Model Transformation Chains and Model Management for End-to-End Performance Decision Support

Mathias Fritzsche[1] and Wasif Gilani[2]

[1] SAP AG, Architecture and Innovation Services
Modeling and Taxonomy
Germany
mathias.fritzsche@sap.com

[2] SAP Research Belfast
Enterprise Intelligence
United Kingdom
wasif.gilani@sap.com

Abstract. The prototypical Model-Driven Performance Engineering (MDPE) Workbench from SAP Research permits multi-paradigm decision support for performance related questions in terms of what-if simulations, sensitivity analyses and optimizations. This support is beneficial if business analysts are designing new processes, modifying existing ones or optimizing processes. The functionality is provided as an extension of existing Process Modelling Tools, such as the tools employed by process environments like the jCOM! or the SAP NetWeaver Business Process Management (BPM) Suites as well as classical enterprise software like SAP Business Suite or Open ERP.

By evaluating our workbench for real world cases we experienced that business processes may span different environments, each employing different Process Modelling Tools. The presence of heterogeneous tools influences the end-to-end performance of the overall process. Thus, the MDPE Workbench essentially needs to take the complete process into account. In this paper, a model transformation chain and a model management architecture is explained to enable such functionality. This architecture combines results from our previous publications, outlines these results in more detail and explains them in the context of end-to-end processes. Furthermore, the work is evaluated with an industrial business process which spans three different Process Modelling Tools.

1 Introduction

We experienced that performance related decision support for business processes is especially useful in cases of a high degree of complexity in resource intensive processes, such as processes with layered use of resources or complex workflows. Also the complex statistical distribution of the history data or the plan data related to a process, like a planned workload, demand performance decision support. Business performance related decision support therefore needs to be

© Springer-Verlag Berlin Heidelberg 2011

considered as one integral part of process environments, especially of *Process Modelling Tools*. Such tools are considered as part of such process environments and are the focus of this paper.

In our previous publications we described the Model-Driven Performance Engineering (MDPE) Workbench [1,2,3] which enables the integration of multiple *Performance Analysis Tools* into Process Modelling Tools in order to enable support for decisions which have influence on the process performance. For instance, decision support related to throughput and utilization of resources can be provided by discrete event simulation tools, like AnyLogic [4]. Such tools also enable predictions related to the gross execution time of process instances. This enables to answer questions like "How will the execution time of a Sales Ordering Process be affected in June by adding an additional process step into the process in May?". Additionally, we did experiments with analytical Performance Analysis Tools, such as the LQN tool [5] and the FMC-QE tool [6,7]. Analytical tools are beneficial to answer sensitivity related questions in a short computation times, such as "Which are the most sensitive resources (humans or hardware) of the process for the overall performance?". Moreover, optimization tools can be integrated for performance related decision support, for instance, in order to decide at which point of time a certain business resource is needed to meet processing targets and to optimize utilization of resources. For optimization, existing libraries and tools can be used, such as OptQuest [8] which is employed by the AnyLogic tool as well.

Thus, the integration of different Performance Analysis Tools in Process Modelling Tools via the MDPE Workbench permits multi-paradigm decision support for process modelling. Additionally, we also experienced the need for decision support, which spans across multiple Process Modelling Tools. An example use case would be a business process provided by an enterprise software vendor, like SAP, which is extended with sub-processes. These can be maintained by independent software vendors. In such cases, different parts of the process might be modelled with different Process Modelling Tools which are normally based on different process modelling language, such as BPMN [9], jPASS! [10] or SAP proprietary languages. Thus, the performance related decision support needs to abstract these different languages.

In this paper, we describe the model transformation chain that we have implemented in order to integrate multi-paradigm decision support into multiple Process Modelling Tools. This transformation chain has raised the need for a model management architecture to support tracing of performance analysis results back to the process models, management of process model annotations and administration of the different tools.

The remainder of the paper is structured as follows: The next section describes an industrial example of a Process Modelling Tool chain. In Sections 3 and 4, a solution is proposed which comprises a model transformation chain and a model management approach. The success of this approach is evaluated in Section 5. Finally, in Sections 6 and 7, an overview on the related work and conclusions are provided.

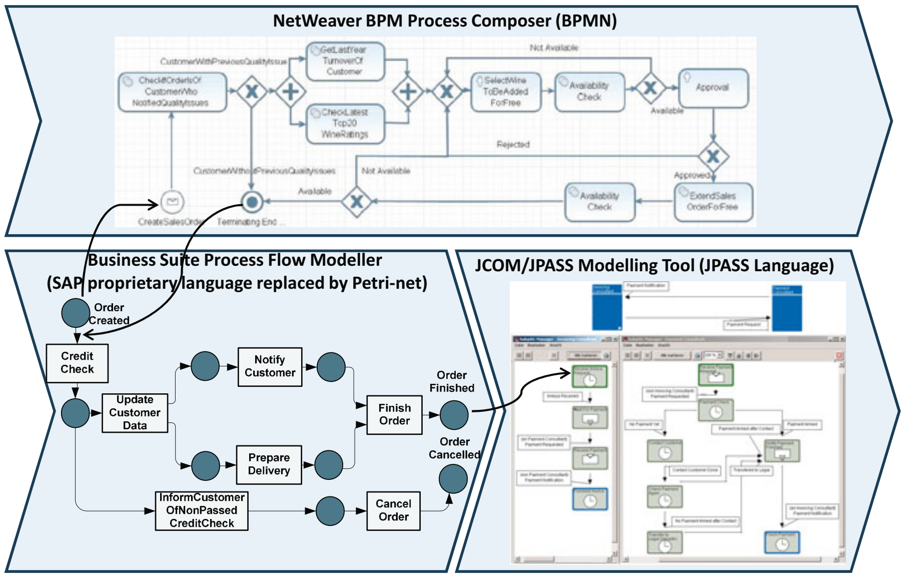
NetWeaver BPM Process Composer (BPMN)
CheckIfOrdersOf CustomerWho NotifiedQualityIssues
CustomerWithPreviousQualityIssue
GetLastYear TurnoverOf Customer
CheckLatest Top20 WineRatings
Not Available
SelectWine ToBeAdded ForFree
Availability Check
Approval
Available
Rejected
Approved
CustomerWithoutPreviousQualityIssues
Not Available
Available
Availability Check
ExtendSales OrderForFree
CreateSalesOrder
Terminating End
Business Suite Process Flow Modeller (SAP proprietary language replaced by Petri-net)
Order Created
Credit Check
Update Customer Data
Notify Customer
Prepare Delivery
Finish Order
Order Finished
Order Cancelled
InformCustomer OfNonPassed CreditCheck
Cancel Order
JCOM/JPASS Modelling Tool (JPASS Language)

Fig. 1. Case Study: A business process spanning across three different Process Modelling Tools

2 Case Study: A Business Process Spanning Multiple Process Modelling Tools

Our case study involves a *Wine Seller* who gets wine supply from several suppliers, and thereafter sells the wine to customers. The *Sales and Distribution Organization* of the *Wine Seller* is supported by a standard software product implementing standard back-end processes, such as Business ByDesign [11] or Business Suite [12]. The back-end process under consideration is called "Sales Order Processing".

The lower left part of figure 1 depicts the "Sales Order Processing" process. Please note that the "Sales Order Processing" sub-process is modelled as so called "Process Flow Model". This is a SAP proprietary and, at the time of writing, not published modelling language. Therefore, the language has been replaced in the figure with a Petri-net [13].

Moreover, as can be seen in the lower right part of Figure 1, this process also triggers an external process for the "Invoice Processing", which is outsourced by the Wine Seller to an external company. In our case study, this external company uses a tool called *jPASS!*, which is the Process Modelling Tool of the jCOM! Business Process Management (BPM) Suite [14]. The jPASS! Process Modelling Tool enables top-down modelling of business processes with the help of the jPASS! modelling language [10,15].

The *Sales and Distribution Organization* of the Wine Seller additionally required an extension of the standard business process so that an extra free bottle of wine could be added to orders of customers who reported a quality issue in their previous order.

This raises the need for an extended version of the "Sales Order Processing" in this case. It is, however, not desirable to change the business process directly in the back-end because the application should be independent of the software vendors life cycle, SAP and jCOM! in our case. Therefore, a third technology is needed that could use the platform back-end business logic. To meet this requirement, the process called "Wine Under Special Treatment" has been developed, which is depicted by the upper part of Figure 1 and was originally introduced in [16].

For the implementation of this process, the BPMN [9] based Process Modelling Tool called "Process Composer", which is a part of the NetWeaver BPM [17] tooling, has been employed. This tool is specifically meant for applying extensions on top of existing back-end processes, such as the described extensions.

We experienced that a domain expert, with no performance expertise, generally cannot predict the performance related consequences if such an extension is deployed. In the context of the wine seller case study, one performance related issue would be to analyse if the additional manual approval steps introduced with the NetWeaver BPM result in an increased end-to-end time than defined as an objective and if so, how to improve the situation.

We also experienced that such analyses need to be done with a combination of different Performance Analysis Tools, for instance, a discrete event simulation tool for what-if questions, an analytical performance analysis tool for sensitivity related questions and an optimization tool to improve the process within user provided constraints.

Moreover, the complete end-to-end process spanning three different Process Modelling Tools needs to be considered since the whole process would be influenced by the newly developed composite process.

In the following sections, these needs are addressed.

3 A Model Transformation Chain for MDPE

In this section, the transformation chain of Figure 2 is proposed to interconnect chains of Process Modelling Tools with multiple Performance Analysis Tools.

This chain has a number of *Process Models* and *Process Model Annotations* as input. This input is transformed via a number of model to model transformations (see *M2Ms* in the Figure), a few model to text transformations (see *M2Ts* in the figure) and a number of intermediate models, into a *Performance Analysis Tool Input.* Most transformations create a trace model (see *Trace Models* in the Figure) that enable visualization of performance analysis results based on the original process models.

In the remainder of the section, the transformation chain is explained in more detail by describing the different intermediate models of this chain. This description is divided into two parts. The first part, which deals with the abstraction

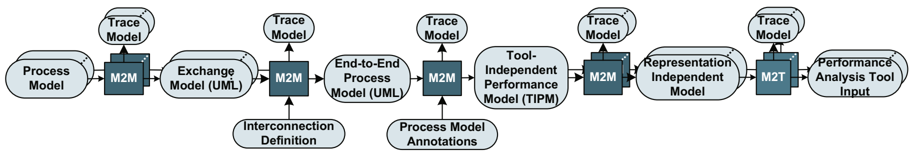
Trace Model
Process Model
M2M
Exchange Model (UML)
End-to-End Process Model (UML)
Interconnection Definition
Process Model Annotations
Tool-Independent Performance Model (TIPM)
Representation Independent Model
M2T
Performance Analysis Tool Input

Fig. 2. MDPE transformation chain as block diagram [18]

of Performance Analysis Tools, is explained in the following paragraph. Then, the abstraction of Performance Modelling Tools is provided as the second part.

3.1 Abstraction of Performance Analysis Tools (TIPM and Representation Independent Model)

An abstraction of Performance Analysis Tools is reflected by the so called Tool-Independent Performance Model (TIPM) which we defined in a joint work together with TU-Dresden, XJ-Technologies. The TIPM is related to the so called Core Scenario Model [19]. A simplified version of the TIPM meta-model is shown in Figure 3. In the following paragraph, the meta-model is explained in detail.

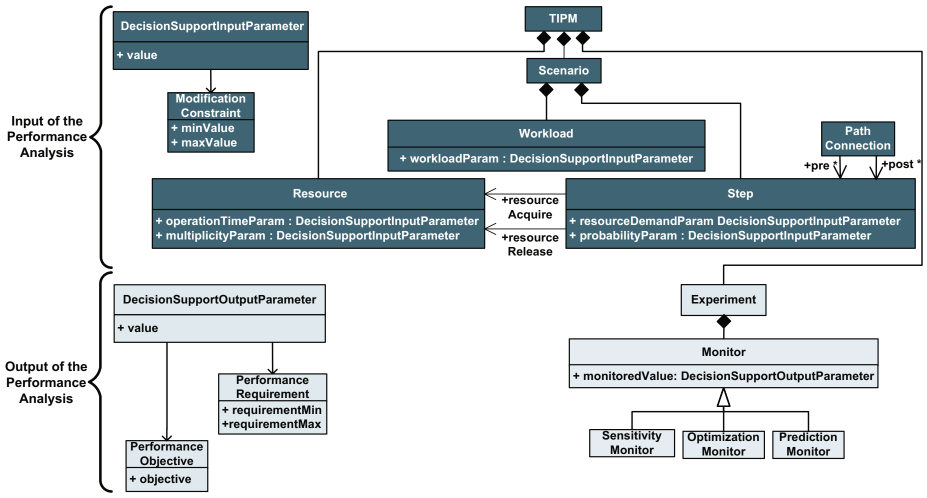
Input of the Performance Analysis
DecisionSupportInputParameter
+ value
Modification Constraint
+ minValue
+ maxValue
TIPM
Scenario
Workload
+ workloadParam : DecisionSupportInputParameter
Path Connection
+pre *
+post *
Resource
+ operationTimeParam : DecisionSupportInputParameter
+ multiplicityParam : DecisionSupportInputParameter
+resource Acquire
+resource Release
Step
+ resourceDemandParam DecisionSupportInputParameter
+ probabilityParam : DecisionSupportInputParameter
Output of the Performance Analysis
DecisionSupportOutputParameter
+ value
Performance Requirement
+ requirementMin
+requirementMax
Performance Objective
+ objective
Experiment
Monitor
+ monitoredValue: DecisionSupportOutputParameter
Sensitivity Monitor
Optimization Monitor
Prediction Monitor

Fig. 3. Tool-Independent Performance Model (TIPM) (simplified)

A TIPM can be distinguished into two parts. The first part contains “Experiments” which include a list of “Monitors”. These monitors are visualized with the light colour in Figure 3. Monitors are filled with values based on the performance analysis results, such as simulation results (see “PredictionMonitor” in the figure) which comprises latencies, utilizations and queue lengths. The latency specifies the time required in a simulation between two “Steps”. The queue length and utilization attributes are filled with the queuing behaviour of the simulated

"Resources". The monitored values are of type "DecisionSupportOutputParameter" which can be associated with "PerformanceRequirements" and "PerformanceObjectives". These meta-classes represent user provided knowledge, to further analyse the performance analysis results, for instance, for the assessment if a requirement will be met in the future. User provided knowledge is taken from the *Process Model Annotations* (see Figure 2) which are explained in a later part of this section.

The part of the TIPM meta-model which is shaded in dark grey (see Figure 3) has to be filled before a performance analysis can be executed by a Performance Analysis Engine. This part of the TIPM combines the behavioural information from Process Models with Performance Parameters. The behavioural information, consumed from an *End-to-End UML Model* (see Figure 2) which is explained in the next paragraph, is defined in the TIPM as a graph with "Steps" (nodes) and "PathConnections", which are used to interconnect the "Steps".

Performance parameters, which are also taken from the process model annotations, are represented in the TIPM as attributes of different meta-classes. For instance, the parameter multiplicity (see "multiplicityParam" attribute in the "Resource" meta-class) indicates how many units are available in a pool of resources, e.g. 10 employees in the Philadelphia sales office.

Performance parameters are also used in order to create "Scenarios" in the TIPM. An example of such a "Scenario" is the execution of the previously introduced business process of the Wine Seller for a certain sales unit (e.g. *Philadelphia* and *Chicago*) or a certain type of process instances (e.g. *instances of sales orders below 1000 Euro* and *instances of orders of 1000 Euro or above*). Resources can be shared among multiple "Scenarios" (see Figure 3), such as the case that 10 employees for marketing are shared between the sales unit in Philadelphia and in Chicago. In this specific case, the TIPM would contain the business process of Section 2 twice, but the marketing resources only once. Of course, "Scenarios" can also be used to simulate resource sharing situations between different business processes.

All performance parameters are of type "DecisionSupportInputParameter". Such values can have a reference to a "ModificationConstraint", again based on process model annotations, which define possible variations of a performance parameter, for instance, in order to restrict the solution space for the optimization of a parameter to answer questions like "How many employees are needed to meet processing targets and to optimize utilization of resources?".

In the case that a Performance Analysis tool is not TIPM based and its input data structure is different to the TIPM structure, a transformation between the TIPM and the tool input is required. For such transformations, two concerns can be separated:

– Structural concern: One is required to perform a structural transformation from the TIPM structure to the Performance Analysis Tool structure. It is, for instance, necessary to generate a structure of AnyLogic library objects if the simulation tool AnyLogic [4] is employed as performance Analysis Tool. This structure includes all AnyLogic objects and their connections. This step,

therefore, needs to generate all necessary objects together with additional objects required to connect everything into a working model.
- Representation concern: The Performance Analysis Tools use a specific concrete syntax, such as a concrete XML format, as input. For the purpose of serialization it is necessary to apply the formatting so that the generated data can be read by the Performance Analysis Tool.

Thus, a so-called *Representation Independent Performance Analysis Model* (see Figure 2) is needed as an intermediate model to keep the transformations between a TIPM and Performance Analysis Tools clean from serialization details. Due to this intermediate models, two separate M2M transformation are introduced, one for the structural and another for the representation concern.

The structural model-to-model (M2M) transformation translates from the abstract syntax of the TIPM to the representation independent model. The second model-to-text (M2T) transformation then translates the representation independent model into the *Performance Analysis Input*. For example, if a XML based representation is used, this input would reflect a structure of XML Attributes and XML Nodes. Fortunately, most modern Performance Analysis Tools use a XML language to represent their input. One can therefore normally benefit from the already available serialization functionality of most transformation tools, such as the ATL tooling, to serialize XML conformant text.

In the following subsection, the intermediate models to abstract Process Modelling Tools are described.

3.2 Abstraction of Process Modelling Tools (Exchange Model and End-to-End Process Model)

Transformations from Process Models and Performance Parameters to a TIPM are complex as the structure of the TIPM meta-model is most likely different from the meta-model structure of the process modelling languages. This is due to the fact that Process Models normally express Petri-net [13] like behaviour whereas the TIPM additionally expresses "Scenarios", "Resources", etc., and the related associations.

We, however, also experienced that the structures of the process modelling languages, such as BPMN, jPASS!, Process Flow, etc., are related as they can normally be translated into each other.

Therefore, it is beneficial to add an *Exchange Model* in the Model Transformation Chain (see Figure 2) in order to express process behaviour. We have not chosen Petri-nets itself due to the following reasons: A UML to TIPM transformation was available from the initial proof of concept phase of the MDPE related research described in [3]. At that time, UML had especially been chosen as it enabled us share models with external partners in public funded research project, such as the MODELPLEX project [20]. Additionally, one can apply formally defined Petri-net semantics [13] to UML Activity Diagrams, as discussed in more detail in [21]. Finally, UML Activity diagrams permit to not only support active systems but, in future versions of the MDPE Workbench, also reactive systems as explained below.

Eshuis [22] analysed the process modelling related concept of workflows, which are used to define "operational business process[es]". According to Eshuis, models which do not consider interactions with the environment, such as timing events in BPMN, are interpreted as closed and active systems, whereas models which support external events are interpreted as open and reactive systems. He also explains that Petri-nets especially well support the first type, whereas UML Activity Diagrams also support the open and reactive systems. In our current implementation we only consider closed systems, i.e. we don't support most of the BPMN event types. By employing UML Activity Diagrams as exchange language, we are, however, prepared for implementing the support for the missing events in the future.

Due to the UML Activity Diagrams that we use as Exchange Models, the required transformations to adapt a new Process Modelling Tool are less complex than generating a TIPM directly. This is due to the fact that an complex already available UML to TIPM transformation can be reused. This also helps to prevent investing significant duplicated effort in implementing the similar functionality of transforming from Petri-net like behaviour model to a TIPM.

Figure 2 shows that we employ UML Activity Diagrams not only as Exchange Models, but also for an *End-to-End Process Model.* This model simply merges the different sub-processes into a single end-to-end process. The M2M transformation which generates this end-to-end model takes, in addition to the UML models for the sub-processes, also an *Interconnection Definition* as input. This definition is needed to specify how the different UML models are connected, for example, via a map which relates the final nodes of one sub-process with the initial nodes of another sub-process.

3.3 Challenges Based on the Model Transformation Chain

It has been explained that a transformation chain is needed to interconnect the different tools.

However, means are still needed to represent the transformation chain in a systematic way, so that, for example, different adapters for Process Modelling Tools and Performance Analysis Tools can be activated and deactivated with little effort and the modular transformation can still be extracted.

Related to this, it is necessary to associate trace models of a transformation chain instance with the correct intermediate models of the chain.

In addition to that, the process model annotations have to be managed by the model transformation chains. This is caused by the fact that the performance parameters and the user provided knowledge in terms of performance objectives, requirements and constraints is annotated based on the process models at the beginning of the chain, but only taken as an input for the transformation which generates the TIPM. This can be, for instance, the third transformation step in the transformation chain. Thus, an approach is required which makes it possible to keep annotations independent from a concrete intermediate model in a model transformation chain. Hence, the pollution of the proposed intermediate models and model transformations with the annotation data should be avoided.

Concluding, an approach is required which allows representation of different kinds of models, such as trace models, annotation models, intermediate models and also model transformations[1]. Additionally, for the purpose of navigation between the different models, an approach to manage relationships between the different models, such as trace relationships or annotation relationships, is necessary. In the following section, a solution for this need is provided.

4 A Model Management Approach for MDPE

In this section, first a meta-model is described which permits to define models and relationships between these models. Second, an architecture is proposed which employs this meta-model to simplify administration of the MDPE transformation chain, tracing and model annotation.

4.1 Representation of Models and Model Relationships

In order to systematically deal with different kinds of models and the relationships among them, Bézivin's megamodelling approach [23] enables to define all these relationships in another model. Such a global model could be interpreted for model navigation tasks, such as for the navigation from an intermediate model to the related trace models. Bézivin's basic meta-model for such a global model is depicted by the upper part of Figure 4. This meta-model enables to express different kinds of "Relationships" among different types of "Models", such as "Terminal Models". Thus, a number of process models and other kinds of models of the transformation chain are terminal models that can be described using the basic megamodelling concepts.

The upper right part of Figure 4 shows other types of terminal models, which are considered by the megamodelling approach, such as *Transformation Models* and *Weaving Models.*

However, besides of transformation models, annotation models and tracing models are also used in the transformation chain (see Figure 2). Even if annotation models and tracing models are based on weaving models, a distinguishing between both types is necessary to simplify end-to-end tracing and transformation chain based annotation functionality.

The "ModelAnnotation" and "ModelTrace" relationships have been introduced to allow navigation among models related through annotation and traceability information. The last one is associated with a "TraceModel" which can be related to a specific target model through associations inherited-from "DirectedRelationship" and "ModelWeaving". In a similar way, "ModelAnnotation" is associated with an "AnnotationModel" and can be related to a specific target model. The "ModelTransformation" meta-class in Figure 4 represents a single transformation. A "ModelTransformation" can be specified in terms of a "TransformationModel", such as an ATL transformation.

[1] In this paper, we assume that transformation scripts conform-to meta-models, such as it is the case for ATL scripts.

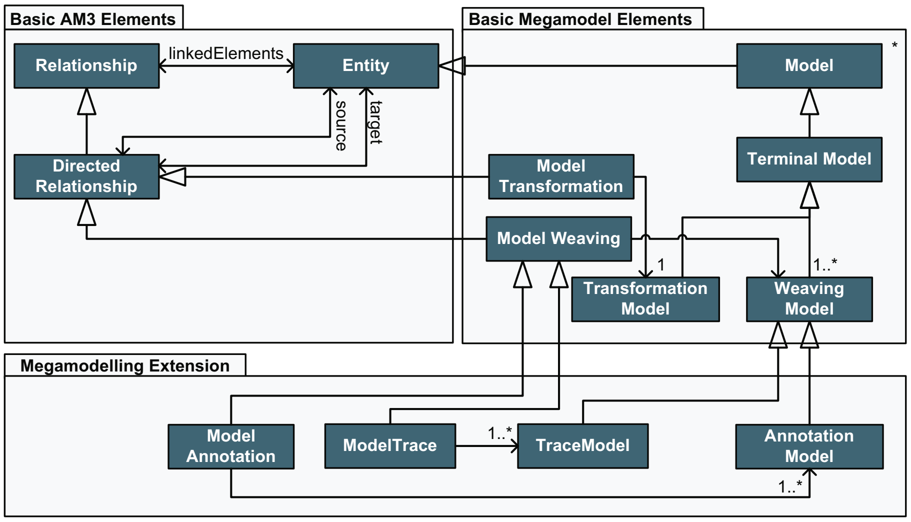

Fig. 4. Extract of the MDPE Metamodel extension of the Megamodel

The following section explains how the megamodel is employed within the MDPE Workbench [2,24,25].

4.2 Magamodelling Based Model Navigation

Based on the refined megamodel, the architecture of the *Model Navigation Agent* of the MDPE Workbench has been defined. This architecture is depicted in Figure 5.

The figure shows that, beside of other relationships, the *ModelTransformation (Relationship)*, *ModelTrace(Relationship)* and *ModelAnnotation(Relationship)* are represented in the *Refined Megamodel.* These relationships and the different *MDPE Modelling Artefacts* are used by the *Transformation Chain Controller*, *Tracing Controller* and *Annotation Controller* as depicted in the figure.

In the following paragraphs, the different controllers are explained in detail.

Transformation Controller: Figure 6 shows that the transformation relationship is set in the megamodel by an *Administration Tool.* The purpose of this tool is to enable users to select the currently active Process Modelling Tool and Performance Analysis Tool. The administration tool therefore stores the currently active Transformation Chain into the megamodel. This specification is consumed by a *Transformation Controller*, which executes the different *Transformation Models* and outputs the final Tool Input for Performance Analysis Tools based on a number of *Process Models.* The transformation outputs a number of *Trace Models* as by-products of the transformation [25]. The Transformation Controller also sets the *Tracing Relationships* between Process Models, Intermediate Models (e.g. the TIPM) and the final input for the Performance Analysis Tool into the megamodel.

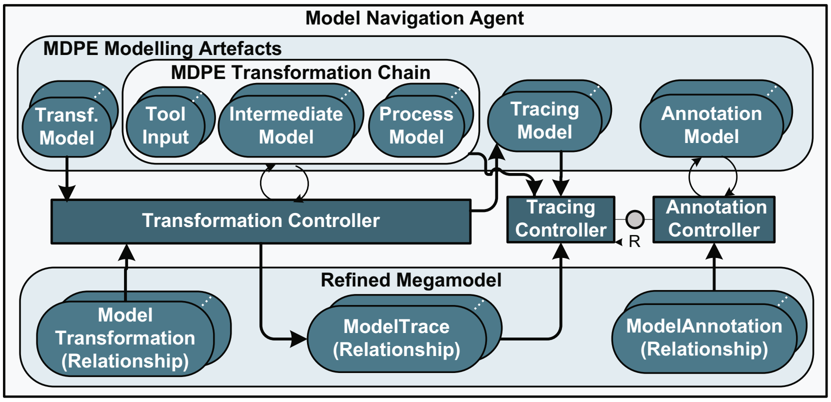

Fig. 5. Model Navigation Agent

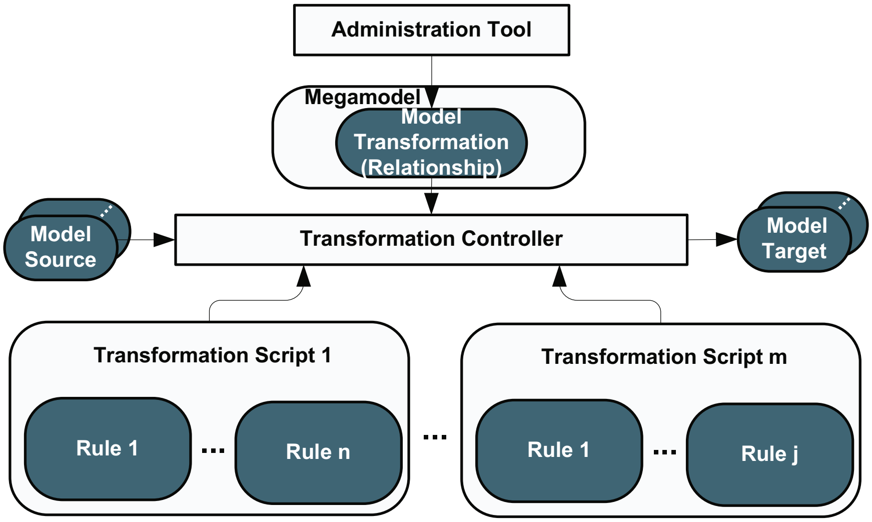

Fig. 6. Megamodel based Transformation Controller

Tracing Controller: The tracing relationships between trace models and models of the transformation chain are consumed by the *Tracing Controller*, which is depicted in Figure 7. Due to the *Model Trace* relationships in the megamodel, this agent is able to navigate backward through an instance of a model transformation chain containing i sets of intermediate, source and target models as shown in the figure. The agent is therefore also able to create an end-to-end trace model from an arbitrary Set_h of models to a Set_k of models. The resulting trace model is called the *End-to-End Trace Model* in the figure. The implementation of this end-to-end trace model generation has been done with the ATL transformation language.

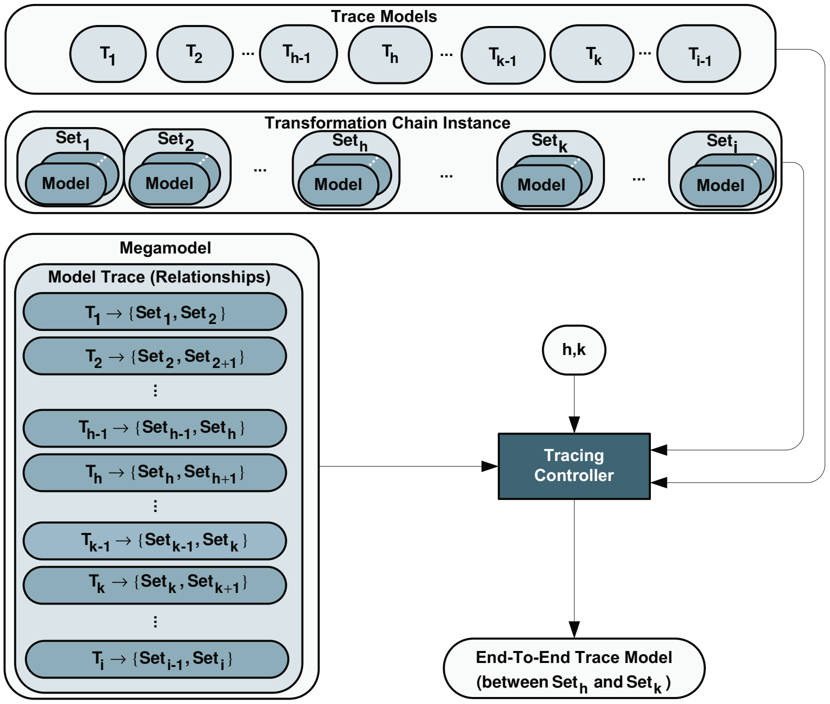

Fig. 7. Megamodel based Tracing Controller

However, some M2M languages, such as ATL, only allow transformation scripts where each single input model is explicitly named. Thus, lists of an infinite length of input models are not always supported. This is, however, beneficial for tools like the MDPE Workbench if one considers possible future extensions of the approach. Additionally, it is only required to activate the “UML2UML Transformation” if a process is spanning across multiple process environments. Thus, only some set-ups of the MDPE Workbench will contain this transformation. Therefore, an end-to-end trace model can be generated via recursively iterating over the on the traces from the different transformation steps, for instance, via a transformation which takes all traces and the megamodel as an input. The megamodel is needed because the transformation needs to know the currently active model transformation chain.

The resulting end-to-end trace model can be utilized for different purposes, for instance, to utilize a model annotation as input for an arbitrary transformation step in an automated model transformation chain (see “R” between Tracing Controller and Annotation Controller in Figure 5), as described in the following subsection.

Annotation Controller: Figure 8 gives an overview of the proposed *Annotation Controller*. This controller takes the end-to-end trace model from the tracing controller as input in order to create an *Annotation of Model in* Set_h based on *Data of Model in* Set_k.

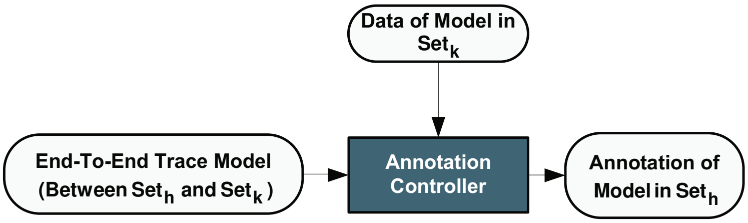
Data of Model in Set$_k$
End-To-End Trace Model (Between Set$_h$ and Set$_k$)
Annotation Controller
Annotation of Model in Set$_h$

Fig. 8. Megamodel based Annotation Controller

The approach can, therefore, be used to annotate content or annotations of an arbitrary model in the Set_k in an automated transformation chain, to another arbitrary model in the Set_h. Note that the Set_h is not necessarily created in the chain before the Set_k.

The combination of the tracing controller and the annotation controller is, however, only beneficial if the following restriction is considered:

In some cases a model in the Set_k cannot be related to a model in the Set_h. This for example happens if not all meta-classes of a model in the Set_k are translated into models of Set_h. In this case it is obviously not possible to relate these elements of the Set_k to elements of the Set_h, which have not been translated. However, such cases are considered as an indication of either an invalid model annotation or an invalid model transformation chain. Thus, the domain specialist needs to be informed by the tooling in such cases.

5 Experiences

For the evaluation of the transformation chain based approach, we had to set up the model transformation chain as shown in Figure 9. This transformation chain takes the three different kinds of process modelling languages as input, which constitute the end-to-end process described in Section 2. All three modelling languages are first transformed to UML (see *BPMN2UML*, *ProcessFlow2UML*, *JPASS2UML* in Figure 9). In a second step, the three UML models are interconnected to formulate an end-to-end UML model (see *UML2TIPM* in Figure 9), which is then, together with the Performance Parameters, transformed to a TIPM.

Three different types of performance related decision support are provided. The AnyLogic tool is used as a discrete event simulation engine and as an optimization engine. Please note that the required transformations are different for both cases as different kinds of experiments have to be created in the AnyLogic tool. Moreover, the simulation based decision support only takes performance requirements into account as Performance Assessment Data, whereas the optimization also considers objectives and constraints. Additionally, we attached the FMC-QE tool in order to get analytic support for sensitivity related decisions. Therefore, the TIPM, which contains the end-to-end business process including its resource related behaviour, is transformed via a Representation Independent Model (e.g. the *AL_SIM Model* in Figure 9) into Tool Input (e.g. the *AL_SIM.xml* in Figure 9).

In the following paragraphs, we describe our experiences:

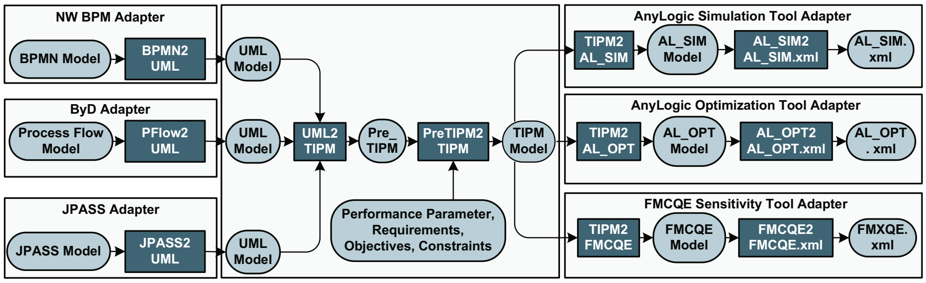

Fig. 9. Example Transformation Chain Set-Up

From the functional point of view, the combination of TIPM and UML as intermediate languages enables to abstract the different process modelling languages. Without the proposed transformation chain, the creation of end-to-end simulation models would have required to manually switch between three different Process Modelling Tools and three different Performance Analysis Engines. Therefore, in case one wants to get end-to-end decision support for the use case that we introduced, nine different tools need to be understood. In general, for the end-to-end decision support spanning across n process environments and m Performance Analysis Tools, $n+m$ tools have to be understood by a domain specialist. In [26] metrics are provided to measure the complexity of using a tool. One of the measures is the number of manual steps; another one is the number of context switches. Obviously, our solution reduces the number of manual steps to a “push-button” activity and one is not required to switch between numerous tools involved.

Due to UML being employed as an Exchange Model, we only had to write the complex part of the transformation between process modelling languages and TIPM once. Thus, as long as a new process modelling language can be mapped to UML Activity Diagrams, those effectively reduce the development effort associated with attaching different Process Modelling Tools. If a process modelling language cannot be mapped to UML Activity Diagrams due to a currently unforeseen process modelling concept, one might still be able to map it, with more effort, to the TIPM. If this is also not possible, significant development effort might be required to modify the TIPM and, in worst case, all transformations from and to the TIPM. This is, however, considered as not very likely.

Additionally, the TIPM as Generic Performance Analysis Model also enabled to provide decision support based on multiple Performance Analysis Engines, such as the analytic FMC-QE tool and the simulation engine AnyLogic. In case a new Process Modelling Tool is attached to the MDPE Workbench, the existing transformations between TIPM and Performance Analysis Tools can be reused. Again, if the TIPM needs to be modified for a new Performance Analysis Tool due to a currently unforeseen performance analysis concept, significant effort might be needed, in particular if the new TIPM version induces the need to update existing transformations. This is, however, considered as not likely.

Moreover, the Transformation Chain Management permits to manage the different modelling artefacts, such as annotation models and process models, across the transformation chain. This enables one to use, for instance, annotation models that reference the process models at any step in the transformation chain, and by tracing Assessment Results backwards through the chain in order to visualize them, also as annotation models, based on the original process models.

Adding a new step to the Transformation Chain is, therefore, not an issue anymore. This was, for instance, needed when we added the “UML2UML” transformation step (see Figure 9). The additional transformation step was one prerequisite in order to realize end-to-end decision support which spans the tool chain built by the Process Flow modelling tool, the Process Composer and the jPASS! tool.

However, the performance parameters are currently specified manually, which is not sufficient in cases where historic performance parameters, such as how long a special process step took in the past, are available as process instance data. Therefore, in order to fully support end-to-end decision support spanning across the three BPM tools, it is not only required to abstract three different modelling languages with a model transformation chain, but also to abstract three different sources for such historic process instance data.

Additionally, we are lacking a mechanism to deal with the confidentiality of the business sub-processes. It might, for instance, not be appropriate for the third party of our use case to provide their business process description to the wine seller company. Thus, solutions are required to ensure confidentiality, for instance, by preventing that the employees of the wine company have visibility to the invoicing process.

6 Related Work

Some model-driven scenarios which involve chains of multiple transformations can be found in the literature, such as the twelve-step transformation chain in the interoperability scenario for business rules presented in [27] or the five step chain in the interoperability scenario for code clone tools presented in [28]. Compared to these works, reasoning for the transformation chain of the MDPE Workbench has been provided to integrate chains of Process Modelling Tools and Performance Analysis Tools. Additionally, an architecture for model management has been provided which enables end-to-end tracing, annotation and administration of automated model transformation chains of arbitrary lengths.

The main concepts behind the concept of intermediate languages that we have employed for our transformation chain can also be found in other software transformations. For example, the abstraction of Performance Analysis Tools and Process Modelling Tools with performance specific language TIPM and the behaviour specific language UML Activity Diagrams is related to formats like PDG or SSA [29,30]. The combination of UML Activity Diagrams and the TIPM can be seen as means for a source- or target-independent intermediate (pivot) format to reduce the number of required transformations. This is similar to the use of byte code to abstract from the processor architecture.

From the application point of view, the closest related work to our knowledge is the integrated performance simulation functionality within some process environments of the BPM domain, such as EMC Documentum Process Suite [31]. However, the offered simulation-based functionality is only at a basic level [32]. To the best of our knowledge, the proposed approach is the only one which provides, based on a model transformation chain and an architecture for model management, support for chains of process environments, each employing different Process Modelling Tools. Our approach further enables to benefit from the know-how and functionality contained in a sophisticated performance decision support system, which makes use of sophisticated model simulations, optimizations, and static analyses, and also a combination of them.

7 Conclusion

In this paper, the combination of a model transformation chain and a model management architecture has been proposed in order to provide end-to-end performance related decision support for business processes spanning multiple Process Modelling Tools. The proposed architecture enables to interconnect multiple Process Modelling Tools with multiple Performance Analysis Engines.

As a next step we anticipate to extend the current model transformation chain in order to provide functionality for the abstraction of history data logs which are provided by process environments. This would allow a better integration of the MDPE Workbench into these environments. Additionally, we plan to extend the existing model management approach to permit dealing with the confidentiality of the business sub-processes.

Disclaimer

The information in this document/This software is proprietary to SAP. No part of this document/software may be reproduced, copied, or transmitted in any form or for any purpose without the express prior written permission of SAP AG.

This document/software is a preliminary version and was created only for research purposes. This document/software contains only intended strategies, developments, and functionalities of the SAP product and is not intended to be binding upon SAP to any particular course of business, product strategy, and/or development. Please note that this document/software is subject to change and may be changed by SAP at any time without notice.

SAP assumes no responsibility for errors or omissions in this document/software.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material. This document/software is provided without a warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. This limitation shall not apply in cases of intent or gross negligence.

The statutory liability for personal injury and defective products is not affected. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third-party Web pages nor provide any warranty whatsoever relating to third-party Web pages.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves information purposes only.

References

1. Fritzsche, M., Picht, M., Gilani, W., Spence, I., Brown, J., Kilpatrick, P.: Extending BPM Environments of your choice with Performance related Decision Support. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 97–112. Springer, Heidelberg (2009)
2. Fritzsche, M., Johannes, J., Assmann, U., Mitschke, S., Gilani, W., Spence, I., Brown, J., Kilpatrick, P.: Systematic usage of embedded modelling languages in automated model transformation chains. In: Gašević, D., Lämmel, R., Van Wyk, E. (eds.) SLE 2008. LNCS, vol. 5452, pp. 134–150. Springer, Heidelberg (2009)

3. Fritzsche, M., Johannes, J.: Putting Performance Engineering into Model-Driven Engineering: Model-Driven Performance Engineering. In: Giese, H. (ed.) MODELS 2007. LNCS, vol. 5002, pp. 164–175. Springer, Heidelberg (2008)
4. XJ Technologies: AnyLogic — multi-paradigm simulation software (2009), `http://www.xjtek.com/anylogic/`
5. Franks, R.G.: PhD Thesis: Performance Analysis of Distributed Server Systems. Carlton University (1999)
6. Zorn, W.: FMC-QE: A new approach in quantitative modeling. In: Proceedings of the 2007 International Conference on Modeling, Simulation & Visualization Methods (MSV 2007), pp. 280–287. CSREA Press (2007)
7. Porzucek, T., Kluth, S., Fritzsche, M., Redlich, D.: Combination of a discrete event simulation and an analytical performance analysis through model-transformations. In: Proceedings of the International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2010), pp. 183–192. IEEE Computer Society, Los Alamitos (2010)
8. Rogers, P.: Optimum-seeking simulation in the design and control of manufacturing systems: Experience with optquest for arena. In: Proceedings of the 2002 Winter Simulation Conference, WSC 2002 (2002)
9. Object Management Group: Business Process Modeling Notation Specification, Final Adopted Specification, Version 1.0 (2006), `http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf`
10. jCOM1 AG: jpass! - subjektorientierte prozessmodellierung (2009), `http://www.jcom1.com/cms/jpass.html`
11. SAP AG: Sap solutions for small businesses and midsize companies (2009), `http://www.sap.com/solutions/sme/businessbydesign/index.epx`
12. SAP AG: Sap business suite - integrated enterprise applications help lower costs, improve insight, and capture opportunities (2009), `http://www.sap.com/solutions/business-suite/index.epx`
13. Peterson, J.L.: Petri Net Theory and the Modelling of Systems. Prentice-Hall, Englewood Cliffs (1981)
14. jCOM1 AG: Process management - jcom1 (2009), `http://www.jcom1.com/`
15. Fleischmann, A.: Distributed Systems, Software Design & Implementation (1995)
16. Fritzsche, M., Gilani, W., Fritzsche, C., Spence, I., Kilpatrick, P., Brown, T.J.: Towards utilizing model-driven engineering of composite applications for business performance analysis. In: Schieferdecker, I., Hartman, A. (eds.) ECMDA-FA 2008. LNCS, vol. 5095, pp. 369–380. Springer, Heidelberg (2008)
17. SAP AG: Components & tools of sap netweaver (2009), `http://www.sap.com/platform/netweaver/components/sapnetweaverbpm/index.epx`
18. Knöpfel, A., Gröne, B., Tabeling, P.: Fundamental Modeling Concepts: Effective Communication of IT Systems. John Wiley & Sons, Chichester (2006)
19. amd Murray Woodside, D.B.P.: An intermediate metamodel with scenarios and resources for generating performance models from uml designs. Software and Systems Modeling 6(2), 163–184 (2007)
20. Information Society Technologies: Sixth Framework Programme, Description of Work: MODELling solution for comPLEX software systems (MODELPLEX) (2006)
21. Dehnert, J.: PhD Thesis: A Methodology for Workflow Modeling: From business process modeling towards sound workflow specification. TU-Berlin (2003)
22. Eshuis, R.: PhD Thesis: Semantics and Verification of UML Activity Diagrams for Workflow Modelling. Centre for Telematics and Information Technology (CTIT), University of Twente (2002)

23. Bézivin, J., Jouault, F., Rosenthal, P., Valduriez, P.: Modeling in the large and modeling in the small. In: Aßmann, U., Liu, Y., Rensink, A. (eds.) MDAFA 2003. LNCS, vol. 3599, pp. 33–46. Springer, Heidelberg (2005)
24. Fritzsche, M., Bruneliere, H., Vanhoof, B., Berbers, Y., Jouault, F., Gilani, W.: Applying megamodelling to model driven performance engineering. In: Proceedings of the International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2009), pp. 244–253. IEEE Computer Society, Los Alamitos (2009)
25. Fritzsche, M., Johannes, J., Zschaler, S., Zherebtsov, A., Terekhov, A.: Application of tracing techniques in model-driven performance engineering. In: Proceedings of the 4th ECMDA Traceability Workshop (ECMDA-TW), pp. 111–120 (2008)
26. Keller, A., Brown, A.B., Hellerstein, J.L.: A configuration complexity model and its application to a change management system. IEEE Computer Society Transactions 4, 13–27 (2007)
27. Fabro, M.D.D., Albert, P., Bzivin, J., Jouault, F.: Industrial-strength rule interoperability using model driven engineering. In: Proceedings of the 5mes Journes sur l'Inginierie Dirige par les Modles (2009) (to appear)
28. Sun, Y., Demirezen, Z., Jouault, F., Tairas, R., Gray, J.: A model engineering approach to tool interoperability. In: Gašević, D., Lämmel, R., Van Wyk, E. (eds.) SLE 2008. LNCS, vol. 5452, pp. 178–187. Springer, Heidelberg (2009)
29. Ferrante, J., Ottenstein, K., Warren, J.: The program dependence graph and its use in optimization. ACM Transactions on Programming Languages and Systems (TOPLAS) 9, 319–349 (1987)
30. Cytron, R., Ferrante, J., Rosen, B., Wegman, M., Zadeck, F.: Efficiently computing static single assignment form and the control dependence graph. ACM Transactions on Programming Languages and Systems (TOPLAS) 13, 451–490 (1991)
31. Associates, B.S.: The BPMS Report: EMC Documentum Process Suite 6.0 (2008), http://www.bpminstitute.org/whitepapers/whitepaper/article/emc-documentum-process-suite-6-0-1/news-browse/1.html
32. Harmon, P., Wolf, C.: The state of business process management (2008), http://www.bptrends.com/surveys_landing.cfm