Austin, TX, USA, 2015-02-04

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

Work-in-Progress Paper (WiP)

International Conference on Performance Engineering (ICPE) 2015

Andreas Brunnert[1], **Alexandru Danciu[1]**, Helmut Krcmar[2]

[1]fortiss GmbH, [2]Technische 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

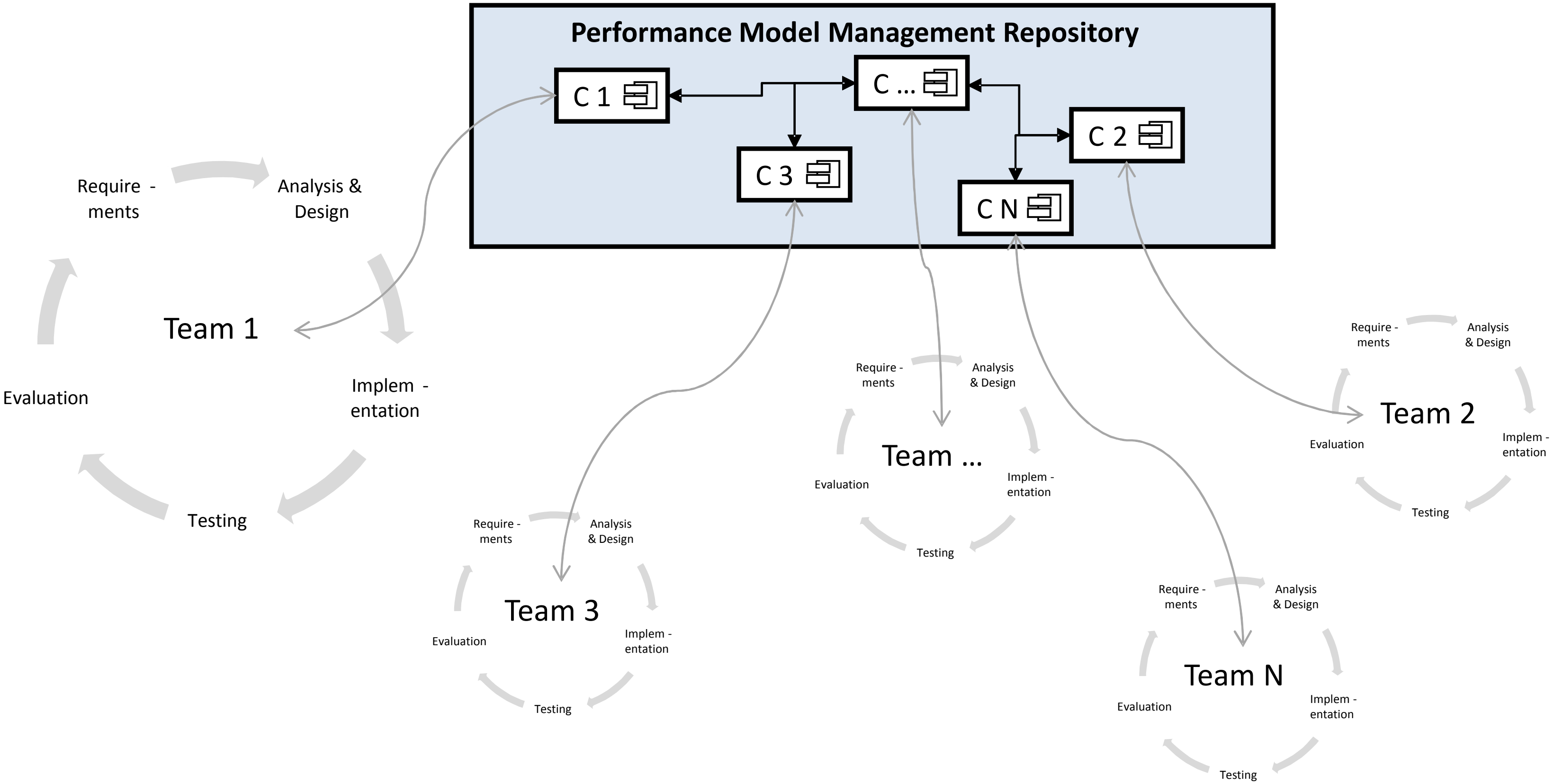
Motivation & Vision

- 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.

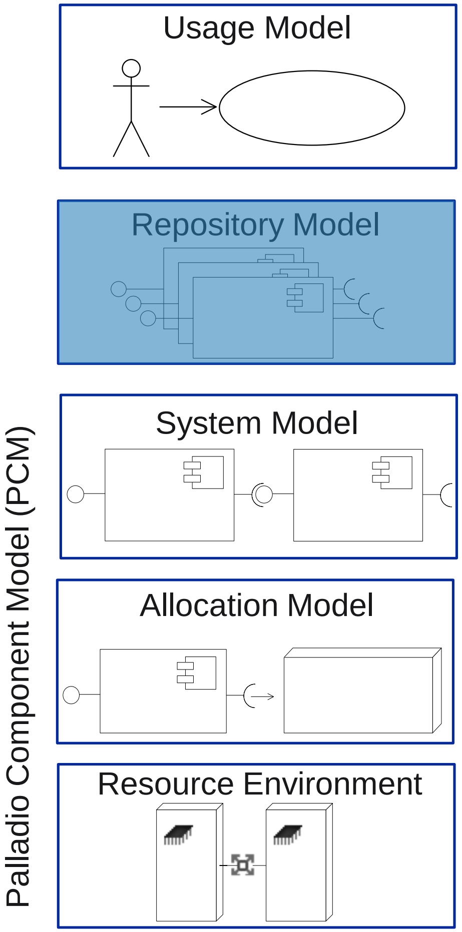


Agenda

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

Basic Technologies

PCM as Meta-Model



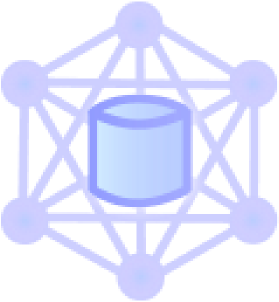
- Workload
- Components, Interfaces, Relationships, Control Flows, Resource Demands
- System composed of components within the repository model
- Mapping of system components to hardware servers
- Specifies available servers, networks, …

Basic Technologies

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

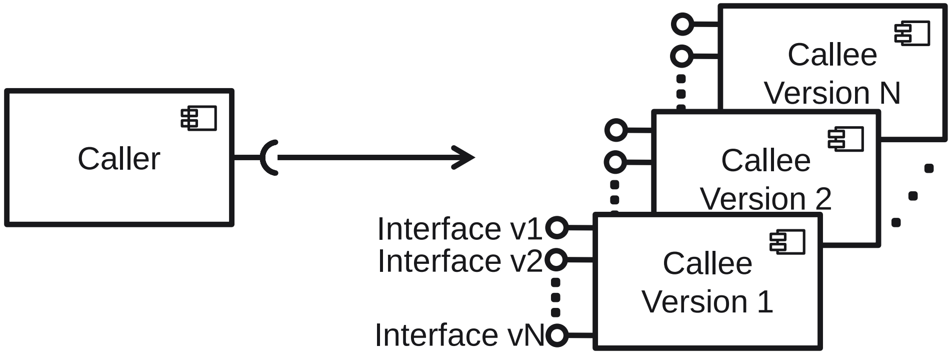
[1] http://eclipse.org/emfstore/

Agenda

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

Component (Version) Dependencies

Extension of the PCM Meta-Model



- 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

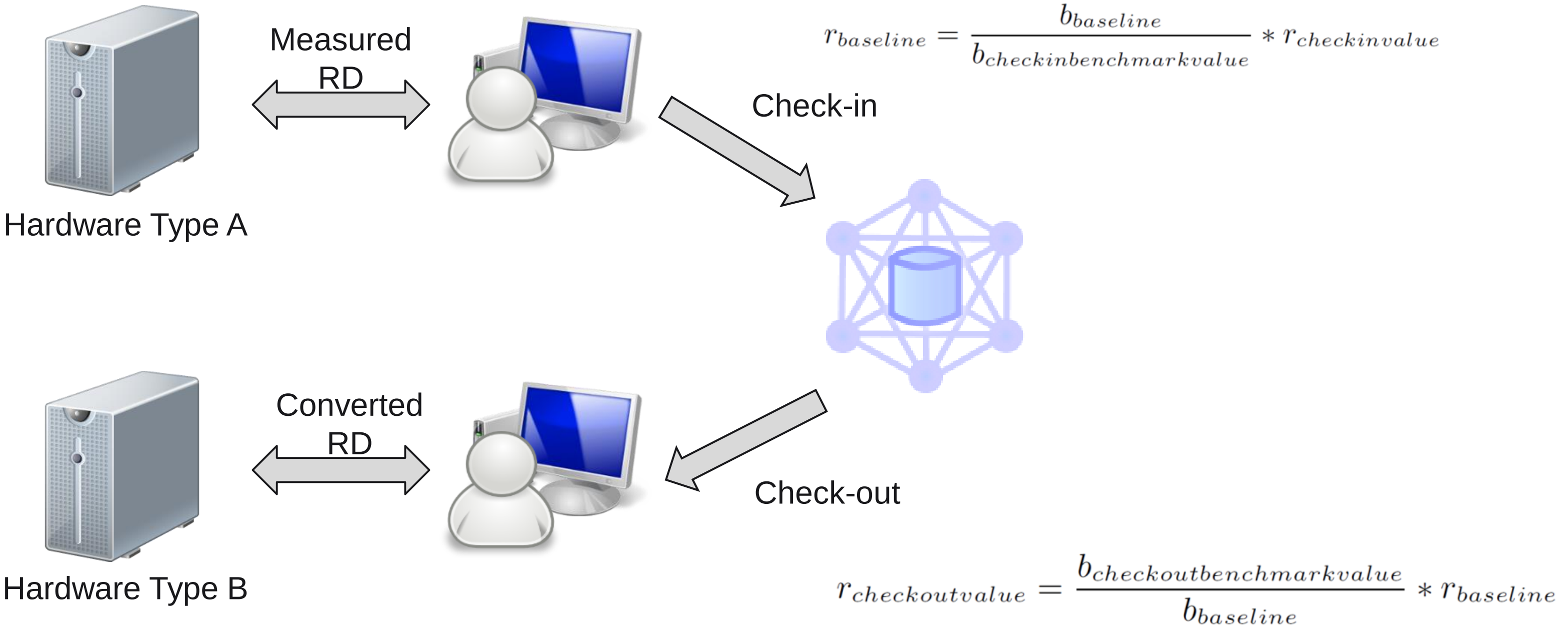
- Motivation & Vision
- Basic Technologies
- Component (Version) Dependencies
- Handling Resource Demands
- Related Work
- Outlook

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



$$r_{baseline} = \frac{b_{baseline}}{b_{checkinbenchmarkvalue}} * r_{checkinvalue}$$

$$r_{checkoutvalue} = \frac{b_{checkoutbenchmarkvalue}}{b_{baseline}} * r_{baseline}$$

Agenda

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

Related Work

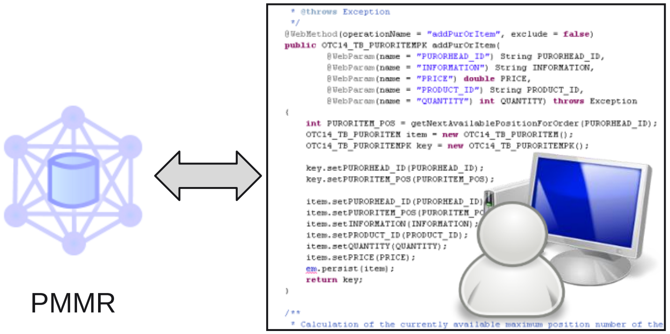
- Several approaches for versioning model artifacts exist in literature *(Altmanninger et al. 2009)*
  - These approaches do not address the specific requirements which arise from the versioning of performance models of individual components
- Woodside et al. (2007) proposed the Performance Knowledge 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



```
 * @throws Exception
 */
@WebMethod(operationName = "addPurOrItem", exclude = false)
public OTC14_TB_PURORITEMPK addPurOrItem(
        @WebParam(name = "PURORHEAD_ID") String PURORHEAD_ID,
        @WebParam(name = "INFORMATION") String INFORMATION,
        @WebParam(name = "PRICE") double PRICE,
        @WebParam(name = "PRODUCT_ID") String PRODUCT_ID,
        @WebParam(name = "QUANTITY") int QUANTITY) throws Exception
{
    int PURORITEM_POS = getNextAvailablePositionForOrder(PURORHEAD_ID);
    OTC14_TB_PURORITEM item = new OTC14_TB_PURORITEM();
    OTC14_TB_PURORITEMPK key = new OTC14_TB_PURORITEMPK();

    key.setPURORHEAD_ID(PURORHEAD_ID);
    key.setPURORITEM_POS(PURORITEM_POS);

    item.setPURORHEAD_ID(PURORHEAD_ID)
    item.setPURORITEM_POS(PURORITEM_PO
    item.setINFORMATION(INFORMATION);
    item.setPRODUCT_ID(PRODUCT_ID);
    item.setQUANTITY(QUANTITY);
    item.setPRICE(PRICE);
    em.persist(item);
    return key;
}

/**
 * Calculation of the currently available maximum position number of the
```

References

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

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.

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.

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.

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.

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.

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.

S. Kounev. Performance Engineering of Distributed Component-Based Systems - Benchmarking, Modeling and Performance Prediction. Shaker Verlag, Ph.D. Thesis, Technische Universit at Darmstadt, Germany, Aachen, Germany, 2005.

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

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 Università di Genoa, 2011.

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.

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.

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.

CONTACT US

Alexandru Danciu,
Andreas Brunnert
performancegroup@fortiss.org
pmw.fortiss.org

Basic Technologies

Performance (Meta-)Models

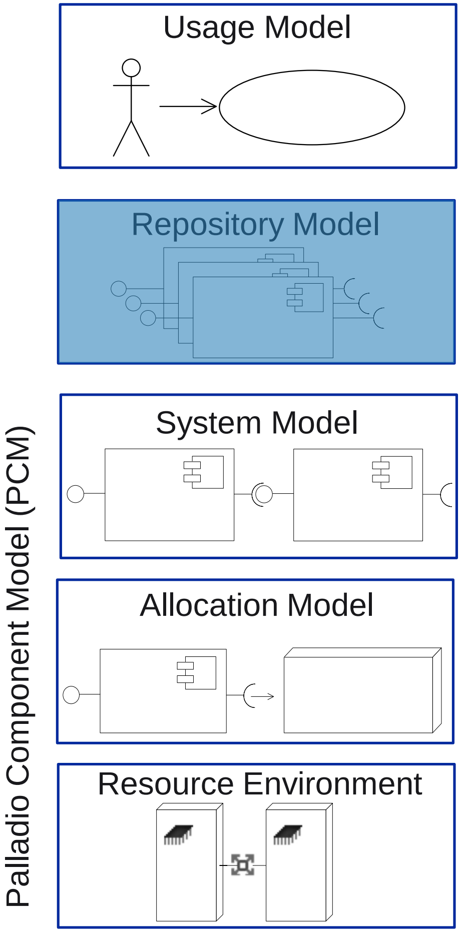
- LQN, QNs, QPNs depict workload, hardware environment and performance-relevant aspects of an application in one monolitic model
  - Hard to change a single aspect without modifying the whole model

- 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

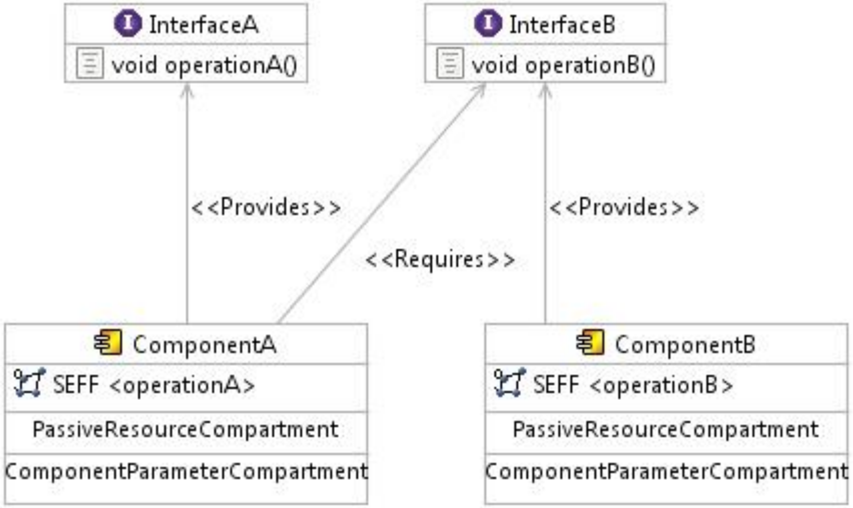
- We propose the use of the Palladio Component Model (PCM) as meta-model for the component performance models managed in a PMMR (*Becker et al. 2009*)

Implementation of the PMMR

PCM as Meta-Model

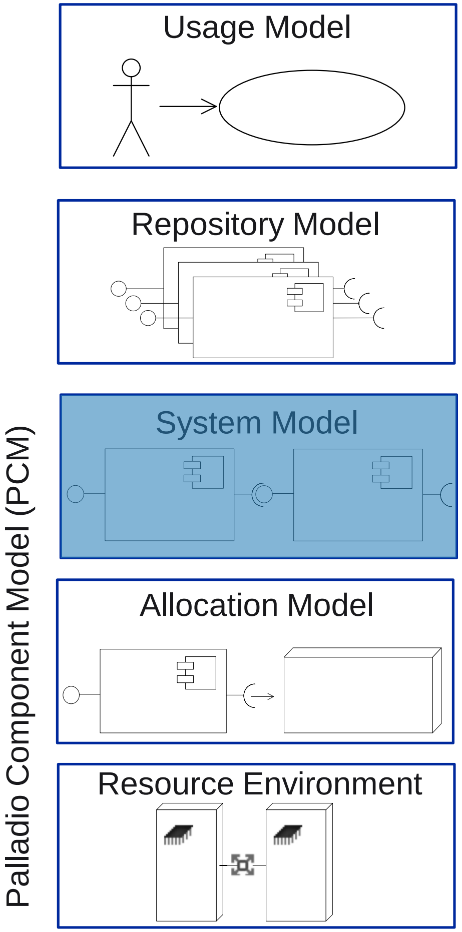


- Repository models are created by component developers

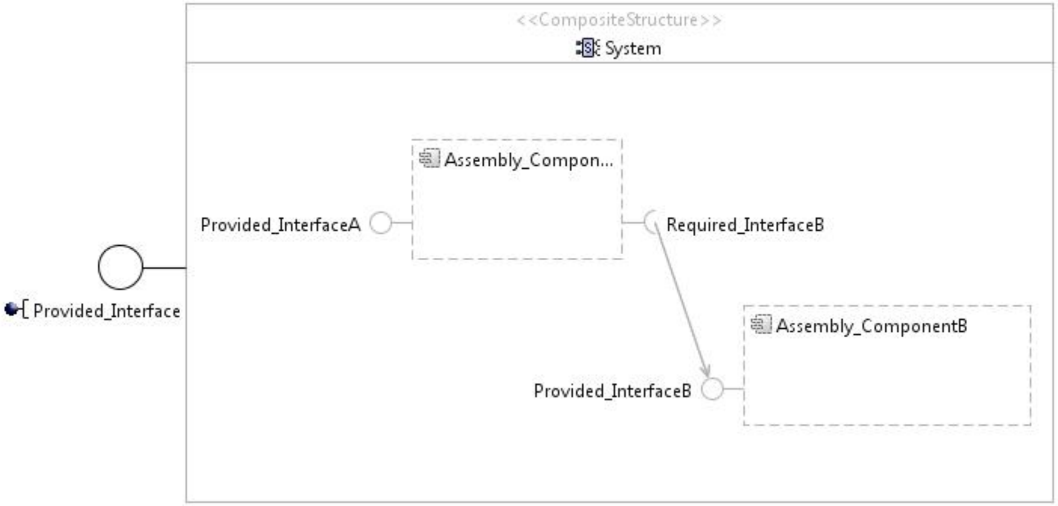


Implementation of the PMMR

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.
  - *$r_{baseline}$ denotes baseline resource demand*
  - *$b_{baseline}$ denotes baseline hardware resource benchmark score*
- *During check-in:*
  - *$r_{checkinvalue}$ denotes resource demand measured by the user*
  - *$b_{checkinbenchmarkvalue}$ denotes benchmark score of the hardware resource*

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

- *During check-out:*
  - *$r_{checkoutvalue}$ denotes resource demand calculated relative to benchmark score*
  - *$b_{checkoutbenchmarkvalue}$ denotes benchmark score of the target hardware resource*

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