Karlsruhe Institute of Technology

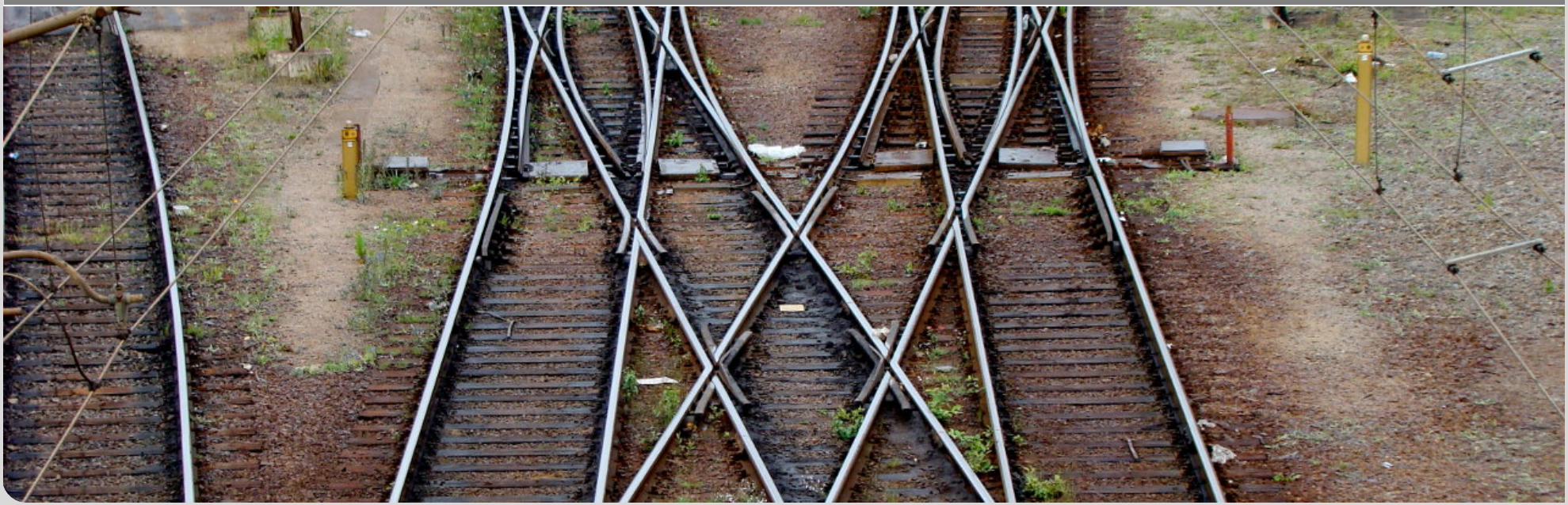
Descartes research

Controlling the Palladio Bench using the Descartes Query Language

Kieker/Palladio Days 2013

Fabian Gorsler, Fabian Brosig, Samuel Kounev | 2013-11-27

DESCARTES RESEARCH GROUP



Our Motivation

Service X	?ms
Service Y	?ms
Service Z	?ms
Resource A	??%
Resource B	??%

Figure 1 : The Performance Prediction Process

Our Motivation

Service X	?ms
Service Y	?ms
Service Z	?ms
Resource A	??%
Resource B	??%

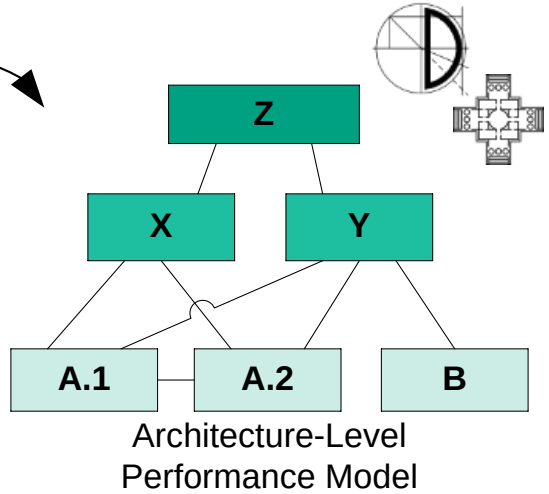


Figure 1 : The Performance Prediction Process

Our Motivation

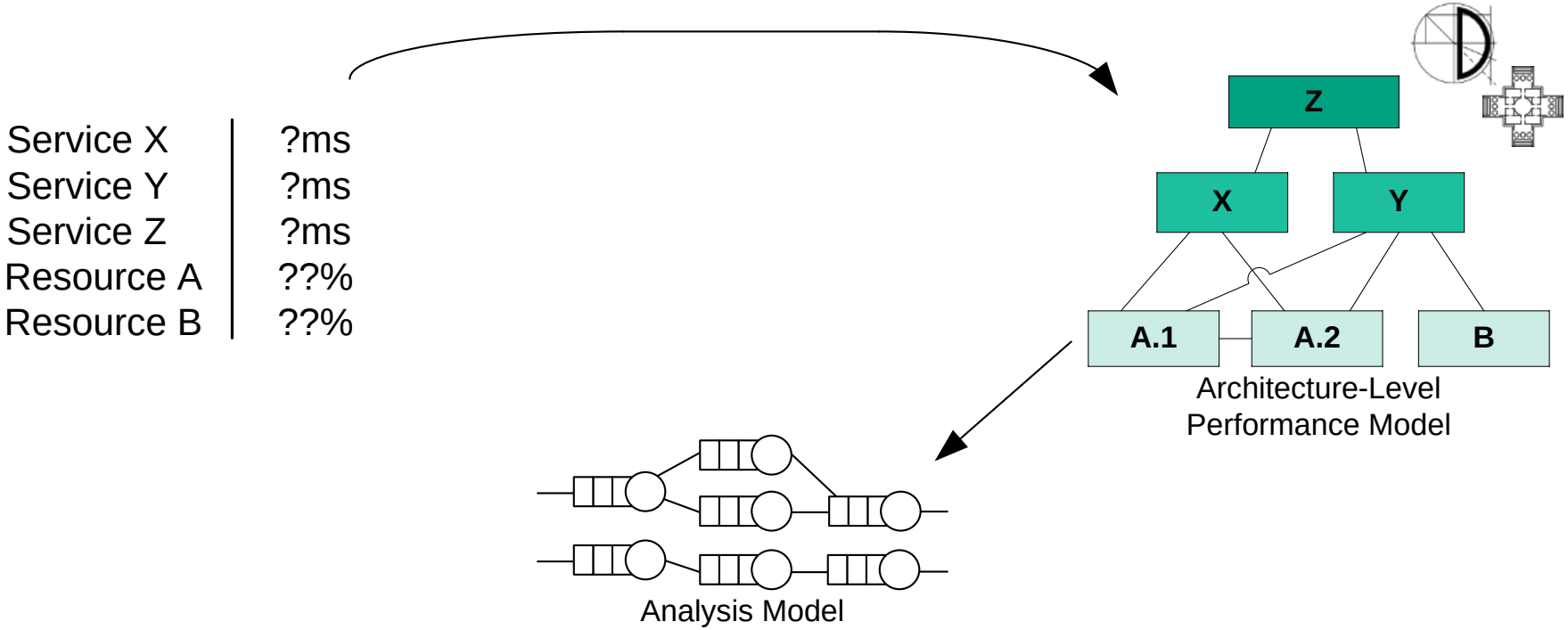


Figure 1 : The Performance Prediction Process

Our Motivation

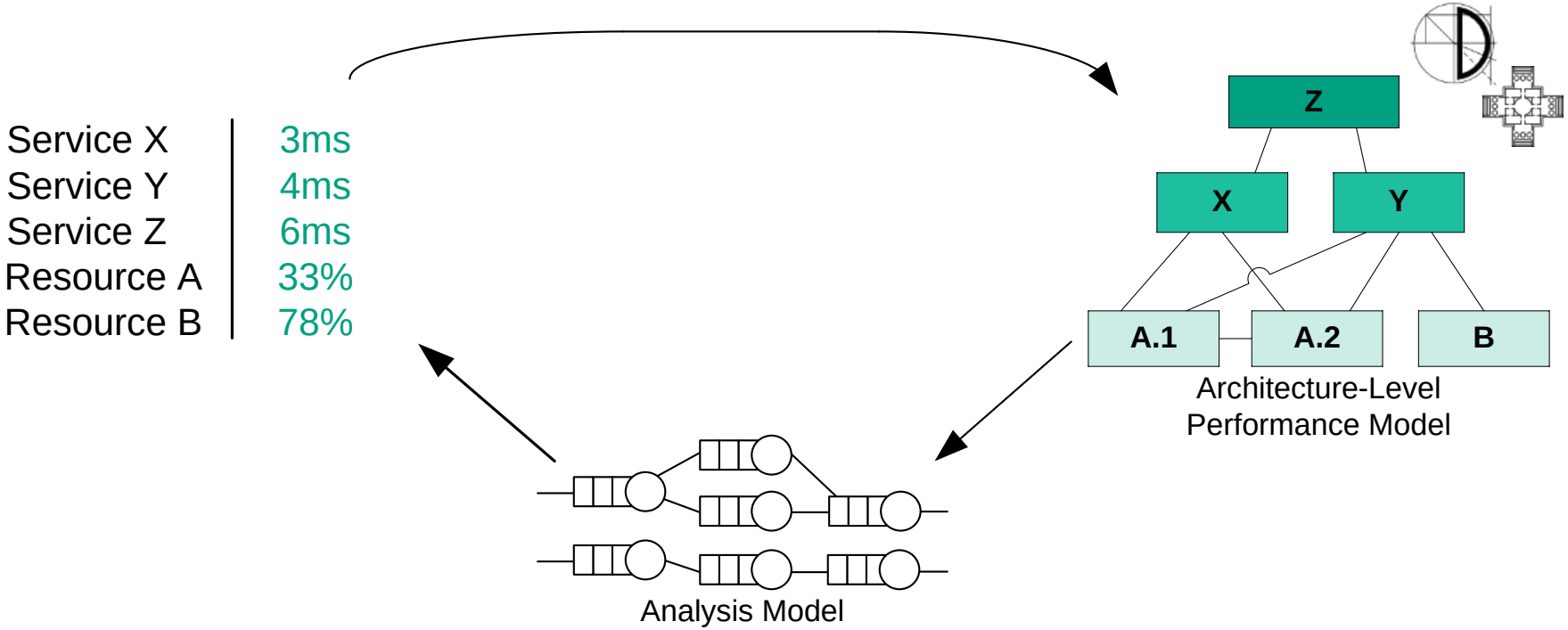


Figure 1 : The Performance Prediction Process

Our Motivation

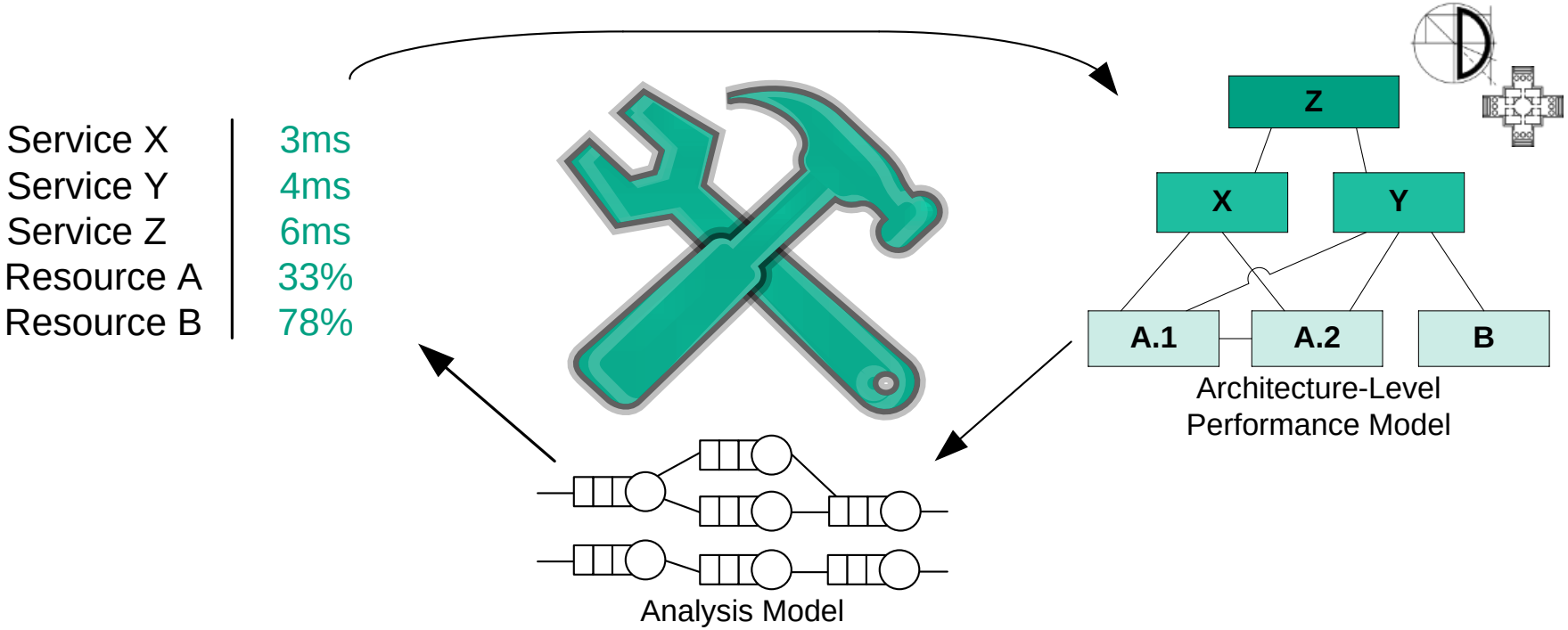


Figure 1 : The Performance Prediction Process

Our Approach

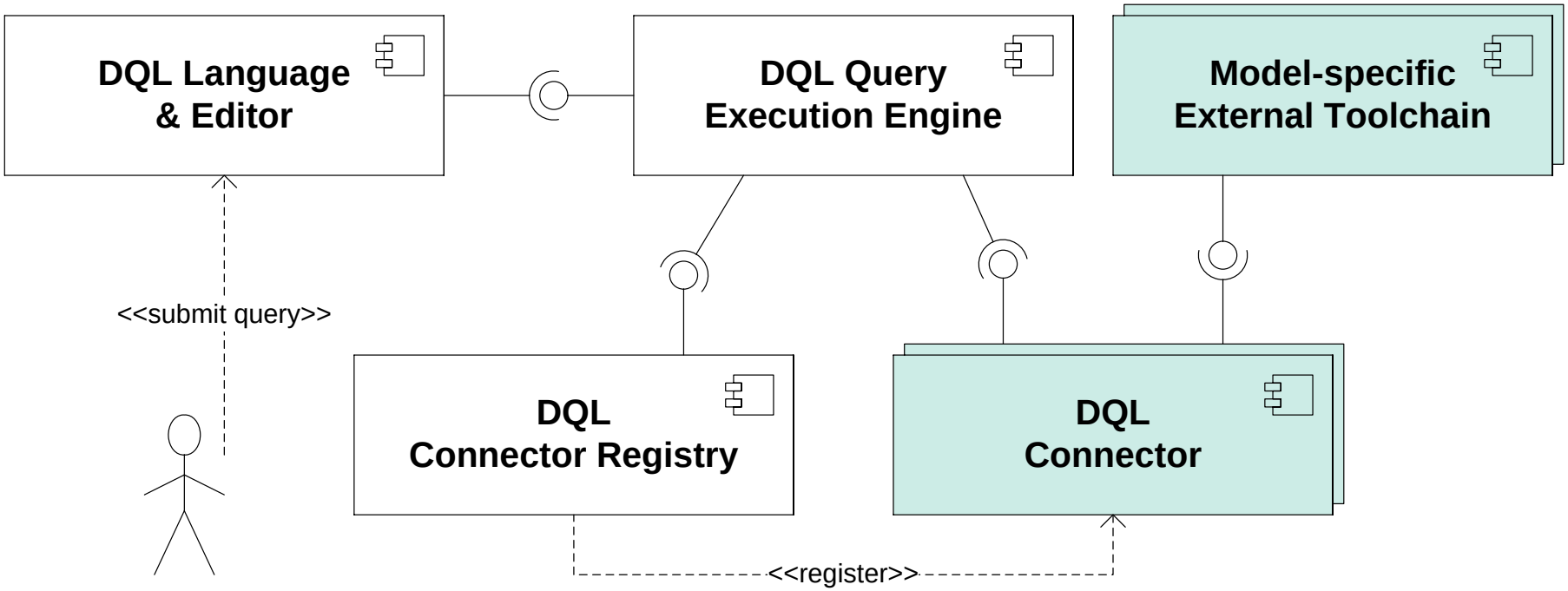
What is the Descartes Query Language (DQL)?

- A *declarative query language* [Gorsler, 2013]
- *Independent* of a specific modeling or prediction formalism
- An approach *to integrate* existing tools and techniques
- An interface *to unify* other approaches' interfaces
- Built on top of an *extensible architecture*

...and what it is not?

- Neither an approach for performance predictions, ...
- ...nor a model transformation approach

Architecture of DQL



MediaStore and Outline

MediaStore

- MediaStore is an example of a three-tier web application
- Demonstrates capabilities of Palladio Component Model (PCM)
- Contains an *AppServer* and a *DBServer* with *CPUs* and *HDDs* each

$\Rightarrow$ Is DQL usable in such a scenario?

Outline

1. Determine performance-relevant entities and metrics
2. Trigger a simulation and extract performance metrics
3. Trigger an experiment series through Degrees of Freedom (DoFs)

Finding Entities

- Obtain information about *all performance-relevant entities*

⇝ Interpret the model instance

Finding Entities

- Obtain information about *all performance-relevant entities*

⇝ Interpret the model instance

```
LIST ENTITIES
USING pcm@'mediastore.properties';
```

Finding Entities

- Obtain information about *all performance-relevant entities*

⇝ Interpret the model instance

```
LIST ENTITIES
USING pcm@'mediastore.properties';
```

Exemplary Result:

Type	Id	Alias	Metric	Value
Resource	id1	AppServer_CPU	./.	./.

Backgrounds: Type Mapping

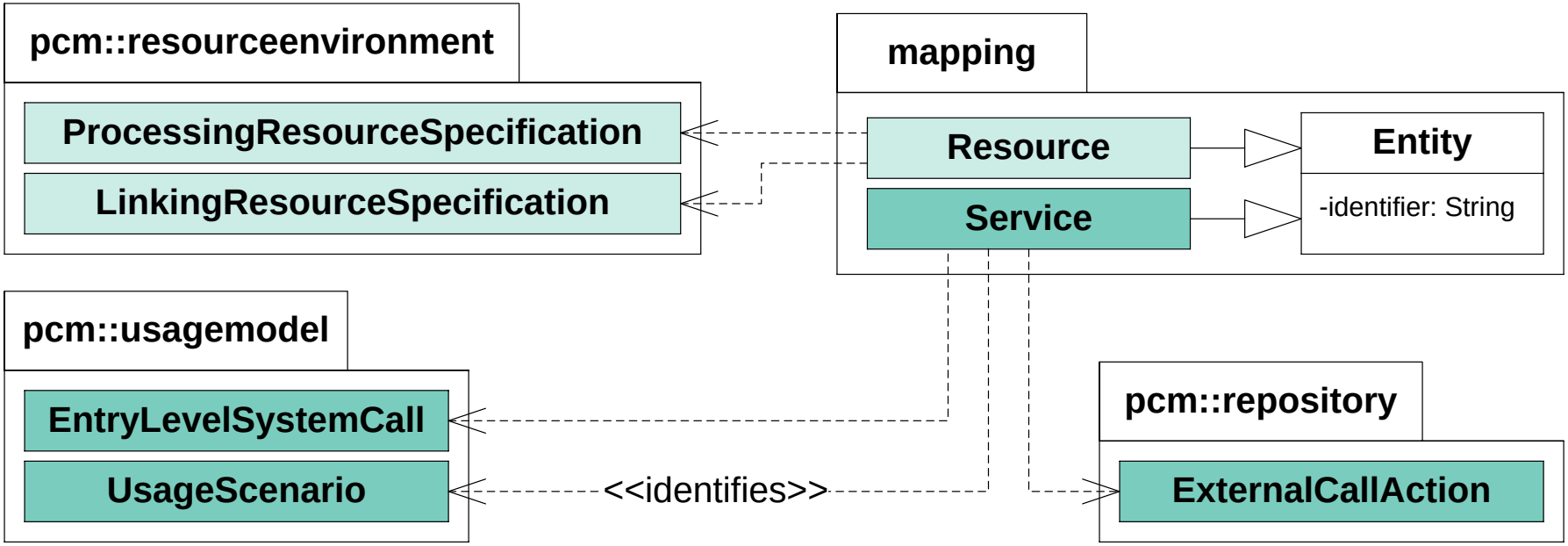


Figure 2 : PCM ← DQL based on [Reussner et al., 2011]

- Direct EMF/Ecore accesses
- Loosely-coupled using *identifier values*

Determining available Metrics

- Obtain information about *available performance metrics*

⇝ Interpret the referenced entities

Determining available Metrics

- Obtain information about *available performance metrics*

⇝ Interpret the referenced entities

```
LIST METRICS
 (RESOURCE 'id1' AS AppServer_CPU,
  RESOURCE 'id2' AS DBServer_CPU
  RESOURCE 'id3' AS DBServer_HDD)
USING pcm@'mediastore.properties';
```

Determining available Metrics

- Obtain information about *available performance metrics*

⇝ Interpret the referenced entities

```
LIST METRICS
 (RESOURCE 'id1' AS AppServer_CPU,
  RESOURCE 'id2' AS DBServer_CPU
  RESOURCE 'id3' AS DBServer_HDD)
USING pcm@'mediastore.properties';
```

Exemplary Result:

Type	Id	Alias	Metric	Value
Resource	id1	AppServer_CPU	Utilization	./.

Triggering a Simulation

- Request performance metrics for model entities

⇝ Trigger a simulation run and extract results

Triggering a Simulation

- Request performance metrics for model entities

⇝ Trigger a simulation run and extract results

```
SELECT AppServer_CPU.utilization, DBServer_CPU.utilization,
       DBServer_HDD.utilization
FOR RESOURCE 'id1' AS AppServer_CPU,
    RESOURCE 'id2' AS DBServer_CPU,
    RESOURCE 'id3' AS DBServer_HDD
USING pcm@'mediastore.properties';
```

Triggering a Simulation

- Request performance metrics for model entities

⇝ Trigger a simulation run and extract results

```
SELECT AppServer_CPU.utilization, DBServer_CPU.utilization,
       DBServer_HDD.utilization
FOR RESOURCE 'id1' AS AppServer_CPU,
    RESOURCE 'id2' AS DBServer_CPU,
    RESOURCE 'id3' AS DBServer_HDD
USING pcm@'mediastore.properties';
```

Exemplary Result:

Type	Id	Alias	Metric	Value
Resource	id1	AppServer_CPU	Utilization	0.3671

Bonus: Aggregates in DQL

- Computation of aggregated performance metrics
- Functionality *independent* of a DQL Connector

Bonus: Aggregates in DQL

- Computation of aggregated performance metrics
- Functionality *independent* of a DQL Connector

```
SELECT MEAN(AppServer1_CPU.utilization,
            AppServer2_CPU.utilization)
FOR RESOURCE 'idA1' AS AppServer1_CPU,
    RESOURCE 'idA2' AS AppServer2_CPU,
USING pcm@'mediastore.properties';
```

Backgrounds: Trigger and Extraction

- *Palladio Component Model (PCM) Experiment Automation* to trigger simulations [Merkle, 2011]
  - Part of the PCM Incubator, not yet stable
- Static mapping of available performance metrics
  - `Resource`: *demanded time* and *utilization*
  - `Service`: *response time*
- Direct access to performance metrics through *SensorFramework*
  - Workaround for unique identification of `Resource` instances

More Automation? More Automation!

- Evaluate DoFs, i.e. variability, in model instances
- Automate experiment series with a single query

$\rightsquigarrow$ Alter the model instance and trigger $i \times j \times \ldots$ simulations

More Automation? More Automation!

- Evaluate DoFs, i.e. variability, in model instances
- Automate experiment series with a single query

$\rightsquigarrow$ Alter the model instance and trigger $i \times j \times \ldots$ simulations

```
SELECT AppServer_CPU.utilization, DBServer_CPU.utilization,
       DBServer_HDD.utilization



FOR RESOURCE 'id1' AS AppServer_CPU,
    RESOURCE 'id2' AS DBServer_CPU,
    RESOURCE 'id3' AS DBServer_HDD
USING pcm@'mediastore.properties';
```

More Automation? More Automation!

- Evaluate DoFs, i.e. variability, in model instances
- Automate experiment series with a single query

$\rightsquigarrow$ Alter the model instance and trigger $i \times j \times \ldots$ simulations

```
SELECT AppServer_CPU.utilization, DBServer_CPU.utilization,
       DBServer_HDD.utilization

FOR RESOURCE 'id1' AS AppServer_CPU,
    RESOURCE 'id2' AS DBServer_CPU,
    RESOURCE 'id3' AS DBServer_HDD
USING pcm@'mediastore.properties';
```

More Automation? More Automation!

- Evaluate DoFs, i.e. variability, in model instances
- Automate experiment series with a single query

$\rightsquigarrow$ Alter the model instance and trigger $i \times j \times \ldots$ simulations

```
SELECT AppServer_CPU.utilization, DBServer_CPU.utilization,
       DBServer_HDD.utilization
EVALUATE DOF
  VARYING 'id4' AS ClosedWorkloadPopulation <100, 200>
          'id5' AS ActionReplication <2, 8>
FOR RESOURCE 'id1' AS AppServer_CPU,
    RESOURCE 'id2' AS DBServer_CPU,
    RESOURCE 'id3' AS DBServer_HDD
USING pcm@'mediastore.properties';
```

DQL Editor and Result Visualization

Java

File Edit Navigate Search Project Run Window Help

Quick Access | Java

PalladioDoFQuery.dql

```
SELECT AppServer_CPU.utilization, DBServer_CPU.utilization,
       DBServer_HDD.utilization
EVALUATE DOF
    VARYING  '_TyV-MFBwEd6ActLj8Gdl_A' AS ClosedWorkloadPopulation <100, 200>
             '_Q8jwMEg9Ed2v5eXKEb0Q9g'  AS ActionReplication <2, 8>
FOR RESOURCE '_5uTBUBpmEdyxqpPYxT_m3w@CPU' AS AppServer_CPU,
    RESOURCE '_tVi40Dq_EeCCbpF63PfiyA@CPU' AS DBServer_CPU,
    RESOURCE '_tVi40Dq_EeCCbpF63PfiyA@HDD' AS DBServer_HDD
USING pcm@'mediastore.properties';
```

Query

PalladioDoFQuery. | PalladioDoFQuery. | PalladioDoFQuery. | PalladioDoFQuery.

Entity Type	Entity Alias	Entity Identifier	Metric Name	Resu	Result Valu	Result Accu
D DoF	Replication	_Q8jwMEg9Ed2v5eXKEb0Q9g	Parameter Setting	true	2.000000	1.0000
D DoF	Workload	_TyV-MFBwEd6ActLj8Gdl_A	Parameter Setting	true	100.000000	1.0000
R Resource	AppServer_CPU	_5uTBUBpmEdyxqpPYxT_m3w@	utilisation	true	0.603704	1.0000
R Resource	DBServer_CPU	_tVi40Dq_EeCCbpF63PfiyA@CPI	utilisation	true	0.500486	1.0000
R Resource	DBServer_HDD	_tVi40Dq_EeCCbpF63PfiyA@HD	utilisation	true	0.714469	1.0000

Results

Writable | Insert | 9 : 35

Conclusion

- ✓ DQL Connector ready to use with PCM
- ✓ Approach to unify the interfaces to PCM, instead of:
  - EMF + PCM EA + SensorFramework
- ✓ Further automation with DoFs
  - Relies on PCM EA
- ✓ Use Case: API for PCM
  - Can be embedded or exposed as a centralized instance

- Future work on *goal-oriented queries*

References I

Gorsler, F. (2013).
Online Performance Queries for Architecture-Level Performance Models.
Master's thesis, Karlsruhe Institute of Technology (KIT).

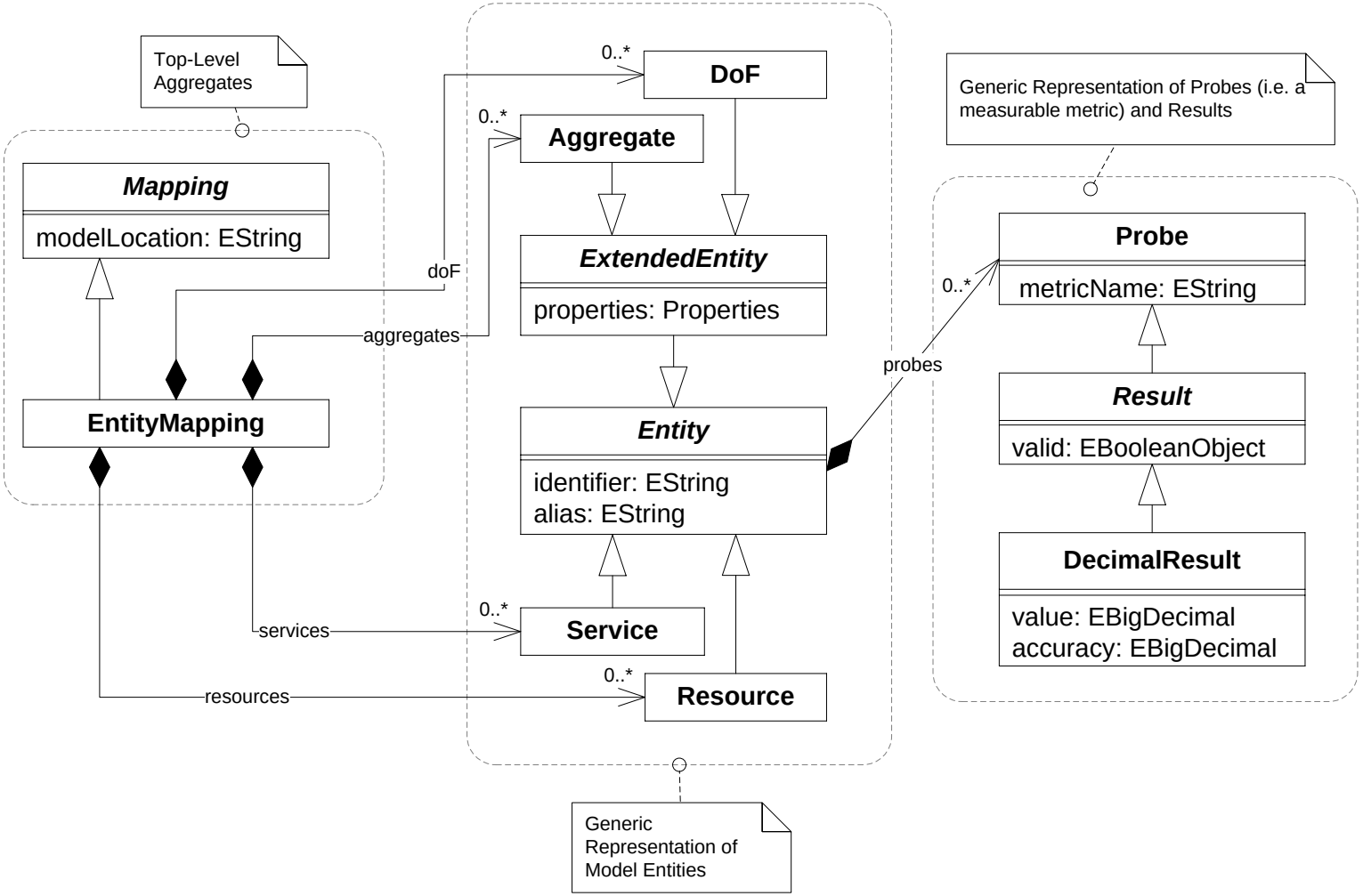
Merkle, P. (2011).
Comparing Process- and Event-oriented Software Performance Simulation.
Master thesis, Karlsruhe Institute of Technology (KIT).

Reussner, R., Becker, S., Burger, E., Happe, J., Hauck, M., Koziolek, A., Koziolek, H., Krogmann, K., and Kuperberg, M. (2011).
The Palladio Component Model.
Technical report, Karlsruhe Institute of Technology (KIT), Karlsruhe.

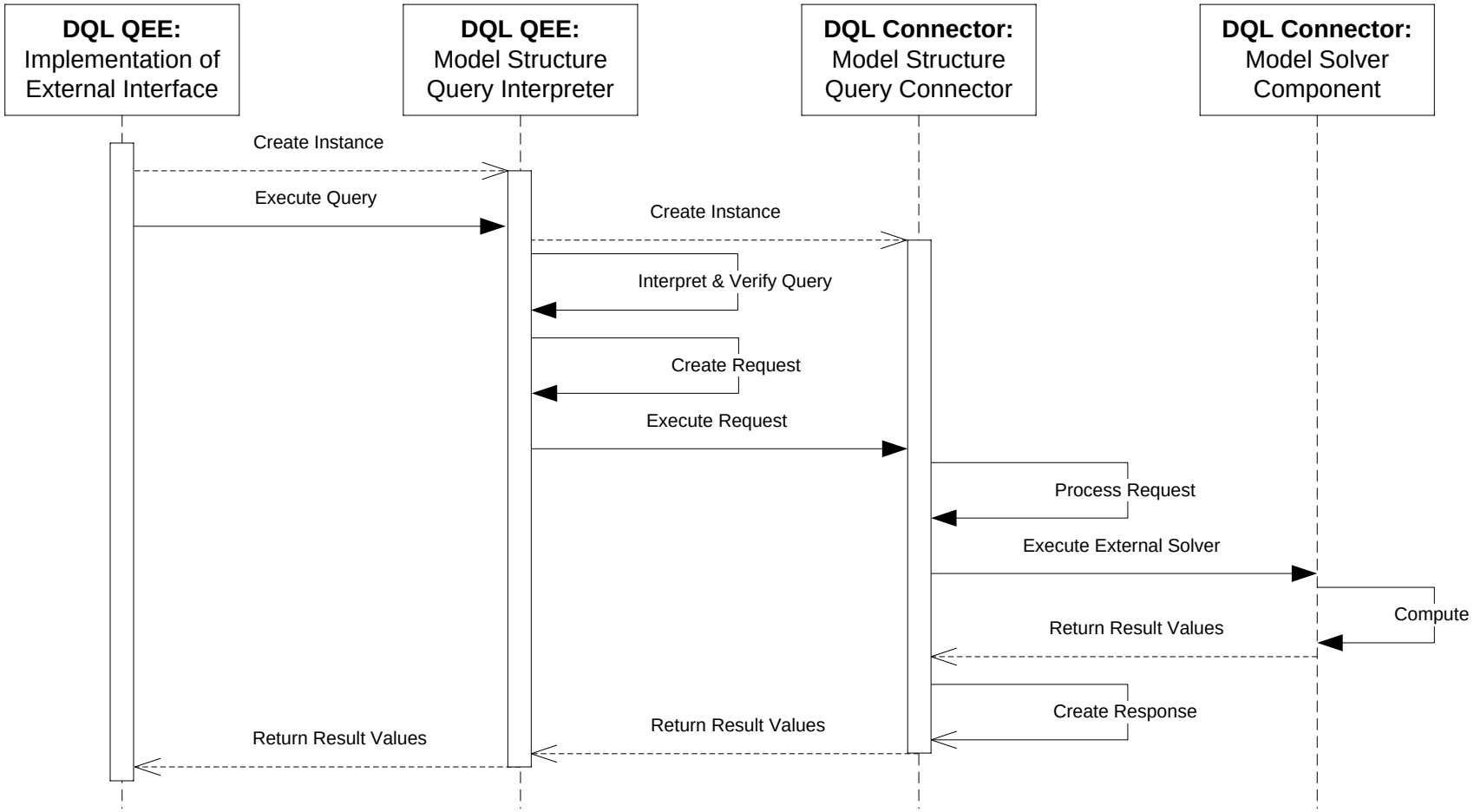
Backup: Mapping Meta-Model I

- To interchange data, requests and results need a representation
- Support of different modeling approaches requires abstraction
- Encapsulation of structural model information simplifies interfaces

Backup: Mapping Meta-Model II



Backup: Component Interaction I



Backup: Component Interaction II