A MODEL-DRIVEN APPROACH TO MANAGING AND CUSTOMIZING SOFTWARE PROCESS VARIABILITIES

Fellipe Araújo Aleixo[1,2], Marília Aranha Freire[1,2], Wanderson Câmara dos Santos[1] and Uirá Kulesza[1]
[1]*Federal University of Rio Grande do Norte (UFRN), Natal, Brazil*

[2]*Federal Institute of Education, Science and Technology of Rio Grande do Norte (IFRN), Natal, Brazil*

Keywords: Software Process, Software Product Lines, Model-Driven Development.

Abstract: This paper presents a model-driven approach to managing and customizing software process variabilites. It promotes the productivity increase through: (i) the process reuse; and (ii) the integration and automation of the definition, customization, deployment and execution activities of software processes. Our approach is founded on the principles and techniques of software product lines and model-driven engineering. In order to evaluate the feasibility of our approach, we have designed and implemented it using existing and available technologies.

1 INTRODUCTION

Nowadays, the importance of using software processes is already consolidated and is considered fundamental to the success of software development projects. Large and medium software projects demand the definition and continuous improvement of a software process in order to promote the productive development of high-quality software. Customizing and evolving existing software processes to address the variety of scenarios, technologies, culture and scale is a recurrent challenge required by the software industry. It involves the adaptation of software process models for the reality of their projects. Besides, it must promote the reuse of past experiences in the definition and development of software processes for the new projects. The adequate management and execution of software processes can bring a better quality and productivity to the produced software systems.

In this context, automated tools supporting the definition, customization and deployment are increasingly necessary. Although there are already many existing tools to specify processes (IBM 2010) (EPF Project 2009), there is a strong need to develop tools, technologies and techniques that help: (i) the management of components and variations of such processes; and (ii) the automatic composition and derivation of these elements to generate a customized process for a project. Furthermore, we know that the definition of a software process is a complex activity that requires much experience and knowledge of many areas and disciplines of software engineering. Our main research question is thus related to: how a software organization can reuse existing software processes by rapidly and automatically allowing their customization for new projects?

In this paper, we propose an approach that supports: (i) the variability management of software processes; and (ii) the automatic product derivation of customized specifications of software processes. Besides, it also allows automatically transforming these customized software processes to workflow specifications, which can be deployed and executed in existing workflow engines. Our approach is founded on the principles and techniques of software product lines (Pohl, Bockle and Van der Linden 2005) and model-driven engineering (Kleppe, Warmer and Bast 2003). In order to evaluate the approach feasibility, we have implemented it using several model-driven technologies. The software processes are specified using Eclipse Process Framework (EPF). The variability management and product derivation of software processes has been implemented as an extension of an existing product line tool, called GenArch (GenArch Plugin 2009). Finally, ATL and Acceleo (OBEO 2009) transformation languages are adopted to transform

Araújo Aleixo F., Aranha Freire M., Câmara dos Santos W. and Kulesza U. (2010).
A MODEL-DRIVEN APPROACH TO MANAGING AND CUSTOMIZING SOFTWARE PROCESS VARIABILITIES.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Information Systems Analysis and Specification*, pages 92-100
DOI: 10.5220/0002910300920100

EPF process to jPDL workflow language specifications in order to enable the deployment and execution of software processes in the JBoss BPM workflow engine.

The remainder of this paper is organized as follows. Section 2 presents existing research work on software processes reuse by identifying several challenges in the variability management of software processes. Section 3 gives an overview of the main elements and functionalities of our approach. Section 4 describes the approach implementation using existing model-driven technologies. Finally, Section 5 presents the conclusions and points out future work directions.

2 SOFTWARE PROCESS REUSE

Over the last years, several approaches have been proposed that explore the development of software product lines (SPLs) (Pohl, Bockle and Van der Linden 2005). The main aim of these approaches is to maximize reuse and minimize costs by promoting the identification and management of commonalities and variabilities (common and variable features) of software families. Software product line engineering promotes the effective reuse of software artifacts based on the organization of similar artifacts according to commonalities and variabilities (Rombach 2005). A common and flexible architecture is designed to address the commonalities and variabilities of the SPL. Finally, a set of reusable assets is implemented following the SPL architecture. After the design and implementation of the SPL architecture and code assets, which is called domain engineering, new applications (products) can be easily derived by reusing and customizing the code assets developed for the SPL architecture. Currently, there are some existing tools, such as Gears (Gear/BigLever Software 2009), pure::variants (Pure::Variants 2009) and GenArch (GenArch Plugin 2009), which automate the automatic derivation of new applications/products from existing code assets. They facilitate the streamline selection, composition and configuration of code assets.

In the software development process scenario, recent work has been developed to allow the reuse of process assets, in the same way that code assets can be reused. The Eclipse Process Framework (EPF Project 2009) is one of these initiatives. It facilitates the definition of software processes using: (i) the UMA (Unified Method Architecture) metamodel; (ii) a supporting tool (EPF Composer); and (iii) content (process asset) that can be used as the basis for a wide range of processes. The EPF Composer allows authoring, configuring and publishing methods. You can add, remove and change process elements according to your team and project needs. In other words, the EPF Composer allows software development processes be extended and customized in a simple way (Haumer 2007).

Although the EPF already provides some support to specify and define software processes, it does not allow the representation and automatic customization of existing software processes. Next, we present some recent research work that proposes the adoption of SPL techniques to enable the automatic management, reuse and customization of software processes.

Rombach (Rombach 2005) presents the first studies to describe the term Software Process Line. His proposal suggests the organization of families of similar processes. It has been applied in small domains and points out the feasibility of applying this approach in more general contexts. However, his work does not define any approach or tools to effectively promote the reuse of software processes.

Xu et al (Xu, et al. 2005) present a definition of a standardized representation and retrieval of process components. The focus is on: (i) the specific components organization of a process and its representation; and (ii) the recovery process definition based on the reuse of existing components. The main drawback of their approach is that it requires high expertise for the representation and retrieval of components.

Barreto et al (Barreto, Murta and Rocha 2009) propose an approach to the componentization of legacy software processes. Their strategy aims to facilitate the achievement of expected results for maturity models. This work states that make processes reusable is a costly task, because many different situations must be provided and addressed by components, lines and features. The work is restricted to the definition of reusable process fragments, and it does not propose any automation for the effective reuse.

3 A MODEL-DRIVEN APPROACH FOR PROCESS DEFINITION, CUSTOMIZATION AND EXECUTION

In this section, we present an overview of our

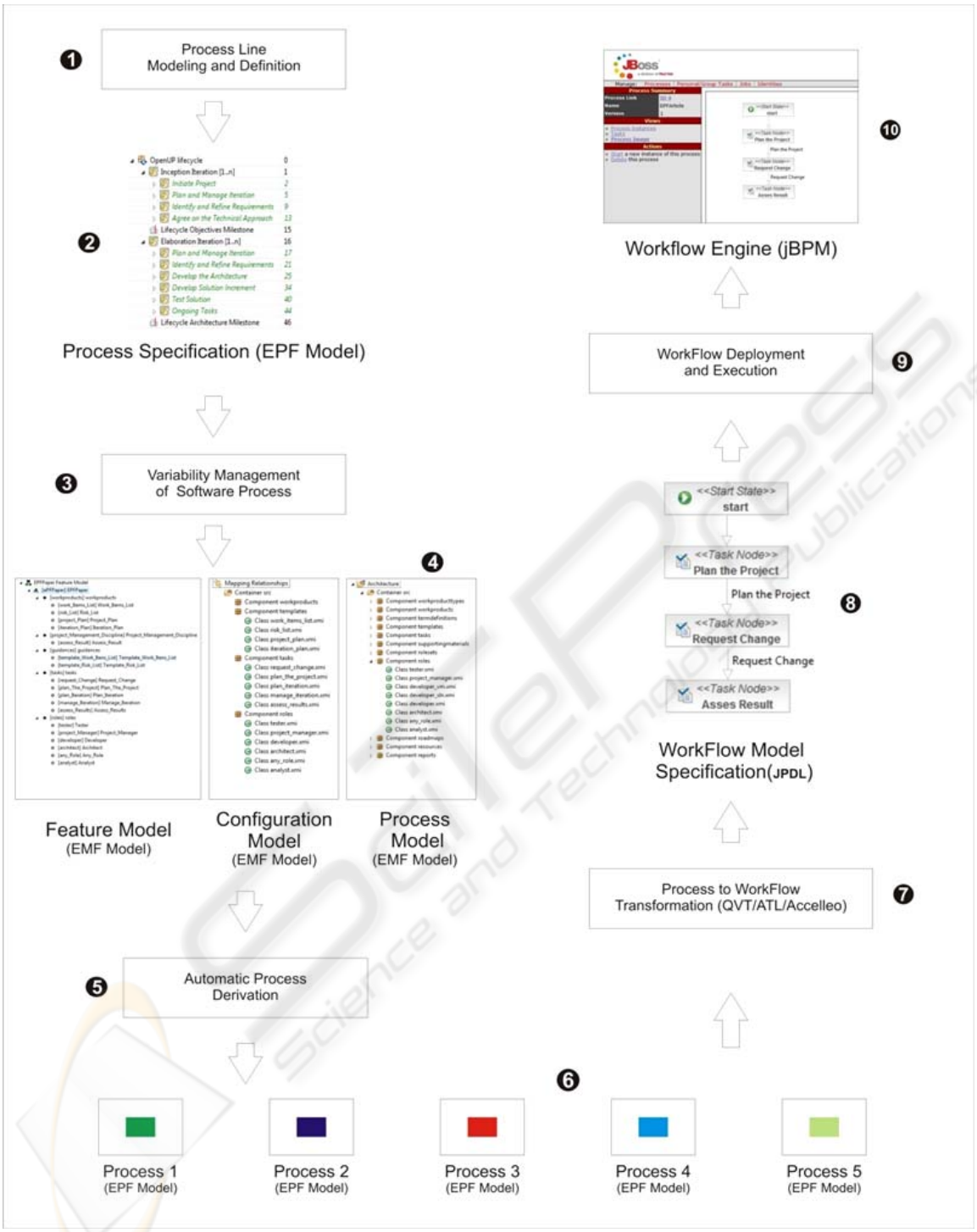

Figure 1: Approach Overview.

approach for process definition, customization and execution. It is founded on the principles and techniques of software product lines and model-driven engineering. Figure 1 illustrates the main elements of our approach and their respective relationships. Next we briefly explain the activities of the proposed approach.

The first stage of our approach is the software process modelling and definition (steps 1 and 2 in Figure 1). Existing tools such as EPF provides

support to address it by using the UMA metamodel. After that, our approach concentrates on the variability management of software process elements. This second stage consists on the creation of a variability model (e.g. feature model) that allows specifying the existing variabilities of a software process (steps 3 and 4). A product derivation tool can then be used to allow the selection of relevant features from an existing software process, thus enabling the automatic derivation of customized specifications of the software process addressing specific scenarios and projects (steps 5 and 6). Finally, our approach supports the automatic transformation of the software process specification to a workflow specification (steps 7 e 8) in order to make possible their deployment and execution in a workflow engine (steps 9 and 10). Through these transformations, the sequence of activities of the process is mapped to a workflow definition.

In order to evaluate the feasibility of our approach, we have designed and implemented it using existing and available technologies. Figure 1 also provides an overview of the implementation of our approach. The process specification is supported by EPF composer using the UMA metamodel (step 2 in Figure 1).The variability management of the EPF specifications is addressed by GenArch product derivation tool (Cirilo, Kulesza and Lucena, A Product Derivation Tool Based on Model-Driven Tecniques an Annotations 2008) (E. Cirilo, U. Kulesza and R. Coelho, et al. 2008b) (Cirilo, Kulesza and Lucena, Automatic Derivation of Spring-OSGi based Web Enterprise Applications 2009). This tool was extended to explicitly indicating which variabilities in a feature model are related to different process fragments from an EPF specification (step 4). The tool uses this information to automatically derive customized versions of a software process (step 6). Finally, we have implemented model-to-model transformations (M2M) codified in ATL/QVT (OMG 2009) to allow the translation of the EPF specification of an automatically customized process to JPDL model elements (step 7). This JPDL specification is then processed by a model-to-text (M2T) transformation implemented using Acceleo language (OBEO 2009) to promote the generation of Java Server Faces (JSF) web forms from a JPDL workflow specification (step 8). These web forms can then be deployed and executed in the JBoss Business Process Management (jBPM) workflow engine. Section 4 describes our approach in action by detailing a customization example of a software process.

Our approach brings several benefits when compared to other existing research work (Barreto, Murta and Rocha 2009) (Rombach 2005) (Xu, et al. 2005). First, it promotes the variability management of existing software processes by allowing to explicitly specifying which process fragments (activities, guides, roles, tasks, etc) represent variabilities (optional and alternative) to be chosen and customized when considering specific projects. Second, it allows automatically deriving, deploying and executing software processes in workflow engines by supporting the systematic transformation of process specifications to workflow specifications. Last but not least, the approach is flexible enough to allow the adoption of process and workflow specifications defined in different languages and notations, as well as to promote the adoption of different tools to process definition, automatic derivation, deployment and execution.

4 IMPLEMENTING THE MODEL-DRIVEN APPROACH

In this section, we present the approach implementation by exploring the adopted techniques to managing software process variabilities and deploying software processes in workflows engines.

4.1 Managing Variabilities in Software Processes

Figure 2 presents a fragment of a case study developed in the context of research and development projects of a technical educational organization (Aleixo, et al. 2010). It illustrates three projects of software development, which are: (i) an integrated academic management information system, called SIGA; (ii) a professional and technological education information system, called SIEP; and (iii) an enterprise system development project, called PDSC. Each project used a customized version of the OpenUP process (EPF Project 2009). The detailed analysis of these OpenUP customizations allowed us identifying and modelling the commonalities and variabilities of this process family. Due to restriction space, in this paper we only focus on the project management discipline.

Figure 2 presents the details of the plan project task of the project management discipline. Some steps of this task were performed in every project – the commonalities, such as: (i) establish a cohesive team; (ii) forecast project velocity and duration; (iii)

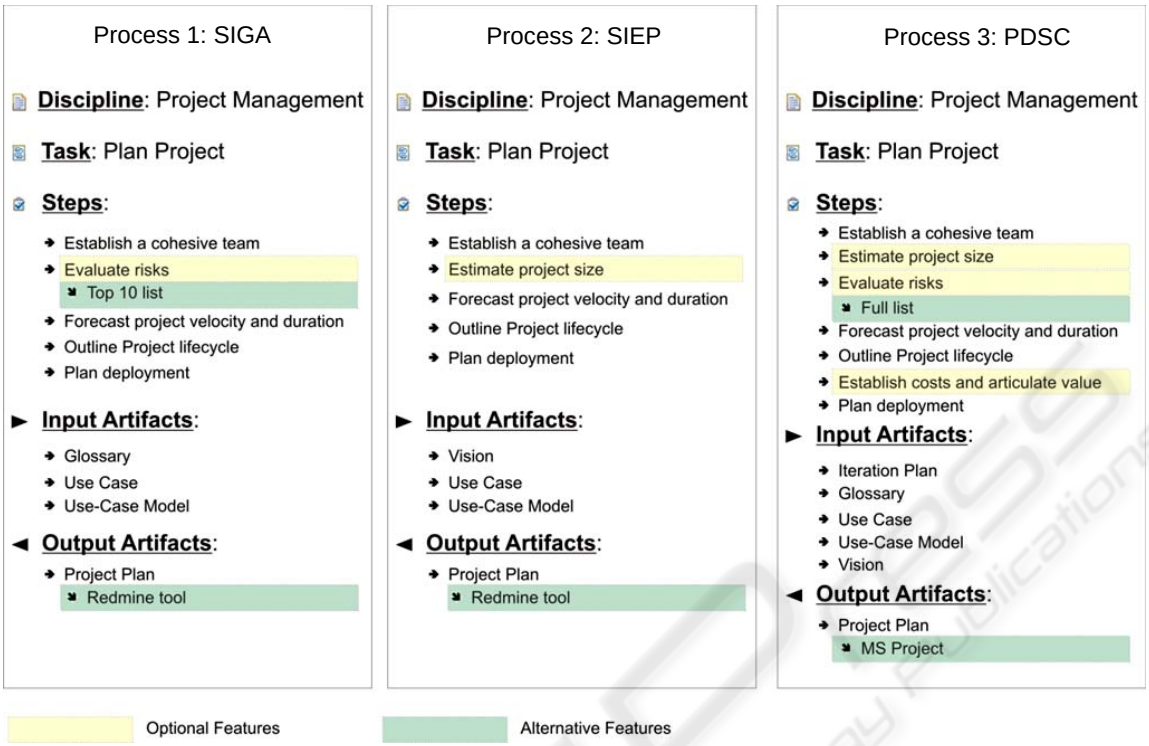

Figure 2: Fragment of Case Study Result.

outline project lifecycle; and (iv) plan deployment. Some steps can be executed or not (optional features), such as: (i) evaluate risks; (ii) estimate project size; and (iii) establish costs and articulated value. Some steps include the use of specific artefacts, which should demand the change of original document template provided by the OpenUP (alternative features). Examples of such alternative templates are: (i) risk list template – that can be top 10 or full list; and (ii) project plan template – that can be specified using the Redmine or MS-Project tools. Figure 3 shows the correspondent feature model for this fragment of the project management discipline.

The variability management in a software process is based on the used representation notation. One of most cited notation is the SPEM (OMG 2010), an initiative of the OMG. In our work, we have adopted an evolution of SPEM, called Unified Method Architecture – UMA (Eclipse Foundation 2010), which is supported by the Eclipse Process Framework – EPF (Eclipse Foundation 2009). EPF was used to specify a software process line that models a family of processes that shares common and variable features. The software process line maintains all the process elements that can be found in any process to be customized from it. It allows systematically reusing common process content elements and fragments of existing software processes.

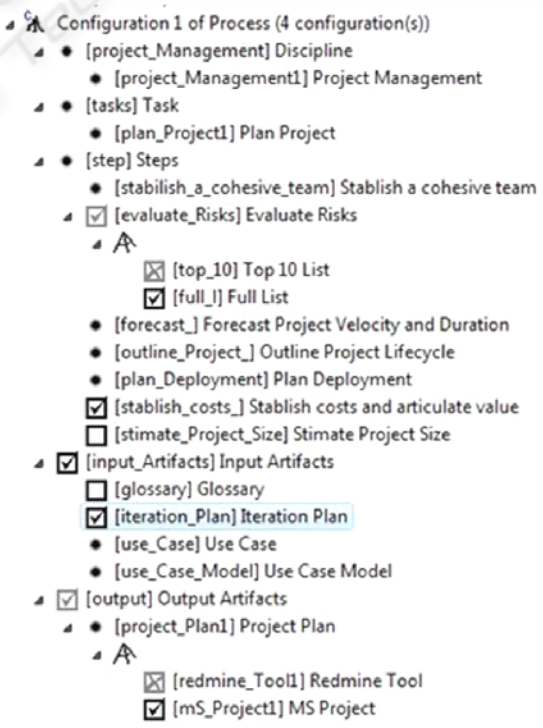

Figure 3: Feature Model Resultant for the Case Study.

The variability management of the software process line is supported by a product derivation tool. This tool enables us to relate variability models (e.g. feature models) to the process content elements. This is a similar strategy adopted by existing product derivation tools to manage the variabilities of software product lines. In our approach, we have adapted an existing product derivation tool, called GenArch, to support the variability management of software process lines. The original implementation of GenArch provides three models: (i) feature model – that represents the commonalities and variabilities; (ii) architecture model – that represents all the code assets implemented for a software product line; and (iii) configuration model – that defines the mapping between features and code assets in order to enable the automatic product derivation. To enable the usage of GenArch in the software process line context, we replaced our architecture model by the EPF process model. It allows specifying how specific variabilities (optional and alternative features) from a feature model are mapped to existing elements from a process specification. Figure 5(A) shows an example of the variability management of process lines for project management process activities. As we can see, the feature model is used to indicate the optional, alternative and mandatory features of an existing process line.

The configuration model defines the mapping of these features to process elements. The complete configuration model is automatically produced from feature variabilities annotations that are inserted in the EPF process specification.

Figure 4 shows an example of feature annotation inside the Assess_Result activity from an EPF specification. As we can see, each annotation defines the name (Assess_Result), parent (tasks) and type (optional) of the feature that the related artefact represents.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- @Feature(name=Asses_Results, parent=tasks,
                type=optional) -->

<org.eclipse.epf.uma:TaskDescription xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:org.eclipse.epf.uma=
        "http://www.eclipse.org/epf/uma/1.0.5/uma.ecore"
    xmlns:epf="http://www.eclipse.org/epf" epf:version="1.5.0"
    xmi:id="_a3uz4LBYEdm7Eph_l9Cn9w"
    name="assess_results,_0l53cMlgEdmt3adZL5Dmdw"
    guid="_a3uz4LBYEdm7Eph_l9Cn9w"
    changeDate="2007-05-01T13:24:08.202-0300"
    version="1.0.0">
  ...
```

Figure 4: Feature Annotation in an EPF specification.

The following process variabilities have been found in the process line case study that we have already modelled and specified: (i) optional and alternative activities in process workflows; (ii) optional and alternative steps from process activities; (iii) optional and alternative specification templates for any specific tool or activity; and (iv) optional and alternative technology developer guides that provides principles and guidelines to adopt specific technologies (CASE tools, modelling and programming languages, API libraries, components and frameworks, and so on). Besides, we are currently exploring fine-grained variabilities (parameters, variables, text portions) that can occur as part of the descriptions of process activities and steps.

Due to restrictions space, this paper does not present additional details about these variabilities. Additional information about process line variabilities modelling can be found in (Aleixo, et al. 2010). After specifying the mapping between variabilities in the feature model to the process elements from an EPF specification, GenArch tool can automatically derive customized versions of a software process line. This stage is similar to what is called product derivation (Clements 2002) in software product line approaches.

During the process derivation, the process engineer chooses the desired variabilities (optional and alternative features) in a feature model editor. Next, the GenArch tool processes the mappings specified in the configuration model to decide which process elements will remain in the final customized process according to the variabilities selection. Resolution of feature constraints and process component dependencies are also computed by the tool during this step of process customization. Finally, after all this processing, the tool is responsible to produce the only EPF specification that represents a customized process to be adopted by a specific project. After the feature selection, the Genarch can be used to generate a new process that makes sense in the features selected in the feature model. Figure 5(B) illustrates two examples of feature selection (configuration1, configuration2) that are processed by GenArch tool to produce two different set of project management activities for specific projects (SIGA, SIEP, and PDSC).

4.2 Deploying and Executing a Software Process in a Workflow Engine

Nowadays, organizations are investing in

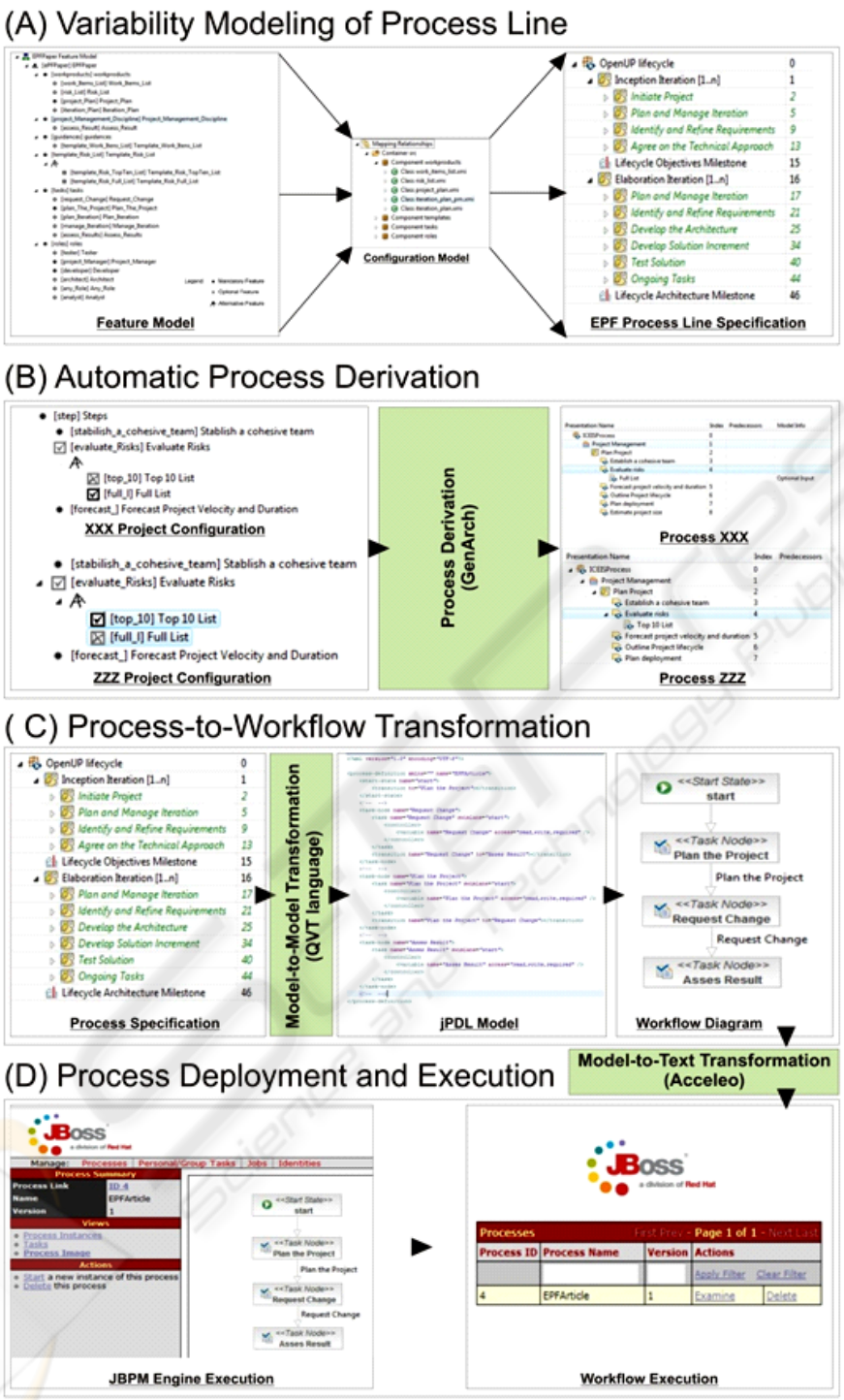

Figure 5: Approach Implementation.

information technology to support their processes. With this increasing need to control and improve processes, we include the concept of Business Process Management (BPM), which in essence is the union of resources in information technology to the analysis of business management focused on improving business processes.

Our approach allows automatically deploying and executing a customized software process automatically derived by GenArch in the jBPM

workflow engine. jBPM (Hat 2009) is a framework of JBoss that allows the creation of business solutions based on business processes using graphical notations and graph-oriented programming. It also provides a workflow engine. We use the jBPM engine to run and monitor software process activities, which were previously defined in EPF process specification and customized by GenArch tool. In our approach, we have implemented transformations that automatically convert the EPF process to the jPDL workflow specification language. This language is used to specify business processes graphically in terms of tasks, wait states, timers, automated actions, and so on. This model-to-model transformation (EPF process specification to jPDL specification) was implemented using the ATLAS Transformation Language (ATL) inside the Eclipse platform. ATL is an implementation of the QVT (Query/Views /Transformations) transformation language (OMG 2009), which is the OMG's standard language for expressing queries, views and transformations on MOF models. Figure 5(C) shows how an EPF customized specification produced as result of the variability management of the process line (Figure 5 A and B) can be automatically translated to jPDL workflow specifications. It can observed that many activities ("plan the project", "request change" and "assess result") are present in both textual and graphical jPDL specification.

The jBPM enables from a definition of a jPDL workflow model, the creation of Java Server Faces forms implementations to monitor the process flow. This monitoring functionality is responsible to store information about the tasks and or decisions taken during the process execution. In order to generate a process definition archive, in jPDL schema, and the related JSF forms for the jPDL workflow specification, we implemented a model-to-text transformation using Acceleo (OBEO 2009). This is a code generation tool that allows transforming models to code. We also generated the "forms.xml" file, which is a XML file that matches each specific task node to a JSF form. All of these files were generated in a jPDL Eclipse project. Through of simple configurations, this project can be deployed in the jBPM workflow engine.

After the deployment of the process workflow in the jBPM engine, the user can request the start of a new instance of the process. Figure 5(D) shows the result of the deployment of the process previously customized and generated by GenArch tool. When starting the execution of a new instance of the process, the user can visualize the actual state of the specific process – that presents details of the activity that have to be done. After the execution of each activity, the user notifies the workflow engine that requests the user to enter some information about the activity in a specific JFS form. All the information is stored in a specific database, related to the process instance. Finally, the workflow engine shows that a new activity is now required. All these steps are repeated for each activity until the end of the process, when the end state of the workflow was reached.

5 CONCLUSIONS

In this paper, we presented a model-driven approach to managing and customizing software processes variabilities. Our approach also provides support to the execution of the customized process in a workflow engine. The approach has been implemented and validated using existing model-driven and software product line technologies. The main benefits of our approach are: (i) the variability management of software processes that directly contributes to productivity improvement when producing customized software processes to related projects; and (ii) the integration and automation of the process definition, customization, deployment and execution. Additionally, our approach has been designed in a flexible way that allows its easy adaptation to deal with new technologies (e.g., new process or workflow specification notations or languages, new model-driven technologies).

As a future work, we intend to apply and evaluate our approach to more extensive and complex software process customization scenarios. We are currently refining the approach to apply it in an industrial scenario of a company that defines and reuses its processes using the Rational Unified Processes (RUP) framework. Additional details about the approach and its implementation can be found in (Aleixo, et al. 2010).

ACKNOWLEDGEMENTS

This work was supported partially by Brazilian Research Council (CNPq) under grants: No. 313064/2009-1, No. 552010/2009-0, and No. 480978/2008-5.

REFERENCES

Aleixo, Fellipe A., Marília A. Freire, Wanderson C. Santos, and Uirá Kulesza. 2010. http:// softwareprocesslines.blogspot.com/ (accessed 01 2010).

Barreto, A. S, L.G.P Murta, and A. R Rocha. "Componentizando Processos Legados de Software Visando a Reutilização de Processos." Ouro Preto: Anais do VIII SBQS, 2009.

Cirilo, Elder, U. Kulesza, and C. Lucena. "Automatic Derivation of Spring-OSGi based Web Enterprise Applications." *ICEIS*, 2009.

Cirilo, Elder, U. Kulesza, and C. Lucena. "A Product Derivation Tool Based on Model-Driven Tecniques an Annotations." *Journal of Universal Computer Science*, 2008, nº 8 ed.

Cirilo, Elder, U. Kulesza, R. Coelho, C. J.P. Lucena, and A. von Staa. "Integrating Component and Product Lines." *ICSR*, 2008b.

Clements, Paul. *Software Product Lines: Practices and Patterns.* Boston: Addison-Wesley, 2002.

Eclipse Foundation. *Eclipse Process Framework (EPF) Composer 1.0 Architecture Overview.* 2010. http://www.eclipse.org/epf/composer_architecture/ (accessed 01 2010).

Eclipse Process FrameWork. 2009. http:// www.eclipse.org/epf/ (accessed 08 2009).

EPF Project. *EPF.* 2009. http://www.eclipse.org/ epf/ (accessed November 2009).

Gear/BigLever Software. *Gears.* 2009. http://www.biglever.com (accessed November 2009).

GenArch Plugin. *Generative Architectures Plugin.* 2009. http://www.teccomm.les.inf.puc-rio.br/genarch/ (accessed November 2009).

Hat, J. *JBPM.* 2009. http://labs.jboss.com/jbossjbpm/ (accessed 09 2009).

Haumer, P. *Eclipse Process Framework Composer:Part 1: Key Concepts.* http://www.eclipse.org/ epf/general/EPFComposerOverviewPart1.pdf, 2007.

IBM. *Rational Method Composer.* 2010. http://www-01.ibm.com/software/awdtools/rmc (accessed 01 2010).

Kleppe, Anneke G., Jos B. Warmer, and W. Bast. *MDA Explained: The Model Driven Architecture: Pratice and Promisse.* Addison-Wesley, 2003.

OBEO. *Acceleo: MDA generator.* 2009. http:// www.acceleo.org/pages/home/en (accessed 10 2009).

OMG. *OMG: QVT Specification.* 2009. http:// www.omg.org/spec/QVT/1.0/ (accessed 10 2009).

Software & Systems Process Engineering Metamodel Specification (SPEM). 2010. http://www.omg.org/ spec/SPEM/2.0/.

Pohl, Klaus, G. Bockle, and F. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques.* New Yourk: Springer Berlin/Heidelberg, 2005.

Pure::Variants. 2009. http://www.pure-systems.com (accessed November 2009).

Rombach, Dieter. "Integrated Software Process and Product Lines." In *Unifying the Software Process Spectrum*, 83-90. Berlin / Heidelberg: Springer, 2005.

Xu, Ru-Zhi, H. Tao, C. Dong-Sheng, X. Yun-Jiao, and Q. Le-Qiu. "Reuse-Oriented Process Component Representation and Retrieval." *The Fifth International Conference on Computer and Information Technology*, 2005.