|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | A Model-Driven Approach to MAnaging and Customizing software process Variabilities | | Marília Aranha Freire1,2, Fellipe Araújo Aleixo1,2, Uirá Kulesza1 | | 1Federal University of Rio Grande do Norte (UFRN), Natal, Brazil  [uira@dimap.ufrn.br](mailto:uira@dimap.ufrn.br) | | 2Federal Institute of Education, Science and Technology of Rio Grande do Norte (IFRN), Natal, Brazil  [marilia.freire@ifrn.edu.br](mailto:marilia.freire@ifrn.edu.br), [fellipe.aleixo@ifrn.edu.br](mailto:fellipe.aleixo@ifrn.edu.br) | |

Keywords: Software Process, Software Process Lines, Metrics.

Abstract: This paper presents a .

# Approach_Overview.jpg1 INTRODUCTION

Nowadays,

In this context,

In this paper,

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 THE based approach

This work is based in an approach defined by Aleixo et al. [REF -ICEIS]. The overview of the initial definition of the approach could be seen in Figure X. The main goal of this approach is to allow the derivation of specific software development process specification according with the Eclipse Process Framework – EPF. This software development process fits the specific needs of a software development project. The approach applies the software product line concepts [REF] to software development process, reinforcing the premise that software processes are software too [REF].

As shown in Figure X, the original approach could be defined in three stages: (i) process definition, (ii) process derivation, and (iii) process deployment and execution. In the initial stage a specific software process family is selected, as a set of software process that shares the same structure and some common process elements [REF]. After the analysis of each member of the software process family, identifying the commonalities and variabilities among the individuals, an EPF process specification is created and the variation points are annotated in the specification. These annotations could be recognized by a product derivation tool, adapted to this purpose, and interpreted to generate the models necessary to the product derivation: (i) process model, (ii) variability model, and (iii) configuration model [REF].

The second stage is intended to the software process derivation. The execution of this stage is motivated by a new software development project. After the analysis of the particularities for this new software process, a feature selection is done in an instance of the variability model. With this feature selection, a product derivation tool could be used to derivate such specific process. The result of this derivation is an EPF process specification according to the selected features.

In the third stage, the approach provides the deployment and execution of the software process specification with the support of a workflow engine. To able this intent, the process specification have to be transformed in a workflow specification. The transformation between these two models is possible with the support of model-to-model transformation language as operational QVT [REF]. To able the generation of the deployment files, beyond the workflow specification are needed some well-known information about the future execution of the software process, information such as: (i) number of iterations in each phase (assuming that the process is based in the Unified Process – UP), (ii) people playing specific roles, (iii) milestones dates, (iv) release dates, etc. The workflow specification and the well-known execution information are used in a model to text transformation to generate the workflow deployment files. These generated files are likely to be deployed in a workflow engine.

After the deployment in the workflow engine, a new instance of this process could be started. The process execution could be monitored through the workflow engine console, which coordinates the interaction of the process stakeholders and the process instance execution. The workflow engine console indicates which the actual activity that has to be done in the process execution, and also could store some information about the execution, such as: (i) start and end date of an activity, (ii) some related observations, (iii) produced artifacts, etc.

# 3 Extended APPROACH FOR compose metrics in a PROCESS DEFINITION, CUSTOMIZATION AND EXECUTION

In this section, we present an overview of our approach that integrates software metrics in a process definition, customization and execution. It is based on the approach outlined in section 2 of this document. illustrates the main elements of our approach and their respective relationships. This new approach combines a metric model with a process model to enable the automatic metrics collection during the execution of a software process. Next we briefly explain the stages of the proposed approach.

Initially we have to specify phase like the original one. In this phase we have two new activities: the metrics specification and the metrics selection. The first activity refers to metrics addition in the software process. It is realized when the process definition using tools such as EPF, like the based approach. After that, the second new activity is concerned to the metrics selection that we can get from the process feature model that now includes options concerning the possible metrics previously defined.

On the second stage we need to model the composition of metrics in a metric model. This modelling is made specifying for each metric selected which process tasks need to be intercepted to collect data automatically during process execution. In the end of this phase we have two independents models: process and metrics.

Finally, the third stage is the automatic transformation of the software process specification to a workflow specification. Here, we integrate metrics actions into the execution of the base process tasks performing a sort of *weaving*. Through these transformations, the sequence of activities of the process and the process metrics are mapped to a workflow definition. This step makes possible the deployment and execution of automatic metrics collection during the process execution in a workflow engine.

In order to evaluate the feasibility of our approach, we have designed and implemented it using the based approach technologies. also provides an overview of the implementation of our approach. The process and metrics specification is supported by EPF composer. The variability management of the EPF specifications is addressed by GenArch product derivation tool. The process modelling is realized using a simplified metric model that will be explained throughout this paper.

Finally, the model-to-model transformations (M2M) codified in ATL/QVT (OMG 2009) to allow the translation of the EPF specification and metric model composition to JPDL model elements. 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. These web forms can then be deployed and executed in the JBoss Business Process Management (jBPM) workflow engine. Section 4 describes this new approach in action by detailing an example of metrics integrated into the software process.

This work promotes the automatic metrics collect..

Falar dos benfícios da abordagem????

**4 Implementing the MDD based COMPOSITION**

4.1 Adding Metrics in the Software Process

The starting point for implementing this extended approach is including the metrics into de process definition. This inclusion aims to define activities relating to the metrics calculation within the process, as well as in analyzing and defining the most common types of metrics used in software development projects. At moment, this approach is limited the definition of basic productivity metrics applied in projects.

This adding was implemented using the EPF Composer, only inserting news elements in a traditional software process definition.

As described in section 2, after define the process line comes the variability management phase where now besides be able to select features that represent process activities, may also be selected activities relating to the collection of metrics that will be held during project execution. The feature model now includes the metrics selection as optional and alternatives features.

Figure :The Basic Metric Metamodel

After the metric inclusion, the process execution must perform the automatic metric collect. These automatic metrics collect tasks have to be intercalated with the process tasks for execution.

4.2 Metric Metamodel Composition

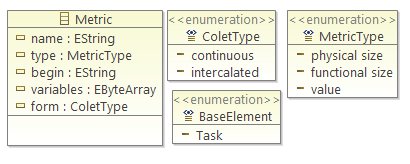
Here we present a metamodel for metric, which describes where the metric collect can be plugged into the workflow that represents a process execution.

In our first implementation, we intend to support only tasks like intervention points for the calculation of the metrics. Therefore, it is necessary to specify for each metric which tasks which it must operate through a simplified model of metrics.

Figure X shows the basic metamodel for metric and process composition. It describes only the main information for the transformation composition.

The parameter “name” contains the metric name. The parameter “type” contains the type of metric that can be one of the following: physical size, functional size, design size and value, adopting the taxonomy presented in []. ...

The attributes “begin” and “endd” store the first and final task name that this metrics will intercept, respectively.

The property “form” represents the type of task time collect. The collectType *continuous* means that the task can´t be paused and the time counted is the time from start until the end of task and the collectType *intercalated* means that the task can be paused and the final time is the sum of each timeslice spend on it. This information will be useful in future when we are generating the process workflow.

At end, the property variables can be used for adding other important information for the process execution but it still is not being used.

4.3 The Metric with Process MDD Composition

The metrics and software process integration is made at the moment of workflow model generation, when the model to model transformation is performed, the UMA to JPDL transformation described in the base approach.

After the intercept points of the metrics are defined in the model metrics, all tasks related metrics are associated with actions that are triggered by events at the beginning and end of the each task execution. This integration represents a transversal composition with the original approach. This composition is performed with QVTO transformation language which receives a process model and a metric model as input and produces an execution process model (workflow model) as output. The metric metamodel of input was presented in the previous section and the process metamodel is UMA. For the output, the workflow metamodel is the JPDL, following the same base approach technology.

At the end of this stage, we have a workflow model that is a weaving of the process and metrics models.

4.4 The composite model execution

As the original approach is still needed the model to text transformation which is responsible for configuration and the user interface files generation allowing the workflow execution in the jBPM engine.

At this point it is important to supply certain information necessary to deploy the workflow, such as the number of iterations, the roles and actors, as well as the specialization of the metric collect action code when necessary.

**5. The MDD Transversal Composition in Action**

In order to evaluate the approach feasibility, we have applied the transversal composition of metrics in an *OpenUp* process family.

The productivity metrics relates with the CDU effort were chooser and divided at the level of a basic four-activity phase structure: requirements, design, development and testing. However, the measurements are performed at the granularity of tasks associated with the activities of each of these phases.

**6. Conclusions**

**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**