Towards Performance Related Decision Support for Model Driven Engineering of Enterprise SOA applications

Mathias Fritzsche, Wasif Gilani
SAP Research CEC Belfast
[mathias.fritzsche | wasif.gilani]@sap.com

Ivor Spence, T. John Brown, Peter Kilpatrick, Rabih Bashroush
Queen's University Belfast
[i.spence | tj.brown | p.kilpatrick | r.bashroush]@qub.ac.uk

Abstract

Model Driven Performance Engineering (MDPE) enables early performance feedback in a MDE process, in order to avoid late identification of performance problems which could cause significant additional development costs. In our past work we argued that a synchronization mechanism between development and performance analysis models is required to adequately integrate analysis results into the development process enabling performance related decision support. In this paper we present a solution for this requirement. We present a new multi-view based approach and its implementation enabling systematic performance related decision support. We currently apply our research on the model driven engineering of process orchestrations on top of SAP's Enterprise Service Oriented Architecture (Enterprise SOA).

1. Introduction

Increasingly complexity of software systems, characterized here in terms of attributes such as size, distribution, heterogeneity and dynamicity, create a high need for an early identification of possible performance problems in order to avoid significant additional development effort. We deal with providing a solution for addressing the performance related issues in the earlier stages of software development, and applying our work to highly distributed applications built on top of SAP's SOA platform called Enterprise SOA [13], [14].

In our previous work [1], we proposed Model-Driven Performance Engineering (MDPE) for early performance feedback. The process supports earlier initial performance feedback with minimal effort as well as maximal performance feedback with extended (but still cost-efficient) effort by utilization of Model Driven Engineering (MDE) concepts. Hence, MDPE enables earlier performance feedback to address the challenges of short time to market by taking into account the increased complexity in software development.

We identified the requirement of a synchronization mechanism, between the development models[1] and performance analysis models, in order to adequately integrate the performance analysis results into the development process. This requirement is extended here with the notion of providing performance related decision support based on analysed performance view models.

An example of a performance view model is the Core Scenario Model (CSM) proposed in [2], which combines performance relevant model knowledge and performance measurements of a usage scenario. This information has still to be interpreted, as mentioned in [3]: *"We must […] learn how to combine measurement data interpretation with model interpretation and to get the most out of both*". A first step towards this kind of interpretation is taken in [4], in which a metric is introduced for the detection of bottleneck sources for decision support, in order to apply improvements and realistically estimate their effectiveness. The decision support in that work is based on a metric called Bottleneck Strength providing a first step towards combining measurement interpretation and model interpretation.

[1] We use the term development model in this paper to distinguish between models as development artefacts and performance view models.

978-0-7695-3141-0/08 $25.00 © 2008 IEEE
DOI 10.1109/ECBS.2008.52

The contribution of this work is to propose an approach enabling systematic performance related decision support for non-performance experts in terms of what in a design and in a resource mapping has to be changed to get better results with regard to performance objectives and modification constraints.

The approach combines three different performance related views enabling effective performance assessments. The approach is presented as an extension of our previous idea of MDPE by providing stepwise performance assessment and is described in section 3. We implemented the approach by utilizing an Eclipse based implementation of a systematic model annotation approach (see section 4) and currently apply our research for the MDE of process orchestrations (see section 2).

2. Application

The requirements for our approach are motivated by the Enterprise SOA architecture [13], [14]. A simplified view of this architecture is depicted in Figure 1.

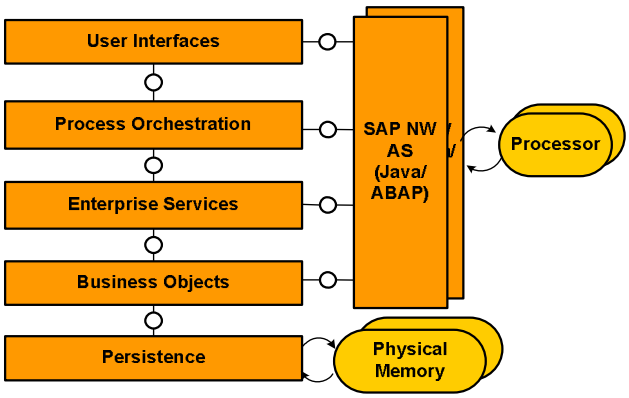

Figure 1: Enterprise SOA high-level architecture as Block Diagram [10]

As can be seen in the figure, the architecture is structured in layers accessible as software resources. The functionality provided by the different layers can be deployed in one or several instances of the *SAP NetWeaver Application Server* which are running on physical resources which are *Processors*.

The Persistence layer uses distributed data repositories that may consist of multiple databases using physical memories.

Business Objects on top of the persistence layer encapsulate semantic data, such as Sales Order data, and provide methods to manipulate them. Business Objects enable Business Processes and provide one or more *Enterprise Services* which are technically implemented as WebServices extended with proprietary features [14]. Enterprise Services can be provided not only by SAP specific Business Objects but also by 3rd party objects.

The Process Orchestration Layer defines the business control logic. It is the role of Enterprise Services to provide access to business specific data or functionality that can be used to compose business processes. In the current architecture two kinds of process orchestrations are possible depending on the lifecycle of the orchestrated process. Back-end process orchestration is done to define processes with longer lifecycles whereas front-end orchestration is done to compose processes with shorter lifecycles. In our current work we deal with the model driven engineering of processes with minimum user interaction.

Following models specify the orchestrated processes and the underlying architecture:

- Models of orchestrated front-end processes
- Models of underlying back-end processes

Models and measured performance data of building blocks of a system enables performance analysis at design time conforming to the original MDPE process, [1] or other processes utilizing MDE for performance engineering such as [15] and [16]. Alternatively, performance analysis at runtime can be performed by measuring indices of a system, such as utilization of resources.

We identified the problem that it is difficult to deal with the interpretation of performance analysis results for orchestrated processes on top of the complex Enterprise SOA architecture. One reason for that is the layered architecture consisting of the Persistence, Business Objects, Enterprise Service, Process Orchestration and User Interface Layer, where a bottleneck in one layer may in fact result in a bottleneck in another layer by push-back which makes interpretation difficult [4]. Additionally, the high degree of flexibility for deploying the system on physical resources and the integration of 3rd party services complicates performance analysis. Hence, an approach is required to enable interpretation of performance analysis results.

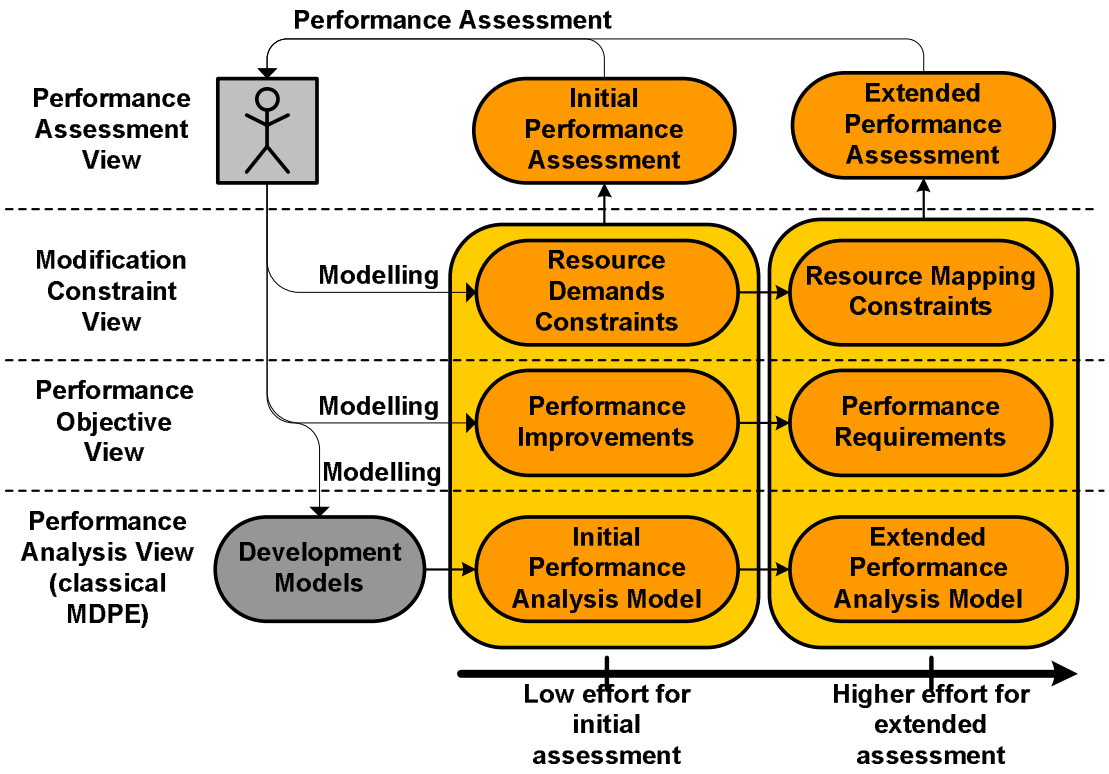

Figure 2: Multiple views

The approach has to be in some sense intelligent to adequately integrate the analysis results into development models, such as models of orchestrated business processes on top of Enterprise SOA. Therefore, the approach should provide decision support for non-performance experts in terms of what in a design and in a resource mapping has to be changed to get better results with regard to performance objectives. We have refined this requirement in terms of the following issues addressed in this paper:

- **Information filtering**: We should only provide relevant information with respect to the modification constraints and performance objectives provided by users of our approach. In this paper we define performance objectives as performance requirements and performance improvements. Performance improvements are concerned with maximizing the resource utilization and mimizing the response time of the modeled system.
- **Information interpretation**: We are required to provide help in interpreting measurement data, performance models, and performance prediction results related to performance objectives and modification constraints by providing intelligent performance related metrics delineating how the performance can be improved.
- **Systematic model synchronization:** We should provide an approach for systematic integration of performance metrics into development models in the MDE process.
- **Assessment visualization:** The solution should enable visualization support for graphical representation of identified performance metrics on development models.

3. Proposed multi-view based approach

We propose utilization of different views for calculating metrics of interest to the user. Figure 2 depicts all views considered in our approach.

A description of the semantics of the different performance related views is given below.

3.1 Performance Analysis View

The *Performance Analysis View* is a viewpoint on the system encapsulating performance-related characteristics and execution parameters of a system. Hence, the Performance Analysis View is used to *calculate* the metrics providing performance related decision support. Based on the stepwise MDPE approach we consider the *Initial Performance Analysis Model* and the *Extended Performance Analysis Model*. The former one is based on development models

annotated with resource demands and probabilities of paths. It enables initial performance feedback in terms of upper and lower bounds in the absence of factors due to contention of resources. In order to give performance related decision support, we use the Initial Performance Analysis Model as one input for *Initial Performance Assessment*.

The Extended Performance Analysis Model requires more detailed information and hence more effort by the modeller. In more detail, it additionally takes into account factors due to contention for resources, enabling more detailed scenario specific analysis. We consider here that the Extended Performance Analysis contains information about utilization of resources based on measurements at runtime or, conforming to the original MDPE approach, based on performance prediction techniques at design time. The Extended Performance Analysis Model is used as one input for *Extended Performance Assessment*.

3.2 Performance Analysis View

The *Modification Constraint View* specifies the configuration options with respect to possible resource mappings and in the future also with respect to design alternatives. This view enables decision support which is realizable and hence useful. We currently employ *Resource Demand Constraints* as input for Initial Performance Assessment and *Resource Mapping Constraints* as input for Extended Performance Assessment. With Resource Demand Constraints we currently consider resource demands as fixed, e.g. for the specification of resource demands of third party services, or as variable. Resource mapping constraints consider resources as duplicable or single-only resources.

The Modification Constraint View is used to *filter* the resulting performance assessment view for user needs.

3.3 Performance Objective View

The *Performance Objective View* concerns how the modeled system should perform. This view can be split into the specification of *Performance Requirements* and *Performance Improvement*. Performance Improvements are currently concerned with maximizing the resource utilization and mimizing the response time of the modeled system. We consider specifying the Performance Improvements in the first step as an input for Initial Performance Assessment. Specifications of Performance Requirements, which are specific for factors due to contention of resources, are considered as an input for Extended Performance Assessment.

The Performance Objective View is also used to *filter* the resulting performance assessment view for user needs. In the current implementation we only support one metric for Initial Performance Assessment and one metric for Extended Performance Assessment. In the future we anticipate using the Performance Objective View additionally to *compute* metrics of interest by calculating dependencies between the performance objectives and design decisions within the development models. Those dependencies should either be directly visualized to a user as a metric or used to calculate how the optimal configuration with respect to design and resource mapping alternatives should look like by taking performance objectives and modification constraints into account.

3.4 Performance Assessment View

We claim that the combination of the former views enables calculation of performance related metrics and patterns, thereby enabling decision support by automatically taking performance objectives and modification constraints into account. Hence, the approach enables the automatic generation of a *Performance Assessment View* from the information provided by other views.

The Performance Assessment View provides performance related decision support for non-performance experts in terms of what in a design and in a resource mapping has to be changed to get better results with regard to performance objectives and modification constraints. It therefore provides help in information interpretation and filtering as stated in section 1. Conforming to the stepwise MDPE approach we consider Initial Performance Assessment which provides performance related decision support in the absence of concrete usage scenarios including information about factors due to contention of resources. In the second step we consider Extended Performance Assessment taking additional factors due to contention of resources, resource related requirements and resource related constraints into account. For both steps of Performance Assessment we calculate metrics from the other three views to provide decision support.

In order to gain first hand experience with our approach we selected one metric per assessment step: Step Performance Importance (SPI) as Initial Performance Assessment and Bottleneck Strength as defined in [12] as Extended Performance Assessment.

- The SPI metric depicts the impact of processing time changes of a process step. It therefore depicts

the importance of decreasing resource demands of a process step or increasing of resource quality or quantity on the overall performance. SPI is calculated for each step in a behaviour model from the probabilities of paths available in the Initial Performance Analysis Model as follows:

S = *Step in a behavior model*

n_s = *number of possible paths to Step S*

p_i = *probability of path i*

$$SPI_s = \sum_{i=0}^{n_s} p_i$$

A high SPI value of a step in a behavior model indicates a high impact on the overall performance if the resource demand of the process step is decreased or the resource quality or quantity is increased. The results are shown only for these steps which are defined as *variable* in the Modification Constraint view and which are marked as Improvable in the Performance Objective View.

- Bottleneck Strength can be calculated if models showing layered use of resources, such as shown in section 2, are available and if they are containing information about resource utilization. For our current implementation we use a refined and slightly extended version of the Core Scenario Model (CSM) [2] to have an instance of an Extended Performance Analysis Model. The Bottleneck Strength (BStrength) metric is defined in [12]:

R = *hardware or software resource of a Step*

r = *another resource which is requested by R*

$$Shadow(R) = \arg\max_{r requested by(R)} utilizatio n_r$$

$$BStrength_R = \frac{utilizatio n_R}{utilizatio n_{Shadow(R)}}$$

BStrength enables bottleneck characterization for layered resource consumption. The resource with the largest BStrength value and utilization close to 100% is interpreted as the bottleneck. A more detailed description is provided in [12] about how this metric has to be interpreted to support design and resource mapping decisions. Currently we use the Modification Constraint View and the Performance Objective View to filter the visualization of BStrength values. Consistent with the SPI metric we only visualize BStrength for those parts in a model that are not fulfilling Performance Requirements defined within the Performance Objectives or which are marked as Improvable in the Performance Improvements and where the Resource mapping Constraint does not prevent the use of more resources. We identified that both selected metrics are a first step towards performance related decision support for non-performance experts in terms of what in a design and in a resource mapping has to be changed to get better results with regard to performance objectives and modification constraints. Anyways, we additionally identified the need to delineate dependencies between performance objectives and design/resource mapping alternatives.

To summarize, the Performance Analysis view is mainly used to fulfill the requirement of *information interpretation* because it is mainly about interpreting measurement data, performance models, and performance prediction results. In the future the Performance Objective View will target this requirement as well. The requirement of *information filtering* is currently mainly fulfilled based on the Performance Objective and Modification Constraint View. *Systematic model synchronization* and *assessment visualization* is fulfilled by the profile based model annotation approach which is described in the following section. This section introduces the architecture of our approach that enables systematic performance related decision support for non-performance experts in terms of what in a design and resource mapping has to be changed to get better results with regard to performance objectives.

4. Proposed implementation

To gain initial experience with our approach we implemented an extension of MDPE for performance related decision support by utilizing a systematic model annotation approach. The following subsections give an overview of the proposed architecture.

4.1 Overall architecture

Figure 3 depicts the high-level architecture of the proposed approach. The three performance assessment related views (Performance Objectives, Modification Constraints and Performance Analysis) are integrated by a *Composition Engine* to an *Assessment Computation Model* which is used internally within *Decision Support Engine*. This model is technically a composite model of the original *Development Models* and the performance assessment related views. The

Composition Engine and Assessment Computation Engine are described in more detail in subsection 0. The internally used Assessment Computation Model is used as input for the *Assessment Computation Engine* calculating the *Performance Assessment View*.

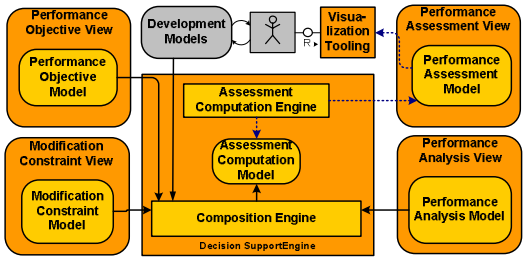

Figure 3: High-level Architecture of our approach as Block Diagram [10]

The user of the performance related decision support uses the *Visualization Tooling* to access the Performance Assessment View and to determine design and resource mapping decisions for the original development models. We anticipate utilizing the model metric visualization tooling described in [19] to visualize performance related metrics out of our current and future work to realize user centric design decision support based on the original development models.

Figure 4 depicts the information flow from specification of different views (1), composing them to an internally used Assessment Computation Model (2), compute customized performance metrics (3) and visualize them for users of the approach (4).

The following subsection delineates the concrete models we use to specify view points.

4.2 Currently used view-point models

For our initial implementation we support UML2.0 models as development models due to the available tool support. In our current example we used UML Activity Diagrams modeling Process Orchestrations on top of Enterprise Services as introduced in section 2, and Deployment Diagrams. Both types of models are annotated with performance data conforming to the UML SPT profile [17].

In order to obtain an Initial Performance Analysis View, we annotated the UML Activity Diagram with resource demands of Actions and probabilities of paths.

Following this, we added information concerning contention of resources to the Activity Diagram and the Deployment Diagram to transform them via ATL transformation [11] to the Extended Performance Analysis Model. In more detail, we generated stepwise two kinds of Extended Performance Analysis Model: A Tool Independent Performance Model (TIPM) and a Tool Specific Performance Model (TSPM) as described in [1]. The TIPM is defined as a refined and slightly extended version of the Core Scenario Model (CSM) [2]. The TSPM has been used as input for the simulation tool AnyLogic [12]. In the future we anticipate generating input for other simulation tools as well to get a broader set and therefore more useful simulation results [1]. The resulting information from the simulation about utilization of resources has been annotated back to the TIPM which has been then used as input to the Extended Performance Assessment View.

In order to specify the Modification Constraints View and Performance Objective View we defined initial UML profiles.

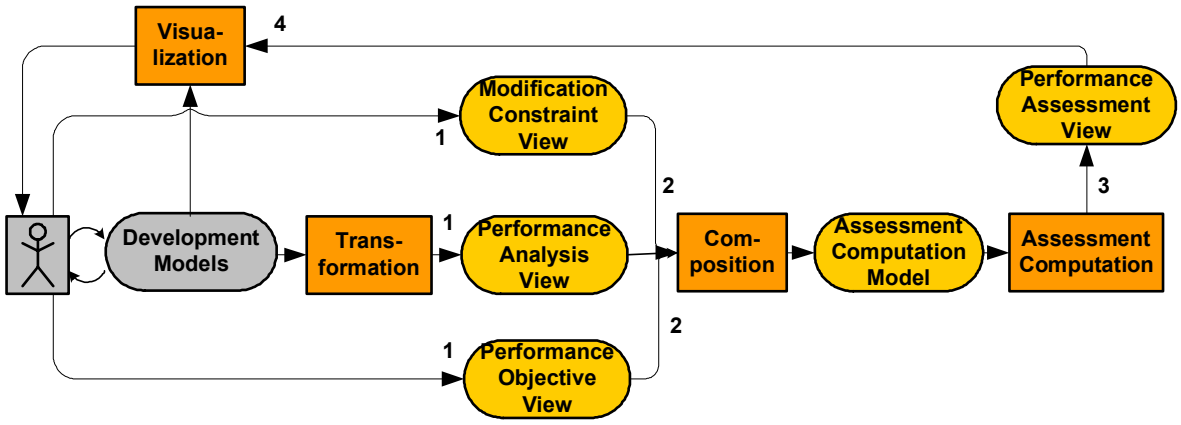

Figure 4: Information Flow as Block Diagram [10]

The Modification Constraints Profile consists of two stereotypes which can be attached on *UML.ExecutableNodes*: *ResourceDemandsConstraint* which is defined by the *value* that can be *fixed* or *changeable* and *ResourceMappingConstraint* which is defined by the integer values *minMultiplicity* and *maxMultiplicity*. In the future we anticipate extending this profile to be more expressive.

In order to express Performance Requirements as part of the Performance Objectives we could have used the UML SPT profile [17] since it is possible to express all performance values as *required*. Since we do not need the full expressiveness of SPT, and would like to merge the view points of Performance Objectives and the Performance Analysis into one profile and also to express Performance Improvements, we defined a UML Profile containing the Stereotypes *ExecutionTimeRequirement* and *ResourceRequirement* which can be applied on *UML.ExecutableNodes*. *ExecutionTimeRequirement* specifies the overall time to execute an ExecutableNode. The stereotype is specified by its *maximumExecutionTime* and the Boolean value *Improvable*, which specifies if the value should still be reduced if the *maximumExecutionTime* criterion has been reached. The stereotype *ResourceRequirement* specifies the range of resource utilization to be achieved (*maxUtilization* and *minUtilization*). To express future improvements the Boolean values *improveTowardsMaxUtilization* and *improveTowardsMinExecutionTime* are needed.

4.3 Systematic model annotation

We make extensive use of UML profiles, which are a second-class extension mechanism [5] for UML models. In general, modeling of view-points with UML profiles weakens the separation of concerns principle significantly as shown by [6]. In [7] we argue that the manual application of UML profiles for large models is a time consuming and error prone process. Additionally, it does not adhere to the separation of concerns principle in order to manage complexity by treating each concern in its own space; see also [8]. We apply model modification constraints and performance objectives to development models by specifying them in our Query and Annotation Language (QUAL) [7]. The language enables us to specify model extensions centrally. An Eclipse based infrastructure enables us to perform UML annotations specified in QUAL for a number of model elements related to a number of models in a model repository such as SAP's Modeling Infrastructure (MOIN) [9]. QUAL consists of a model querying part in order to select model elements to annotate, and an execution part to specify the annotation itself. For queries we support syntactic, type and semantic queries. Semantic queries allow us to select model elements which have already been applied with other profiles. This concept has been outlined as very useful to select, for instance, those model elements which have been annotated with the *SPT.Resource* stereotype and specify the utilization of them.

The QUAL approach also includes an extension mechanism in order to perform model annotations that conform to an algorithm specified in Java. This extension mechanism can be used to calculate (see Assessment Computation in Figure 3)

Hence, QUAL as a systematic model annotation approach has been used as implementation of the Composition Engine and of the Assessment Computation Engine (see Figure 3) as well.

5. Related Work

A number of approaches [15, 16, 20, 21, 22, 23, 24 and 25] are available for generating performance analysis models from development models by utilization of model driven techniques. Almost all these approaches follow the approach of deriving performance models from the annotated UML models.

However, these approaches differ in terms of the type of development models they take as input, and the performance models they output, which are then employed for performance prediction. They further differ in terms of the automation degree they offer. A very comprehensive survey of the different performance engineering tools/techniques is provided in [26] and [27]. Most of the available approaches demand performance expertise from their users. Our work addresses this need by integrating performance objectives and modification constraints, thereby providing decision support for non-performance experts, based on development models.

Furthermore, in our proposed architecture, the performance assessment results are visualized based on the development models. Theoretically we could use bidirectional model transformations [18] for integrating performance assessment results into development models, but our approach requires calculation of metrics and therefore the functionality provided by the QUAL approach (see section 0) is employed. QUAL completely automates the annotation of development models and calculation of performance assessment metrics, which has largely to be done manually in the existing approaches.

For the specification of the Modification Constraint View and the Performance Objective View we could have used specialized models instead of

UML profiles but the QUAL approach enables us to utilize the available tool support for UML profiles and to perform a straight forward composition of the proposed views by not weakening the separation of concerns principle.

6. Conclusion and future work

We presented an approach enabling performance related decision support for non-performance experts in terms of what in a design and in a resource mapping has to be changed to get better results with regard to performance objectives and modification constraints. We additionally proposed an architecture integrating this approach in MDE. We currently apply the approach for MDE of process orchestrations on top of SAP's Enterprise SOA.

Our approach utilizes the Performance Analysis View, the Modification Constraint View and the Performance Objective View which enables valuable feedback about how design and resource mapping decisions are related to performance objectives.

The approach enables information filtering to only provide information which is relevant for the user by taking performance objectives and modification constraints into account. We proposed initial performance assessment metrics enabling information interpretation for non-performance experts. In order to provide a systematic synchronization between the performance assessment and development models we proposed an architecture based on the systematic model annotation approach QUAL.

In the future we anticipate extending the expressiveness of the views we proposed. For the Modification Constrain View we will work on the specification of design alternatives. We also identified that delineating dependencies between performance objectives and design/resource mapping alternatives is required. We anticipate to either visualize those dependencies to a user as a metric or to calculate how the optimal configuration with respect to design and resource mapping alternatives should look like by taking performance objectives and modification constraints into account.

To realize user centric visualization of performance assessment we anticipate using a GIS-like representation of metrics such as proposed and implemented by [19].

In order to gain experiences with our approach for different domains we plan industrial case studies to assess business performance on the one hand and hosting scenarios on the other hand.

7. References

[1] Mathias Fritzsche, Jendrik Johannes, "Putting Performance Engineering into Model-Driven Engineering: Model-Driven Performance Engineering", MoDeVVa'07 at the Model Driven Engineering Languages and Systems, ISBN 2-7261-1294 3, 2007, pp. 77–87, (selected to appear in Springer-Verlag LNCS format).

[2] Dorin B. Petriu, Murray Woodside, "An intermediate metamodel with scenarios and resources", Software and Systems Modeling, Springer-Verlag, pp. 163–184, 2007.

[3] Murray Woodside, Greg Franks, Dorina C. Petriu, "The Future of Software Performance Engineering", 29th Int. Conference on Software Engineering, IEEE, 2007.

[4] Greg Franks, Dorina Petriu, Murray Woodside, Jing Xu, Peter Tregunno, "Layered Bottlenecks and Their Mitigation", Conference on Quantitative Evaluation of Systems (QUEST), IEEE, 2006.

[5] Jean Bézivin, Vladan Devedžić, Dragan Djurić, Jean-Marie Favreau, Dragan Gašević, Frederic Jouault, "An M3-Neutral infrastructure for bridging model engineering and ontology engineering", First International Conference on Interoperability of Enterprise Software and Applications, Springer, 2005, pp. 159-171.

[6] Farid Mehr, Ulf Schreier, "Modelling of Message Security Concerns with UML", 9th International Conference on Enterprise Information Systems, 2007.

[7] Farid Mehr, Mathias Fritzsche, Ulf Schreier, "QUAL: A Query and Annotation Language for the UML models of Service-oriented Applications", submitted to the International Journal of Business Process Integration and Management (IJBPIM).

[8] David L. Parnas, "On the Criteria to be Used in Decomposing Systems into Modules", Communication of the ACM, Vol.15, N°12, 1972.

[9] Michael Altenhofen, Thomas Hettel, Stefan Kusterer, "OCL Support in an Industrial Environment", LNCS Volume 4364, Springer-Verlag, 2007, pp. 169-178.

[10] Andreas Knoepfel, Bernhard Groene, Tabeling, P., "Fundamental Modeling Concepts: Effective Communication of IT Systems", John Wiley & Sons, 2006.

[11] ATLAS Group, "ATLAS transformation language", URL http://www.eclipse.org/m2m/atl/, 2007.

[12] XJ Technologies, "AnyLogic — multi-paradigm simulation software", URL http://www.xjtek.com/anylogic/, 2007.

[13] Dan Woods, Thomas Mattern, “Enterprise SOA: Designing IT for Business Innovation”, O’Reilly Media Inc, 2006.

[14] Robert Heidasch, “Get ready for the next generation of SAP business application based on Enterprise Service-Oriented Architecture (Enterprise SOA)”, SAP Professional Journal, 2007.

[15] Vittorio Cortellessa, Antinisca Di Marco, Paola Inverardi, “Software performance model-driven architecture”, SAC ’06: Proceedings of the 2006 ACM symposium on Applied computing, ACM Press, 2006, pp. 1218–1223.

[16] Andrea D’Ambrogio, “A model transformation framework for the automated building of performance models from UML models”, WOSP ’05: Proceedings of the 5th international workshop on Software and performance, ACM Press, 2005, pp. 75–86.

[17] OMG, “UML profile for schedulability, performance, and time specification”, URL http://www.omg.org/docs/formal/03-09-01.pdf, 2005

[18] Perdita Stevens, “Bidirectional Model Transformations in QVT: Semantic Issues and Open Questions”, Model Driven Engineering Languages and Systems, 2007, pp. 1-15.

[19] Christian Lange, Michel Chaudron, “Combining Metrics Data and the Structure of UML Models using GIS Visualization Approaches”, International Conference on Information Technology: Coding and Computing, 2005

[20] Wagh Ramrao, “Transformation of UML design model into performance model: A model driven framework”, ECOOP Student Workshop, 2006.

[21] Lloyd G. Williams, Connie U. Smith, “PASA: An architectural approach to fixing software performance problems“, In Software Engineering Research and Performance Engineering Services, 2002

[22] Andrew Bennett, A. J. Field, “Performance Engineering with the UML Profile for Schedulability, Performance and Time: a Case Study”, 12th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04), 2004

[23] Moreno Marzolla and Simonetta Balsamo, “UML-PSI: the UML Performance Simulator”, In IEEE First International Conference on Quantitative Evaluation of Systems (QEST), 2004

[24] Vittorio Cortellessa, Michele Gentile, and Marco Pizzuti., “XPRIT: An XML-based tool to translate UML diagrams into execution graphs and queueing networks”, In IEEE First International Conference on Quantitative Evaluation of Systems (QEST), 2004.

[25] Elena Gómez-Martínez and José Merseguer, “ArgoSPE: Model-Based Software Performance Engineering”, In 27th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, 2006

[26] Simonetta Balsamo, Antinisca Di Marco, P. Inverardi, M. Simeoni, “Model-Based Performance Prediction in Software Development: A Survey”, IEEE Transactions on Software Engineering, vol. 30, 2004.

[27] Sabri Pllana, Ivona Brandic, Siegfried Benkner, “Performance Modeling and Prediction of Parallel and Distributed Computing Systems: A Survey of the State of the Art”, In 1st International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'07), 2007.