Scenario-based process modeling with GRETA

Dirk Fahland[1] and Matthias Weidlich[2]

[1] Humboldt-Universität zu Berlin, Germany
fahland@informatik.hu-berlin.de
[2] Hasso-Plattner-Institute, University of Potsdam, Germany
matthias.weidlich@hpi.uni-potsdam.de

Abstract. Designing understandable business process models is one of the key factors to successful business process management. Current modeling practices advocate the use of block-oriented concepts and subprocesses to structure complex process models. However, such guidelines cannot be applied in any case as case studies in process mining have shown. Previously, we proposed the *scenario-based* paradigm to structure models of complex processes in behavioral fragments, i.e., scenarios. This paper presents GRETA as a tool that supports scenario-based process modeling and execution.

1 Introduction

Business process modeling has become an established technique for documenting, understanding, and analyzing business processes of enterprises of different kinds and size. Depending on the domain, process modeling results in several hundred or thousand process models [1]. These models need to be created, understood, and maintained by modelers and stakeholders who are not necessarily experts in computer science.

The BPM community developed modeling practices to create understandable models [2]. Typically large process models are structured hierarchically using subprocess of reasonable size or block-oriented concepts. However, important lessons from process mining tell that actual processes do not follow such structuring. Typically, flow of information connects various parts of a business process leading to "spaghetti-like" process models [3] for which structuring into subprocesses or block-oriented concepts is hard, if possible at all, e.g., Fig. 3 on the right.

In [4], we proposed the *scenario-based* modeling paradigm to structure behavior of processes into *behavioral fragments* of any chosen shape. In this paradigm, a process model is a set of *scenarios*. Each scenario denotes a finite acyclic process fragment consisting of several actions, some of them ordered in parallel. A distinguished *history* of the scenario expresses when the scenario may occur, i.e., how to *continue a process instance* that exhibited the scenario's history. We formalized these notions in the model of *oclets* [5] based on Petri nets. Figure 1 depicts two example oclets *depart standard* and *depart devices* (ignore the highlighted nodes). Each oclets's history is drawn above the horizontal line. Thus, *depart standard* triggers *depart devices* because of *present equipment*. Moreover, *depart standard* and *depart devices* describe different continuations after *present equipment*, i.e., both oclets describe *alternative continuations* of a

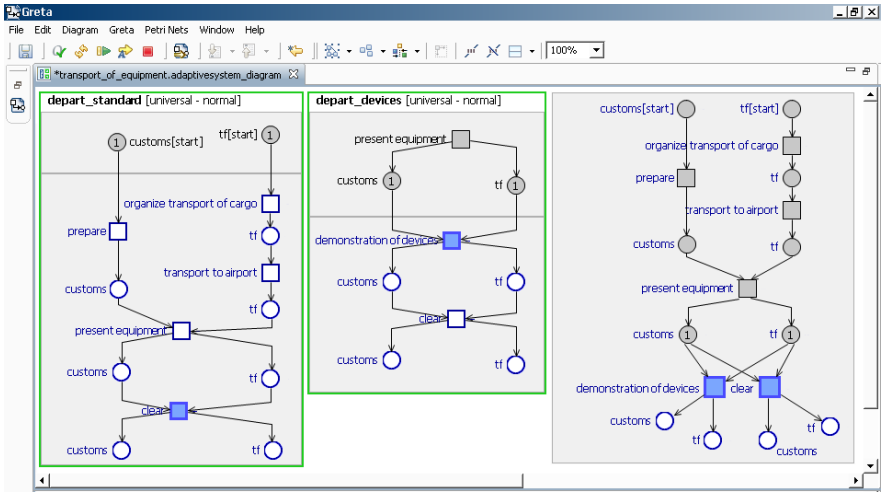

Fig. 1. Screenshot of GRETA with two oclets and a process instance constructed from these oclets.

process instance. This interplay allows to model complex business processes in terms of behavioral scenarios.

A process modeler reads each oclet as a "self-contained story" that occurs in the process. So, a scenario-based process model structures a complex process in terms of the "stories in the process" rather than blocks or subprocesses. Each story can, in principle, be understood in isolation. The difficulty of this approach is to design all oclets of the process so that they "fit together". This paper presents the tool GRETA that supports scenario-based process modeling in a *graphical editor* with animated *process execution*. Using GRETA, a process modeler can create and improve a complex model iteratively. Section 2 elaborates on the use case in more detail. Section 3 explains how GRETA supports scenario-based process modeling. We compare GRETA's key features to existing tools and conclude the paper in Section 4.

2 Use case: modeling unstructured processes

We illustrate the need for scenario-based modeling by a process that is run by the Task Force Earthquakes of the German Research Center for Geosciences (GFZ). The main purpose of the task force is to coordinate the allocation of an interdisciplinary scientific-technical expert team after catastrophic earthquakes worldwide [4]. In particular, we focus on the "Transport of Equipment" process of the task force. It specifies how the transport of scientific equipment from Germany to the disaster area is handled.

In order to elicit a process model for the "Transport of Equipment" process, typically, a process analyst conducts workshops with members of the task force and interviews them about the process. As a result, the analyst is faced with descriptions of how the transport of equipment is done in general. These descriptions resemble *stories* as they relate to the concrete experience of the team members gained during recent missions.

In our example, one story relates to preparation activities enacted in Germany, such as organizing transport of cargo and the actual transportation to the airport. Another story

relates to the activities that are done immediately after arrival in the disaster area, e.g., buying maps and renting vehicles. Once the standard processing has been clarified, the analyst discusses exceptional cases with the team members. Again, these descriptions can be seen as stories that describe 'what if' scenarios. In our exemplary process, for instance, customs might require that the equipment is presented before the decision on clearance is taken. Based thereon, the analyst abstracts the actual process model containing the complete standard processing along with all exceptional cases. In case of the "Transport of Equipment" process, the resulting model shows a complex structure as illustrated in Fig. 3 on the right.

3 Tool support for scenario-based modeling

How to model processes with GRETA**.** The scenario-based modeling approach differs from the typical approach discussed in Sect. 2 in the final step. Instead of creating one Petri net that copes with all eventualities, the modeler expresses each story of the "Transport of Equipment" process as a scenario. These scenarios are then assessed for continuations and dependencies. GRETA supports scenario-based process modeling with *oclets* by a graphical editor. To express a scenario in GRETA, the modeler creates a new oclet and models the scenario's behavior as a Petri net in the oclet's lower compartment as shown in Fig. 1 on the left and middle. To express when the scenario may occur, the modeler describes the *history* that triggers the scenario in the oclet's upper compartment. Each oclet is created *locally*, focusing on its story and its local history.

Typically, standard scenarios such as *depart standard* of Fig. 1 are modeled first. Then, variants and exceptions like *depart devices* are introduced one oclet at a time. The modeler may refine the oclets iteratively to achieve consistency of the overall processing. To this end, GRETA provides animated execution of oclets. The modeled oclets can be executed step-wise until a situation is reached in which a particular oclet should "fit", i.e., can be executed. If this is not the case, GRETA allows pausing the animation, changing the oclet's history, and resuming the animation at the current state. Thus, GRETA supports the modeler effectively in relating different scenarios to each other.

The tool GRETA**.** Having illustrated how GRETA supports scenario-based process modeling we now explain some technical aspects of the tool. We designed GRETA in a model-driven approach: data structures (oclets, process instances) and the graphical editor were modeled in respective languages of the *EMF/GMF framework* of *Eclipse* (www.eclipse.org). These models were translated to executable code which we extended manually to improve usability; details are given in [6]. GRETA exposes its data structures and user interface as *plugins* as illustrated in Fig. 2.

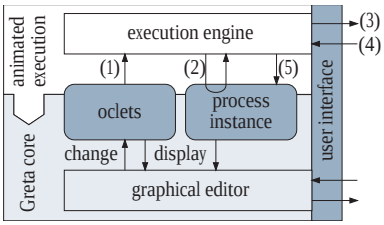

Fig. 2. Architecture of GRETA.

GRETA's *animated execution plugin* provides an *execution engine* for scenario-based models. When started, the engine (1) determines the *enabling condition* of each action in the denoted oclets. For instance, in Fig. 1 action *demonstration of devices* is enabled iff *present equipment* occurred. Then, the engine (2) checks which actions are enabled in the

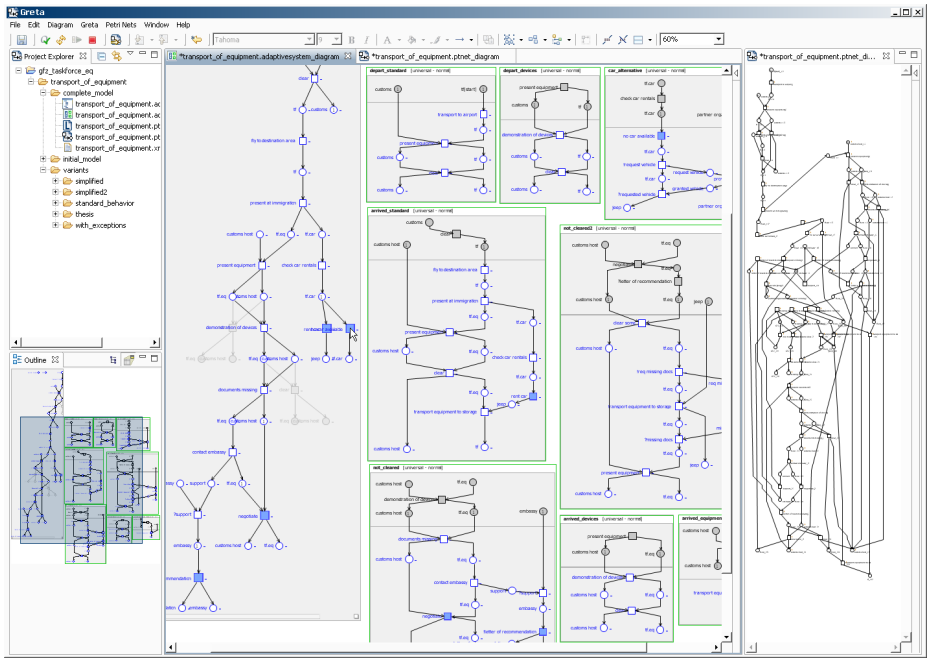

Fig. 3. The complete model of the "Transport of Equipment" process of the "Taskforce Earthquakes" (in the middle) and the Petri net model that can be synthesized with GRETA (on the right).

current process instance and (3) presents all enabled actions to the user by highlighting them. In Fig. 1, the current process instance is depicted on the right as gray shaded nodes with thin borders. Actions *demonstration of devices* and *clear* are enabled and highlighted as possible extensions of this process instance. The user (4) picks one enabled action for execution by clicking on it, and the engine (5) extends the process instance by the chosen action. The user may pause, stop, or reset the execution at any time to change the modeled oclets as she wishes.

Scalability. GRETA can handle models of complex processes consisting of dozens of scenarios with various exceptions. Fig. 3 shows a typical modeling situation in GRETA. The central window shows all oclets of the "Transport of Equipment" process of the "Taskforce Earthquakes" and a process instance on the left with four enabled actions. The two actions on the right are alternatives, all other actions are enabled concurrently. All enabled actions originate in different oclets as highlighted.

As an extension, GRETA supports *anti-oclets* which describe behavior that *must not* occur in the process. Further plugins for GRETA allow the modeler to *check consistency* of the model (i.e., whether each action of each oclet can actually be triggered by some other oclet), to *verify soundness* of a process model, and to automatically *synthesize a classical Petri net-based process model* such as the one depicted in Fig. 3 on the right. The synthesized Petri net exhibits exactly the behavior modeled in the oclets. Though the net is not necessarily "elegant" because the modeled real-world process has many exceptions and alternatives and does not follow a block structure. In Fig. 3, the oclets are structurally simpler and hence easier to comprehend than the Petri net.

4 Discussion and Conclusion

This paper presented GRETA as a *proof of concept prototype tool* that demonstrates the feasibility of *scenario-based process modeling and execution*. Currently, we are successfully applying GRETA in a case study where we model the process of the "Taskforce Earthquakes" with oclets as explained in Sect. 2 and 3. The case study itself is work in progress; some more information can be found in [4, 7]. GRETA and all its source code together with several examples are available for download at `http://service-technology.org/greta`.

Related tools and approaches. First and foremost, our work relates to approaches of scenario-based process modeling. Desel et al. advocated to model a business process in terms of its partially ordered runs [8] or expressions over finite scenarios [9]. The VipTool allows to synthesize a Petri net-based process model from such a scenario-based model [9]. Oclets extend these ideas and provide a mechanism to control the chaining of scenarios in terms of local histories. GRETA directly executes oclet-based models. Both ideas are influenced by *Live Sequence Charts* and the *play engine* [10]. In comparison to these, oclets directly serve business process modeling as they are based on Petri nets.

A closely related approach are *proclets* [11]. A proclet is a small workflow net in which actions send and receive data via channels. A process model is a set of proclets that are instantiated and coupled along their channels according to an underlying business object model. In [12], process behavior is described with a set of business object life cycle models. In both approaches, interaction between process artifacts follows from a data model, whereas *oclets* focus on control flow and use the notion of a *history* to describe when a behavior may occur; anti-oclets provide additional expressive power.

A high degree of variation of a common business process might be addressed using concepts of flexible process management. Research projects like ADEPT [13] and WASA [14] developed process management systems that enable ad-hoc modification of a process model for certain process instances. The YAWL workflow engine provides *worklets* [15], i.e., subprocesses that can be chosen and instantiated at runtime. The main differences between GRETA and existing execution engines root in the underlying semantic model: oclets are defined on the semantic model of *distributed runs* and the process instance's *history* defines which actions are enabled [5]. Existing execution engines operate on *sequential runs* and decide enabling of actions based only on the current *state*: history information is not available. The main difference to ADEPT is that ADEPT requires symmetrical, block-structured process models [13]. In contrast, oclets are unconstrained and describe process flow "as is". So, integrating oclets into an existing engine, e.g., as a plugin, remains a challenging and interesting task.

Finally, oclets can be seen as reusable patterns from which a process model is derived. Patterns have been proposed to business process modeling on various abstraction levels, e.g., control-flow patterns [16], business semantics aware action patterns [17], activity patterns that represent micro workflows [18]. Such patterns might be leveraged for modeling support, for instance, to accelerate process modeling and minimize modeling errors [19]. In general, oclets are more specific than the aforementioned patterns. They focus on partial scenarios and dynamically constructing process instances from scenarios whereas patterns aim at reuse or modeling support in a broader context.

Future work. In future work, we aim at lifting oclets to high-level modeling languages, such as BPMN. Here, the question of appropriate notions of a history has to be answered. Further on, the execution of scenarios involves the decision on a dedicated continuation once multiple oclets are activated. Such a decision should be supported by a tool suggesting the best continuation relative to a quality criterion (e.g., execution time).

Acknowledgements. We thank the referees for valuable comments and suggestions.

References

1. Rosemann, M.: Potential pitfalls of process modeling: part A. Business Process Management Journal **12**(2) (2006) 249–254
2. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7pmg). Information & Software Technology **52**(2) (2010) 127–136
3. van der Aalst, W.M.P., Rubin, V., Verbeek, H.M.W., van Dongen, B.F., Kindler, E., Günther, C.W.: Process mining: a two-step approach to balance between underfitting and overfitting. Software and System Modeling **9**(1) (2010) 87–111
4. Fahland, D., Woith, H.: Towards process models for disaster response. In: Workshops of the BPM'08. Volume 17 of LNBIP., Springer-Verlag (2008) 244–256
5. Fahland, D.: Oclets - scenario-based modeling with Petri nets. In: Petri Nets 2009. Volume 5606 of LNCS., Paris, France, Springer-Verlag (June 2009) 223–242
6. Wolf, M.: Erstellung einer modellbasierten Laufzeitumgebung für adaptive Prozesse. Diplomarbeit, Humboldt-Universität zu Berlin (September 2008)
7. Weidlich, M., Zugal, S., Pinggera, J., Fahland, D., Weber, B., Reijers, H., Mendling, J.: The impact of change task type on maintainability of process models. In: ER-POIS 2010, held in conjunction with CAiSE 2010. (2010) to appear.
8. Desel, J.: Validation of process models by construction of process nets. In van der Aalst, W.M.P., Desel, J., Oberweis, A., eds.: BPM. Volume 1806 of LNCS., Springer (2000) 110–128
9. Bergenthum, R., Desel, J., Mauser, S., Lorenz, R.: Construction of process models from example runs. T. Petri Nets and Other Models of Concurrency **2** (2009) 243–259
10. Harel, D., Marelly, R.: Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine. Springer-Verlag (2003)
11. van der Aalst, W.M.P., Barthelmess, P., Ellis, C.A., Wainer, J.: Workflow Modeling using Proclets. In: CoopIS 2000. Volume 1901 of LNCS., Springer (2000) 198–209
12. Küster, J.M., Ryndina, K., Gall, H.: Generation of business process models for object life cycle compliance. In: BPM 2007. Volume 4714 of LNCS., Springer (2007) 165–181
13. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. Computer Science - R&D **23**(2) (2009) 81–97
14. Vossen, G., Weske, M.: The WASA2 object-oriented workflow management system. In: SIGMOD Conference 1999, ACM Press (1999) 587–589
15. Adams, M., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Worklets: A service-oriented implementation of dynamic flexibility in workflows. In: OTM. Volume 4275 of LNCS., Springer (2006) 291–308
16. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases **14**(1) (2003) 5–51
17. Smirnov, S., Weidlich, M., Mendling, J., Weske, M.: Action patterns in business process models. In: ICSOC. Volume 5900 of LNCS. Springer (2009) 115–129
18. Thom, L.H., Reichert, M., Iochpe, C.: Activity patterns in process-aware information systems: Basic concepts and empirical evidence. IJBPIM **4**(2) (2009) 93 – 110
19. Gschwind, T., Koehler, J., Wong, J.: Applying patterns during business process modeling. In: BPM. Volume 5240 of LNCS., Springer (2008) 4–19