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

Work-in-Progress Paper (WiP)
International Conference on Performance Engineering (ICPE) 2015

Andreas Brunnert¹, Alexandru Danciu¹, Helmut Krcmar²

¹fortiss GmbH, ²Technische Universität München
Agenda

• Motivation & Vision
• Basic Technologies
• Component (Version) Dependencies
• Handling Resource Demands
• Related Work
• Outlook
Agenda

• Motivation & Vision
• Basic Technologies
• Component (Version) Dependencies
• Handling Resource Demands
• Related Work
• Outlook
Motivation & Vision

• Performance models are still not in widespread industry use (Koziolek 2010, Mayer et al. 2011):
  – Creation effort often outweighs their benefits (Brunnert et al. 2013, Kounev 2005)
  – Several approaches for automatic generation (Balsamo et al. 2004, Brunnert et al. 2013, Smith 2007)

• Challenge for applying performance models in industrial practice is the organizational complexity (Brunnert et al. 2014, Schmietendorf et al. 2002):
  – Components of enterprise applications are often under the control of different teams within one or more organizations
  – Teams adhere to different release cycles for their components
  – Challenge to keep a performance model consistent and in sync
Motivation & Vision

- To introduce an integration server for performance models to support the collaboration of distributed teams within an organization.
Agenda

- Motivation & Vision
- Basic Technologies
- Component (Version) Dependencies
- Handling Resource Demands
- Related Work
- Outlook
Basic Technologies
PCM as Meta-Model

- Workload
- Components, Interfaces, Relationships, Control Flows, Resource Demands
- System composed of components within the repository model
- Mapping of system components to hardware servers
- Specifies available servers, networks, …
Required Enhancements

• PCM repository models are represented by single files that are hard to maintain by different teams concurrently.

• Multiple PCM repository models with outdated component specifications exist, as multiple component versions need to be maintained at the same time by different teams.

We propose to use EMFStore\(^1\) as PMMR server:

– The PCM meta-model is based on the Eclipse Modeling Framework (EMF)
– EMFStore implements the required versioning features for models based on the Ecore meta-model

\(^1\) [http://eclipse.org/emfstore/](http://eclipse.org/emfstore/)
Agenda

• Motivation & Vision
• Basic Technologies
• Component (Version) Dependencies
• Handling Resource Demands
• Related Work
• Outlook
We propose to extend the PCM meta-model to specify require references across component versions

- The Palladio-Bench also needs to be extended to support the user while interacting with different component versions
Agenda

- Motivation & Vision
- Basic Technologies
- Component (Version) Dependencies
- Handling Resource Demands
- Related Work
- Outlook
Handling Resource Demands

Hardware-specific Resource Demands

- PCM repository model components can contain hardware-specific resource demands
- Resource demands stored in a PMMR are specified relative to a common baseline

\[
R_{baseline} = \frac{b_{baseline}}{b_{checkinbenchmarkvalue}} \times R_{checkinvalue}
\]

\[
R_{checkoutvalue} = \frac{b_{checkoutbenchmarkvalue}}{b_{baseline}} \times R_{baseline}
\]
Agenda

• Motivation & Vision
• Basic Technologies
• Component (Version) Dependencies
• Handling Resource Demands
• Related Work
• Outlook
Related Work

• Several approaches for versioning model artifacts exist in literature (Altmanninger et al. 2009)
  – These approaches do not address the specific requirements which arise from the versioning of performance models of individual components

• Woodside et al. (2007) proposed the Performance Knowledge Base (PKB) as a central performance repository
  – The PKB is intended to store measurement and model prediction results in a PKB instead of the models itself
  – PKB should allow to build performance models on demand
  – PMMR is designed so that performance models can be stored in it directly

• Koziolek (2010) argues that central performance model repositories (called model libraries) "... could allow rapid performance predictions ...“.
Agenda

• Motivation & Vision
• Basic Technologies
• Component (Version) Dependencies
• Handling Resource Demands
• Related Work
• Outlook
Outlook

• Abstraction Level:
  – Choose abstraction level to reduce the amount of components that need to be represented
  – Level of detail for storing white-box and high-level black-box models

• Evaluation:
  – Experimental setup to validate the feasibility of the approaches
  – Representative software development project to validate the intended improvements

• Integration:
  – Danciu et al. (2014) propose an approach to support developers with insights on the response times of the component they are currently developing
References


Alexandru Danciu, Andreas Brunnert
performancegroup@fortiss.org
pmw.fortiss.org
Basic Technologies
Performance (Meta-)Models

• LQN, QNs, QPNs depict workload, hardware environment and performance-relevant aspects of an application in one monolithic model
  – Hard to change a single aspect without modifying the whole model

• Architecture-level performance models (e.g. the Palladio Component Model (PCM)) allow to specify these aspects independently from each other
  – Used to represent resource profiles
  – Several existing ways to create such models based on static, dynamic or hybrid analysis

• We propose the use of the Palladio Component Model (PCM) as meta-model for the component performance models managed in a PMMR (Becker et al. 2009)
Implementation of the PMMR

PCM as Meta-Model

- Repository models are created by component developers

[Diagram showing the interaction between InterfaceA and InterfaceB, and the relationships between ComponentA and ComponentB, including their operations and compartments.]
Implementation of the PMMR

PCM as Meta-Model

- System models are created by system architects
Handling Resource Demands
Using Benchmark Scores

• Scores are specified for all relevant hardware resources.
  – \( r_{\text{baseline}} \) denotes baseline resource demand
  – \( b_{\text{baseline}} \) denotes baseline hardware resource benchmark score

• During check-in:
  – \( r_{\text{checkinvalue}} \) denotes resource demand measured by the user
  – \( b_{\text{checkinbenchmarkvalue}} \) denotes benchmark score of the hardware resource

\[
    r_{\text{baseline}} = \frac{b_{\text{baseline}}}{b_{\text{checkinbenchmarkvalue}}} \times r_{\text{checkinvalue}} \tag{1}
\]

• During check-out:
  – \( r_{\text{checkoutvalue}} \) denotes resource demand calculated relative to benchmark score
  – \( b_{\text{checkoutbenchmarkvalue}} \) denotes benchmark score of the target hardware resource

\[
    r_{\text{checkoutvalue}} = \frac{b_{\text{checkoutbenchmarkvalue}}}{b_{\text{baseline}}} \times r_{\text{baseline}} \tag{2}
\]