Process Model Editing Support Using Eclipse Modeling Project Tools

René Wörzberger and Thomas Heer

Department of Computer Science 3
RWTH Aachen University, Germany
{woerzberger, heer}@i3.informatik.rwth-aachen.de

Abstract: Process models of manifold kinds are extensively used in the field of Business Process Management (BPM). Providing appropriate visualization and editing support for process models is of major importance. This particularly includes support for the adherence to certain modeling constraints. In this paper we report how we leverage tools and frameworks of the Eclipse Modeling Project in order to rapidly develop an editor for different kinds of interrelated process models.

1 Introduction

Process management is a key factor for commercial success in modern companies. Stakeholders in process management must be provided with *appropriate tools* to enable them to *define* processes but also to *interact with* and *monitor* processes. This holds true for the management of complex development processes but also for business processes.

In the past our group has dealt with complex *development processes* in the area of chemical engineering [NM08]. In this context, we have developed the *process management system AHEAD* [HJK+08]. This tool can be used to *visualize* and *edit* the complex graphical structure of dynamic development processes on a *medium grained level* which fits the needs of a process manager. Furthermore, certain regulations for development process are enforced by AHEAD.

Currently, we are *transferring* our research work into industrial practice [HNWH08]. In cooperation with AMB-Informatik, an IT-service provider for the insurance group Generali, we develop tools for the management of *dynamic business processes*, e.g., business processes that cannot be completely predefined but *structurally evolve* during *execution*.

Compared to the past work in the AHEAD-project, there are some *similarities* to those requirements, which we are currently dealing with. Also in the area of business processes different *kinds* of *models* are used, many of which are best represented in a *graphical* way. Business process models cannot be freely edited but have to *adhere* to certain *constraints*, which stem from *technical* requirements or *professional* regulations.

There are also some important *differences* of our current transfer work compared to the past AHEAD project. The AHEAD tool has been built *from scratch* yet using *progres-*

sive tools like the *graph rewriting* specification language and development environment PROGRES [SWZ99], which can be seen as a predecessor of contemporary model driven development tools. However, PROGRES as well as AHEAD are still *academic prototypes* with limited maturity and platform independence and are rather agnostic of existing standards compared to the Eclipse Modeling Project. In our current research project we *cannot* implement a process management system from scratch but have to *extend* the *existing* business process management system, which is used by our partner AMB-Informatik, namely products of the IBM WebSphere family [WEH08]. In particular, we have to extend the visualization and editing support of process models which are processed by the IBM WebSphere products.

In this paper we show how we leverage the Eclipse Modeling Framework (EMF), Eclipse Graphical Modeling Framework (GMF) and the Object Constraint Language (OCL) of the Eclipse Modeling Project (EMP) to rapidly develop a *graphical* editor. This editor supports editing of *several kinds* of process models and furthermore helps the user to adhere to several kinds of *constraints* which are imposed by process modeling.

The *structure* of the paper is as follows: In Section 2 we delineate different process model kinds of our process editor as well as modeling constraints, some of which stem from interrelationships between different model kinds. Section 3 provides an insight into the application of the Eclipse Modeling Project for our editor. Related works are discussed in Section 4. We conclude with some remarks in Section 5.

2 Editor for Process Models

Figure 1 depicts a screenshot of our process model editor, which we could *rapidly* develop using the above mentioned tools of the Eclipse Modeling Project. In business process management process models of different kinds on different *layers of abstraction* have to be visualized and edited. These are key requirements for our editor. The figure shows three process model examples each of which is of a different kind.

In our approach, we model *abstract process knowledge* in a graphical way as depicted by the upper model in Figure 1. Process knowledge can, e.g., be derived from company specific or legal regulations. We call the graphical language for this kind of models *Business Process Compliance Language (BPCL)*. As the name suggests, other model kinds have to comply with the knowledge expressed in BPCL models. For instance, the BPCL model from the top of Figure 1 particularly states the following compliance constraint for claim handling processes of insurance companies: The existence of a check amounts (A) activity implies that also a check coverage (C) activity is executed; hence there is an existence-connection between the A-node and the C-node.

The process model in the middle of Figure 1 is of a different kind than the upper model. It is modeled in a *simplified* version of *WS-BPEL* abbreviated with *SimBPEL*. WS-BPEL is a standard for process definitions which are precise enough to be executed in a process management system [AAA+07]. The IBM WebSphere tools, which we extend, rest upon the WS-BPEL-standard. In SimBPEL certain WS-BPEL elements are not modeled, e.g.

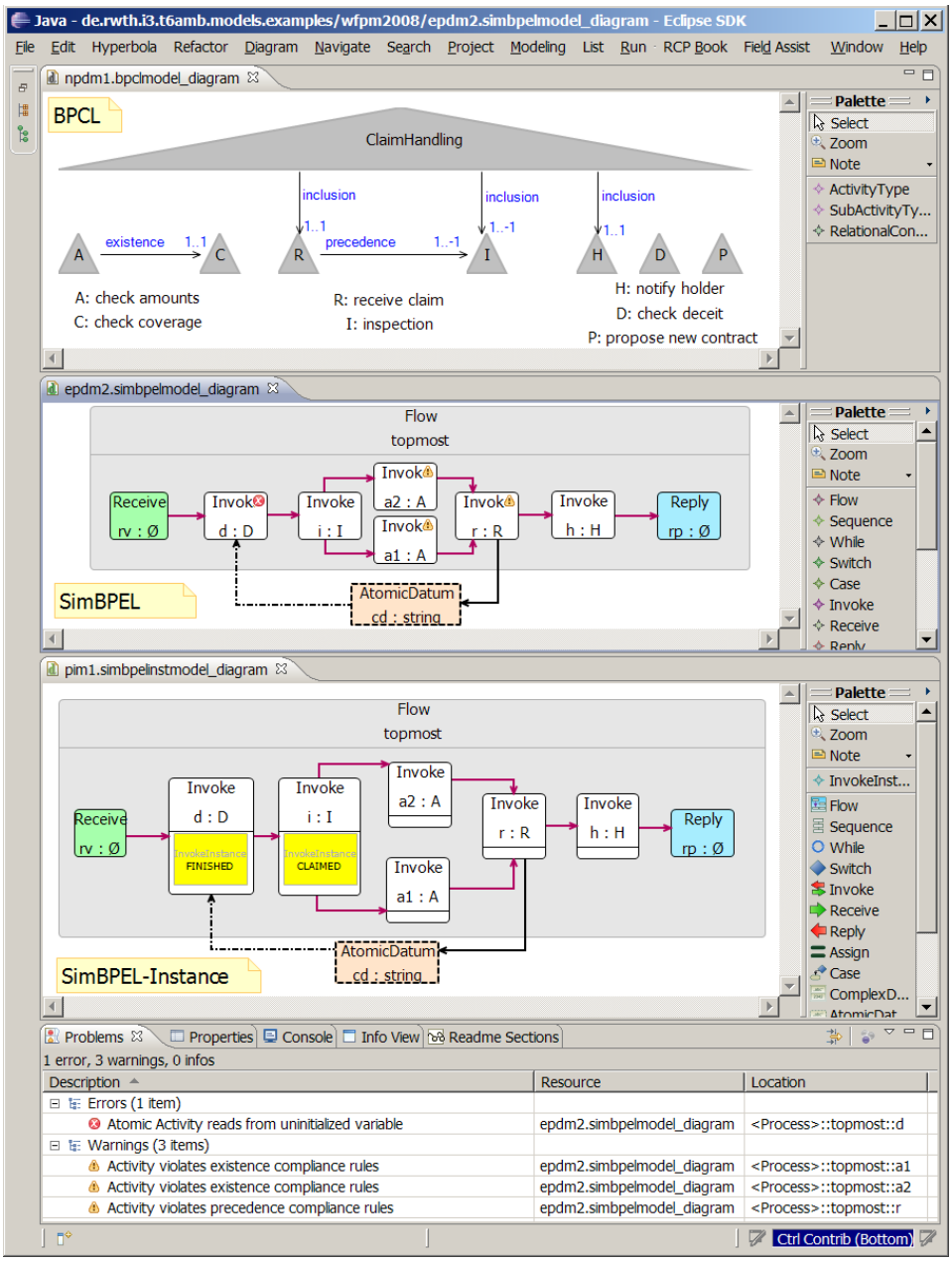

Figure 1: Editor for related process models of different kinds

port types, whereas others are explicitly modeled in contrast to WS-BPEL, e.g., the read (write) access of invoke activity d (r) from (to) the datum cd.

The bottom of the figure shows a *SimBPEL-Instance* model, which is of another important process model kind. These models do not reflect process definitions but process instances during their execution. SimBPEL-Instance models are similar to SimBPEL models but additionally contain elements that constitute the current state of a certain process instance. In particular, activities are associated with activity instances (yellow rectangles) that carry the state of the respective activity. Additionally, SimBPEL-Instance models do not just visualize process instances but can also be *structurally modified* in order to perform dynamic modifications to process instances.

Besides the mere editing functionality the editor also detects certain flaws in the models. In particular, violations of certain *correctness* constraints are detected, e.g., that activity d reads from datum cd before it is written by some activity. Correctness violations are classified as errors and marked with an error marker (⊗).

Furthermore, the *interrelations* of the models are exploited in order to detect violations of *compliance* constraints. For instance, the SimBPEL model does not comply with the BPCL model, since there are two check amounts (A) activities in the process model but no check coverage (C) activities, which violates the compliance constraint described above. Violations of compliance constraints are signaled with a warning marker (⚠).

3 Use of the Eclipse Modeling Project

In the *implementation* of the process model editor we made extensive use of the Eclipse Modeling Project. Figure 2 illustrates the *coarse-grained architecture* of the editor, which is embedded in an overall technical context.

3.1 Abstract Syntax

The *abstract syntax* of our models is defined by Ecore [BSM+04] meta-models. On the one hand, the meta-models are kept *separated* from each other in different packages each of which corresponds to a certain model kind. On the other hand, the meta-models are coupled with *meta-model crossing meta-references*. SimBPEL meta-models and BPCL meta-models are rather *loosely coupled*. They are just connected by a meta-reference that associates an activity (SimBPEL) with an activity type (BPCL). However, SimBPEL and SimBPEL-Instance *significantly overlap*, since SimBPEL-Instance models are just SimBPEL models with additional state information. Thus, the SimBPEL-Instance meta-model reuses the meta-model of SimBPEL and extends it by some model elements, which reflect the state of a process instance. E.g., activity instances are part of the SimBPEL-Instance meta-model. Each activity is associated not just with exactly one but with zero to many activity instances. This is because we do not consider an activity having a state before it is executed for the first time. Furthermore, activities can be nested in decisions

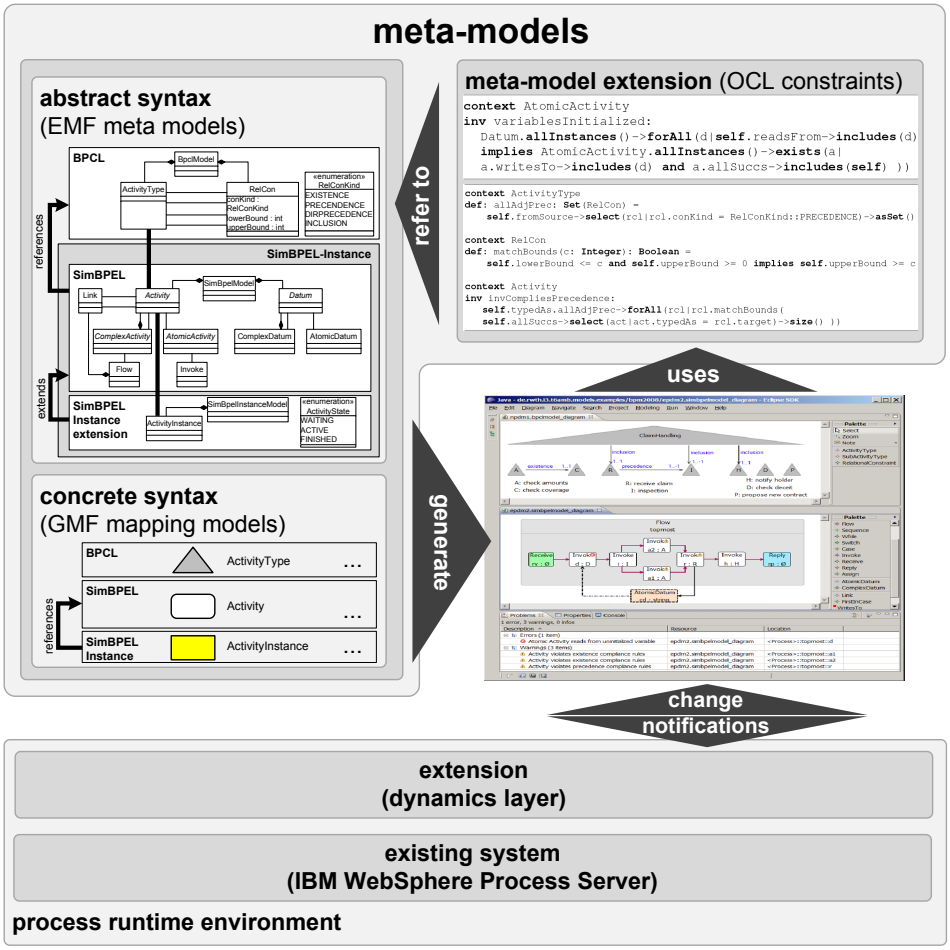

Figure 2: Coarse-grained architecture of the process model editor

or loops and thus be executed not at all, once or arbitrarily many times.

Since the meta-models are not strictly isolated from each other but interrelated, together they can be regarded as one single meta-model. We call this single meta-model the *integrated* meta-model.

3.2 Concrete Syntax

The Eclipse Modeling Framework (EMF) is able to generate *tree oriented standard editors* for model instances of existing meta-models. For our purposes, these standard editors do *not suffice*, since process models of any kind are better rendered in a true graphical

way. Hence, we also use the *Graphical Modeling Framework (GMF)* in order to define a *concrete syntax* for each of our model kinds. Here, we can again *exploit the overlapping* between SimBPEL and SimBPEL-Instance, i.e. the *definitions of the figures* for SimBPEL model elements do not have to be defined again for SimBPEL-Instance but can be *reused* by referencing them. This way, we preserve a *consistent visualization* of SimBPEL models and SimBPEL-Instance models.

3.3 Meta-Model Extension via OCL

Common meta-modeling languages like Ecore are *not expressive* enough to exclude certain situations in model instances. E.g., the SimBPEL model instance of Figure 1 is a valid model instance of its respective meta-model but nonetheless *incorrect* due to the error described in Section 2. Therefore, we *extend* our meta-models by *OCL-constraints* [Obj06]. These OCL-constraints are later on used by the editor to detect such model flaws. We give details on these OCL-constraints in [WKH08] and show how they can also be used in combination with the integrated meta-model in order to detect *compliance* violations of a SimBPEL model with respect to an existing BPCL model.

3.4 Integration with a Process Runtime Environment

The editor is embedded in a *technical context*. Since SimBPEL-Instance models are just useful if they are coupled with real process instances, we have to *integrate* the editor with a process runtime environment. This integration is *bidirectional*: First, changes to process instances, e.g., made by other process participants, have to be reflected in the local SimBPEL-Instance model to keep the respective *user informed* about the current process instance state. Second, changes made to the local SimBPEL-Instance model have to be propagated to the process runtime environment in order to *update* the *process instance*, which is hosted there.

In a *preceding work* [WEH08] we have *extended* the process runtime environment IBM WebSphere Process Server inasmuch that *dynamic* structural *modifications* can be made to process instances even during their execution. Thus, we also have to *propagate complex model changes* made in a SimBPEL-Instance model to the process runtime environment. It is advantageous to have the process instance information *redundantly stored* in a local SimBPEL-Instance model. This way, a process participant can change SimBPEL-Instance models and check them w.r.t. correctness and compliance before committing the changes to the process runtime environment.

4 Related Work

Adherence to constraints in process modeling is a research field on its own. Many research works aim at the *correctness* of process models. Some of them deal with the correctness of executable *process definition* models [VBvdA01, MvdA07] whereas other particularly target at the correctness of *process instance* models [RD98].

Other works focus on the *compliance* of process models regarding separately modeled compliance constraints. In the AHEAD-prototype [HJK+08] so called dynamic task nets, i.e., highly dynamic process models on a medium grained level, can be checked w.r.t. compliance against constraints modeled in variants of UML class diagrams [Sch02]. AHEAD was also implemented in a model driven fashion, i.e., major parts were specified in the graph rewriting language PROGRES [SWZ99]. The DECLARE project [PSA07] covers compliance constraints that resemble ours and which are also defined in graphical models like our BPCL models. However, DECLARE-models are used for process execution whereas we rely on the WS-BPEL standard for execution and use BPCL models merely for compliance checks.

5 Conclusion

In this paper we have reported about the *application of Eclipse Modeling Project* tools for *rapidly developing* a process model editor. This editor supports editing of *different process model kinds* and accounts for *correctness* constraints of models as well *compliance* relationships between different models.

Our work differs from related works inasmuch as we implemented both correctness and compliance checks in the *same way* using model driven development tools. We are confident that this approach is sufficiently *flexible* to *incorporate* new correctness checks as well as new types of compliance checks. Due to the employed Eclipse Modeling Project *revisions* of the meta-models almost *automatically* yield an *updated* editor.

Model driven development is *important for academia* in two ways: First, it is still a young discipline and hence an *active research area*. Second, model driven development tools fit the needs for *rapid prototyping*. This is important for informatics research, where requirements and particular meta-models tend to be *experimental* and need to be *adapted frequently*. Thus, prototypes like our editor cannot be built with acceptable effort just using basic development tools.

References

[AAA+07] Alexandre Alves, Assaf Arkin, Sid Askary, Charlton Barreto, Ben Bloch, Francisco Curbera, Mark Ford, Yaron Goland, Alejandro Guízar, Neelakantan Kartha, Canyang Kevin Liu, Rania Khalaf, Dieter König, Mike Marin, Vinkesh Mehta, Satish Thatte, Danny van der Rijn, Prasad Yendluri, and Alex Yiu. Web Services Business

Process Execution Language v2.0. Technical report, Organization for the Advancement of Structured Information Standards (OASIS), 2007.

[BSM+04] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T. Grose. *Eclipse Modeling Framework*. The Eclipse Series. Addison-Wesley Professional, 1st edition edition, 2004.

[HJK+08] M. Heller, D. Jäger, C.-A. Krapp, M. Nagl, A. Schleicher, B. Westfechtel, and R. Wörzberger. *An Adaptive and Reactive Management System for Project Coordination*, volume 4970 of *Lecture Notes in Computer Science*, chapter 3.4, pages 300–366. Springer, 2008.

[HNWH08] M. Heller, M. Nagl, R. Wörzberger, and T. Heer. *Dynamic Process Management Based upon Existing Systems*, volume 4970 of *Lecture Notes in Computer Science*, chapter 7.7, pages 711–726. Springer, 2008.

[MvdA07] J. Mendling and W. M. P. van der Aalst. Formalization and Verification of EPCs with OR-Joins Based on State and Context. In John Krogstie, Andreas L. Opdahl, and Guttorm Sindre, editors, *CAiSE*, volume 4495 of *Lecture Notes in Computer Science*, pages 439–453. Springer, 2007.

[NM08] M. Nagl and W. Marquardt, editors. *Collaborative and Distributed Chemical Engineering: From Understanding to Substantial Design Process Support*, volume 4970 of *Lecture Notes in Computer Science*. Springer, 2008.

[Obj06] Object Management Group (OMG). *Object Constraint Language (OMG) Specification - Version 2.0*, May 2006.

[PSA07] M. Pesic, M.H. Schonenberg, and W.M.P. van der Aalst. DECLARE: Full Support for Loosely-Structured Processes. In *EDOC '07: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*, page 287, Washington, DC, USA, 2007. IEEE Computer Society.

[RD98] M. Reichert and P. Dadam. ADEPTflex-Supporting Dynamic Changes of Workflows Without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

[Sch02] A. Schleicher. *Management of Development Processes: An Evolutionary Approach*. PhD thesis, RWTH Aachen University, 2002.

[SWZ99] A. Schürr, A. Winter, and A. Zündorf. The PROGRES Approach: Language and Environment. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Handbook on Graph Grammars and Computing by Graph Transformation – Volume 2: Applications, Languages, and Tools*, pages 478–550. World Scientific, 1999.

[VBvdA01] H. M. W. Verbeek, T. Basten, and W. M. P. van der Aalst. Diagnosing Workflow Processes using Woflan. *The Computer Journal*, 44(4):246–279, 2001.

[WEH08] R. Wörzberger, N. Ehses, and T. Heer. Adding Support for Dynamics Patterns to Static Business Process Management Systems. In Cesare Pautasso and Éric Tanter, editors, *Software Composition*, volume 4954 of *Lecture Notes in Computer Science*, pages 84–91. Springer, 2008.

[WKH08] R. Wörzberger, T. Kurpick, and T. Heer. Checking Correctness and Compliance of Integrated Process Models. In *Proceedings of the 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2008. (to appear).