Towards a Generic Hybrid Simulation Algorithm Based on a Semantic Mapping and Rule Evaluation Approach

Christoph Prackwieser, Robert Buchmann,
Wilfried Grossmann, and Dimitris Karagiannis

University of Vienna, Research Group Knowledge Engineering
Währingerstrasse 29, 1090 Vienna, Austria
{prackw,rbuchmann,dk}@dke.univie.ac.at,
wilfried.grossmann@univie.ac.at

Abstract. In this paper we present a semantic lifting methodology for heterogeneous process models, depicted with various control flow-oriented notations, aimed to enable their simulation on a generic level. This allows for an integrated simulation of hybrid process models, such as end-to-end models or multi-layer models, in which different parts or subprocesses are modeled with different notations. Process simulation outcome is not limited to determining quantitative process measures as lead time, costs, or resource capacity, it can also contribute greatly to a better understanding of the process structure, it helps with identifying interface problems and process execution requirements, and can support a multitude of areas that benefit from step by step process simulation - process-oriented requirement analysis, user interface design, generation of business-related test cases, compilation of handbooks and training material derived from processes.

Keywords: business process modelling, simulation, hybrid process models, semantic lifting.

1 Introduction

Process simulation is often seen as a method to determine such quantitative process measures as lead time, costs, or resource capacity. However, simulating a process and graphically animating or highlighting its relevant paths, also contributes to a better understanding of the process, helps with identifying interface problems, and supports communication of altered process implementations (possibly deriving from the introduction of new software, underlying technology or from organizational changes). Despite the fact that process simulation is a well-known analysis method, it is surprisingly not very widely used in practice, on one hand due to quantity and quality of input data, on the other hand due to the models themselves. This may be caused by the diversity of business process modeling languages generating heterogeneity of the process repository even within the same company, and thus requiring extensive alignment in order to setup the input for simulation. Each language tries to find the right level of abstraction, expressing concepts that are fit to different contexts, while

M. Wang (Ed.): KSEM 2013, LNAI 8041, pp. 147–160, 2013.

ignoring others. Some modelers may see shortcomings where others see strengths as a recent argumentation of Prof. Reisig [1] proves.

With our approach, we aim to overcome the legacy of notation heterogeneity, under the assumption that different individuals or units contribute with fragments using different control flow-oriented process modeling notations to a hybrid model encompassing a cross-organizational, cross-functional or multilayered process. Best practices from supply chain management, such as SCOR [2] recommend a tight integration and visibility of processes, from the customer's customer to the supplier's supplier.

The presented approach allows for simulating such heterogeneous models, while preserving their graphical representation, thus reducing additional re-modeling efforts and increasing the original model creator's identification with the simulation result.

With the rise of quite universally usable modeling notations like Business Process Modeling Notation (BPMN) or Unified Modeling Language (UML) this problem may be defused over time (under the assumption of universal acceptance) but there are still a lot of legacy models in companies and not every modeling requirement is satisfied with such standardized modeling notations [3]. Models that were created for special project goals in another modeling notation could be linked and simulated together to form, for example, a cross-functional end-to-end process, including business-focused and IT-related process models. This also reduces the initial effort for model creation.

The structure of the paper is as follows: After this introduction we discuss related literature and similar approaches in Section 2, where the problem is also formulated in the context of hybrid process modeling. In Section 3, we describe our approach, both conceptually and with a proof of concept, along with some implementation notes regarding the model transformation required for running a simulation, and the simulation itself. A conclusive SWOT analysis closes this paper.

2 Problem Statement and Background

2.1 Problem Positioning

In literature the term "hybrid" is used for a variety of model characteristics or simulation functionalities [4]. In this paper, a process model is called a "hybrid process model" when at least one of its process fragments or sub-processes is modeled in a different modeling notation. The analysis of end-to-end processes concerns a variety of domains and, therefore, involves different stakeholders. Although every domain has to contribute to the overall enterprise goals, each one has its own circumstances and objectives that lead to the application of specialized approaches and instruments to deal with each domain's requirements [5]. This general observation is also true for modeling languages. Although there are standardized modeling methods like BPMN or UML, special process-oriented application scenarios such as compliance management, risk management, or platform-specific process automation are not sufficiently supported. Hybrid process models can appear in a horizontal-oriented form as an end-to-end process, a vertical-oriented multi-layer process, or a combination of both.

Horizontal-oriented hybrid process models can come up when domain specific process models of two or more departments/units are linked together subsequently to form a cross-functional end-to-end process. Vertical hybrid models can be a result of a classical functional decomposition method wherein different modeling notations are used to depict various levels of granularity or stakeholder perspectives on the model. A strict vertical orientation means that the whole model is built up by sub-process models that are linked to form a hierarchy and are not aligned subsequently. The link is established from a process step to a corresponding sub-process, which describes the step in greater detail. Another application example described by Muehlen and Recker as "emerging tendency in industry to separate business-focused process modeling from implementation-oriented workflow implementation" [6] where workflow models typically use implementation platform-specific modeling languages or special dialects of standardized languages, such as BPMN. Also for classical, not workflow oriented software implementation a business focused process model could have links to a user interface flow model to show the software support of specific business-oriented tasks. In this context, we state the problem through its following components:

Assumption: A manufacturing company coordinates a virtual enterprise with multiple possible configurations of its supply chain. **Open issues:** The coordinating company needs to evaluate end-to-end processes for specific virtual enterprise instances. The participating business entities are willing to provide process visibility, but they use different tools and languages to model their processes;

Assumption: A manufacturing company has its process map designed according to a multi-layered framework such as SCOR. **Open issues:** The company aggregates processes from its multiple SCOR layers to simulate their cost but uses different notations for high level supply chain and low level workflows, as required by auditing and documentation purposes.

Proposed solution: A methodology (enabled at meta-modeling level) of annotating process elements represented with different notations but having overlapping semantics, in order to enable the detection of work order along simulated process paths (while preserving the original notations).

2.2 Literature and Related Work

A starting point for our research interest was the "Hybrid Method Engineering" approach introduced by Karagiannis and Visic [3] as part of the "Next Generation Modeling Framework" (NGMF). On a conceptual level the approach deals with the merging of two or more modeling methods into a single hybrid metamodel, which is described in more detailed [7] using an encapsulation mechanism. They also present a corresponding technical realization platform and an organizational structure. Their approach is based on Karagiannis and Kühn's [8] proposed structure of modeling method components. Therein, different modeling languages of a hybrid method form together with modeling procedures a modeling technique. The modeling technique and dedicated mechanism and algorithms are the main parts of a modeling method in turn. These mechanism and algorithms are automated functionalities, which are performed on the models in order to use or evaluate them. To define a hybrid

metamodel they suggest the usage of a metamodeling language which is defined by a meta-metamodel. In this approach, a generic algorithm such as universal simulation, which should work on all involved modeling languages, must be implemented on a meta^2-model whereof all language specific sub metamodels are instantiated.

Whereas there are several publications describing transformation from one modeling language to another often motivated by Model Driven Architecture (MDA) [9] approaches, quite few sources are concerned with the conceptualization of generic (simulation) algorithms for hybrid models. A good overview of modeling language transformations involving block-oriented languages, especially BPEL, is provided in [10]. Other papers describe, for example, transformations from BPMN to BPEL [11], from UML to BPEL [12], and from BPEL to EPC [13]. Because of the numerous analysis methods and simulation algorithms applicable on Petri Nets, there are also some publications concerning transformations from BPEL to Petri Nets [14]. To deal with simulation tools that are merely capable of processing models in one specific language, different modeling method transformation approaches are presented by various authors. Rozinat, Wynn et al. [15] use a simulation tool that is based on Colored Petri Nets (CPN) so they describe how to convert YAWL models by preserving simulation relevant data into Petri Nets.

3 The Proposed Approach

3.1 The Simulation Core Concepts

As stated earlier, we strive to enable a generic simulation algorithm which has **the key requirement** of being applicable on control flow oriented process models while leaving the simulated model notations unchanged in order to improve the user's identification with (and understanding of) simulations results. To simulate different process modeling languages using one generic algorithm there is a need for a semantic mapping of specific modeling language object types to the metamodel on which the simulation algorithm relies.

There are alternative realization options to fulfill the key requirement. One could hardcode all possible object types of different predefined modeling languages into the simulation algorithm. However, this alternative lacks flexibility, as with every newly added domain-specific language program code of the simulation algorithm has to be changed. Another possibility would be to transform the graphical modeled process each time immediately before a simulation run takes place into a "super notation" known by the simulation algorithm. The problem hereby is that the user has to depend on this automatic transformation step and cannot check if the meaning of the process model is translated correctly in the intermediate format. A token animation, for example, to depict waiting queues is difficult to implement because the simulation and animation algorithm are not executed on the basic models directly.

In our approach, the algorithm works directly on the graphical model, which is enhanced by a layer of generic formulated information about the objects' semantics, flow control, and resource assignment. This information is generated during a separate transformation step and stored in specific attributes of each modeling object

(however this transformation is much less complex than a complete redesign of models in another notation). Thus, an experienced user is able to alter the output of the transformation step before the simulation takes place; the user therefore, has the ability to formulate the input for the simulation specific to individual needs.

To identify necessary semantic core concepts of a generic simulation algorithm which have influence on the control flow of a process, we analyzed various graphical process modeling notations. To be sure to support both horizontal and vertical oriented hybrid models, we selected modeling languages that are used for at least one of the following domains: a) Strategic core processes; b) Business processes; and c) IT-specification, process automation.

We compiled the list of the major business process modeling methods with graphical notations out of a structured literature study [16-19]. Included notations are Process Maps and Value Added Chain Diagrams (VACD), Business Process Modeling Notation (BPMN), Event-Driven Process Chains (EPCs), UML Activity Models, ADONIS Business Process Management Systems (BPMS) [20], Process Schematics of the Integrated Definition Method 3 (IDEF3), Flow Chart Notation, Task Flow User Interface Modeling and Business Process Execution Language for Webservices (BPEL4WS).

Table 1. Semantic concepts of flow objects relevant for simulation

Notation	Semantic Concepts of Flow Objects Relevant for Simulation								
	Activity	Sub Process	Start	End	XOR	AND	Merge	Event	Neutral
Value Chain Models	x	x	x	x					x
UML 2.0 Activity Diagrams	x		x	x	x	x	x		
EPC	x	x	x	x	x	x	x	x	x
ADONIS BPMS	x	x	x	x	x	x	x	x	x
BPMN 2.0	x	x	x	x	x	x	x	x	
IDEF3	x	x			x	x	x		
Flow Charts	x	x	x	x	x	x	x		
Mask Flow Diagrams	x	x	x	x	x			x	
BPEL	x	x	x	x	x	x	x	x	

Table 2. Simulation-related semantic concepts

Semantic Concept	Keyword	Description
Activity	ACT	Describes an executable task in a process. Each execution consumes execution time of a resource such as a role or IT system and causes costs.
Sub Process	SUB	Is a placeholder for a linked process fragment. It is used for structuring of process models or reuse of process fragments.
Start	STA	Depicts the arrival of new entities into the process model. The simulation generates corresponding process token which are transfered through the model
End	END	Marks the end of a process path. Each arriving token will be consumed.
XOR	XOR	Depicts an exclusive or decision of different outgoing control flow branches
AND	AND	Depicts the possible parallel execution of the same process instance. Therefore the token will be cloned at the AND object. Outgoing control flow branches can be annoted with condition which leads to an OR functionality
Merge	UNI	All at the latest AND object cloned tokens are merged to one token. This synchronisation can lead to waiting time.
Event	EVE	Depicts an intermediate event which controls the flow of tokens regarding predefined condtions. A flow interruption can lead to waiting times
Neutral	NTR	It is within the controlflow but has no effekt on token flow or simulation result itself. It can be used for example for cross-references between processes.

In this simulation approach, we consider the control flow only. Information flows like message handling in BPEL4WS are not supported so far. Furthermore, the simulation algorithm is currently applicable for event-based graphical modeling notations only where transitions and connecting control flows are modeled explicitly.

To extract the semantic core pattern of simulation perspective, we analyzed each modeling notation and extracted all of the simulation relevant patterns in regard to flow control and influence on simulation results. As a result of this study we compiled a list of nine core patterns, as we have illustrated in Table 1. In Table 2, all identified simulation-related sematic concepts are explained. The listed keywords are used for rules formulation. The mapping of object types to semantic simulation concepts is not sufficient. To gain more flexibility and cover as much semantics as possible, we add a rule mechanism to determine the next object in the control flow.

One of the preconditions of our approach is that each modeled object is mapped to exactly one simulation core concept. The introduction of another semantic concept would solve this problem, but we want to keep the number of semantic simulation concepts as low as possible. Therefore we developed a rule language that enhances the semantic meaning of an object with annotation indicating the next object in the control flow, to be taken as input by the simulation algorithm. The basic syntax is the following:

_KEYWORD_*RuleForDeterminationOfSubsequentObject*

3.2 The Rules Syntax

We use the rules to identify the subsequent object in a process, as an input for the simulation algorithm. However, the rules mechanism is not limited to the representation of the control flow only, so we can also utilize it to express conditions for the assignment of executing resources to an activity or to formulate a distribution of entry arrival in start objects.

To represent a rule in our concept we use the OMG Production Rule Representation as a basic syntax where a production rule is defined as a "**statement of programming logic** that specifies the execution of one or more actions in the case that its conditions are satisfied" [21]. A production rule consist of one or more conditions in the form of logical formulas and one or more produced actions in the form of action expressions [22]. A typical conditional rule is represented as:

```
IF condition THEN producedAction ELSE alternativeAction
```

This syntax is capable of expressing rules of the following types:

Flow rules. In control flow-oriented process models the chronological subsequence of tasks is typically depicted by connecting arrows (directed edges) from a predecessor to its immediate successor object. A flow rule that is annotated at the preceding object reproduces the information about the successor given by the outgoing arrow. The modeled arrows remain in the process model; although they are not necessary for the simulation anymore, they are very important for a quick understanding of the process structure and the simulation results. We are aware, though, that this enhancement is redundant information, but by introducing an

additional layer, we gain the flexibility to describe the control flow independent of the modeling languages, to be used by the generic simulation algorithm. Therefore, the evaluation of the flow rules has to lead to the designated successor regarding the control flow of a process model. For the identification of the subsequent object a unique identifier, we use the unique internal ID of an object provided by the meta-modeling platform.

One of the most frequently used control flow structures in process management is a sequential flow. To express this structure in a simple form, we introduce the following short form syntax here in conjunction with a keyword (see Table 2):

```
_KEYWORD_IDOfSubsequentObject
```

This is the same as:

```
_KEYWORD_IF TRUE THEN IDOfSubsequentObject
```

In **Fig. 1**, there are two examples for sequential flows and the respective mapping results shown. On the left side, a BPMN task with ID 101 has the semantic meaning of an activity and a subsequent object ID 102. The right side shows a Value Chain that has no connecting arrows at all, but the subsequence is given by the horizontal position of the objects. Also, the process objects have a semantic meaning of activities. The flow rule subsequence is derived from the object's graphical position.

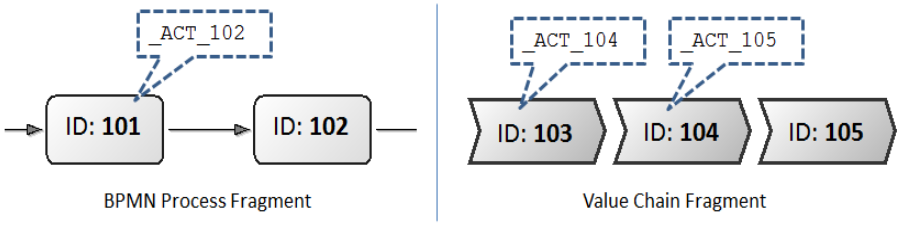


Fig. 1. Flow rule examples for a sequential control flow

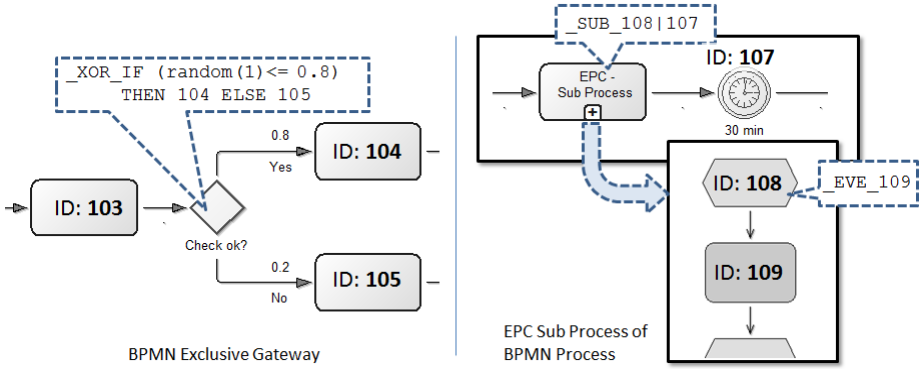


Fig. 2. Flow rule examples for gateways and subprocesses

A typical example for the application of a flow rule is the occurrence of a XOR in a process model. As depicted in **Fig. 2** on the left side the Exclusive Gateway in BPMN notation corresponds to an XOR and is expressed in Flow Rule notation as a programmatic condition. The statement "random(1)" generates a randomly distributed number which is used to decide on the subsequent object, according to the given branch probability for each token at simulation runtime. On the right side, there is a sub-process object in a BPMN process, which calls for a process in EPC notation. This is an example of a hybrid process model. BPMN sub-process objects corresponds to the semantic concept Sub Process "SUB". The syntax *'IDOfSubProcessObject | IDOfSubsequentObject'* commands the simulation algorithm to simulate the sub-process beginning with the object *IDOfSubProcessObject* first. After running through the sub-process and returning to the calling process, the process will then proceed with the object *IDOfSubsequentObject.* These IDs can also be the results of (nested) rules.

Resource rules. Some process modeling notations allow for the declaration of activity executing resources (for example, EPCs and; ADONIS BPMS). Resources are often concepts like roles, persons, organizations, tools, or IT-systems, such as software applications or hardware. A simulation analysis that yields to quantitative results like lead time, capacity consumption, or process costs should include these resources. Therefore, every activity has to be evaluated with an average execution time. Every time this activity is executed, the assigned resource is occupied for the whole execution time. Rules can be used to assign resource IDs to activities.

Start rules. Objects that map to the semantic simulation concept "Start" are the starting point for new process instances and depict the arrival of new entities in the process model. The amount of arriving entities per period can follow a random distribution, deterministic or conditional [23]. The result of the rule evaluation is the quantity of tokens which have to be generated by the simulation algorithm in the respective period or at a specific simulation event. The rule language supports the access to global simulation variables, such as the simulation clock. This allows, for example, a time-conditioned control of process starts.

```
IF (Period <= 20) THEN 1 ELSE 2
```

This rule commands the simulation algorithm to create one process instance in each of the first 20 periods; afterwards, there have to be created two process instances per period.

3.3 Transformation

To turn a hybrid process model into a simulate-able model, a mapping for each modeling object to exact one semantic simulation core concept combined with a flow rule formulation has to be accomplished and stored in a designated object attribute. This task can be executed manually, but for often-used modeling languages, a software-assisted algorithm is required.

Depending on the syntax and semantics of the initial modeling notation, the transformation algorithm has to handle different situations. In case the semantics of

the modeling notations matches the simulations core semantic, like in some flowchart notations, only a 1:1 mapping of modeling object types to simulation concepts is necessary. However, with increasing complexity and expressiveness of initial modeling notations, the transformation algorithm has to not just recognize the object type, but it must also interpret the context of an object. As an easily understandable example is that an AND from the EPC notation could be of the semantic type AND if it has more than one outgoing connectors, or the semantic concept of a merge if there is just one outgoing connector.

In some cases, further user input is required. For example, in a value chain, a process object might have more than one outgoing connectors; this could be interpreted as either the start of a parallel or an alternative path. Besides the transformation of objects and control flows, there might also be a need for transforming a resource assignment. In the EPC notation, resources are modeled as separate objects connected by a relation to the specific function. In ADONIS BPMS, there are object references linked from specified attributes of activities to resource objects like roles. In BPMN, there is no standardized resource assignment, but a user can decide to adopt the organizational object assigned to the respective lane.

As there are no defined modeling standards, the formulation of start-rules has to be made manually.

3.4 Simulation

Static modeling objects, like activities or events, are traversed by a temporary entity's act on them. The simulation algorithm's main task is to control the time advance, to manage the interaction of static and temporary objects, and to collect results. During a simulation run, temporary objects handled by the algorithm are called tokens. The algorithm provides data for the visualization component to visualize the token path throughout the model, and updates the simulation log with collected data. The result component relies on the log file to create reports on various aggregation levels.

Thus, validity of results is dependent on a) the correctness of the original models; b) the availability and correctness of the data to be collected from the model (which should be execution data stored in attributes of the modeling objects, like time, cost); c) the successful transformation of the hybrid process model, as described in the previous section. There has to be at least one object with a semantic meaning of "Start" including a valid start rule. To execute the algorithm, the start model has to be selected, and the number of intervals or process instances that should be simulated has to be provided.

In this paper, we concentrate on the method of enabling a generic simulation via the semantic mapping. Therefore, we will not describe the functional principles of the algorithm in deep detail. As state changes in business processes happen at precise time points, we use a Discrete Event Simulation Algorithm [24] as a basis for our approach. To provide a suggestive token animation, we chose the time slicing method, with fixed intervals to move the model forward in time. Roughly, for each time interval, the following steps are executed by the algorithm:

- At the beginning of each time interval, the algorithm identifies all objects that are mapped to the semantic simulation concept "Start." The algorithm evaluates in cooperation with the rule engine each start-rule of these objects and creates a corresponding number of new tokens, which represent process instances;
- Afterwards, the algorithm traverses each active token in the model. The status of a token can be "idle" or "ready." A status of "idle" indicates that the corresponding work order is still being executed by an activity or that the token has to wait due an event, synchronization in hand, or occupied resource. A token with the "ready" status will be moved by the algorithm to the subsequent object, according to the flow rule of the token's current object, until the token status changes to "idle" or the token is terminated by an end object;
- For report generation and further applications, every step is logged during the simulation run;
- All token movements are depicted by a visualization component.

3.5 Proof of Concept

Simulation results are not only useful for time, capacity, or cost based analysis. A study of the dynamic behavior of a process model helps to understand the wanted and unwanted interactions between process instances better and creates an additional time-based and animated visualization on the otherwise static models.

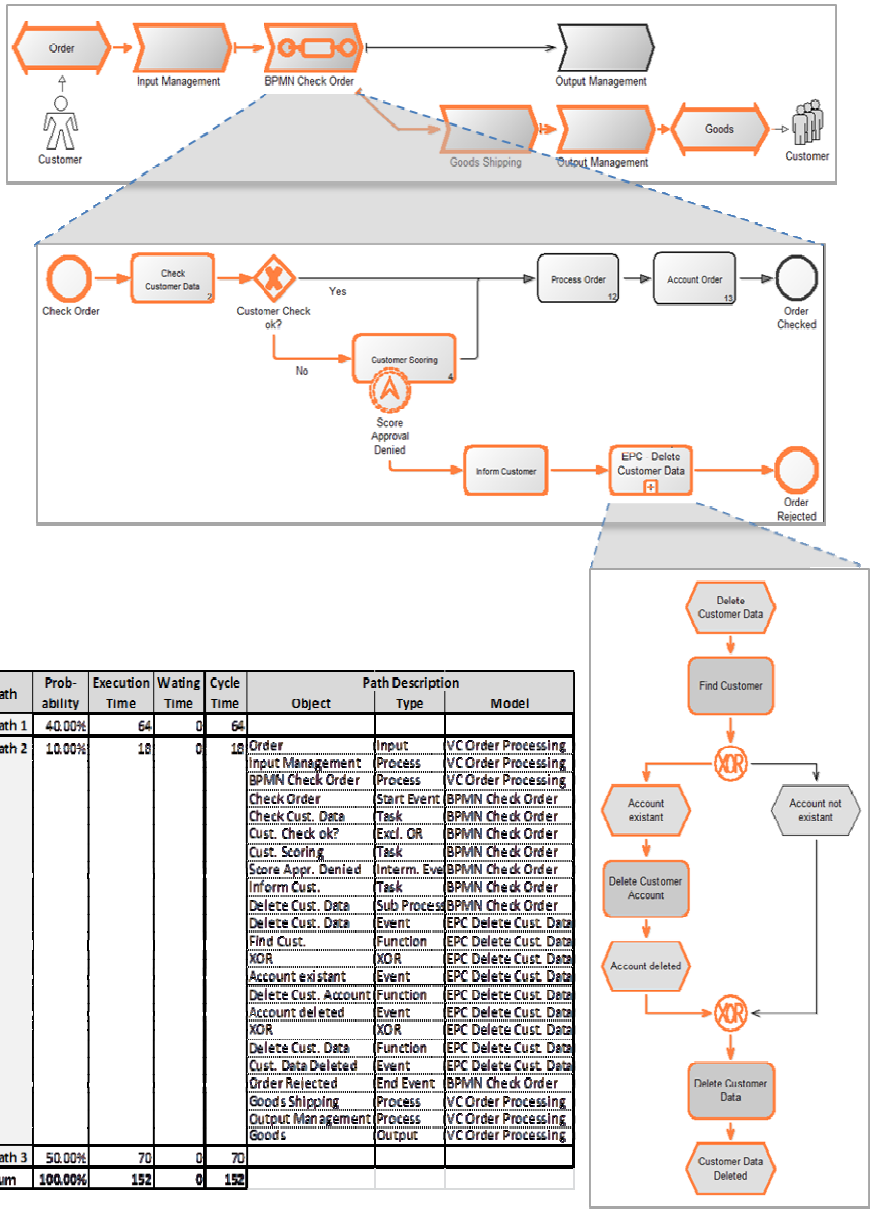
One advantage of simulation in contrast to analytic methods is the possibility to identify and investigate specific process paths throughout the end-to-end process model. Each distinct chronological sequence of modeling objects that are traversed by at least one process instance (case) is a process path.

As a first evaluation step of this approach we integrated different modeling notations such as Value Chain, BPMN and EPC in a meta-modeling platform (the list from Table 2 should be supported with minimal additional effort once the transformation is in place). Further on, we implemented the transformation, simulation and visualization algorithms by using the scripting language of the platform.

Fig. 3 shows screenshots of this implementation. The different models are connected, building a vertical hybrid model from subprocesses using different notations. The application of the simulation algorithm results in a number of possible paths, whereas three are depicted in the result list. Path 2 is highlighted throughout the process models, and all traversed objects and the resulting execution, waiting and cycle times are shown in the table.

In combination with these attributes, the path information could be used to generate auxiliary value:

- process-oriented handbooks or training material;
- business-related test cases;
- process-oriented requirement analysis;
- analysis of IT systems used throughout specific process paths for business continuity management;
- user interface design, especially to present applications behavior regarding specific processes.



Path	Prob-ability	Execution Time	Wating Time	Cycle Time	Path Description		
					Object	Type	Model
Path 1	40.00%	64	0	64			
Path 2	10.00%	18	0	18	Order	Input	VC Order Processing
					Input Management	Process	VC Order Processing
					BPMN Check Order	Process	VC Order Processing
					Check Order	Start Event	BPMN Check Order
					Check Cust. Data	Task	BPMN Check Order
					Cust. Check ok?	Excl. OR	BPMN Check Order
					Cust. Scoring	Task	BPMN Check Order
					Score Appr. Denied	Interm. Eve	BPMN Check Order
					Inform Cust.	Task	BPMN Check Order
					Delete Cust. Data	Sub Proces	BPMN Check Order
					Delete Cust. Data	Event	EPC Delete Cust. Data
					Find Cust.	Function	EPC Delete Cust. Data
					XOR	XOR	EPC Delete Cust. Data
					Account existant	Event	EPC Delete Cust. Data
					Delete Cust. Account	Function	EPC Delete Cust. Data
					Account deleted	Event	EPC Delete Cust. Data
					XOR	XOR	EPC Delete Cust. Data
					Delete Cust. Data	Function	EPC Delete Cust. Data
					Cust. Data Deleted	Event	EPC Delete Cust. Data
					Order Rejected	End Event	BPMN Check Order
					Goods Shipping	Process	VC Order Processing
					Output Management	Process	VC Order Processing
					Goods	Output	VC Order Processing
Path 3	50.00%	70	0	70			
Sum	100.00%	152	0	152			

Fig. 3. Hybrid process model and simulation results

4 Conclusive SWOT Evaluation

In this paper, we presented a semantic mapping methodology acting as an alignment, to enable generic simulation algorithms to run on different modeling notations without altering the graphical representation of the models. A first evaluation of the practicality of this approach was done by implementing it on the ADOxx metamodeling platform [25] and simulating some hybrid process models. A SWOT evaluation generated the following conclusions:

Strengths: Semantic annotations of hybrid models can be enabled on metamodeling platforms to support the lifting of semantics for different languages to a common ground, without affecting the familiarity of modelers with the concrete syntax. Legacy models, or models that follow different notations due to requirements, may be integrated in this way to provide common working ground for generic algorithms performing model evaluation. The metamodeling approach builds up an intermediate layer which allows unified access to all core semantic concepts necessary for the simulation. This approach simplifies the mapping function from an order of n^2 to an order or n (where n is the number of languages to be mapped) and offers a common semantic backbone for the simulation algorithm without having to transform everything to a common syntax ("super notation").

Weaknesses: State-based modeling languages such as Petri Nets are not currently involved in our experimentation. Further evaluation, especially regarding the completeness of the current list of core concepts, is necessary. This is planned by integrating more control flow-oriented modeling notations and testing the semantic core concepts of this approach with the simulation related concepts of the added modeling notations.

Opportunities: Use cases that require handling hybrid process models can be found particularly in collaborative supply or value chains. Frameworks such as SCOR, as well as the paradigm of virtual enterprises, promote inter-organizational process visibility and evaluation. While it is difficult to guarantee that processes in ad-hoc virtual enterprise configurations are modeled with the same language, our annotation approach potentially preserves heterogeneity and at the same time enables sufficient homogeneity for certain simulation approaches. Other domains may also benefit from the type of simulation enabled by our proposed approach. Some examples are software engineering (requirements analysis driven by processes [26] automatic business-oriented test case generation or prototyping in user interface design) and e-learning (scenario-based training [27] or learning space simulations [28]).

Threats: The problem approached in this paper can potentially be defused by imposing a universal alignment to standard notations (BPMN). However, we consider this assumption to be unrealistic in the near future. Simulation tools such as Arena also support notations abstract enough to cover all control flow concepts in a unified way, but our approach aims to preserve the original models and to provide the semantic lifting in a more transparent manner (enabled on the metamodeling level, by the annotating rule attributes).

References

1. Reisig, W.: Remarks on EgonBörger: "Approaches to model business processes: a critical analysis of BPMN, workflow patterns and YAWL, SOSYM 11: 305-318". Software and System Modeling 12(1), 5–9 (2013)
2. Supply Chain Council, The Supply Chain Operations Reference Model, http://supply-chain.org/resources/scor
3. Karagiannis, D., Visic, N.: Next Generation of Modelling Platforms. In: Grabis, J., Kirikova, M. (eds.) BIR 2011. LNBIP, vol. 90, pp. 19–28. Springer, Heidelberg (2011)
4. van Beek, D.A., Rooda, J.E.: Languages and applications in hybrid modelling and simulation: Positioning of Chi. Control Engineering Practice 8, 81–91 (2000)
5. Clark, T., Sammut, P., Willans, J.: Applied metamodelling: a foundation for language driven development, Ceteva, Sheffield (2008)
6. zur Muehlen, M., Recker, J.: How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 465–479. Springer, Heidelberg (2008)
7. Zivkovic, S., Kühn, H., Karagiannis, D.: Facilitate Modelling Using Method Integration: An Approach Using Mappings and Integration Rules. In: Proceedings of the ECIS 2007, vol. 122 (2007)
8. Karagiannis, D., Kühn, H.: Metamodelling platforms. In: Bauknecht, K., Tjoa, A.M., Quirchmayr, G. (eds.) EC-Web 2002. LNCS, vol. 2455, p. 182. Springer, Heidelberg (2002)
9. OMG Model Driven Architecture, http://www.omg.org/mda/
10. Mendling, J., Lassen, K.B., Zdun, U.: Transformation Strategies between Block-Oriented and Graph-Oriented Process Modelling Languages. Vienna University of Economics and Business (2005)
11. Ouyang, C., Van Der Aalst, W.M.P., Dumas, M., Ter Hofstede, A.H.M.: From Business Process Models to Process-oriented Software Systems: The BPMN to BPEL Way (2006)
12. Gardner, T.: UML Modelling of Automated Business Processes with a Mapping to BPEL4WS. In: Proceedings of the First European Workshop on Object Orientation and Web Services at ECOOP (2003)
13. Mendling, J., Ziemann, J.: Transformation of BPEL processes to EPCs. In: 4th GI Workshop on Event-Driven Process Chains (EPK 2005), CEUR Workshop Proceedings, pp. 41–53 (2005)
14. Hinz, S., Schmidt, K., Stahl, C.: Transforming BPEL to Petri Nets. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 220–235. Springer, Heidelberg (2005)
15. Rozinat, A., Wynn, M.T., van der Aalst, W.M.P., ter Hofstede, A.H.M., Fidge, C.J.: Workflow simulation for operational decision support. Data & Knowledge Engineering 68, 834–850 (2009)
16. Wei, W., Hongwei, D., Jin, D., Changrui, R.: A Comparison of Business Process Modeling Methods. In: IEEE International Conference on Service Operations and Logistics, and Informatics, SOLI 2006, pp. 1136–1141 (2006)
17. Fu-Ren, L., Meng-Chyn, Y., Yu-Hua, P.: A generic structure for business process modeling. Business Process Management Journal 8, 19–41 (2002)
18. Wenhong, L., Alex, Y.T.: A framework for selecting business process modeling methods. Industrial Management & Data Systems 99, 312–319 (1999)

19. List, B., Korherr, B.: An evaluation of conceptual business process modelling languages. In: Proceedings of the 2006 ACM Symposium on Applied Computing, Dijon, France, pp. 1532–1539. ACM (2006)
20. BOC Information Technologies Consulting AG: Method - BPMS (Business Process Management System), http://www.boc-group.com/products/adonis/method-bpms/
21. Production Rule Representation (PRR). Version 1.0, vol. formal/2009-12-01 (2009)
22. Wagner, G.: Rule Modeling and Markup. In: Eisinger, N., Małuszyński, J. (eds.) Reasoning Web. LNCS, vol. 3564, pp. 251–274. Springer, Heidelberg (2005)
23. Tumay, K.: Business process simulation. In: Proceedings of the 1995 Winter Simulation Conference, pp. 55–60 (1995)
24. Banks, J.: Introduction to simulation. In: 1999 Winter Simulation Conference Proceedings, WSC 1999. 'Simulation - A Bridge to the Future' (Cat. No.99CH37038) (1999)
25. BOC Group, ADOxx product page, http://www.adoxx.org/live/
26. Li, G., Jin, Z., Xu, Y., Lu, Y.: An Engineerable Ontology Based Approach for Requirements Elicitation in Process Centered Problem Domain. In: Xiong, H., Lee, W.B. (eds.) KSEM 2011. LNCS, vol. 7091, pp. 208–220. Springer, Heidelberg (2011)
27. Liew, B.Y.T., Tsui, E., Fong, P.S.W., Lau, A.S.M.: Rapid Authoring Platform for Instructional Design of Scenarios (RAPIDS). In: Proceedings of IEEE 12th International Conference on Advanced Learning Technologies, Rome, Italy, pp. 473–475. IEEE (2012)
28. Gao, S., Zhang, Z., Hawryszkiewycz, I.: Supporting adaptive learning in hypertext environments: a high level timed Petri net-based approach. International Journal of Intelligent Systems Technologies and Applications 4(3/4), 341–354 (2008)