Reference: Jürgen Münch, “Transformation-based Creation of Custom-tailored Software Process Mod-els”, Proceedings of the 5th Intl. Workshop on Software Process Simulation and Modeling (ProSim 2004), Edinburgh, Scotland, United Kingdom, pp. 50-56, May 24-25, 2004.

Transformation-based Creation of Custom-tailored Software Process Models

Jürgen Münch
Fraunhofer Institute for Experimental Software Engineering
Sauerwiesen 6, 67661 Kaiserslautern, Germany
Juergen.Muench@iese.fraunhofer.de

Abstract

High quality of complex software at rising cost pressure and dramatically shorter innovation cycles can only be obtained with systematic and efficient development processes. Since software development projects are unique regarding their combination of specific goals and characteristics, providing ‘ideal’ and universal development processes is no solution for real life. Instead, effective and efficient software development processes custom-tailored to a project and based on experience from past projects are required. This is contrary to industrial practice, where reuse-oriented process descriptions and, consequently, the possibility for goal-oriented planning are normally missing.

This article presents a tool-based technique for efficient transformation-based customization of formal process models to project constraints. The technique was evaluated in the context of two case studies, where a considerable increase in efficiency could be observed: Even the more unfavorable case resulted in an effort reduction at a factor of 65 as compared to a manual procedure.

1. Introduction

Customization of software process models to project constraints requires an understanding of process variations and knowledge about when to use which variation. Typically, the following logical steps are needed before the customization activity is performed:

- Possible process alternatives need to be elicited and explicitly described.
- Process alternatives need to be characterized and constraints/rules on their use need to be formulated. This requires a deeper understanding about the appropriateness of the process alternatives for different contexts and their effects in these contexts. One rule could be, for instance, that a threatening analysis process needs to be performed if the project is critical.
- Before the start of the project, a characterization of the project context and its goal is necessary, describing the information needed to select process alternatives.

The core customization activity addressed in this paper is the creation of the custom-tailored process model that describes the appropriate processes for a specific project context. Inputs for this customization activity are process alternatives, customization rules, and a project characterization. The output is a custom-tailored process model with the appropriate content. It should be mentioned that feedback from the application of process models should be used to improve the models, the customization criteria, and the customization procedure.

Typical problems encountered with the customization of software development processes are:

- Insufficient reusability of process descriptions: The reuse of process descriptions created for past projects is limited. The reason is that existing process notations lack constructs for describing generic process information (such as optional product flows). Perry [12] considers inappropriate organization and encapsulation of process fragments as an essential factor restricting reuse of process knowledge. Besides, insufficient or missing knowledge about the scope of validity of software processes and techniques hinders their reuse. This addresses questions such as: What are appropriate situations to use a particular process? To what degree has a development process for similar contexts already been evaluated?
- High customization efforts: The adaptation of process descriptions to project goals and project contexts is a complex task. Local modification of a process description often requires a lot of

corollary global changes that are necessary to guarantee the consistence of the customized process description. Identification and execution of such changes require a lot of effort. El Eman et al. [7] accentuate that particularly changing development techniques can result in substantial modifications of process descriptions. The customization of a maintenance process [14], for instance, showed that the deletion of 4 products implied follow-up changes of the product hierarchy, the product flow, and the processes that consume, produce, or modify these products. Eventually, 68 changes in 26 models (product and process models) had to be executed in order to consistently remove the 4 products. Those comprehensive changes are difficult to perform consistently if informal process descriptions are used. The technique presented in this article largely allows for performing these changes in a widely automated manner.

- Insufficient customization tool support: Existing process modeling tools typically provide editing and analysis capabilities. Few tools provide documentation and/or enactment capabilities. There exist nearly no tools that allow for customization of individually developed process models.

Different research approaches from the software development domain can be adapted to support customization of software process models:

- characterization and selection approaches (e.g., [1],[13],[2]);
- composition, generation, and transformation approaches (e.g., [6]);
- AI planning approaches (e.g., [3],[16]);
- domain engineering approaches (e.g., [9]);
- classical planning approaches such as work breakdown structures (e.g., [15]);
- process pattern approaches (e.g., [8]).

These approaches can be combined for customizing software process models. The tool-supported technique for customization that is the focus of this article is based on the principles of transformation systems. The underlying ideas stem from traditional incremental program transformation systems that are used for compiler construction.

2. Transformation-based Customization

A transformation system maps an object O (normally source code) onto an object Q. For this purpose, transformation rules are applied successively. Each transformation rule is fed with the result of the application of the previous rule, so that a sequence of transformations results:

$$O \rightarrow O_1 \rightarrow O_2 \rightarrow ... \rightarrow O_n = Q$$

The principles of transformation systems are applied for the purpose of process customization in the following way: A transformation T_x transforms a process model into a new process model. The result of a transformation T_x, the process model PM_x, is used as input for the subsequent transformation T_{x+1}. Starting with a basic model PM_0 that needs to be customized (this could be, for instance, a standard like the ESA ECSS-E-40), a final process model PM_n is obtained after a chain of transformations. The final process model is custom-tailored for the project context and can be instantiated in the project plan. As in transformation systems for program descriptions, transformations of process models typically also require some interaction with the user (i.e., the project planner or process engineer). This is usually the case if modifications are necessary that cannot be fully automated or if decisions about alternative process descriptions could not be anticipated in advance. The underlying technique presented here is much more detailed and contributes to several challenges. For example, the technique provides a conflict resolution strategy if different transformation rules contradict each other.

The following examples illustrate the representation of project characteristics and customization rules. Additionally, a concrete project characterization and an excerpt of the customization process are given. The examples are extracted from a customizable version of a German software development standard (V-Model) [5].

The context of projects for which a custom-tailored process model should be created is described by using project characteristics. These characteristics are attributes whose values influence the variation of the process model. It is a challenge to determine which project characteristics have an influence on the process model and to find the minimal set of relevant characteristics. Sometimes technical dependencies or external constraints determine specific process variations (e.g., if the project is highly critical, a threatening analysis must be included in the process). Often, the influence factors are unknown or only insufficiently understood (e.g., which factors determine the selection of an appropriate testing technique in a specific context?). Here, empirical methods can help to determine appropriate impact factors.

In the customization approach presented here, characteristics consist of a type, an identifier, a name, an explanation, and a default value. Characterizing a project in order to create a custom-tailored project model means

assigning values to project characteristics. Examples for project characteristics are the following:

```
type:         Boolean
id:           ext_spec_meet_sys_req
name:         "External specifications meet
              product pattern System Requirements"
explanation:  "The external specifications have
              to meet the System Requirements
              both formally and with regards to their
              contents."
default:      false

type:         Integer
id:           max_effort
name:         "Maximal effort"
explanation:  "Effort is measured in days."
default:      2000

type:         Enumeration
definition:   (system dp_segment swci)
id:           object
name:         "Object under contract"
explanation:  "The object under contract may
              be a system, a DP Segment or a SWCI"
default:      system
```

The customization rules are described in a condition-action format using constraints on the project context as conditions and manipulations of the process model as actions. The transformation of a basic model into a custom-tailored model is done by the subsequent application of rules. The condition of a rule is a Boolean expression on the set of project characteristics. The action consists of one manipulation or a sequence of manipulations that modify the basic model (e.g., delete a process, insert a product, append a process). The implementation of the approach allows for consistent execution of the manipulations. This is a complex task because one change of the model can cause a lot of follow-up changes (e.g., the deletion of a process can lead to adaptations of the product and control flows or to changes in the process aggregation hierarchy). Additionally, manipulations can influence the execution of other rules (e.g., the deletion of a product prevents the inclusion of a process that consumes this product). Examples for rules are given in the following:

```
id:           rule4
name:         "Rule 4: Maintenance project and the
              SWCI integration plan is available"
condition:    sw_maintenance == true
condition:    swci_int_plan_available == true
manipulation: "deleteProcess
              V-Model.
              gen_preliminary_design.
              preliminary_design_rest.
              specify_swci_integration"
manipulation: "deleteProcess
              V-Model.
              assess_formally_swci-integration_plan"
manipulation: "requestEdit V-Model Please change
              the instantiation of the
              swci_integration_plan to accepted."
explanation:  "The generation of the SWCI
              integration plan and its assessment are
              skipped."

id:           rule5
name:         "Rule 5: Actual state has no impact"
condition:    actual_state_has_impact == false
manipulation: "deleteProduct
               V-Model.
              system_requirements.
              sys_actual_record"
manipulation: "disableRule rule1b"
explanation:  "The generation of the actual record
              for the system requirements is deleted."

id:           rule6
name:         "Rule 6: External specifications meet
              product system requirements"
condition:    ext_spec_meet_sys_req == true
manipulation: "insertProcess
              V-Model.
              sys_analyze_req_and_design.
              extract_sys_req.
              Extract_System_Requirements"
manipulation: "insertConnect
              V-Model.
              sys_analyze_req_and_design.
              extract_sys_req.
              ext_specifications
              V-Model.ext_specifications"
manipulation: "insertConnect
              V-Model.
              sys_analyze_req_and_design.
              extract_sys_req.
              system_requirements
              V-Model.
              system_requirements"
manipulation: "deleteProcess
              V-Model.
              sys_analyze_req_and_design.
              sys_analyze_req"
explanation:  "System requirements are directly
```

extracted from the external specifications."

If two or more conditions are mentioned (as, for instance, in rule 4), it means that they are combined by "AND" statements. The manipulation part of rule 4 implies the deletion of two processes and a request for manual manipulation. The latter is necessary in case a customization step cannot be fully automated. Here, the supporting tool provides the process engineer with an appropriate excerpt of the model and asks for manual modification. Rule 5 gives an example of a conflict resolution: After applying rule 5, rule 1b is disabled and cannot be applied anymore during the customization process. Rule 6 is an example of a fully automated customization rule.

The execution of the customization activity is the application of all enabled customization rules. These are all rules whose conditions evaluate to "TRUE" for the given project characterization. The supporting tool offers different strategies to determine a sequence for the rule application. Also, manual selection of the rules to be fired is possible. In the following, an excerpt of a project characterization and the application of a rule is shown:

Characterization of project (excerpt):

SW maintenance is false.
SWCI Integration Plan available is false.
DP Integration Plan available is false.
System Integration Plan available is false.
External specifications meet product pattern System Requirements is true.
Maximal effort is 2000.
Object under contract is system.
Diagnosis interface is available is false.
External specifications meet product pattern DP Requirements is false.
External specifications meet product pattern SW Requirements is false.
HWCI/SWCI structure already specified or defined by means of preliminary analysis is false.
System Architecture is specified or already defined by means of preliminary investigations is true.
DP Architecture is specified or already defined by means of preliminary investigations is false.

Customization activity:

Rule execution:
Rule 6: External specifications meet product system requirements
System requirements are directly extracted from the external specifications.

1. Executing:
   insertProcess
   V-Model.
   sys_analyze_req_and_design
   extract_sys_req
   Extract_System_Requirements

2. V-Model.sys_analyze_req_and_design
   [object_relations]:
   appended process extract_sys_req

3. V-Model.sys_analyze_req_and_design
   [objects]:
   inserted process extract_sys_req:
   Extract_System_Requirements

4. Executing: insertConnect
   V-Model.sys_analyze_req_
   and_design.extract_
   sys_req ext_specifications
   V-Model.ext_specifications

5. V-Model.sys_analyze_req_and_design
   [relations]: inserted connect
   extract_sys_req(...
   ext_specifications=>ext_specifications ...)

6. V-Model.sys_analyze_req_and_design
   [interface_relation]: inserting Connect
   extract_sys_req(...,
   ext_specifications=>ext_specifications...)

7. Executing: insertConnect V-
   Model.sys_analyze_req_and_design.extract_s
   ys_req system_requirements V-
   Model.system_requirements

…

This rule caused 14 changes in the basic model.

3. Customization Tool

The tool that supports the technique uses process models that are described in the formal process modeling language MVP-L [4]. MVP-L is extended a) with a notation for the description of rules that are collected in a rule base, and b) a notation for characterizing project contexts. The tool-based adaptation consists of 4 steps:

- configuration of the basic model, the context classification, and the rule base;
- characterization of the actual project context;

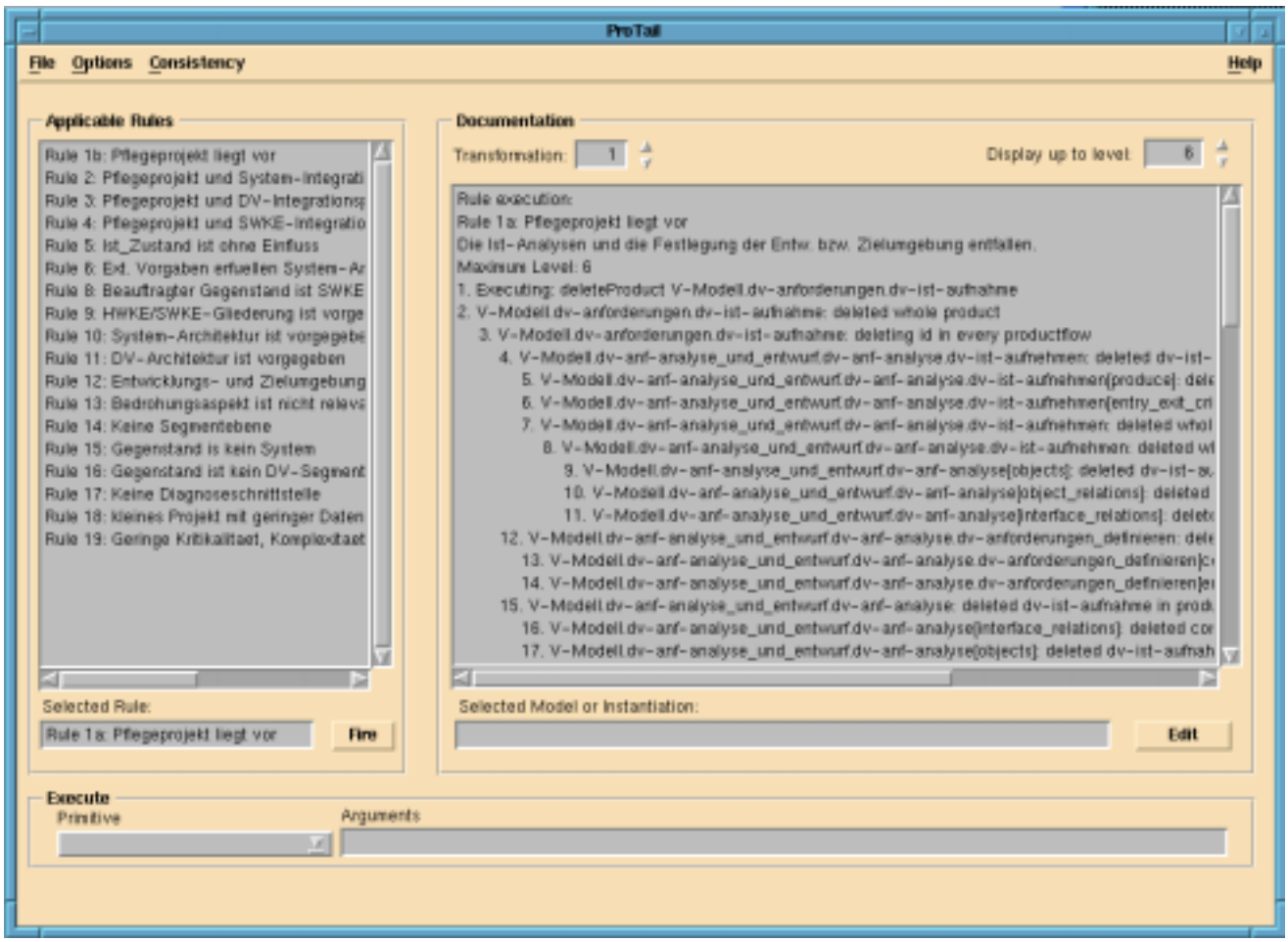

Figure 1. Transformation Panel.

- transformation of the basic model into the custom-tailored model;
- documentation of the adaptations.

The latter can support, for instance, the demonstration of the conformity of the customized model with a standard. Figure 1 shows the transformation panel of the tool. On the left side are the rules that can be executed, on the right side are the documented modifications.

5. Validation

The validation of the approach was done in the context of two case studies. Tool-based customization was compared to a manual approach with respect to effort and number of inconsistencies in the resulting model. The validation showed, for the tool-based approach, a significant effort reduction with the same or less inconsistencies in the result (i.e., the custom-tailored model).

In addition to this validation, a rule-based customization approach is currently being evaluated in the context of a project for the development of wireless Internet services [10]. First experiences show that coming up with alternative processes can be done by descriptive modeling. Finding rules on when to select which process alternative seems to be much more difficult. This requires careful analysis of the commonalities of the variants and their contexts [10][11].

6. Conclusions and Future Research Topics

Process models as described, for example, in standards or handbooks are usually of a generic type in the sense that they are only specified on an abstract level. What is usually missing are statements about the domain or context in which the models are valid as well as (empirically derived) guidelines on how to adapt the models to project-specific goals and characteristics. From the viewpoint of a process engineer, a process support

environment should provide mechanisms to cope with process variants, i.e., the customization of process models should be supported. Important research issues concern an understanding of relevant variation parameters, an appropriate representation of generic process knowledge, the development of customization mechanisms, and the packaging of process models for reuse in further projects. It should be investigated whether internal or external mechanisms are more useful for customization. "Internal" means that the process model itself contains genericity (e.g., optional product flow parameters); "external" means that the process model is an input to an application (e.g., transformator) which allows customization according to project characteristics and goals. A related question is which software reuse approach (e. g., templates, generation, composition, transformation) can be applied to software process models. The prototypical customization tool presented in this paper uses a transformational approach. First experiences with this tool show enormous time savings in constructing the context-specific model. Beyond this, the transformational approach seems to be a good solution for managing the numerous interrelations between process fragments (e. g., control flow, product flow, refinement hierarchies). Challenges for future process customization research are, for instance,

- dynamic replanning and customization during project execution,
- integrated customization of different plans (e.g., process and measurement plans),
- the use of quality models during customization.

Acknowledgement

This work has been partially funded by the European Commission in the context of the WISE project (No. IST-2000-30028). The author would like to thank Markus Schmitz for supporting the implementation of the approach, and Sonnhild Namingha from the Fraunhofer Institute for Experimental Software Engineering (IESE) for reviewing the first version of the article.

References

[1] Linda C. Alexander und Alan M. Davis. Criteria for Selecting Software Process Models. In Proceedings of the 15th Annual International Computer Software & Applications Conference, Seiten 521-528. IEEE Computer Society Press, Tokyo, 1991.

[2] Victor R. Basili und H. Dieter Rombach. Support for comprehensive reuse. Software Engineering Journal, 6(5): 303-316, September 1991.

[3] Ralf Bergmann, Hector Muñoz-Avila, Manuela Veloso und E. Melis. Case-based Reasoning applied to Planning. In M. Lenz, B. Bartsch-Spörl, H.D. Burkhard und S. Wess, Hrsg., Case-Based Reasoning Technology: From Foundations to Applications. Lecture Notes in Artificial Intelligence 1400, Springer Verlag, 1998.

[4] Alfred Bröckers, Christopher M. Lott, H. Dieter Rombach und Martin Verlage. MVP–L Language Report Version 2. Technischer Bericht 265/95, Fachbereich Informatik, Universität Kaiserslautern, 67653 Kaiserslautern, 1995.

[5] Adolf-Peter Bröhl und Wolfgang Dröschel, Hrsg. Das V-Modell. 2. Auflage. R. Oldenbourg Verlag, München 1995.

[6] Faye C. Budlong, Paul A. Szulewski und Ralph J. Ganska. Process Tailoring for Software Project Plans. Technischer Bericht, Software Technology Support Center (STSC), OO-ALCTTISE, Hill AFB, Utah (USA), Januar 1996.

[7] Khaled El Emam, Nazim H. Mahavji und K. Toubache. Empirically Driven Improvement of Generic Process Models. In Proceedings of the 8th International Software Process Workshop, 1993.

[8] Soeli T. Fiorini, Julio Cesar Sampaio do Prado Leite und Carlos José Pereira de Lucena. Reusing Process Patterns. 2nd International Workshop on Learning Software Organizations (LSO), 2000.

[9] M. Lipshutz, R. Creps und M. Simos. Organizational domain modeling (ODM) tutorial, Januar 1997.

[10] Alexis Ocampo, Daniela Boggio, Jürgen Münch, and Gino Palladino, "Towards a Reference Process for Wireless Internet Services", IEEE Transactions on Software Engineering, Special Issue on Wireless Internet Software Engineering, vol. 29, no. 12, pp.1122-1134, December 2003.

[11] Alexis Ocampo, Jürgen Münch, Fabio Bella, "Software Process Commonality Analysis", Proceedings of the 5th International Workshop on Software Process Simulation and Modeling (ProSim 2004), Edinburgh, Scotland, United Kingdom, May 24-25, 2004.

[12] Dewayne E. Perry. Practical Issues in Process Reuse. In Proceesings of the 10th International Software Process Workshop, Frankreich, Juni 1996.

[13] Rubén Prieto-Díaz und Peter Freeman. Classifying Software for Reusability. IEEE Software, 4(1): 6-16, IEEE Computer Society, Januar 1987.

[14] Markus Schmitz, Jürgen Münch und Martin Verlage. Tailoring of large process models on the basis of MVP-L (in German). In Günther Müller-Luschnat, Sergio Montenegro, Ralf Kneuper, eds., proceedings of the 4th workshops of the section 5.1.1 (GI), Berlin, March 1997

[15] Robert C. Tausworthe. The work breakdown structure in software project management. Journal of Systems and Software, 1:181–186, 1980.

[16] Frank Weberskirch. Combining SNLP-like planning and dependency-maintenance. Technischer Bericht, LSA-report LSA-95-10E, Centre for Learning Systems and Applications, Kaiserslautern, 1995.