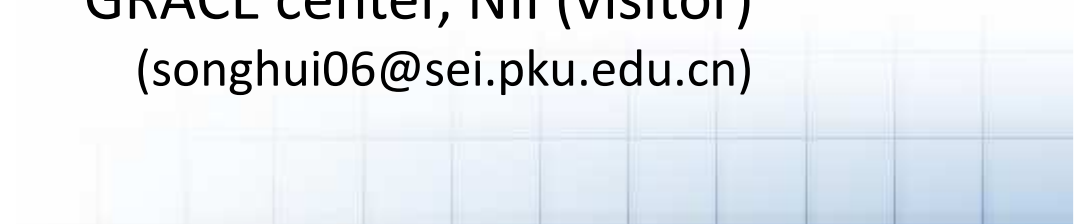


Supporting Architecture-Based Runtime System Management

--An application of bi-transformation

Hui SONG (宋 晖)

Peking University, China
GRACE center, NII (visitor)
(songhui06@sei.pku.edu.cn)



Problem Origin

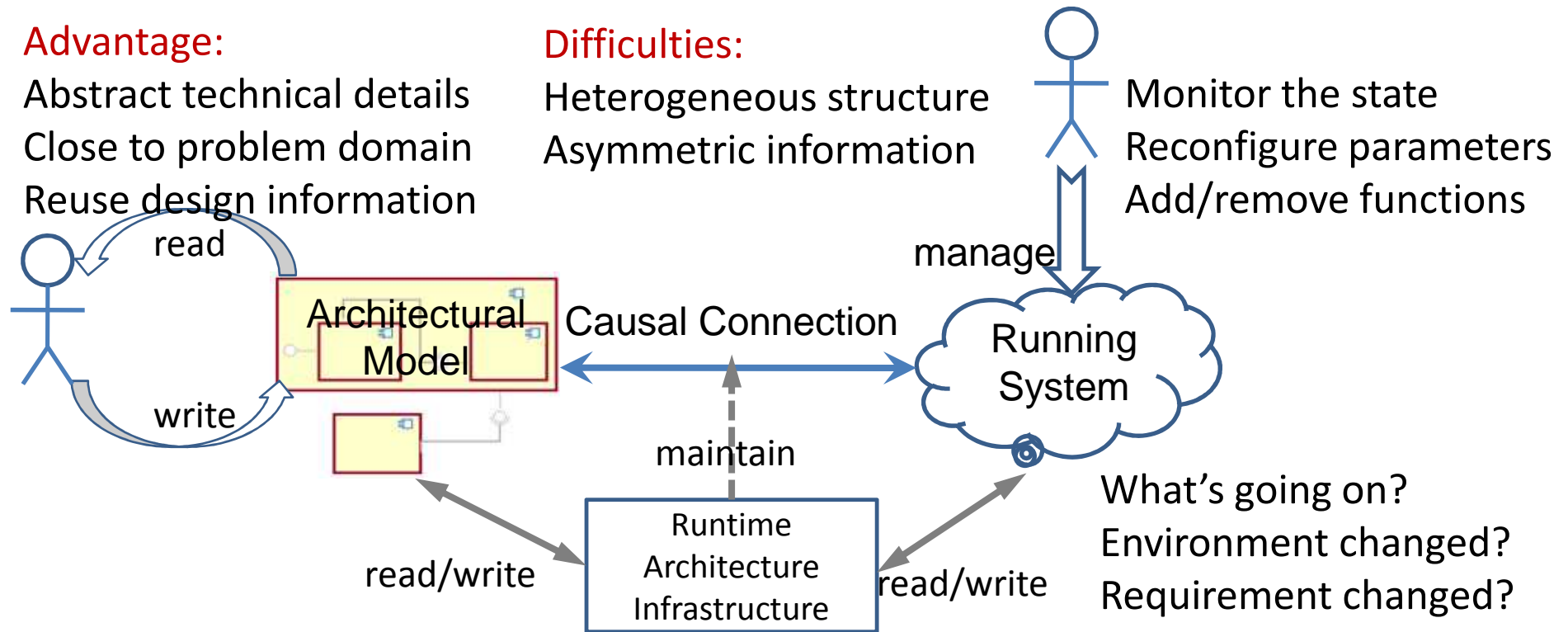
Advantage:

- Abstract technical details
- Close to problem domain
- Reuse design information

Difficulties:

- Heterogeneous structure
- Asymmetric information

- Monitor the state
- Reconfigure parameters
- Add/remove functions



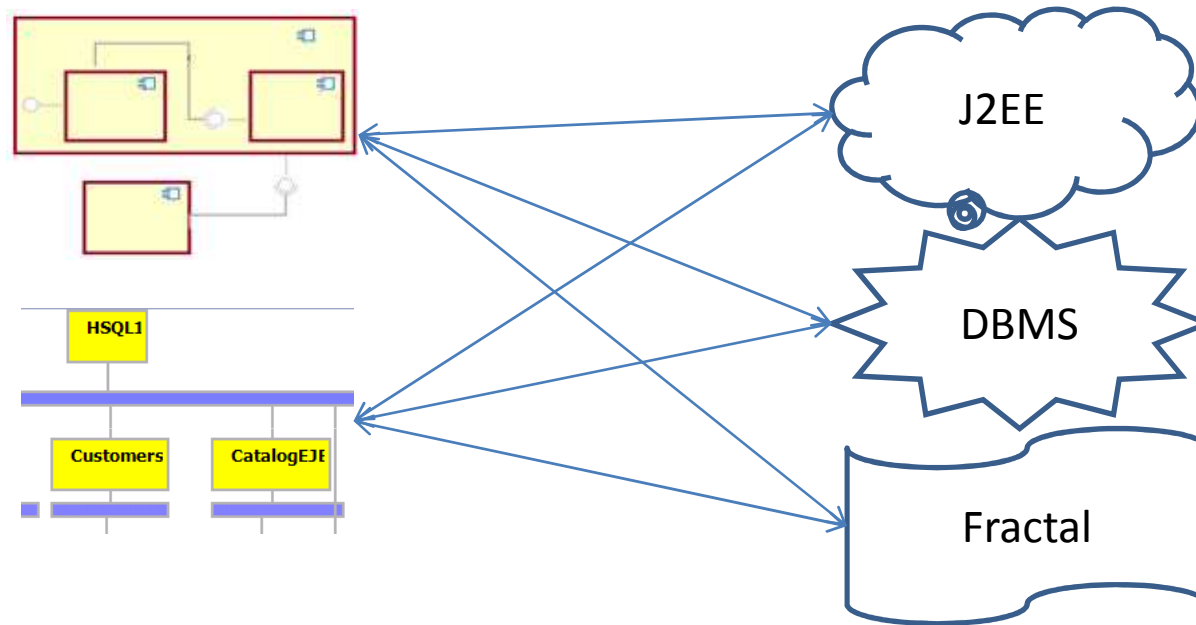
What changes mean on the other side?

Current Status



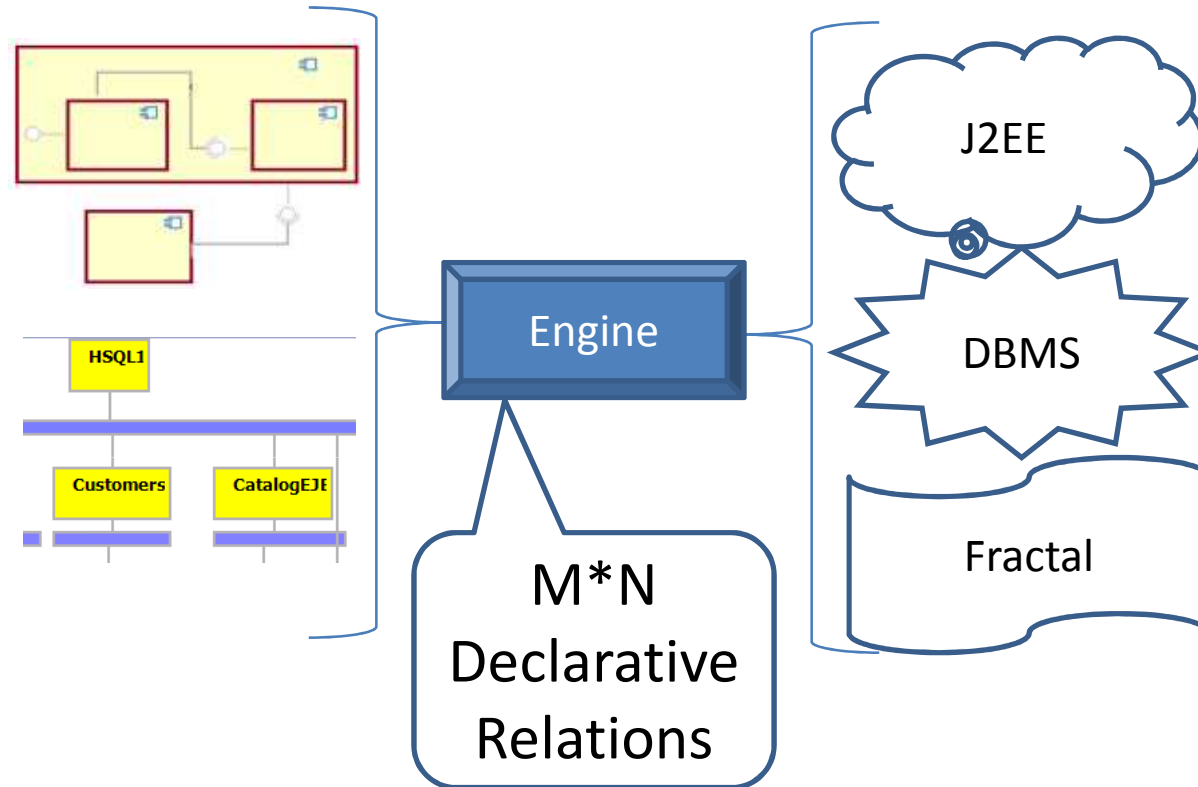
- A hot research topic
 - ICSE 2008 ten years most influential paper award
- Many approaches these years
 - From embedded to enterprise systems
 - Architecture styles for evolution, self-adaptation...
 - Different concerns
- Infrastructures in these approaches
 - Manually constructed, specific infrastructures

We need many infrastructures



- M*N manual constructed infrastructure
 - Considering everything together
- Commonality: propagating changes

How about a framework

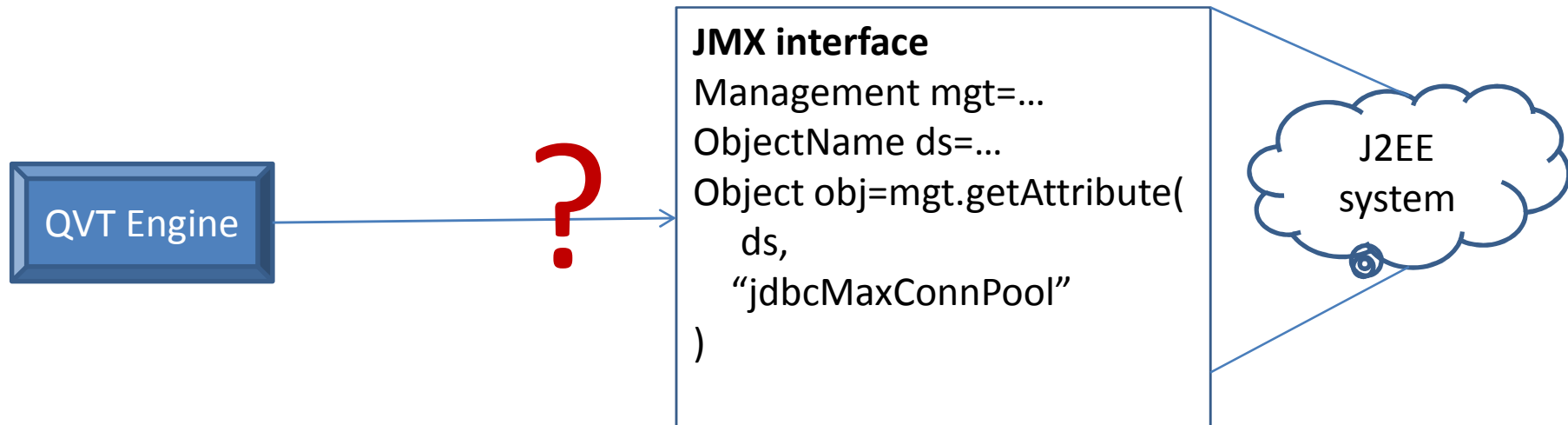


Powerful enough to deal with
the heterogeneity and asymmetry

What we choose in our initial attempt

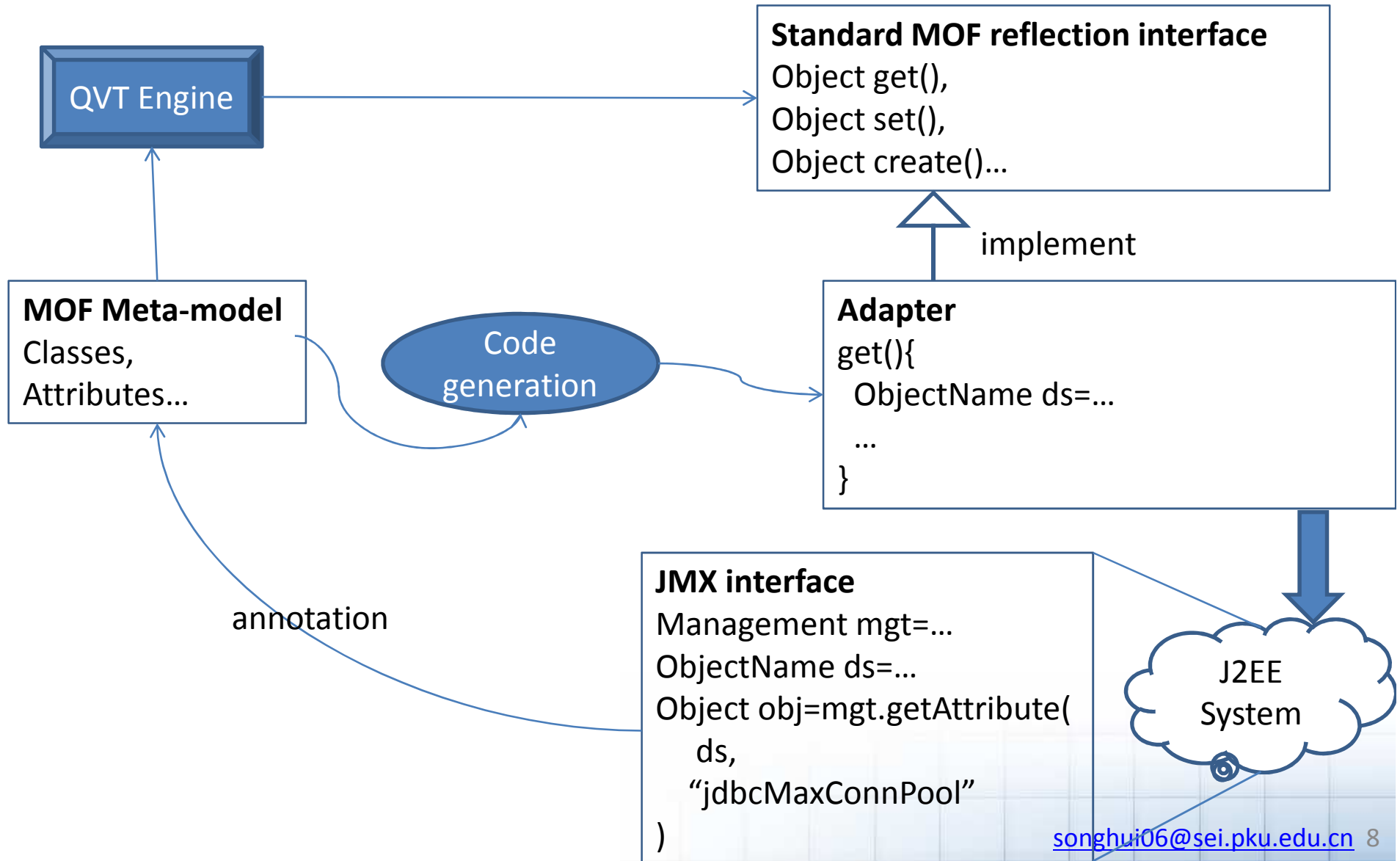
- QVT relational
 - Specifying relations
- QVT bi-transformation engine
 - Auto maintaining the relation
- Can we directly use QVT language? YES.
- Can we directly use QVT engine? NO!
 - Two obstacles
 - Corresponding solutions

Obstacle 1

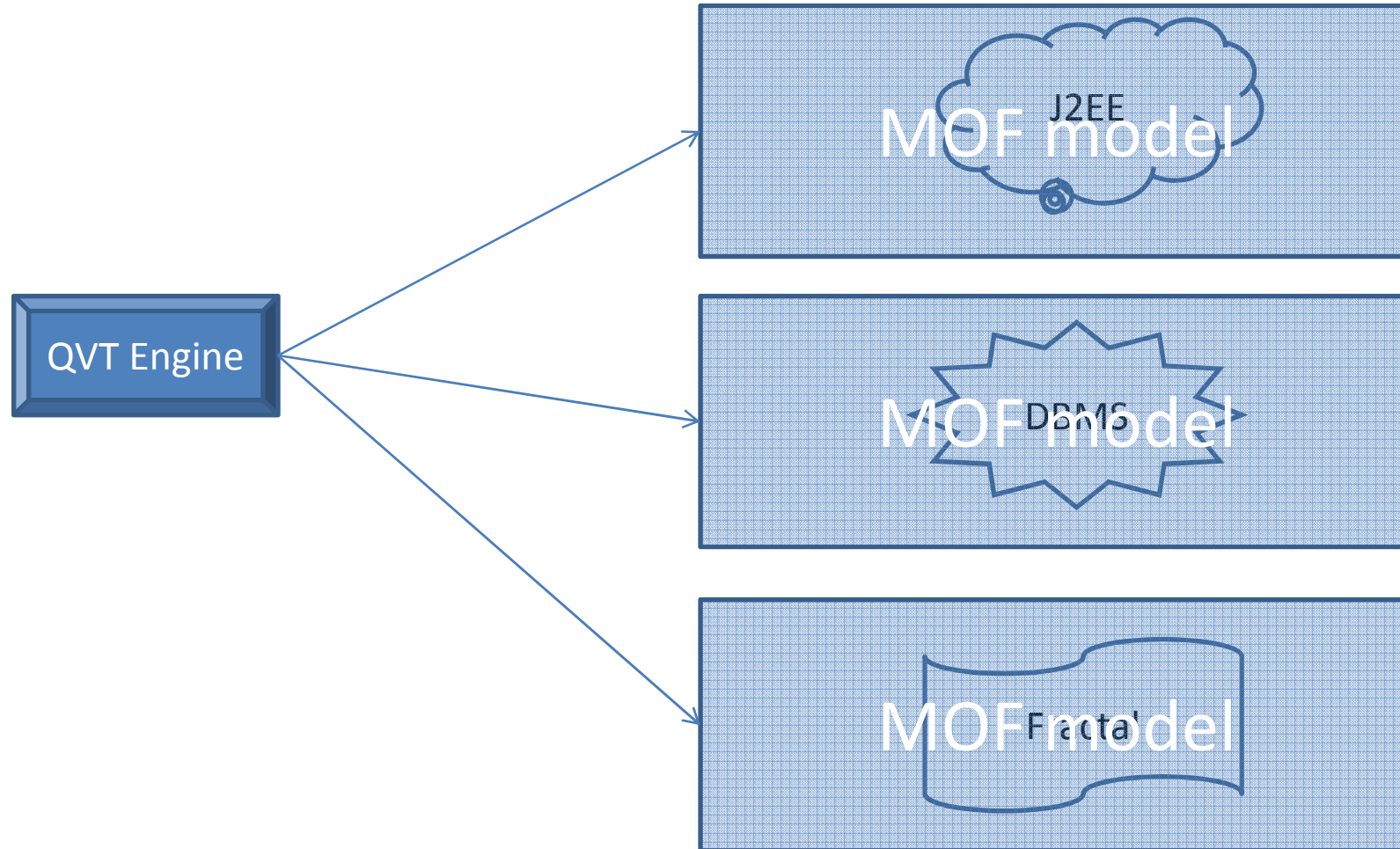


- Common Problem for Applying bi-transformation
 - Various data formats

Our Solution



What does that mean



Example

```
Package (name="JOnASJMX"){  
  ...  
  Class (name="JDBCDataSource"){  
    ...  
    Attribute (name="jdbcMaxConnPool", type=Integer, upperbound=1){  
      Annotation (source="..."){  
        "get"->{  
          $featureName=((Integer)$mainEntry  
            .getAttribute($core, "$feature")).intValue();  
        }  
      }  
    }  
    ...  
  }  
}
```

Meta-model

```
public class JOnASPackageImpl extends EPackageImpl...{  
  ...  
  public Management getMainEntry(){...}  
}  
public class JDBCDataSourceImpl  
  extends CommonWrappingEObjectImpl ...{  
  public int getJdbcMaxConnPool() {  
    jdbcMaxConnPool=((Integer)JOnASPackage.eINSTANCE.getMainEntry()  
      .getAttribute(getCore(), "jdbcMaxConnPool")).intValue();  
    return jdbcMaxConnPool;  
  }  
  public Object eGet(int featureID,boolean resolve,boolean coreType){  
    switch (featureID) {  
      case JOnASPackage.JDBC_DATA_SOURCE__JDBC_MAX_CONN_POOL:  
        return new Integer(getJdbcMaxConnPool());  
      ...  
    }  
  }  
}
```

Generated code

Specific
method

Standard
method

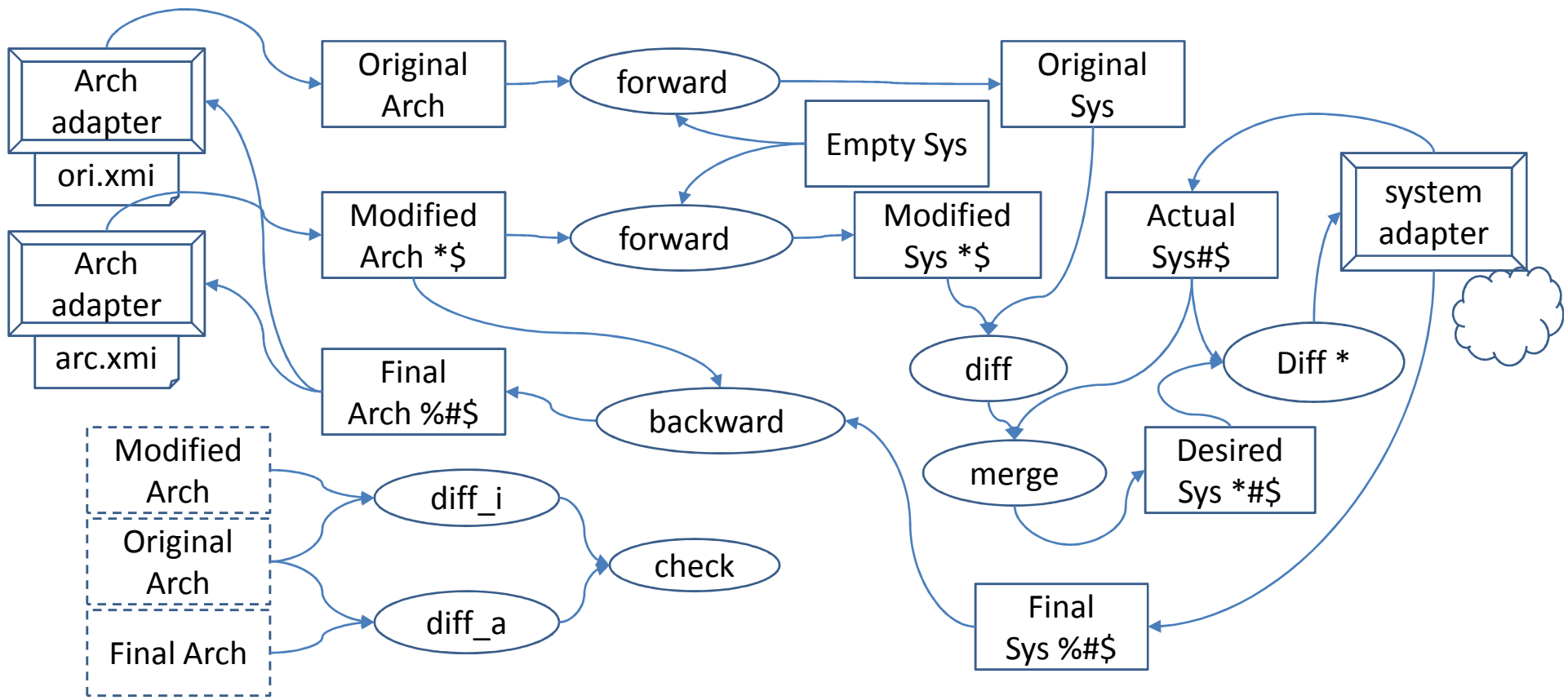
Obstacle 2

- Can we just execute bi-tr on these “models”
 - NO
- Simultaneous change
 - System may change itself
 - Multi-view
- Uncertain modification
 - Systems are always more complex for an outside observer!

Synchronization Algorithm



- Input: original and modified architecture
- Side-effect: change the system state
- Output: new architecture reflects
 - System changes (caused by itself, or other manager)
 - Actual modification result

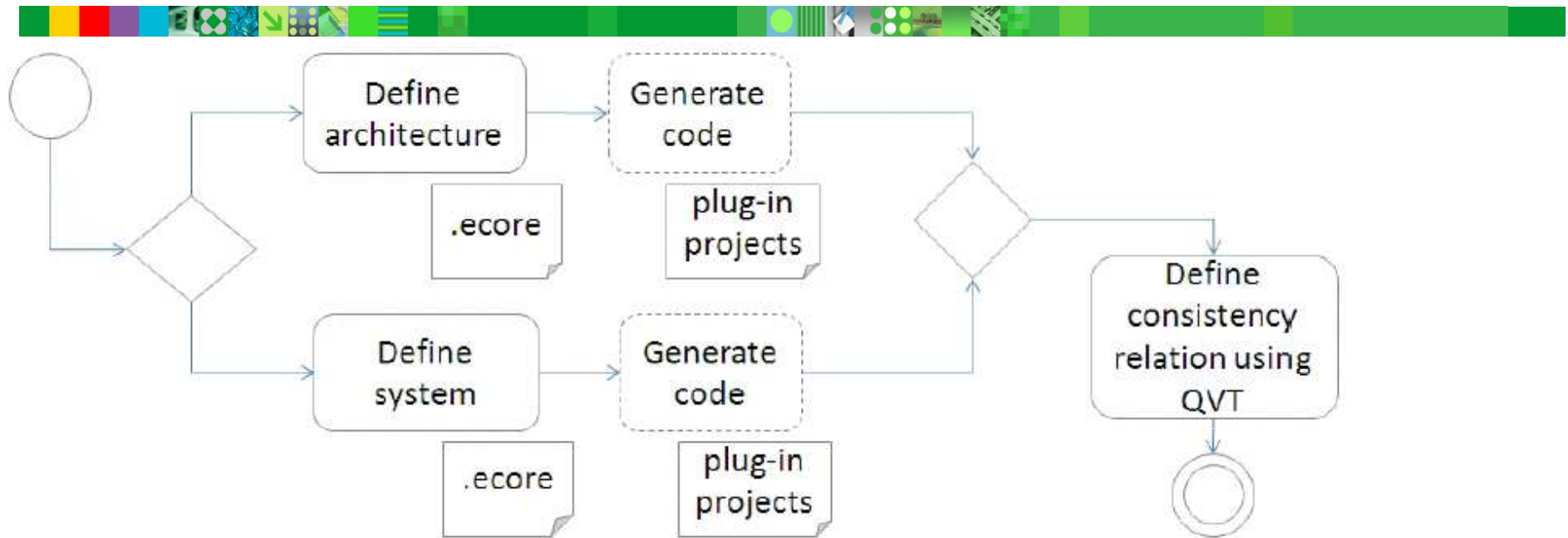


Case Study



- Supporting a typical management case
 - Target system: Java Pet Store on JOnAS server
 - Architecture style: C2
- Effect of the constructed infrastructure
 - Monitor
 - Reconfigure parameters
 - Add new component
 - Re-link components

Development under the Framework



	item	artifact	quantified workload	
Manual	define C2	Ecore model	29	model elements
	define JOnAS	Ecore model	61	model elements
	add annotation	Java code	474	LOC
	define relation	QVT text	207	LOC
Fra. Gen.	for arc	Java code	8761	LOC
	for sys	Java code	18263	LOC
Fra.	common meta-model	Ecore model	17	model elements
	Synch. and utility	Java code	2130	LOC*
*exclude reused libraries and plug-ins				

Advantages of the Framework



- Productive
- Clear and stable behavior, good for maintenance and incremental development
- Separation of concerns, good for reuse

Open Problems

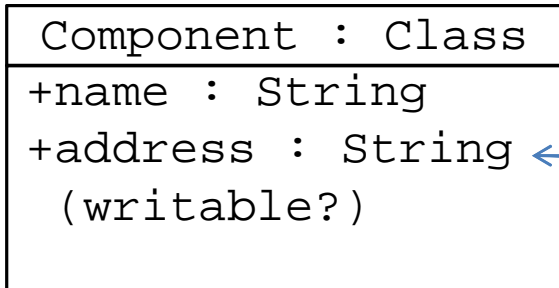


- Foundation
 - Properties of architecture-based management
- Performance
 - Important for automatic management agents
 - Incremental transformation
- Representation of modification
- Static check
 - What can be changed on architecture
 - One architecture style for different systems

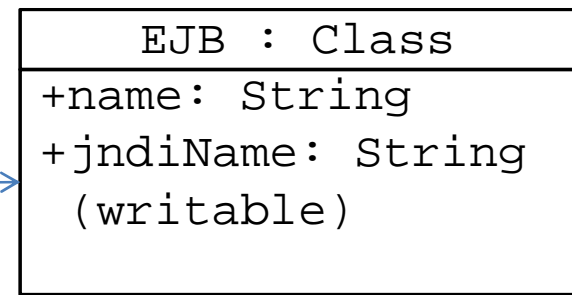
For example



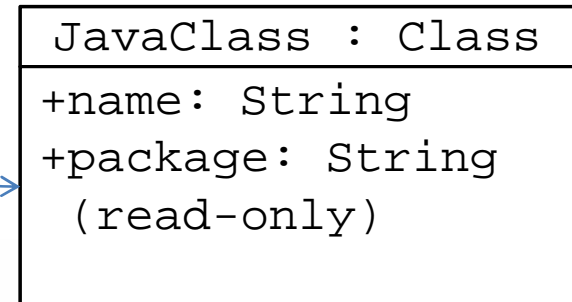
Meta-model for C2 style



JOnAS JMX & Ishmeal



java.lang.reflect



- Derive access right on architecture level from system level and QVT relation

Conclusion



- Architecture-based runtime management
- A framework for architecture-based runtime management, supported by bi-transformation
 - Wrapping various data for bi-transformation
 - A model-to-runtime synchronization algorithm
- Case study
- Open questions

Thank you!

