Towards a Performance Model Management Repository for Component-based Enterprise Applications

Andreas Brunnert, Alexandru Danciu
fortiss GmbH
Guerickestr. 25
80805 Munich, Germany
{brunnert,danciu}@fortiss.org

Helmut Krcmar
Technische Universität München
Boltzmannstr. 3
85748 Garching, Germany
krcmar@in.tum.de

ABSTRACT

This work introduces a Performance Model Management Repository (PMMR) for component-based enterprise applications. A PMMR is a central server that allows managing performance model components in corporate environments. A key challenge when using performance models in such environments is to distribute, update and maintain them. Especially, when software components represented in performance models are under the control of different teams in an organization. Additional problems arise as soon as release cycles for their components are not synchronized. A PMMR helps to address these challenges by introducing a central repository in which different performance model component versions can be managed and maintained. Such capabilities support the collaboration of distributed teams as they can manage their performance model components independently from each other. Performance models of specific component versions can be combined into one performance model as required for the current performance evaluation. We propose to build such a PMMR using the capabilities provided by the Palladio Component Model (PCM) as meta-model and the EMFStore as underlying versioning repository.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: measurement techniques, modeling techniques

General Terms

Measurement, Performance

Keywords

Performance Model Repository, Performance Evaluation, Palladio Component Model, Enterprise Application, Component-based Performance Model

ICPE'15, Jan. 31–Feb. 4, 2015, Austin, Texas, USA.

ACM 978-1-4503-3248-4/15/01 ...$15.00.
http://dx.doi.org/10.1145/2668930.2695526.

1. INTRODUCTION

Performance models are still not in widespread industry use as of today [12, 11]. One of the most cited reasons for this lack of adoption is that the effort required to create performance models often outweighs their benefits [6, 9]. To reduce the modeling effort, several approaches to automatically generate performance models based on static and dynamic analysis have been proposed [2, 6, 15].

An additional challenge for applying performance models in industrial practice is the organizational complexity in corporate IT environments [5, 14]. Performance modeling is especially demanding for component-based enterprise applications as soon as components are under the control of different teams within one or more organizations. Performance models can thus only be created through the cooperation of these teams. It gets even worse if they adhere to different release cycles for their components. In such a scenario it is challenging to keep a performance model consistent and in sync with changes that occur in parallel.

The Performance Model Management Repository (PMMR) concept proposed in this work addresses these organizational challenges. Its primary purpose is that of an integration server for performance models to support the collaboration of distributed teams within an organization (see Figure 1). A PMMR contains architecture-level performance models which represent performance-relevant aspects of a software architecture separately from workload and hardware environment. Only performance-relevant aspects of software architecture are managed in a PMMR. Performance-relevant aspects of component-based enterprise applications are represented by component-based performance models (CBPM). CBPMs represent software components, their relationships, operation behavior and resource demands. A PMMR allows to manage software components in CBPMs independently from each other.

2. PERFORMANCE MODEL MANAGEMENT REPOSITORY

The realization of a PMMR is driven by several research questions. This work in progress paper focuses on the following three:

1. Which existing methodologies and technologies can be used for implementing the PMMR concept?

2. How can the relationships of components and their corresponding versions be represented in a PMMR?

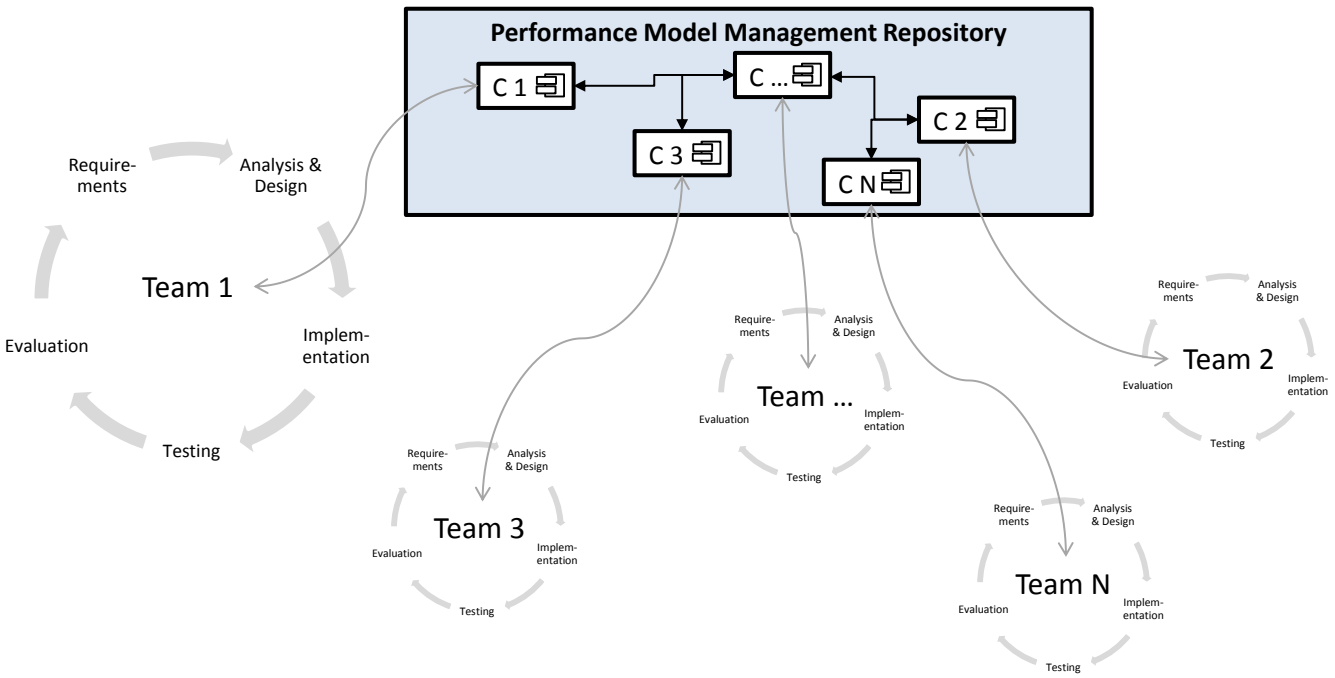

Figure 1: Conceptional architecture of a Performance Model Management Repository (PMMR)

3. How can a PMMR handle resource demand specifications in performance model components that are derived from different hardware environments?

To address the first research question, we propose the use of the Palladio Component Model (PCM) as meta-model for the component performance models managed in a PMMR [3]. A PMMR prototype is being developed on top of the PCM modeling environment (Palladio-Bench[1]) as PMMR client and the EMFStore as PMMR server [8]. Using the PCM modeling methodology and corresponding technologies allows to extend an existing modeling environment and avoids the need to introduce a new one.

PCM is designed with specific organizational roles in mind and consists of several model layers [3, 11]. One of the main models within the meta-model is the PCM repository model. The PCM repository model contains the components of a system, their operation behavior and resource demands as well as their relationships. The content of the PCM repository model is created by component developers. These repository model components are combined into a system model by system architects. A so called deployer can afterwards specify the available servers and resources (e.g., central processing unit (CPU) cores) in a resource environment model. An allocation model specifies how the repository model elements are mapped to these servers. A domain expert specifies the workload on the system represented by the other PCM model layers in the PCM usage model.

All PCM model layers apart from the repository model are intended to represent specific application scenarios. The repository model, on the other hand, is intended to contain reusable components for different application scenarios [3, 11]. The basic purpose of a PCM repository model and a PMMR are thus quite similar. However, the PCM repository model itself can nowadays not address the challenges mentioned in the introduction. PCM repository models are represented by single files that are hard to maintain by different teams concurrently. The result of this difficulty is that multiple PCM repository models with outdated component specifications exist, as multiple component versions need to be maintained at the same time by different teams. The PMMR extends the repository model concept in order to manage repository model components independently.

The PCM meta-model is based on the Eclipse Modeling Framework (EMF)[2]. All PCM-based performance models therefore conform not only to the PCM meta-model but also to the Ecore meta-model defined by EMF. We are leveraging this capability by using the EMFStore, which already implements the required versioning features for models based on the Ecore meta-model. The advantage of using EMFStore compared to other versioning systems is that it is designed to support the semantic versioning of models. Instead of working with textual representations of the models in existing systems such as Apache Subversion (SVN)[3], EMFStore uses the Ecore model elements and their relationships to manage models stored in a repository. For example, a structural change between two model versions is not represented as multiple lines in their textual representation. EMFStore rather stores the change in the Ecore model itself [8].

Users can access, analyze and edit models in the EMFStore using the EMF Client Platform (ECP)[4]. ECP supports collaborative editing of model versions using multiple clients at the same time. By using ECP as a plugin for the Palladio-

[1]http://www.palladio-simulator.com/

[2]http://www.eclipse.org/modeling/emf/

[3]http://subversion.apache.org/

[4]http://www.eclipse.org/ecp/

Bench, model versions in a repository are directly accessible to performance analysts [4]. A PMMR can thus be seen as an organization-wide replacement of the PCM repository model. System architects, deployers and domain experts in distributed teams can use component model versions in a PMMR to build their PCM model cases on demand.

To allow for a comparison of different component model versions, we are also leveraging the fact, that the generated performance models are based on EMF. By using the EMF-Compare framework[5], the differences between two model versions can be analyzed. Such capabilities allow for efficient version-to-version comparisons, to evaluate the performance impact of changes introduced in a specific component version [4].

The second research question is concerned with the representation of PMMR component relationships. Performance model component relationships are specified in PCM by their dependencies. Components in PCM repository models can *require* one or more other components to be usable. As these components are managed independently from each other in a PMMR, these *require* references now need to respect the specific version of a component. A meta-model extension is therefore necessary to specify these relationships across component versions in a PMMR. The Palladio-Bench also needs to be extended to support the user while interacting with different component versions.

Another reason for the difficulty of representing the component relationships is that the PMMR content can be derived from static (e.g., software designs) or dynamic (e.g., runtime measurements) analysis. It is very important to ensure that software components that are dependent on each other in a software architecture are represented in a compatible way in a PMMR. The easiest way to ensure this, is to agree on a common abstraction level for representing software systems in performance models. Following Wu and Woodside [18], we suggest that software systems represented in a PMMR should reflect the actual component subdivision in terms of encapsulated sets of functionality.

A PMMR only contains PCM repository model components in different versions. However, the PCM repository model components can contain resource demands (i.e., CPU or hard disk drive (HDD) demands) that are specified relative to a specific hardware resource. A PMMR should therefore be able to handle the heterogeneity of hardware environments on which different components are deployed. To handle different hardware environments and, thus, to address research question three, all resource demands of repository model components stored in a PMMR are specified relative to a common baseline. Following Menascé and Almeida [13], this common baseline is specified by benchmark scores for hardware resources supported by the PCM meta-model (i.e., CPU, HDD).

Using these benchmark scores in a PMMR allows to transform the resource demands specified in repository model components during PMMR check-ins and PMMR check-outs. Users can specify benchmark scores for all hardware resources referenced by repository model components they are intending to check-in or to check-out. These benchmark scores are then used during check-in to calculate the baseline resource demands relative to the common baseline benchmark scores. For one specific resource type (e.g., CPU) the resource demand ($r_{baseline}$) relative to a baseline hardware resource benchmark score ($b_{baseline}$) during check-in is calculated as shown in Equation 1. In this equation, $r_{checkinvalue}$ denotes the resource demand in a component model that a user checks-in to a PMMR. The benchmark score of the hardware resource used to derive the resource demand, which is also given by the user during check-in, is specified by $b_{checkinbenchmarkvalue}$:

$$r_{baseline} = \frac{b_{baseline}}{b_{checkinbenchmarkvalue}} * r_{checkinvalue} \quad (1)$$

During check-out, the resource demand for the user ($r_{checkoutvalue}$) is calculated relative to the benchmark score provided by the user ($b_{checkoutbenchmarkvalue}$) as follows:

$$r_{checkoutvalue} = \frac{b_{checkoutbenchmarkvalue}}{b_{baseline}} * r_{baseline} \quad (2)$$

This calculation is possible for different hardware resources. For CPU benchmarks it is important that the benchmark can evaluate the performance of a single core (e.g., SPEC CPU2006[6]), otherwise it is much harder to adapt the resource demand from one server to another. Without such a transformation, users of PMMR components would need to know which hardware resources have been used to derive resource demands for the component models. This approach simplifies the reuse of component performance models. If resource demands are estimated instead of measured [15] and no benchmark scores are available, we propose the use of the baseline benchmark scores during check-in and check-out. The check-in and check-out capabilities of ECP for models in the EMFStore need to be extended to support this transformation process.

3. RELATED WORK

Several approaches for versioning model artifacts exist in literature [1]. However, these approaches do not address the specific requirements which arise from the versioning of performance models of individual components.

The work that is most closely related to the PMMR concept is the Performance Knowledge Base (PKB) introduced by Woodside et al. [17]. The PKB is broader in its scope. It is envisioned as central performance repository. The authors propose to store measurement and model prediction results in a PKB instead of the models itself. In this way the PMMR concept differs from the PKB idea as it is designed so that performance models can be stored in it directly. It is not intended to be used as result repository. However, Woodside et al. [17] also note that the PKB should allow to build performance models on demand. These models should be built based on the current state of parameters and a so called model base that builds the foundation for modeling a system.

Koziolek [11] also argues that central performance model repositories (called model libraries) *"... could allow rapid performance predictions ..."*. However, the author does not propose a solution for the realization of such a repository.

4. CONCLUSION AND FUTURE WORK

The proposed approach enables distributed (or even cross-organizational) teams to contribute and maintain perfor-

[5]http://www.eclipse.org/emf/compare/

[6]http://www.spec.org/cpu2006/

mance models concurrently and provides access to a coherent and consistent model of interrelated components.

Future work includes a better integration of existing approaches that support the performance evaluation of component-based enterprise applications using the PCM meta-model with the PMMR concept. For example, an approach proposed in [7] supports developers with insights on the response times of the component they are currently developing by employing the PCM meta-model. The response time of components is calculated based on component reuse and could therefore be derived using the PMMR. As explained in Section 2, existing performance evaluation approaches using PCM need to agree on a common abstraction level for representing software components in performance models before they can be managed by a PMMR.

Another challenge for future work is to define at which level performance models can be abstracted to reduce the amount of components that need to be represented in a performance model. For example, if someone evaluates a specific enterprise application, one might not be interested in a detailed representation of all dependencies of an existing component used by the current application. It would thus be an interesting research direction to evaluate how detailed white-box and high-level black-box models for the same component can be stored in a PMMR [10]. A black-box representation could, for example, only represent the response time behavior of a component in a specific deployment scenario [16]. Whereas a white-box representation would model the component behavior in detail including its dependencies to other components. Clients of a PMMR should be able to choose between such representations during check-out time.

Once the PMMR prototype is completely implemented, it will be evaluated in an experimental setup to validate the feasibility of the approaches described in this work. Afterwards, the PMMR prototype needs to be evaluated in a corporate environment to validate the intended improvements.

5. REFERENCES

[1] K. Altmanninger, M. Seidl, and M. Wimmer. A survey on model versioning approaches. *International Journal of Web Information Systems*, 5(3):271–304, 2009.

[2] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni. Model-based performance prediction in software development: A survey. *IEEE Transactions on Software Engineering*, 30(5):295 – 310, 2004.

[3] S. Becker, H. Koziolek, and R. Reussner. The palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82(1):3 – 22, 2009.

[4] A. Brunnert and H. Krcmar. Detecting performance change in enterprise application versions using resource profiles. In *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '14, New York, NY, USA, 2014. ACM.

[5] A. Brunnert, C. Vögele, A. Danciu, M. Pfaff, M. Mayer, and H. Krcmar. Performance management work. *Business & Information Systems Engineering*, 6(3):177–179, 2014.

[6] A. Brunnert, C. Vögele, and H. Krcmar. Automatic performance model generation for java enterprise edition (ee) applications. In M. S. Balsamo, W. J. Knottenbelt, and A. Marin, editors, *Computer Performance Engineering*, volume 8168 of *Lecture Notes in Computer Science*, pages 74–88. Springer Berlin Heidelberg, 2013.

[7] A. Danciu, A. Brunnert, and H. Krcmar. Towards performance awareness in java ee development environments. In S. Becker, W. Hasselbring, A. van Hoorn, S. Kounev, and R. Reussner, editors, *Proceedings of the Symposium on Software Performance: Descartes/Kieker/Palladio Days 2014*, pages 152–159, November 2014.

[8] M. Koegel and J. Helming. Emfstore: A model repository for emf models. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ICSE '10, pages 307–308, New York, NY, USA, 2010. ACM.

[9] S. Kounev. *Performance Engineering of Distributed Component-Based Systems - Benchmarking, Modeling and Performance Prediction*. Shaker Verlag, Ph.D. Thesis, Technische Universität Darmstadt, Germany, Aachen, Germany, 2005.

[10] S. Kounev, F. Brosig, and N. Huber. The descartes modeling language. Technical report, Universität Würzburg, Institut für Informatik, 2014.

[11] H. Koziolek. Performance evaluation of component-based software systems: A survey. *Performance Evaluation*, 67(8):634–658, 2010.

[12] M. Mayer, S. Gradl, V. Schreiber, H. Wittges, and H. Krcmar. A survey on performance modelling and simulation of sap enterprise resource planning systems. In *The 10th International Conference on Modeling and Applied Simulation*, pages 347–352. Diptem Universitá di Genoa, 2011.

[13] D. A. Menascé and V. A. F. Almeida. *Capacity Planning for Web Services: Metrics, Models, and Methods*. Prentice Hall, Upper Saddle River, New Jersey, 2002.

[14] A. Schmietendorf, E. Dimitrov, and R. R. Dumke. Process models for the software development and performance engineering tasks. In *Proceedings of the 3rd International Workshop on Software and Performance*, WOSP '02, pages 211–218, New York, NY, USA, 2002. ACM.

[15] C. Smith. Introduction to software performance engineering: Origins and outstanding problems. In M. Bernardo and J. Hillston, editors, *Formal Methods for Performance Evaluation*, volume 4486 of *Lecture Notes in Computer Science*, pages 395–428. Springer Berlin Heidelberg, 2007.

[16] A. Wert, J. Happe, and D. Westermann. Integrating software performance curves with the palladio component model. In *Proceedings of the third joint WOSP/SIPEW international conference on Performance Engineering*, ICPE '12, pages 283–286, New York, NY, USA, 2012. ACM.

[17] M. Woodside, G. Franks, and D. C. Petriu. The future of software performance engineering. In *Future of Software Engineering (FOSE)*, pages 171–187, Minneapolis, MN, USA, 2007.

[18] X. Wu and M. Woodside. Performance modeling from software components. *SIGSOFT Softw. Eng. Notes*, 29(1):290–301, 2004.