supporting tools into open datasets in the LOD cloud.

3.1.4. Develop Integration Solutions Activity
The Systems Engineer role will be responsible for developing data integration solutions in order to automate the execution of quality reviews and analysis of software metrics.

Automating quality reviews of software projects is one benefit of applying the LOD approach to software process support tools. Automation requires defining the evaluation criteria as queries on data exposed by the support tools. Automatically checking the adherence of the projects to the process descriptions, standards, and procedures instituted in the organization, is achieved with this approach. Some of the SPARQL queries which can be issued for that purpose and are currently supported in this framework are outlined below:

- Checking the correct application of UML modeling techniques in Software Engineering, for example, by issuing SPARQL queries on data about the diagrams, elements and connectors exposed from visual modeling tools.
- Checking the completeness of documentary work products, for example, by issuing SPARQL queries on data about the articles exposed from wikis.
- Checking the completion of activities of production and management, for example, by issuing SPARQL queries on data about the project versions and issues exposed from issue-tracking systems.

Another benefit of the integration and the LOD approach is related to software measurement, since additional metrics can be obtained by issuing distributed queries on different datasets, regardless of whether or not these metrics are implemented in the own application business logic or are available from the user interface. For example, an indirect metric that can obtained with this integration approach is the number of code builds (compilations) performed in the continuous integration server for every milestone registered in the project monitoring tool. Automating the calculation of this metric is a complex task if no global and integrated view of the information is available, because the required data are scattered in different tools.

3.2. Models and Relations
According to the MDA terminology [15], there are three kinds of models corresponding to the viewpoints on a system: (1) Computation Independent Models (CIMs), which are business or domain models using a vocabulary that is familiar to the subject matter experts, (2) Platform Independent Models (PIMs), which are models that abstract from technical details, and (3) Platform Specific Models (PSMs) with all of the details required to define how a system uses a particular type of platform.

In the SPDEF framework, a set of models at the levels of abstraction defined above are proposed: at the CIM level, the SPEM standard is used for defining a software process; at the PIM level, we have defined two new models intended to represent processes into tools; and at the PSM level, several models of generic and specific tools are proposed.

3.2.1. Process deployment models
From the MDA perspective, the PIM models are views of a system focusing on its operation while hiding the details necessary for a particular platform. In our framework, we consider these models as a means to represent several areas of interest in Software Engineering, such as, management of work products during the software lifecycle, planning and monitoring of software projects, etc. These models are, therefore, independent of the technologies used, enabling multiple tools can be supported from the same PIM and further changes in business factors, as represented in the PIM, can be easily propagated to more detailed models. The PIMs act as intermediary models to bridge from process models to supporting tool models.

With the aim of contextualizing the elements involved in the software process models with respect to the underlying concepts and relationships of the support tools, we
have designed a number of PIM-level models, which are described below.

- **Software Work Product Model (SWPM).** Common work products in software processes are often documents or source code artifacts, which are managed from either specialized or generic tools. For instance, UML modeling tools or version control systems often provide support for the management of certain types of work products, such as software design diagrams or source code modules. However, other generic tools, such as content management systems, document management systems, and collaborative editing systems (e.g. wikis) can also be used to store evidence of the software process execution. With that aim, a model to define work products with a flexible structure and types of its artifacts is included in this framework, as we can see in Figure 3.

- **Software Project Control Model (SPCM).** Planning and monitoring software projects can be carried out with different tools, from systems based on task management to specialized Gantt-based tools or even simply by using generic spreadsheets. Figure 4 depicts the model taking into account the common elements required to control software development or maintenance projects.

### 3.2.2. Generic tool models

There are several families of generic tools that provide support for carrying out certain activities of the software life cycle. The PSM models are closely linked to the structure of the information managed by such support tools. The models corresponding to three families of tools are presented next.

- **Visual Modeling tool Model (VMM).** In Software Engineering, as well as in other disciplines, it is common to use diagram editing tools based on visual languages. In particular, Unified Modeling Language (UML) editing tools enable building software models that allow you to manage much of the work products of a project. From the characterization and analysis of several UML tools, such as Enterprise Architect, Visual Paradigm for UML, and Rational Rose, the model shown in Figure 5 has been designed. This model enables representing the basic information structure of these UML tools, but without excluding other tools commonly used to model software systems or other entities by using other visual languages.

- **WIKI tool Model (WIKIM).** From the analysis of various systems, such as MediaWiki, Confluence, and

---

[Figure 3: Software Work Product Model](#)

[Figure 4: Software Project Control Model](#)

---

[http://www.visual-paradigm.com/](http://www.visual-paradigm.com/)
Figure 5: Visual Modeling tool Model

Figure 6: Wiki Tool Model

DokuWiki the model depicted in Figure 6 was designed.

• Issue Tracking tool Model (ITM). This model (see Figure 7) was designed from the analysis of the features of task management tools and issue tracking systems, such as Redmine, Jira, and Trac.

3.2.3. Specific tool models

There are more and more support tools or integrated platforms which include several features to cover different aspects of the software process. For example, the Redmine tool, primarily aimed at task management, also includes a code version control, a wiki, a forum system, news, and dynamic contents.

In addition to the conceptualization of the internal information structure of these tools, the PSM models are created by reusing the elements of the generic tool models to meet the multiple capabilities that can provide the support tools. In this way, the PSM generic system model can be reused to obtain final PSM models for concrete tools. We can see the resulting models for three specific tools, Enterprise Architect, MediaWiki, and Redmine, in Figure 8.

https://www.dokuwiki.org/dokuwiki
http://www.redmine.org/
https://www.atlassian.com/en/software/jira
http://trac.edgewall.org/
3.2.4. Relations between the models

There is a close relationship between each of the models above presented. Elements of the SPEM software process models are associated with the elements of the SWPM and SPCM deployment models. These latter are associated, in turn, with the elements of the WIKIM, VMM and ITM generic tool models. Ultimately, the models of specific tools use the models defined in the just previous level. Figure 8 depicts these relationships. Additionally, a graphical representation of the mappings between the SWPM model and the WIKIM model is depicted in Figure 9.

3.3. Implementation and tools

The models and relations between the models collected in this framework were implemented in different software artifacts employing the LOD and MDE approaches. This section describes how the models and the relationships described above have been implemented, as well as the tools built for deploying the software process and for opening the data managed by the software management or production support tools.

3.3.1. Metamodels and transformation rules

SPDEF uses the Eclipse implementation of the Unified Method Architecture (UMA) as the CIM-level software process model. The rest of the models included in this framework, both in the PIM level as well as in the PSM level, were implemented using the same technology as UMA, that is, the Ecore metamodeling language.

In order to automatically derive deployment models (PIM) from process models (CIM), we have implemented a set of transformation rules with the Atlas Transformation Language (ATL) language. This language was used similarly for the implementation of the transformation rules between the deployment models (PIM) and the tool models (PSM).

3.3.2. Tool for process deployment

For designing software process models, we use the Eclipse Process Framework (EPF) environment, an a customized version of Eclipse IDE which offers a comprehensive editor for defining software processes and methodologies. For adaptation of the support tools, we implemented software distributed as a set of plugins for Eclipse. It provides a number of tools for editing and transforming the models involved in the stage of process deployment. The software, called Software Process Deployment Toolkit (SPDT) is based on Java technologies and was built using technologies included in the Eclipse Modeling Project.

Some manual tasks for editing and refinement of models and other automatic tasks for derivation and transformation must be performed to adapt the support tools. For the former, a series of tree-based reflexive editors for all models included in the framework are provided. With these editors, the Process Engineer role will be able to customize the automatically generated models so that they fit the needs of the organization and its tools. ATL is used for the automatic derivation of the deployment and tools models, because besides providing a design language for the transformation rules, it has an execution engine for Model-to-Model (M2M) transformations.

To configure the support tools, the System Engineer role will run some specific scripts. The implementation of these scripts depends on the mechanisms offered by the tool to programmatically manipulate the information it manages. In the technical solution implemented to date, a number of artifacts have been developed as required for deploying on MediaWiki and Enterprise Architect tools. In the case of Enterprise Architect, a Java program was built that, from a model conformant to its metamodel, automatically generates a template project containing the packages, UML diagrams, and the elements defined in the input model. To this end, the API itself released by the vendor is used. In the case of MediaWiki, we used another API that enables connecting to a remote wiki installation and creating or modifying articles on it.

3.3.3. Vocabularies, equivalence axioms, and inference rules

To ensure that the information integration solutions are as independent as possible of the specific tools that are to be integrated, they should use shared vocabularies. A fundamental premise in the LOD approach is to reuse, wherever possible, terms from existing vocabularies, rather than reinvent them when publishing Resource Description Framework (RDF) data. This maximizes the probability that the data can be used by applications that may be tuned to well-known vocabularies, without requiring additional modifications.

As a starting point, we proposed using the Description Of A Project (DOAP) vocabulary for describing the ba-
Figure 9: Relationships between different levels of modeling

Figure 10: Mappings between the SWPM and WIKIM models
sic data of the projects managed with forges or software repositories. However, since this vocabulary does not include many aspects of software processes, such as versions or tasks, it was necessary to develop new vocabularies for the deployment models (SWPM and SPCM) and for the generic tools models (VMM, WIKIM and ITM). These vocabularies were defined according to the RDF Schema standard.

Subsequently, it was necessary to implement the relationships between the PIM and PSM level models, in addition to the relationships with terms of the external vocabularies. For this purpose, the equivalence and specialization axioms owl:equivalentClass, owl:equivalentProperty, rdfs:subClassOf and rdfs:subPropertyOf provided by the LOD/Semantic web, were used. However, there is not always a univocal correspondence between the elements of the models at the different levels. For example, a Package in the VMM model can be a realization of a Documentary WorkProduct or a Section in the SWPM model. In order to tackle this kind of relationship, we must use a semantic reasoner or rule engine to infer logical consequences from a series of facts described as RDF triples.

Finally, the vocabularies were published on the Web, using the Neologism tool, and indexed in the de facto LOD directory so that any RDF client can use them.

3.3.4. Tools for opening data
A set of components is needed to expose data from the software process support tools. Thus, the tools provide interfaces that enable managing HTTP requests on resources identified by URIs, as well as query requests via the SPARQL Protocol and the RDF Query Language. These interfaces return the requested information in any of the serialization formats available for RDF.

- LOD Adapter for issue tracking system APIs. First, we implemented a tool, called Abreforjas, to provide a single access mechanism and a common format for software project data hosted on different task management tools. This tool extracts and normalizes the information stored in the software forges, such as Assembla or Redmine. Abreforjas enables a LOD interface for publishing RDF data using the ITM vocabulary with the information of the projects.

- Extension for publishing LOD on MVC web applications. In order to ease the development of RDF interfaces from applications developed with Model View Controller (MVC) web frameworks, two tools were developed: EasyData/Rails and EasyData/Django. Both utilities enable aligning the attributes and entities of the openable application’s models with the properties and classes defined in RDF vocabularies available on Internet.

- Data adapter for publishing LOD from Enterprise Architect. Enterprise Architect is an UML-based editing tool that does not offer free access to its source code, although it has an API in several languages to query and manipulate data about models of the projects. However, at this time, creating a wrap directly on the Enterprise Architect data source was the chosen decision. To implement that adapter, we opted for D2R Server, a linked data-relational mapper.

4. SPDEF Evaluation
In the previous section, we presented a framework for the deployment and evaluation of software processes. This framework provides a method for deployment and evaluation, a number of models and relationships, and a set of support tools. The technical feasibility of the framework was demonstrated by implementing these tools. This section describes how a more rigorous evaluation of the SPDEF framework was carried out through two evaluation methods, according to the classification proposed in [19].

First, a case study consisting in the deployment of the well-known OpenUP software methodology into a number of support tools is presented. The second evaluation presents a representative use scenario, consisting in the technical quality reviews that are commonly performed on a software project.

4.1. OpenUP Case Study
This section describes how the methodology OpenUP can be readily deployed on a number of software development support tools, such as Enterprise Architect and MediaWiki.

This case study aims to validate the research hypothesis RH1. The OpenUP development methodology, donated by IBM to the Eclipse Foundation, is a lightweight version of the Unified Process. This methodology follows an iterative and incremental approach and its life cycle consists of four stages: Inception, Elaboration, Construction and Transition. The OpenUP methodology is available as a library method developed with the EPF tool, so that its process model can be reused.

From the OpenUP process model, we can derive the corresponding work product model, as a result of running the transformation rules defined between the Unified Method Architecture (UMA) and SWPM metamodels by the ATL engine. In the resulting model, a project that follows the OpenUP methodology will contain a series of documents, each of them having a default section and a number of artifacts classified as UML models. Then, it is necessary to carry out a manual refinement of the model to
match the methodology guidelines and then obtain a consistent structure for the work products. Among other settings, new sections were created and a number of artifacts were classified as software models or other specifications (i.e. textual or item lists). Figure 11 shows the resulting work product model.

After refining the model work product model, it is time to generate the corresponding models for the support tools. Since the Enterprise Architect metamodel consists only of the metamodel for visual modeling tools, the derivation was carried out in a single step. This last model can be refined, if desired, before executing the needed scripts for creating a project template (file with *eap* extension), ready to be used in Enterprise Architect. The generated file will be opened with this tool, so that we can visualize all work items needed to develop the software documentation of the software projects under the OpenUP methodology. Figure 12 shows the workbench of Enterprise Architect while editing a UML example diagram.

One of the benefits that the MDE approach offers is to be able to generate multiple platform-specific models from a single input model. Thus, the transformation engine ATL was invoked for deriving a model for MediaWiki from the work product model. The step of derivation and combination of the final model was also performed in a single transformation. From this model, we could launch the needed scripts for customizing the wiki installation. A wiki article with the requirement document of a given project OpenUP can be observed in Figure 13.

This case study illustrates how you can automatically generate templates for Enterprise Architect and MediaWiki. In this way, the structure of the work products under the OpenUP methodology is normalized.

### 4.2 Quality Review Automation Scenario

A detailed scenario using the framework is presented in order to validate the research hypothesis RH2. This scenario illustrates the steps required to automate the carrying out of technical reviews during the development of software projects. To do this, an integration solution is required. This solution, by using the RDF data exposed
from each of the support tools, enable launching SPARQL queries in order to automate the collection and evaluation of evidence about using the techniques, tools and methods in Software Engineering. These tools must be previously endowed with some of the mechanisms of data exposition cited in the previous section and configured with suitable vocabularies.

Figure 14 depicts the data integration solution designed for this example. This solution, based on the Enterprise Information Integration (EII) pattern, was implemented using the Linked Media Framework (LMF) platform, which include data storing, caching, versioning, reasoning, indexing, and querying capabilities, among others.

LMF has a triple local repository in which the vocabularies included in the framework were loaded. In addition, the inference rules were also included in the semantic reasoner provided by the platform, joint with the common rules of reasoning about the axioms of equivalence and specialization of RDF Schema and Web Ontology Language (OWL).

LMF offers a module for transparently fetching and loading RDF resources on demand from a previously registered set of datasets. Therefore, we set the SPARQL endpoint for Enterprise Architect and the Abreforjas endpoint for Redmine, which allows us to extract the required data.

Next, it is essential to make links between each of the resource identifiers (URIs) that the projects have in the several support tools. In Listing 1, a registry of projects of a given fictional organization, implemented as a set of RDF triples, is presented.

Listing 1: RDF implementation of a registry of internal projects

In the snippet of code before, it is shown how a given sample project is linked, using an equivalence axiom, with the corresponding projects hosted in the datasets of Enterprise Architect and Abreforjas. Furthermore, the project template resulting from the previous deployment of the OpenUP methodology on the support tools is also registered. With the data integration solution described above, developers will be able to build new applications intended for conducting quality reviews of software projects. Below, a number of SPARQL queries illustrating the quality check rules are included.

In order to check the correct application of some practices of Software Engineering, such as the UML modeling techniques, or agile project management, a series of SPARQL queries are issued. For instance, with the query in Listing 2, we can retrieve the actors of the system, which are identified during the phase of analysis of a project, that are not associated with any use case. Another example is the query in Listing 3 aimed at knowing if there are unresolved tasks that have been planned for project milestones whose deadline has already expired.

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix doap: <http://usefulinc.com/ns/doap#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

dc:name "JAVA Web App" ;
owl:sameAs <http://ea.my.org/resource/projects/foo> ;

dc:name "Template Project for OpenUp Methodology" ;
owl:sameAs <http://ea.my.org/resource/projects/openUp> ;

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix doap: <http://usefulinc.com/ns/doap#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

dc:name "JAVA Web App" ;
owl:sameAs <http://ea.my.org/resource/projects/foo> ;

dc:name "Template Project for OpenUp Methodology" ;
owl:sameAs <http://ea.my.org/resource/projects/openUp> ;

Listing 1: RDF implementation of a registry of internal projects

In the snippet of code before, it is shown how a given sample project is linked, using an equivalence axiom, with the corresponding projects hosted in the datasets of Enterprise Architect and Abreforjas. Furthermore, the project template resulting from the previous deployment of the OpenUP methodology on the support tools is also registered. With the data integration solution described above, developers will be able to build new applications intended for conducting quality reviews of software projects. Below, a number of SPARQL queries illustrating the quality check rules are included.

In order to check the correct application of some practices of Software Engineering, such as the UML modeling techniques, or agile project management, a series of SPARQL queries are issued. For instance, with the query in Listing 2, we can retrieve the actors of the system, which are identified during the phase of analysis of a project, that are not associated with any use case. Another example is the query in Listing 3 aimed at knowing if there are unresolved tasks that have been planned for project milestones whose deadline has already expired.

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix doap: <http://usefulinc.com/ns/doap#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

dc:name "JAVA Web App" ;
owl:sameAs <http://ea.my.org/resource/projects/foo> ;

dc:name "Template Project for OpenUp Methodology" ;
owl:sameAs <http://ea.my.org/resource/projects/openUp> ;

Listing 1: RDF implementation of a registry of internal projects

In the snippet of code before, it is shown how a given sample project is linked, using an equivalence axiom, with the corresponding projects hosted in the datasets of Enterprise Architect and Abreforjas. Furthermore, the project template resulting from the previous deployment of the OpenUP methodology on the support tools is also registered. With the data integration solution described above, developers will be able to build new applications intended for conducting quality reviews of software projects. Below, a number of SPARQL queries illustrating the quality check rules are included.

In order to check the correct application of some practices of Software Engineering, such as the UML modeling techniques, or agile project management, a series of SPARQL queries are issued. For instance, with the query in Listing 2, we can retrieve the actors of the system, which are identified during the phase of analysis of a project, that are not associated with any use case. Another example is the query in Listing 3 aimed at knowing if there are unresolved tasks that have been planned for project milestones whose deadline has already expired.

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix doap: <http://usefulinc.com/ns/doap#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

dc:name "JAVA Web App" ;
owl:sameAs <http://ea.my.org/resource/projects/foo> ;

dc:name "Template Project for OpenUp Methodology" ;
owl:sameAs <http://ea.my.org/resource/projects/openUp> ;

Listing 1: RDF implementation of a registry of internal projects

In the snippet of code before, it is shown how a given sample project is linked, using an equivalence axiom, with the corresponding projects hosted in the datasets of Enterprise Architect and Abreforjas. Furthermore, the project template resulting from the previous deployment of the OpenUP methodology on the support tools is also registered. With the data integration solution described above, developers will be able to build new applications intended for conducting quality reviews of software projects. Below, a number of SPARQL queries illustrating the quality check rules are included.

In order to check the correct application of some practices of Software Engineering, such as the UML modeling techniques, or agile project management, a series of SPARQL queries are issued. For instance, with the query in Listing 2, we can retrieve the actors of the system, which are identified during the phase of analysis of a project, that are not associated with any use case. Another example is the query in Listing 3 aimed at knowing if there are unresolved tasks that have been planned for project milestones whose deadline has already expired.

15 https://code.google.com/p/lmf/
PREFIX vmm: <http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#>
SELECT ?actorId ?actorName
WHERE{
  <http://integration.my.org/resource/projects/foobar>
  vmm:packages/vmm:embeddedPackages*/ vmm:elements* ?actorId .
  ?actorId vmm:type "Actor" .
  ?actorId vmm:name ?actorName .

  MINUS {
    ?connId vmm:type "UseCase" .
    ?actorId vmm:connectors ?connId
  } .

  MINUS {
    ?connId vmm:type "UseCase" .
    ?actorId vmm:target ?connId
  } .

  MINUS {
    ?connId vmm:type "Association" .
    ?cduId vmm:type "UseCase".
    ?actorId vmm:connectors ?connId .
    ?connId vmm:target ?cduId
  } .

  MINUS {
    ?connId vmm:type "Association" .
    ?cduId vmm:type "UseCase".
    ?actorId vmm:connectors ?connId .
    ?cduId vmm:connectors ?connId
  }
}
ORDER BY ?actorName

Listing 2: SPARQL query for getting the actors who are not associated with any use case

PREFIX itm: <http://spi-fm.uca.es/spdef/models/genericTools/itm/1.0#>
SELECT ?versionName ?versionDueDate ?issueName ?issueCompletedDate
WHERE{
  <http://integration.my.org/resource/projects/foobar>
  itm:versions ?versionId .
  ?versionId a itm:Version .
  ?versionId itm:name ?versionName .
  ?versionId itm:dueDate ?versionDueDate .
  ?versionId itm:issues ?issueId .
  ?issueId itm:name ?issueName .
  ?issueId itm:completedDate ?issueCompletedDate .
  FILTER (?issueCompletedDate > ?versionDueDate)
}
ORDER BY ?issueCompletedDate

Listing 3: SPARQL query to check whether all the tasks belonging to a completed milestone are closed

5. Related works

One of the objectives of this framework is to automatically adapt the support tools. For this purpose, the principles and techniques of the MDE paradigm and opportunities offered by the automatic transformation of models are used, following a process similar to the MDA methodology\[15\]. The aim is, therefore, to customize certain types of support tools, unlike MDA, which aims at building software systems.

Several authors have proposed mechanisms for deploying and executing software processes modeled with SPEM, by transforming these process models into workflow models to ideally be enacted through the general purpose BPM systems, in order to gain the advantages offered by these environments.

Some of the tools in the market, such as IRIS Process Author o EPF, enable automatically generating workitems for the Visual Studio ALM\[16\] platform and Gantt-based planning templates for Microsoft Project\[17\], respectively. There are some other proposals that provide partial support for automating software process. For instance:

<table>
<thead>
<tr>
<th>Link</th>
</tr>
</thead>
</table>

The above queries illustrate some of the opportunities offered by the LOD approach for evaluating software processes. Choosing a vocabulary or others when designing SPARQL queries depends on the desired level of detail for collecting evidence. In the first examples, we used the generic tools vocabularies, VMM and ITM, for retrieving evidence about the use of UML modeling techniques and the tracking of project tasks, respectively.

In order to check whether the work products were elaborated, the SWPM vocabulary was used. Using this vocabulary for queries is especially recommended in contexts where the work products of the projects are managed both in visual modeling tools and in wiki systems. In this way, regardless of the tool used, the way of access to the data will always be uniform.
a modeling environment and deployment based on the MDA methodology is proposed in [21]; (2) [22] describes a tool that embeds the documentation of software processes within the own development environments; and (3) the proposal for semi-automatically generating extensible development environments based on the Eclipse platform, included in [23]. However, unlike the tools and the works above referred, the method for adapting support tools proposed in this framework is independent of the software platforms used and of the specific process or development methodology.

With regard to the models proposed in the framework, modeling of several types of work products has been an approach commonly used in different model-driven web methodologies [24]. However, there is usually the need to adapt and customize well-known methodologies for their application in certain contexts and specific organizations, something which is referred to as process tailoring. The model of work products included in this framework is flexible with respect to the structure and type of its artifacts, so that they can later be instantiated in generic tools, such as wikis, which can be used as a repository to manage documentary work products in a software process, as shown in different papers [25] [26] [27]. Other related work is presented in [28], in which the authors propose a Domain Specific Language (DSL) for generating wiki instances contextualized for organization corporate strategies.

The second objective of this framework is to ease the development of data integration solutions for process evaluation, by using the LOD approach. A partial implementation of this method, but targeted to the field of scientific information systems, is presented in [29]. In the literature we can also find some other papers related to the evaluation of software processes: an approach to collecting and analyzing metrics collected from different data sources [30]; the detection of inconsistencies between the definition of the processes and the data collected from the projects, by using semantic web technologies [31], and the use of techniques of model relaxing and model changing for dynamically adapting process models [32]. Recently, several software provider companies, led by IBM, have been developing a set of open specifications aimed at simplifying the integration of software development tools by using LOD technologies and REST web services [33]. A summary of the related works is collected in Table 1.

6. Conclusions

This paper presents an extensible framework aimed at tackling two common problems in Software Engineering: the lack of alignment between process definitions and support tools, and the high complexity of conducting automated evaluation procedures of processes.

The framework includes a method for the deployment and evaluation of software processes on support tools, based on the application of the principles and technologies of MDE and LOD. This method comes with a series of models, implemented as Ecore metamodels and RDF Schema vocabularies, and a set of relationships between models, implemented as ATL transformations, RDF axioms, and inference rules. The following describes how the hypotheses in this research have been validated.

The working hypothesis RH1 was that inconsistencies between process definitions and execution of the projects could be minimized, in part, by customizing and adapting the support tools and by creating specific templates for them. In order to validate this hypothesis, our framework provides a tool to adapt in a semi-automatic way a number of tools, such as MediaWiki or Enterprise Architect. In that way, the elaborated work products will be more compliant with the defined ones in a particular methodology. Furthermore, we have described a case study on the deployment of the OpenUP methodology into the previous tools.

The hypothesis RH2 was that achieving a global and complete view of the information managed by the support tools would enable automating the quality evaluation in software processes. To validate this hypothesis, some software components for opening RDF data from issue-tracking systems, MVC-based web applications, and modeling tools, such as Enterprise Architect, were implemented. Furthermore, a detailed description of a data integration scenario for automating quality reviews, by issuing SPARQL queries, on software projects was described.

From the perspective of the deployment of process definitions, the works found in the literature propose techniques for deploying processes (based on MDA and others) into specific project management tools or integrated development environments, particularly into the Eclipse platform. The framework proposed in this paper is more general, enabling the deployment into any type of support tools by using models regardless the kind of process or lifecycle model. From the perspective of the process evaluation, this framework combines the use of semantic technologies to integrate heterogeneous systems, in a similar way to some of the papers gathered in the previous section, but in this case, focusing its application in the calculation of metrics and, particularly, on the execution of quality review. Also, our framework provides novel RDF vocabularies for wiki systems, visual modeling tools and task management tools.

The framework proposed for deploying and evaluating software process is open to further extensions, as described here. This framework includes two PIM level models: SWPM for defining work products, and SPCM for project controlling. Both models enable enriching the elements of the SPEM software process models for its further deployment into support tools. However, there are other aspects of the software process, which are only vaguely treated in the SPEM standard, such as configuration management or people management. Designing PIM and PSM models for these aspects and the subsequent deployment into the proper tools, as well as defining relations between the
Table 1: Classification of related works

<table>
<thead>
<tr>
<th>Relation with SPDEF</th>
<th>Tool/Technologies</th>
<th>Objective</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process deployment</td>
<td>MS Visual Studio ALM</td>
<td>Generation of workitems</td>
<td>[34]</td>
</tr>
<tr>
<td>Process deployment</td>
<td>MS Project</td>
<td>Generation of project planning templates</td>
<td>[35]</td>
</tr>
<tr>
<td>Process deployment</td>
<td>Eclipse IDE</td>
<td>Tailoring of supporting environments for MDA processes</td>
<td>[21]</td>
</tr>
<tr>
<td>Process deployment</td>
<td>Eclipse IDE</td>
<td>Embedding process documentation into development environments</td>
<td>[22]</td>
</tr>
<tr>
<td>Process deployment</td>
<td>Eclipse IDE</td>
<td>Generation of development environments</td>
<td>[23]</td>
</tr>
<tr>
<td>Process deployment</td>
<td>Wiki systems</td>
<td>Generation of repositories of work products</td>
<td>[25, 26, 27, 28]</td>
</tr>
<tr>
<td>Process deployment</td>
<td>UML profiles</td>
<td>Metamodel of the work products in web systems development</td>
<td>[24]</td>
</tr>
<tr>
<td>Process deployment</td>
<td>UML language</td>
<td>Model-driven development of software systems</td>
<td>[15]</td>
</tr>
<tr>
<td>Process evaluation</td>
<td>ETL integration pattern</td>
<td>Measurement of software projects</td>
<td>[30]</td>
</tr>
<tr>
<td>Process evaluation</td>
<td>Semantic Technologies</td>
<td>Detection of inconsistencies</td>
<td>[31]</td>
</tr>
<tr>
<td>Process evaluation</td>
<td>Model Theory</td>
<td>Detection of inconsistencies</td>
<td>[32]</td>
</tr>
<tr>
<td>Process evaluation</td>
<td>Semantic Technologies</td>
<td>Integration of software development tools</td>
<td>[33]</td>
</tr>
<tr>
<td>Process evaluation</td>
<td>Semantic Technologies</td>
<td>Integration of scientific information systems</td>
<td>[29]</td>
</tr>
</tbody>
</table>

separate PIMs, are proposed as future lines of research. Furthermore, the PSM models do not support all the capabilities offered by the different types of tools, but only the most commonly used. In the future, these models can be extended with other features.

Currently, the SPDT component for deploying a process only works with the tools Redmine, Enterprise Architect and MediaWiki. It must be borne in mind that, unlike Enterprise Architect, MediaWiki is a generic tool for collaborative editing, so you are not be able to edit models, nor store information in a structured manner. However, this system could be enhanced using the GraphViz [18] extension, which allows user to edit models, or using Semantic MediaWiki [19], an extension that aims at providing additional capabilities, such as the inclusion of semantic annotations and structured data forms, among others.

With regard to the opening data approach, it is expected to apply and extend the framework focusing on the experimentation in Software Engineering. Software forges manage large amounts of information about projects. All that information can be used as an empirical database for experimentation, similar to how some authors have done [36, 37]. Nevertheless, most of the information collected in the forges is often incomplete due to the distributed and decentralized nature of many of the free/open source software projects themselves, as well as the poor application of standards during the development process and the paucity of detail while enriching the information of projects with metadata. Hence, it is planned to improve the mechanisms for opening data in RDF from the support tools, by applying techniques of natural language processing. In addition, we are exploring the use of OLAP cubes and data mining techniques for improving the evaluation procedures. In this way, we will carry out a comprehensive analysis of the real benefits that can be obtained by applying certain techniques, tools, languages or methods in Software Engineering.

Although our framework provides a set of tools and utilities for semi-automatically supporting the deployment and evaluation of the software process, it is required to manually perform several tasks, such as the definition of the process model or the refinement of the deployment and tools models, as well as defining the evaluation criteria as queries on data exposed by the support tools. Both of them require a significant amount of effort. Furthermore, it is worth noting that while the process is now supported by software, software engineers can still ignore the tools that they themselves have prescribed.

An extended description of the method for the deployment and evaluation, the designed artifacts for the models and relationships, the developed tools, and further information about the case studies are available for reading and downloading from the website supporting this framework [20].

Acknowledgements

This work has been sponsored by grants from the Plataforma para el modelado, personalización y benchmarking en la mejora de procesos normalizados (BESTMARK) project (TSI-020100-2011-396) of the Spanish Ministry of Industry, Tourism and Trade.

http://www.mediawiki.org/wiki/Extension:GraphViz
http://semantic-mediawiki.org/

http://spi-fm.uca.es/spdef
Appendix A. Mappings between the PSM and software packages

Several tables with the mappings between the concepts of the generic tool models and the respective software packages are collected in this appendix.

References


[34] Osellus Inc, IRIS Process Author, [Online; accessed June-2014].


Table A.2: Mappings between VMM concepts and some popular tools (using the naming convention of their corresponding APIs)

<table>
<thead>
<tr>
<th>VMM concept</th>
<th>Enterprise Architect</th>
<th>Visual Paradigm for UML</th>
<th>Rational Rose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModelRepository</td>
<td>org.sparx.Repository</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Project</td>
<td>org.sparx.Package</td>
<td>com.vp.plugin.model.IProject</td>
<td>org.eclipse.core.resources.IProject</td>
</tr>
</tbody>
</table>

Table A.3: Mappings between WIKIM concepts and some popular tools

<table>
<thead>
<tr>
<th>WIKIM concept</th>
<th>MediaWiki</th>
<th>Confluence</th>
<th>DokuWiki</th>
</tr>
</thead>
<tbody>
<tr>
<td>WikiDatabase</td>
<td>MediaWiki installation</td>
<td>Space</td>
<td>DokuWiki installation</td>
</tr>
<tr>
<td>WikiContent</td>
<td>Article</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Category</td>
<td>Article in namespace Category</td>
<td>Label</td>
<td>Namespace</td>
</tr>
<tr>
<td>Article</td>
<td>Article in namespace Main</td>
<td>Page</td>
<td>Page</td>
</tr>
<tr>
<td>File</td>
<td>Article in namespace File</td>
<td>Attachment</td>
<td>Wiki markup with a external link</td>
</tr>
<tr>
<td>Image</td>
<td>Article in namespace Media</td>
<td>Image</td>
<td>Media File</td>
</tr>
<tr>
<td>User</td>
<td>Article in namespace Users</td>
<td>User</td>
<td>User</td>
</tr>
<tr>
<td>Section</td>
<td>Wiki markup with heading of level 2</td>
<td>Section macro</td>
<td>Wiki markup with heading of level 3</td>
</tr>
<tr>
<td>SectionContent</td>
<td>Wiki markup</td>
<td>Wiki markup</td>
<td>Wiki markup</td>
</tr>
<tr>
<td>Paragraph</td>
<td>Wiki markup with simple text</td>
<td>Wiki markup with simple text</td>
<td>Wiki markup with simple text</td>
</tr>
<tr>
<td>ItemList</td>
<td>Wiki markup with a bullet list</td>
<td>Wiki markup with a bullet list</td>
<td>Wiki markup with a bullet list</td>
</tr>
<tr>
<td>Item</td>
<td>Wiki markup with a list item</td>
<td>Wiki markup with a list item</td>
<td>Wiki markup with a list item</td>
</tr>
</tbody>
</table>

Table A.4: Mappings between ITM concepts and some popular tools

<table>
<thead>
<tr>
<th>ITM concept</th>
<th>Redmine</th>
<th>Jira</th>
<th>Trac</th>
</tr>
</thead>
<tbody>
<tr>
<td>IssueTrackingDatabase</td>
<td>Redmine installation</td>
<td>Jira installation</td>
<td>-</td>
</tr>
<tr>
<td>Project</td>
<td>Project</td>
<td>Project</td>
<td>Trac installation</td>
</tr>
<tr>
<td>Member</td>
<td>Member</td>
<td>Membership</td>
<td>-</td>
</tr>
<tr>
<td>Version</td>
<td>Version</td>
<td>Version</td>
<td>Version</td>
</tr>
<tr>
<td>Tracker</td>
<td>Tracker</td>
<td>Issue type</td>
<td>Ticket Type</td>
</tr>
<tr>
<td>Issue</td>
<td>Issue</td>
<td>Issue</td>
<td>Ticket</td>
</tr>
<tr>
<td>IssueCategory</td>
<td>Issue category</td>
<td>Issue label</td>
<td>-</td>
</tr>
<tr>
<td>IssueDependency</td>
<td>Related issues</td>
<td>Issue linking</td>
<td>-</td>
</tr>
<tr>
<td>User</td>
<td>User</td>
<td>User</td>
<td>User</td>
</tr>
<tr>
<td>Role</td>
<td>Role</td>
<td>Project Role</td>
<td>Role</td>
</tr>
</tbody>
</table>