Current Trends and Perspectives in Modeling

Between semantic technologies and model reuse

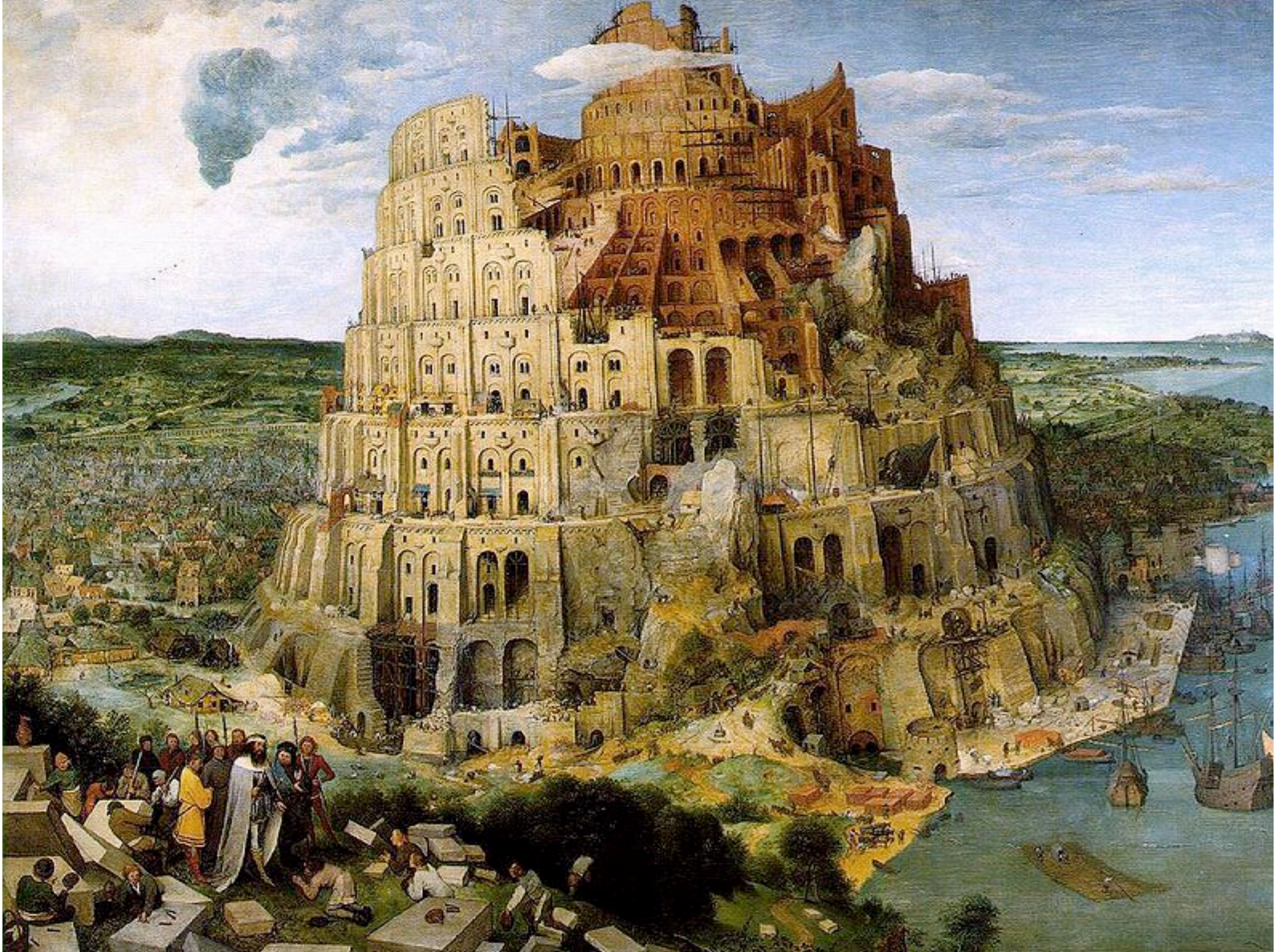
Modellierung Betrieblicher Informationssysteme (MOBIS)

Invited Talk

Prof. Dr. U. Aßmann, J. Johannes, M. Seifert, R. Samlaus
Technische Universität Dresden, Lehrstuhl Softwaretechnologie

16. Sept. 2010

Version 1.0



Different Technological Spaces

Cuneiform Sumarian pictographic language (2800 bC, protowriting)

Ugarit phonetic alphabet (1500 bC, writing)

Schøyen Collection MS 3029.
USA-PD https://en.wikipedia.org/wiki/File:Sumerian_26th_c_Adab.jpg

PD https://upload.wikimedia.org/wikipedia/commons/0/04/22_alphabet.jpg

Current Trends

- 1) Model-Driven Integration of Technological Spaces
  - Bridging the technological spaces of system modeling and ontologies
  - Semantic-oriented modeling
- 2) Model Reuse
  - Model components
  - Model SOC
- 3) Model Synchronization
  - Connecting business and IT-level

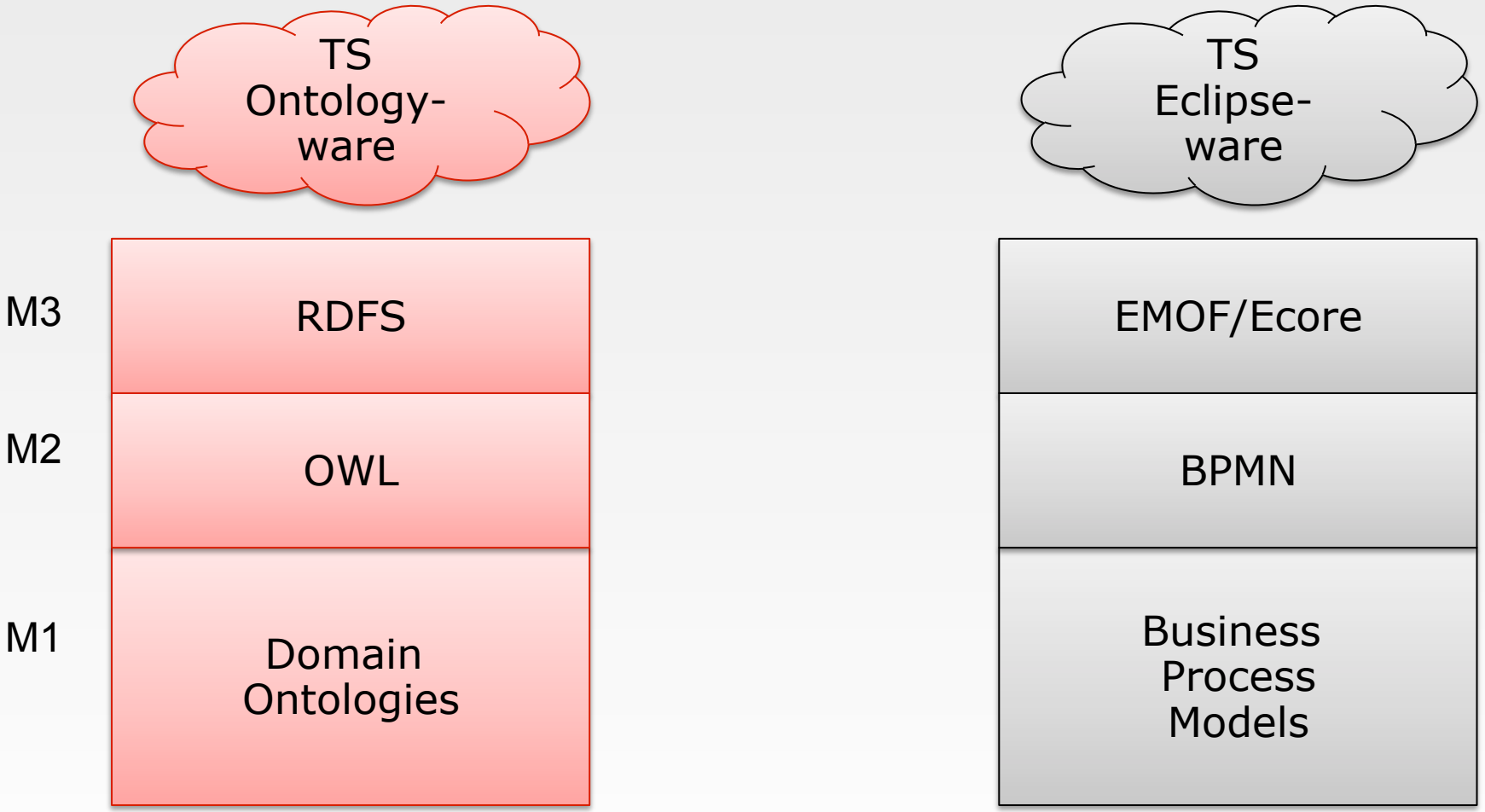
1) Model-Driven Integration of Technical Spaces (MDI)

Syntactic and Semantic Modeling

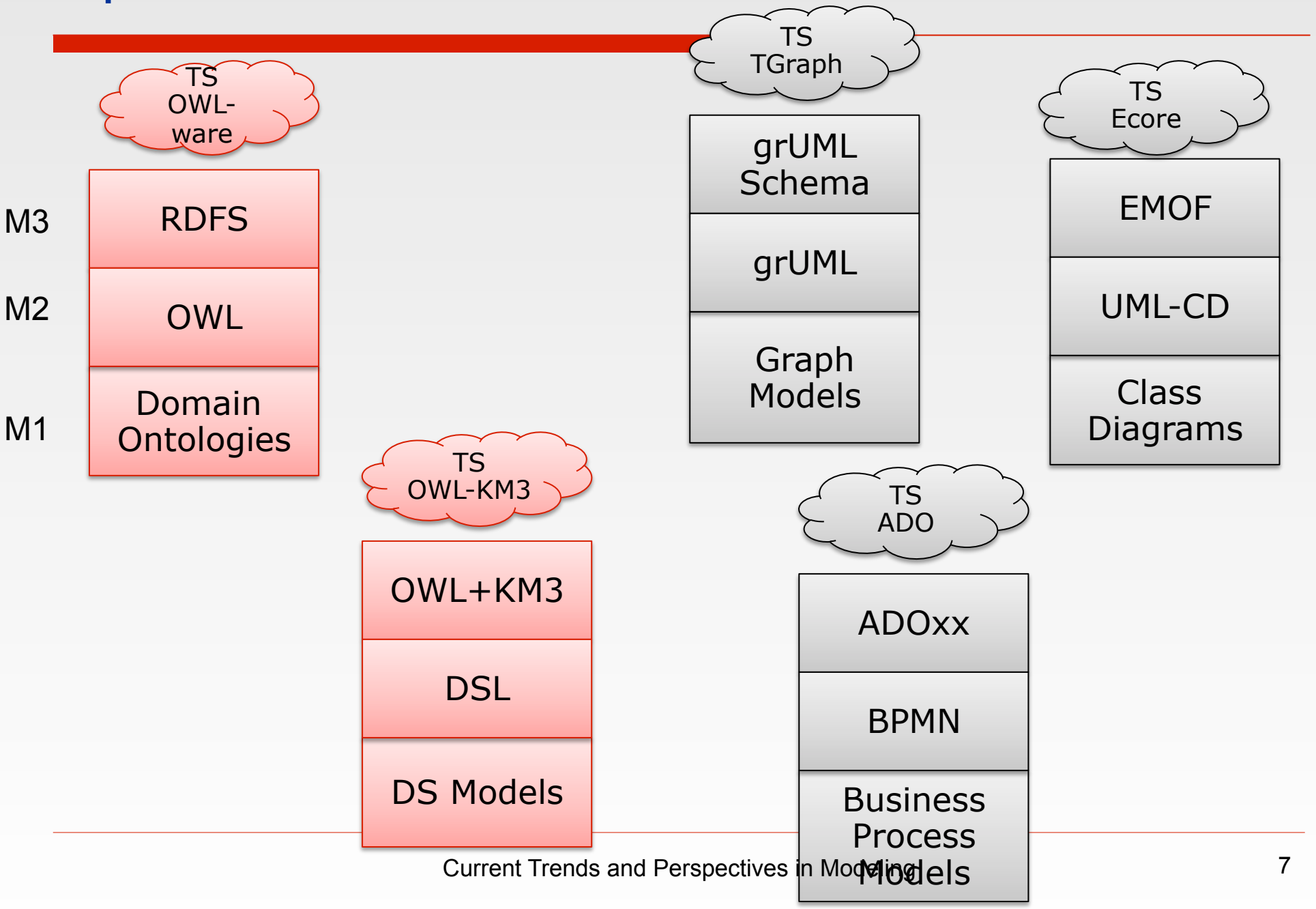
First MDI workshop: http://mdi2010.lcc.uma.es/

Why „Integration“?

- Heterogeneous Technological Spaces (TS)

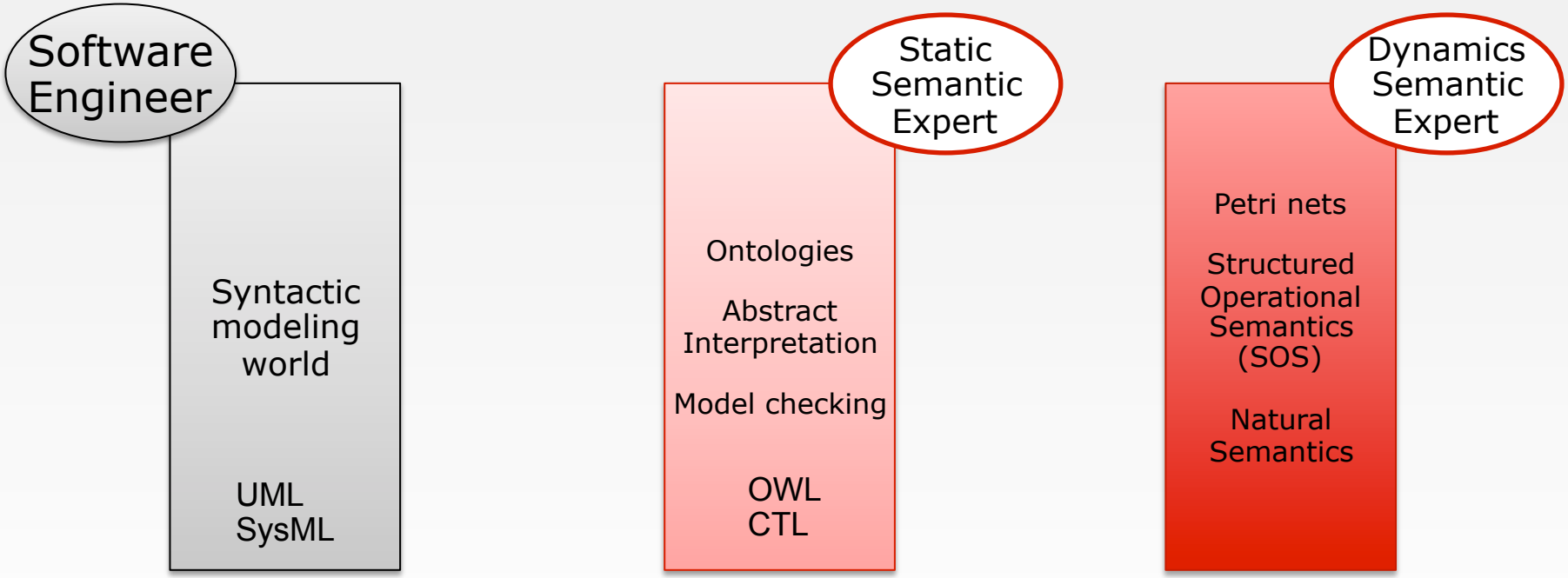


Problem: Many Technological Spaces Exist!



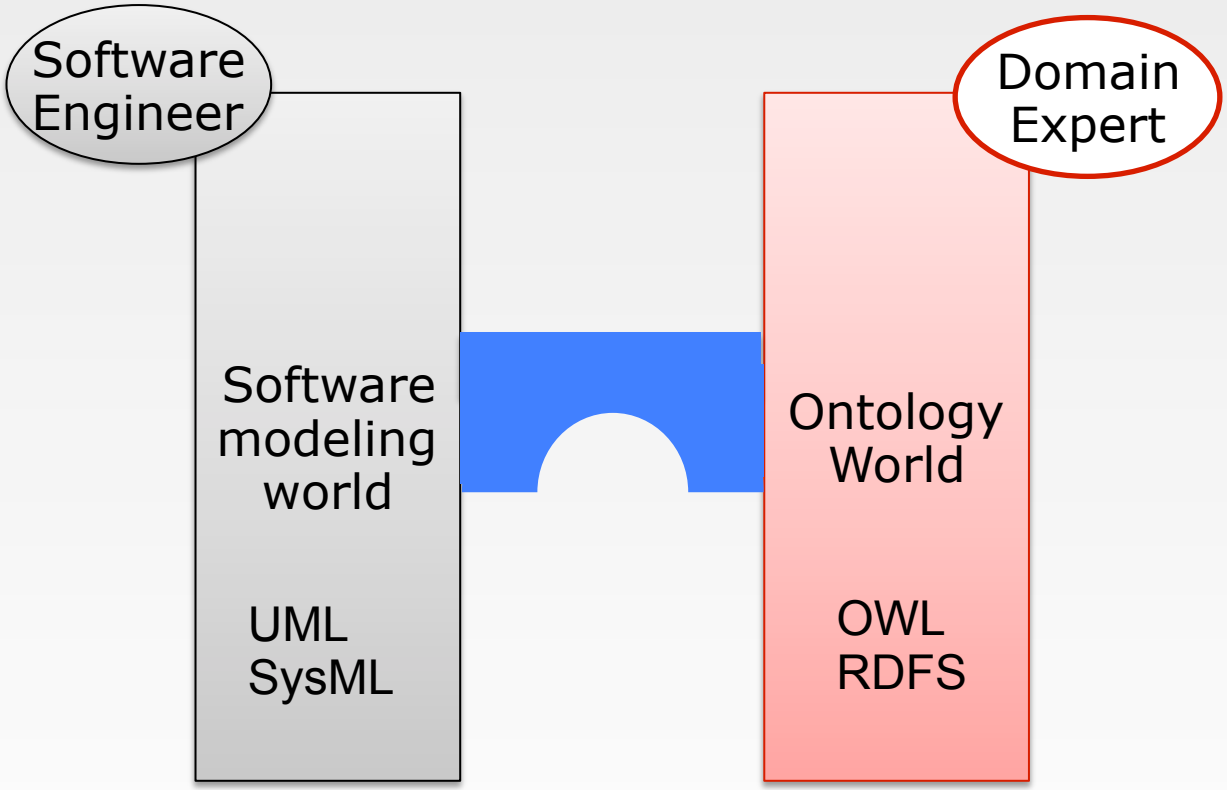
Reason: "Syntactic" and "Semantic" Modeling

- In the UML world, syntactic modeling is dominant (structural models)
  - Structural modeling is needed most, semantics can wait..
  - Different forms of static semantics are done in OWL and other TS
- Dynamic semantics?

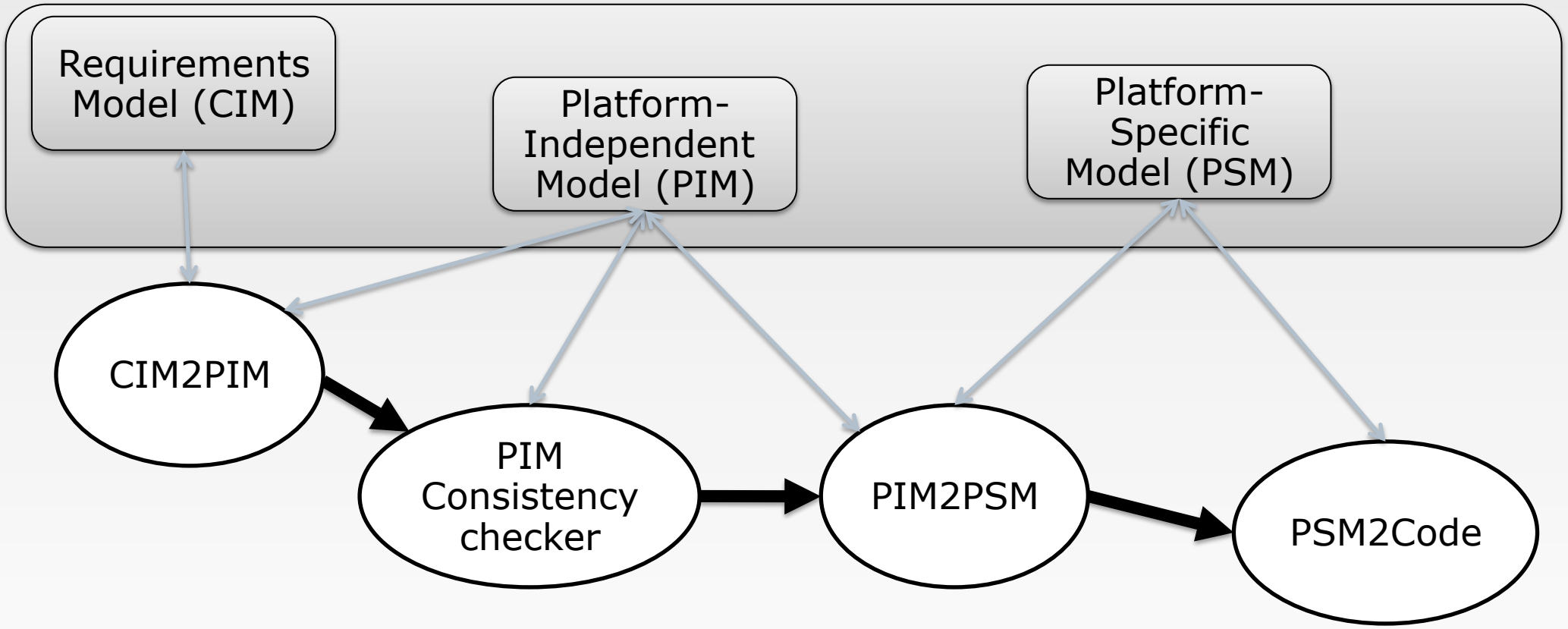


Ex. Marrying Ontologies and Software Technology

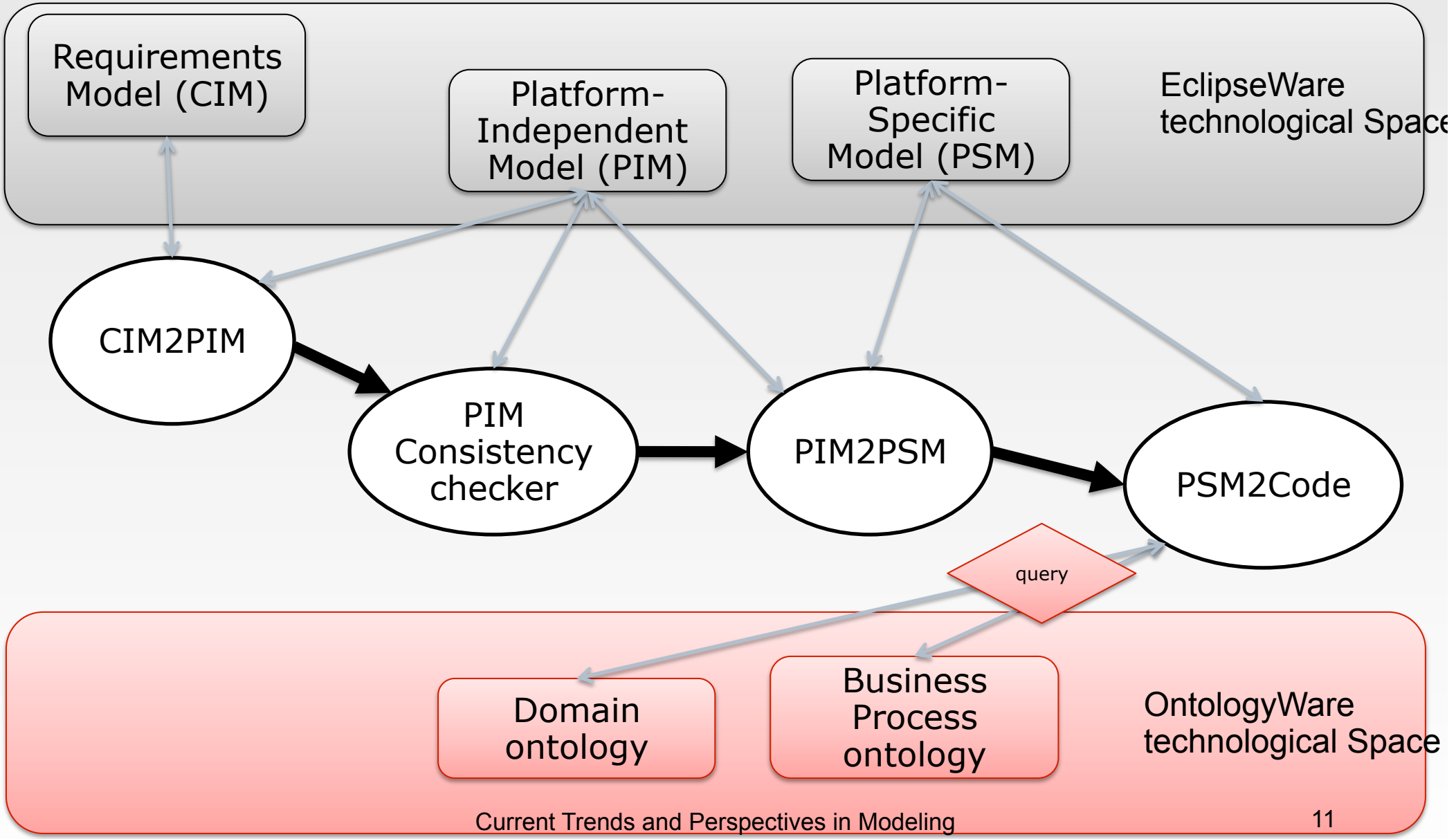
- Bring together domain knowledge and software know-how
- www.most-project.eu



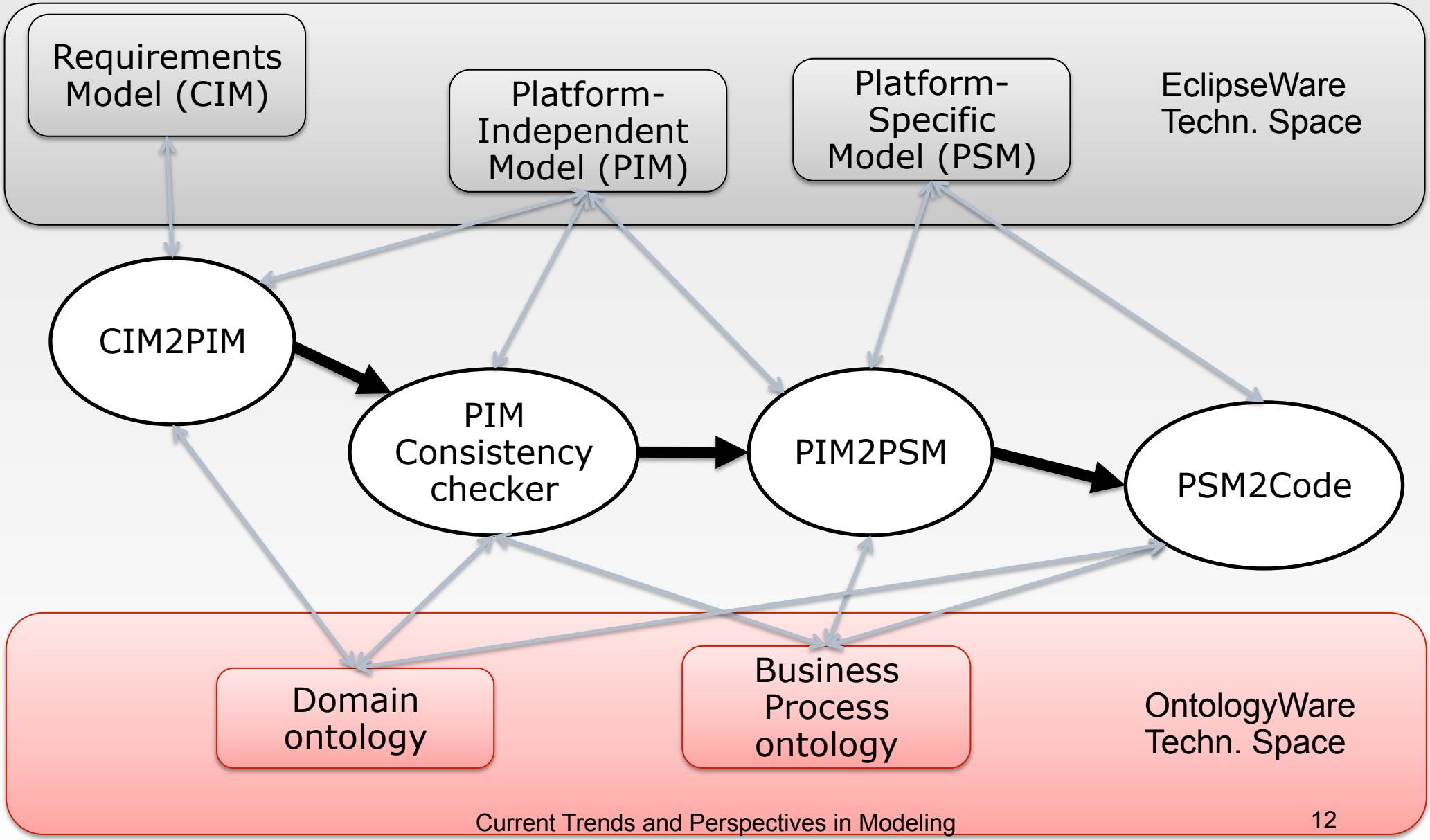
State of the Art Syntactic Modeling: Model-Driven Software Development (MDSD)



State of the Art: Ontology Querying



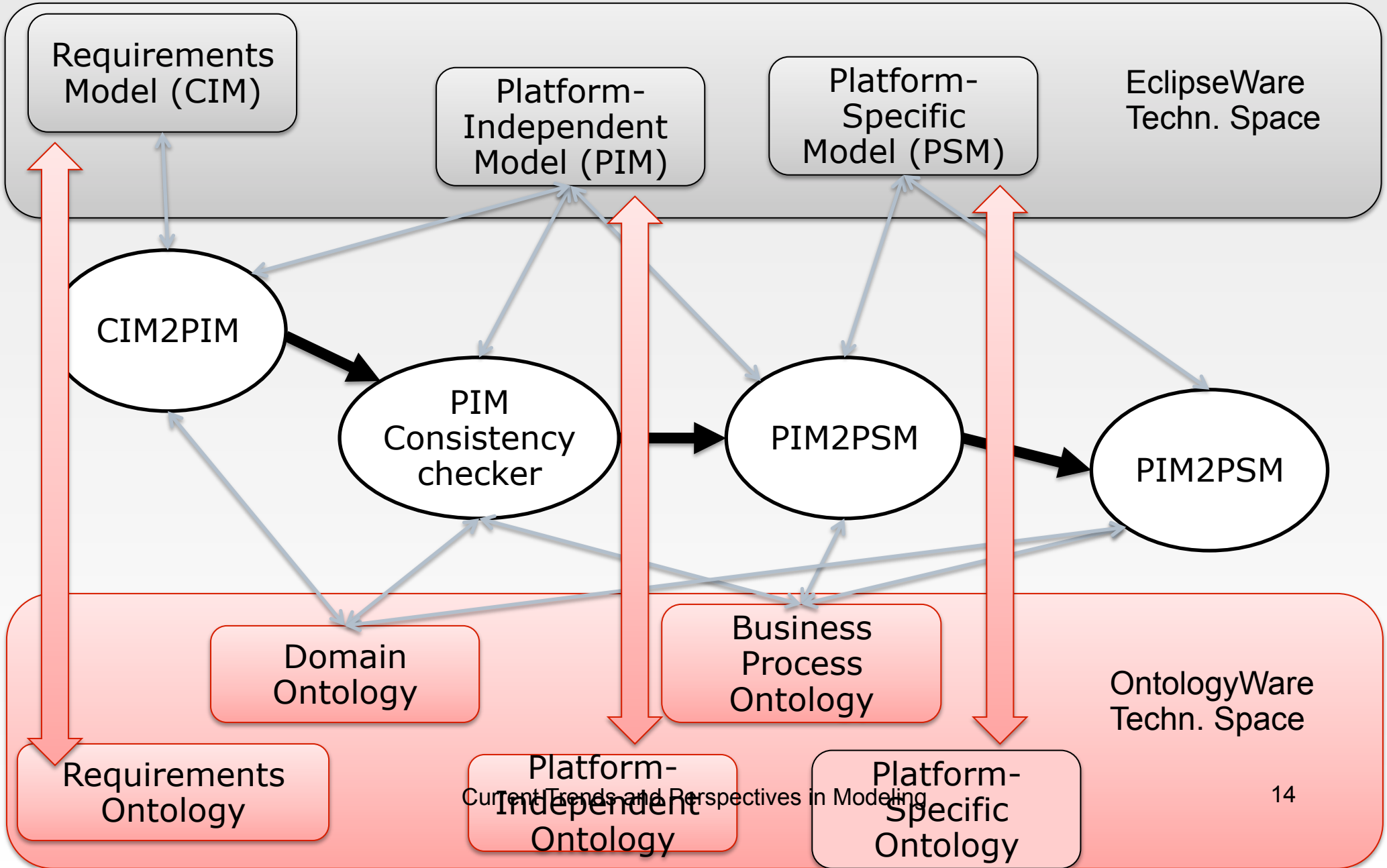
What are „Integrated Ontology Services/Querying“?



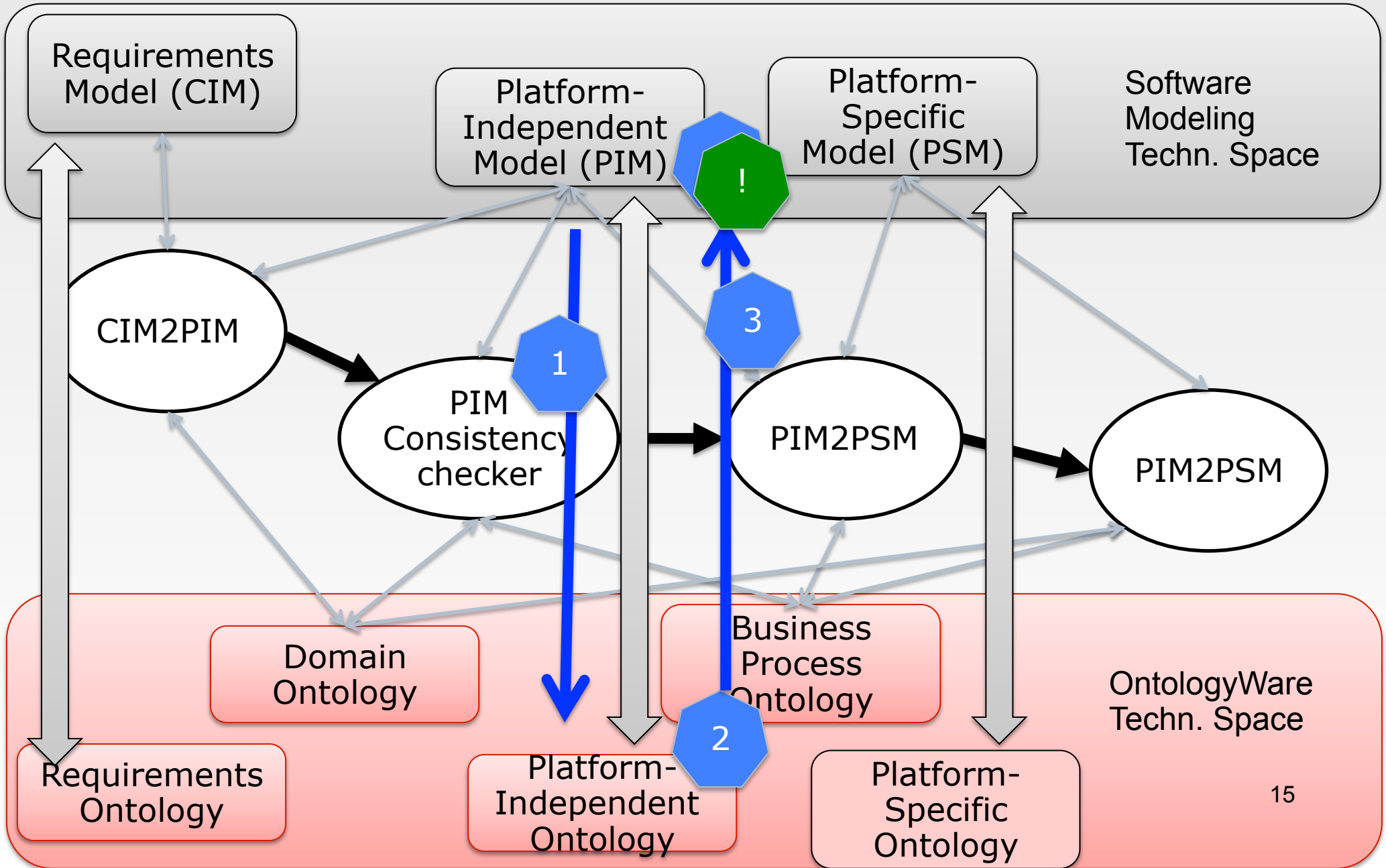
Services Provided by Ontology Technology

- Precise documentation of services provided by ontology technology
- http://www.most-project.eu/admin/xinha/plugins/ExtendedFileManager/images/Deliverables/MOST_D2.5_1_Final.pdf
  - Classification
  - Consistency Checking
  - Explanation
  - Consistency Guidance (Checking with Explanation)
  - Merging
  - Querying
  - Satisfiability
  - Subsumption (with Explanation)
  - Forget

Ontology-Driven Software Development (ODSD)

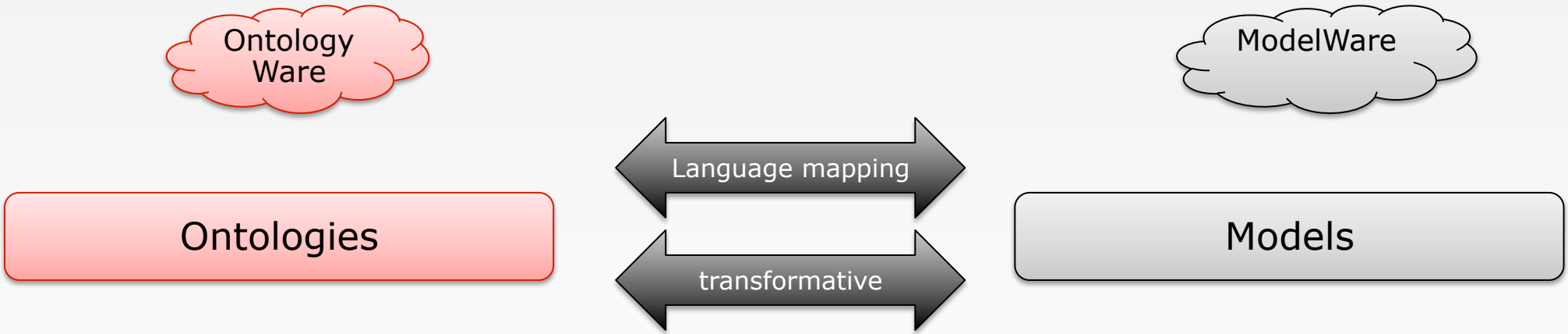


Transformation Bridges enable „Model Transport“ between Techn. Spaces



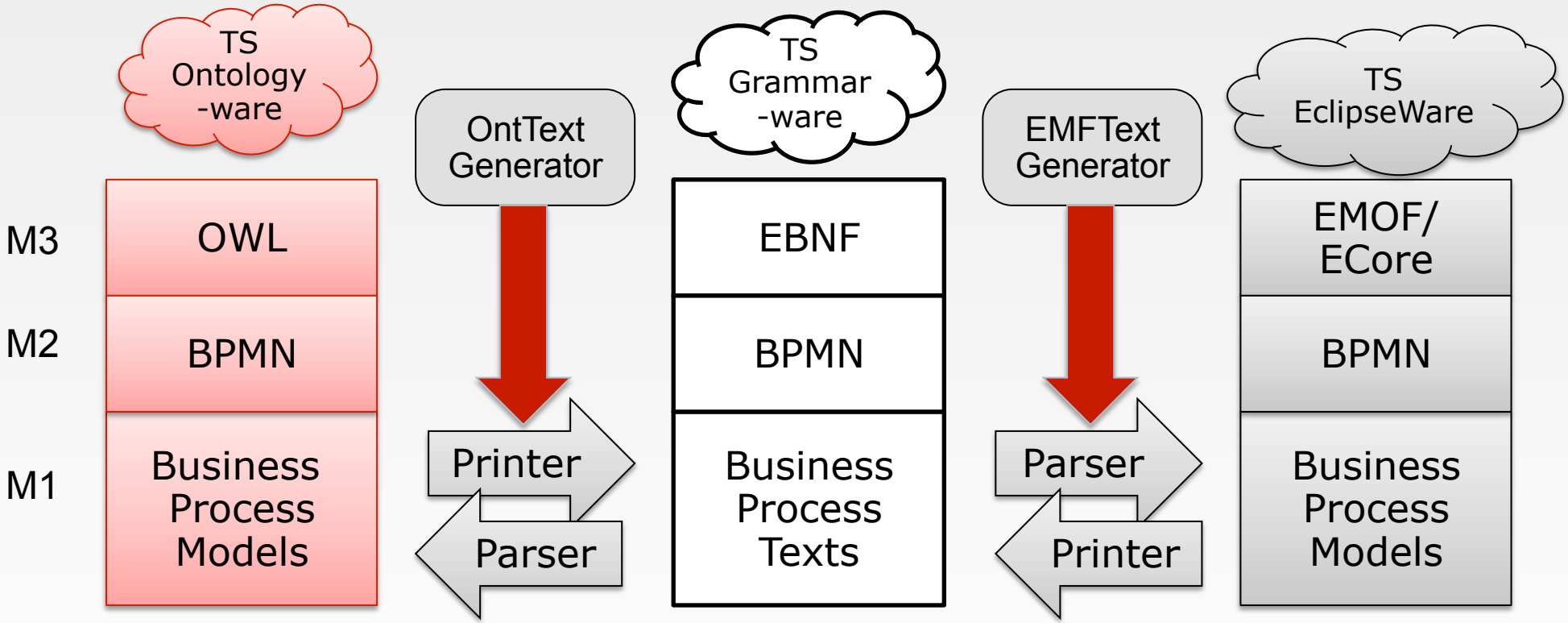
Bridging Technology for Integrated Ontology Services in ODSD

- Modeling bridges
  - Language bridge (between metamodels on M2)
  - Transformation bridge (physical transport to the other space on M1)



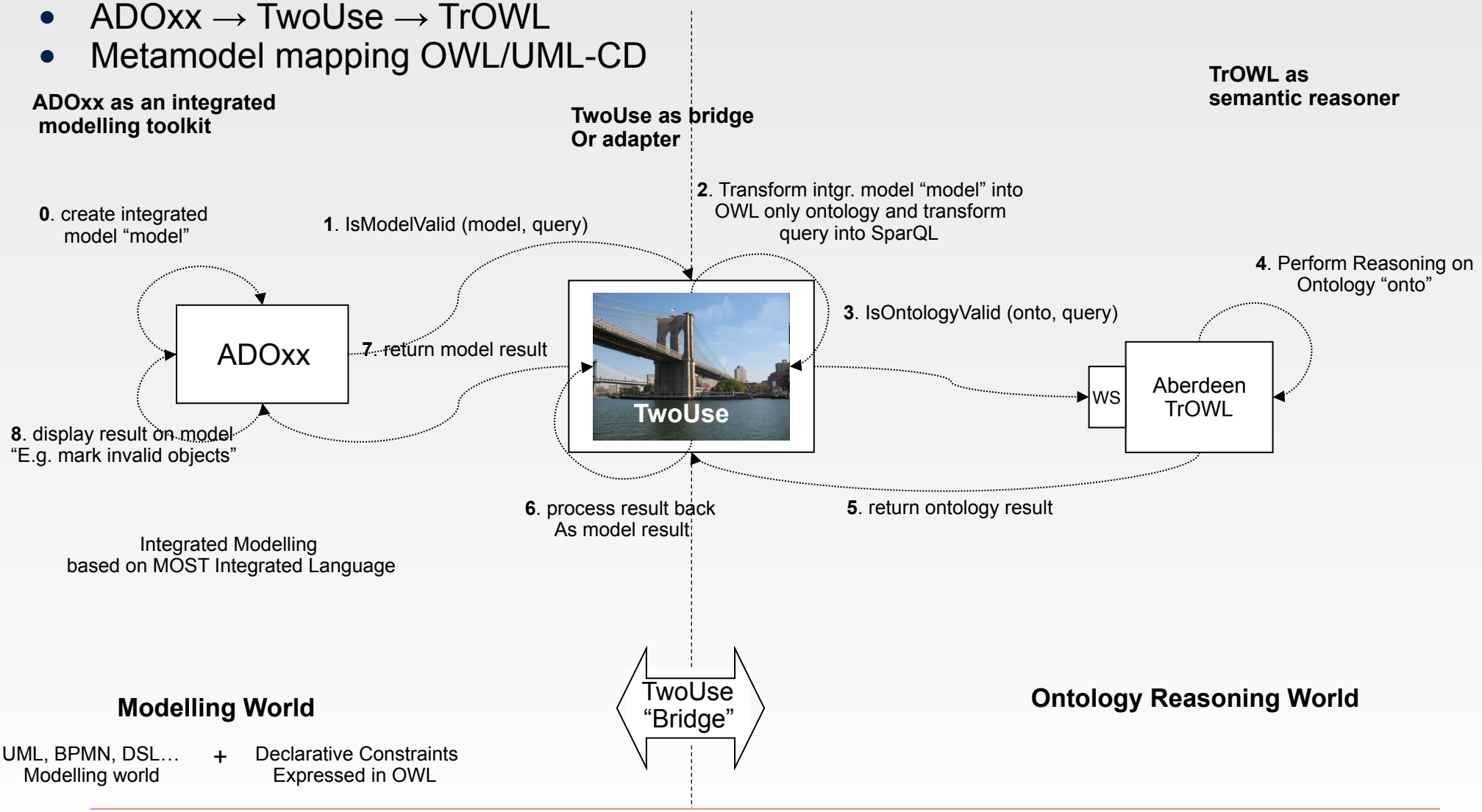
Transformation Bridges

- A **transformation bridge** moves all models from the software modeling tool to the reasoner
  - Generated from a metamodel mapping, e.g., TwoUse



Example: TwoUse-Based Transformation Bridge

- ADOxx → TwoUse → TrOWL
- Metamodel mapping OWL/UML-CD



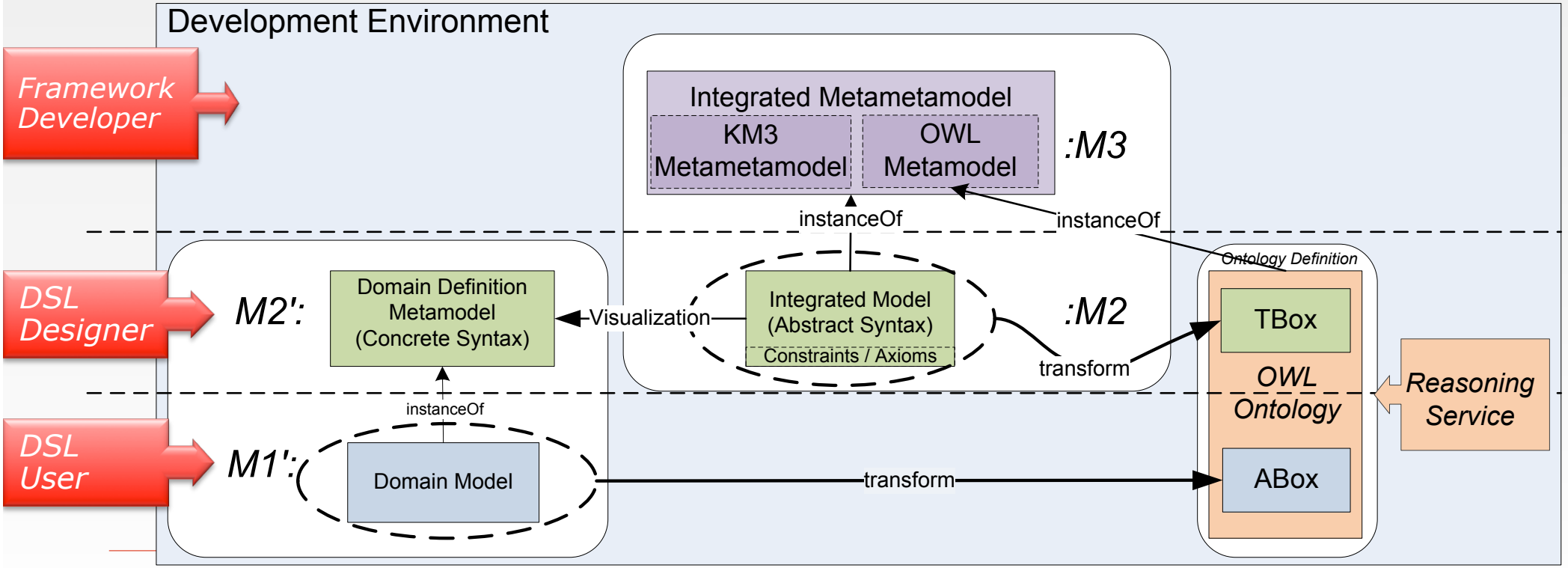
Courtesy F. Parreiras, T. Walter, S. Staab, Uni Koblenz

Ontology-Integrated Modeling

- Tight integration enables *Ontology-Integrated Modeling* with Ontologies and Software Models
- In the same specification:
  - Syntactic structure
  - Semantic constraints
- See Reasoning Web. Semantic Technologies for Software Engineering, 6th International Summer School 2010, Dresden, Germany, August 30 - September 3, 2010. Tutorial Lectures, LNCS 6325, Springer

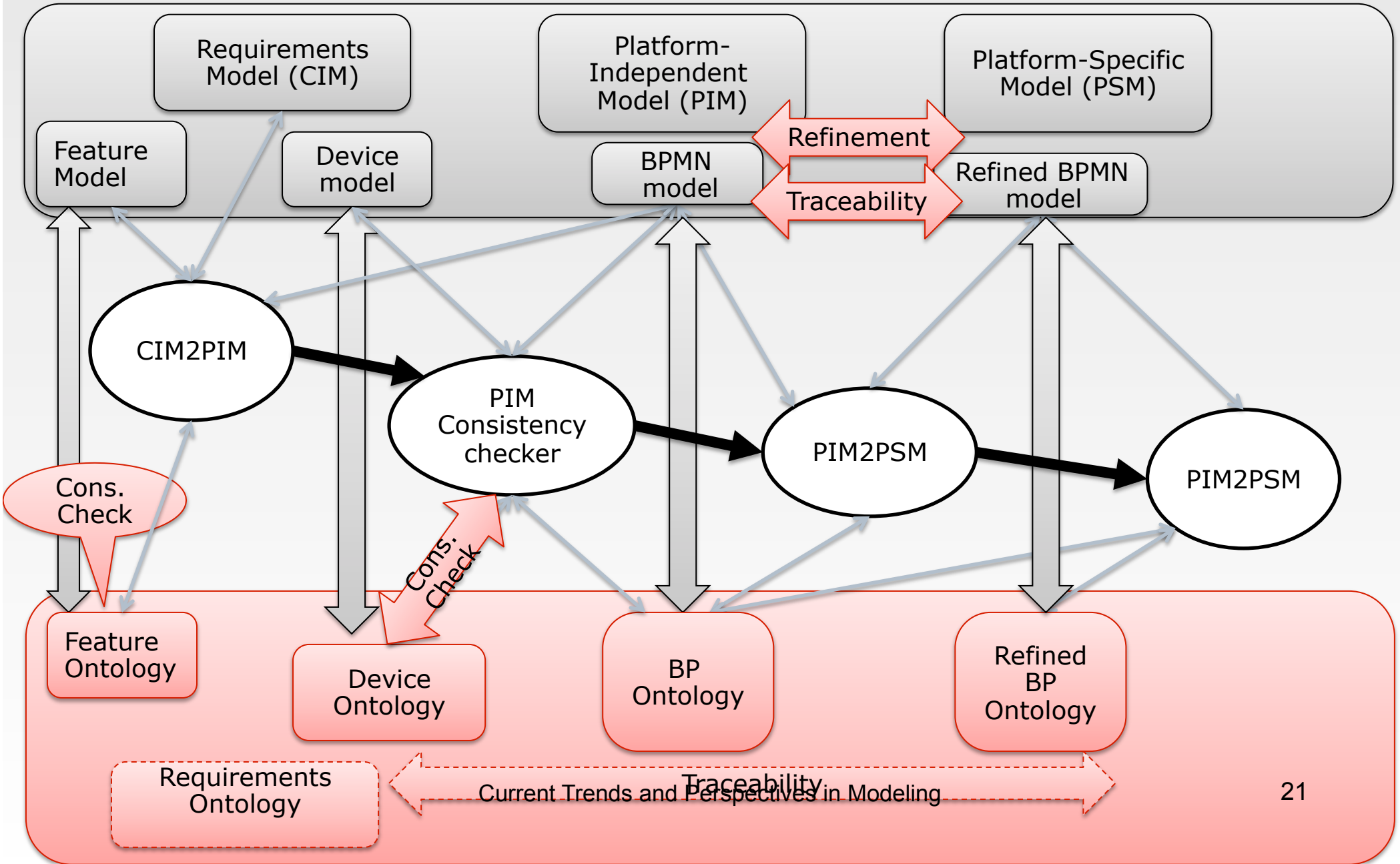
OntoDSL: DSL Modeling with Transformation Bridge

- [picture from Universität Koblenz/Landau, Prof. Staab]
- OntoDSL development environment for DSL for integrated modeling (KM3+OWL)
- Transformation Bridge transports the integrated specifications to ontology space



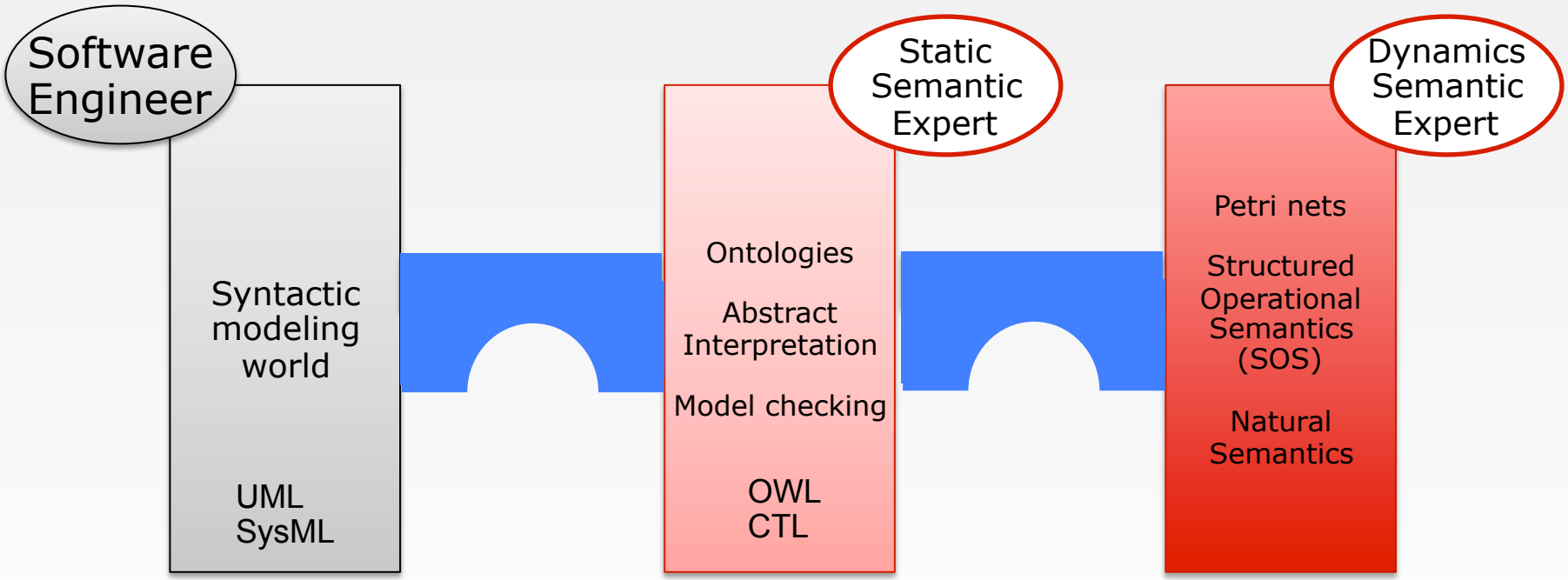
Courtesy T. Walter, S. Staab, Uni Koblenz

Accomplished ODSD Scenarios in MOST



The Future of "Syntactic" and "Semantic" Modeling

- Standalone syntactical TS will remain
- How to bridge them to several semantic technological spaces?



2) Model Reuse and Evolution

Model components for Model Reuse

Model SOC

.. And the tool REUSEWARE (www.reuseware.org)

Model Components – Home

http://www.modelcomponents.com.au/

Google Scholar

Most Visited ST ST Wiki ST-Gruppe Google Google Scholar Google Docs WMedia bibs Software Platforms a... 7.FP NKS Citeseer Doodle Search Courses

Local News

PAG-R... AbsInt:... Lehrst... Find 42 Build d... model ... Existen... Model ...

MCA Model Components (Aust) Pty Ltd

HOME ABOUT US OUR ONLINE-SHOP COMPANY TERMS CONTACT US

Model Components offer a range of Australian produced, model boilers, burner systems and steam engines.

Model Components are an appointed Australian agent for P.M. Research products and the Roundhouse Model Steam Locomotives for SM32/45 & G Scale Garden Railways, we are also an appointed Qld agent for the full product range of Miniature Steam Pty Ltd. We manufacture a range of Model Boilers, Gas Burners and accessories. MCA will also custom build a model boiler and engine combination to suit customer requirements provided it can meet the relevant safety code requirements. We offer a selection of quality hand tools, Taps and Dies and accessories. If you require something not listed please contact us as we may be able to procure it for you through our network of suppliers.

SEARCH

Product Search

SEARCH

ADVANCED SEARCH

NEW PRODUCTS & SPECIALS

Made in Australia by Miniature Steam, these kits feature lost wax castings in Gunmetal and a crankshaft cast from Speroidal Graphite Cast Iron which is very ductile.

> see more detail

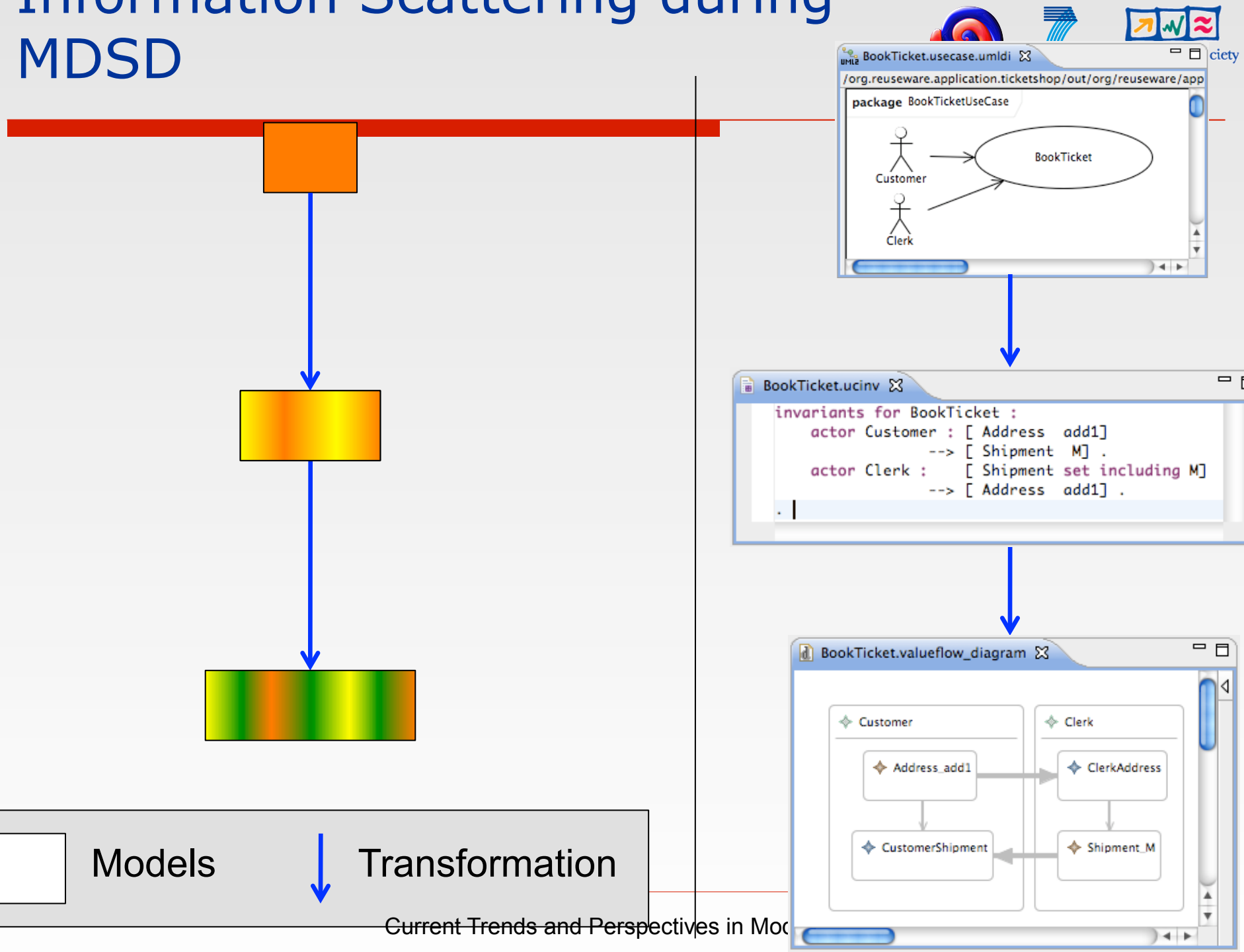
CONTACT US

If you require something not listed please contact us as we may be able to procure it for you through our network of suppliers.

> see more detail

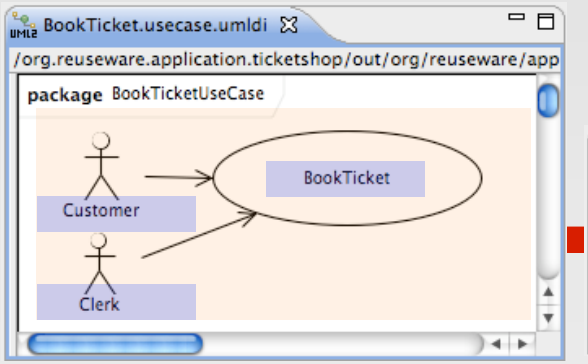
Fertig

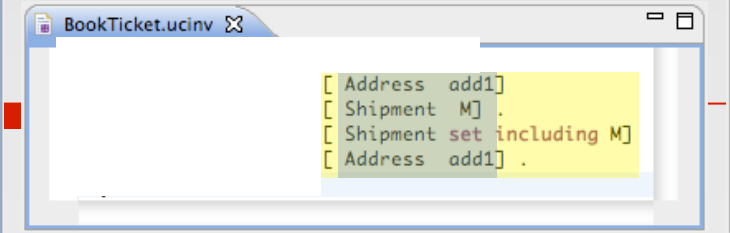
Information Scattering during MDSD

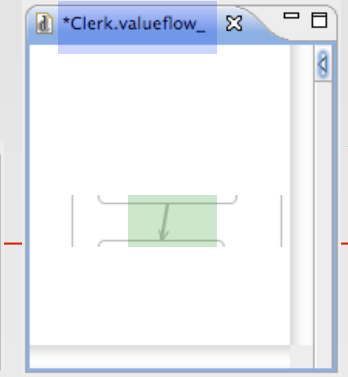


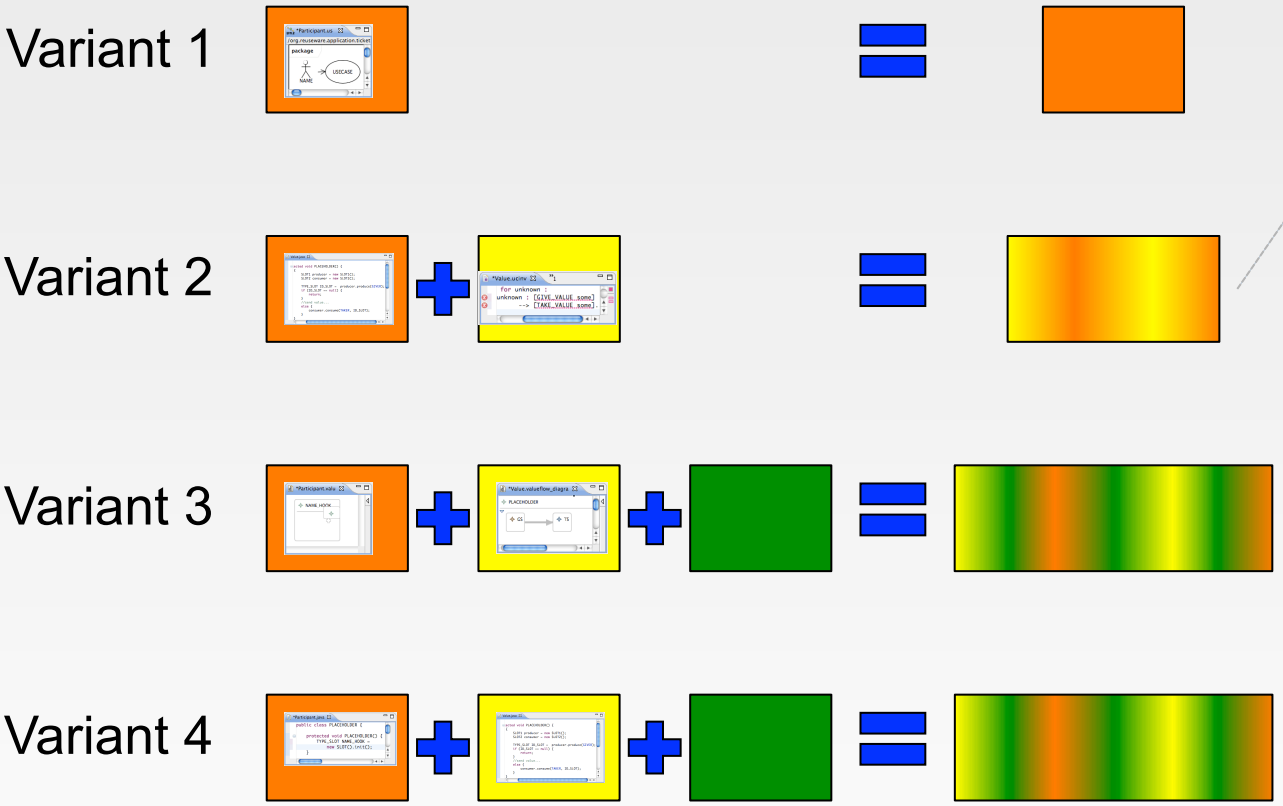
Courtesy J. Johannes, 2010

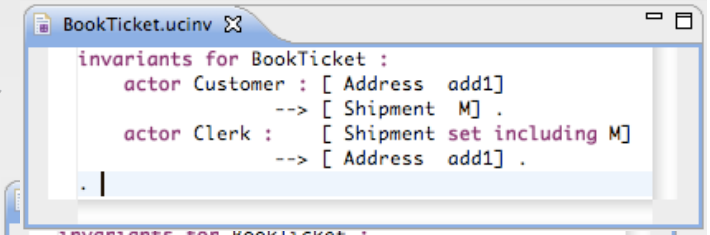
Information Scattering and Duplication

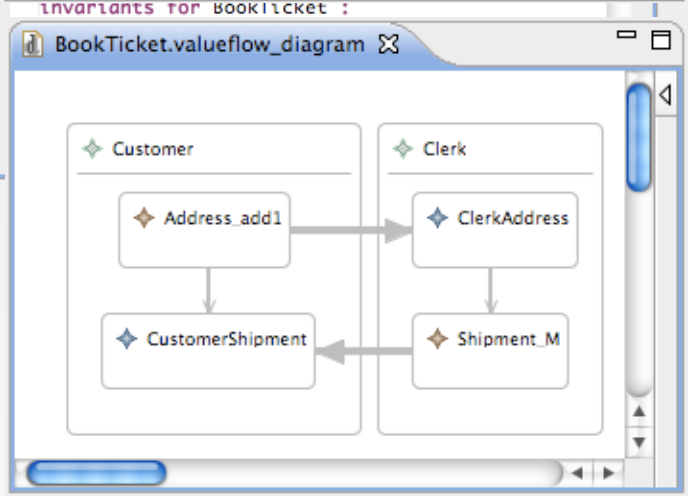


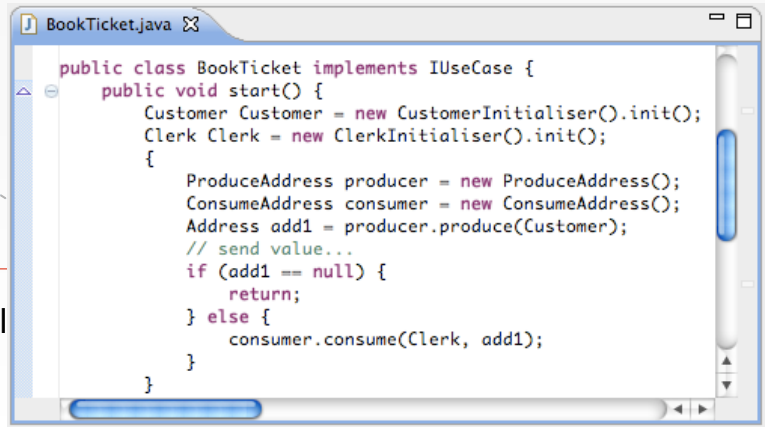








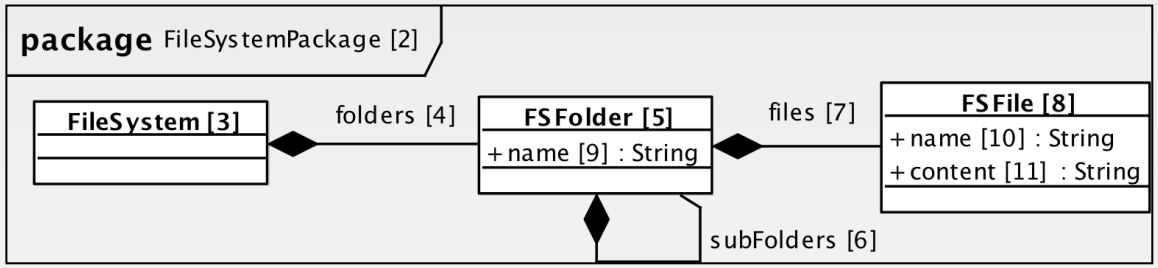


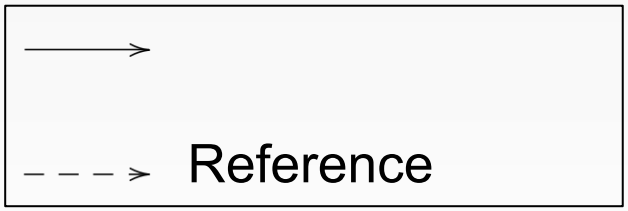


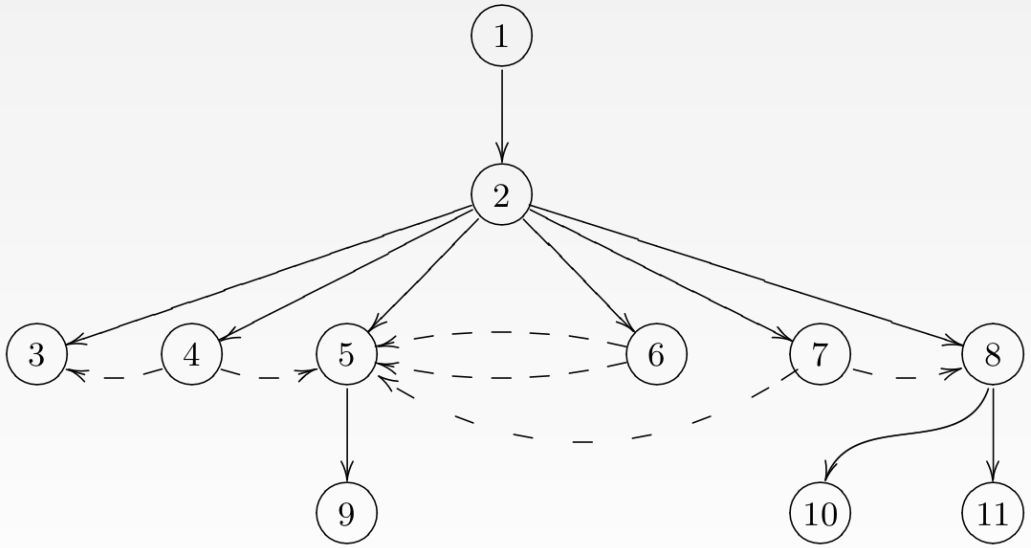
Courtesy J. Johannes, 2010

Model Components Made from Graph Fragments

- Graph Fragment
  - Graph that conforms to an type graph (EMOF metamodel or similar)
- Model component
  - Typed in two dimensions
    - By type graph
    - By variability typing



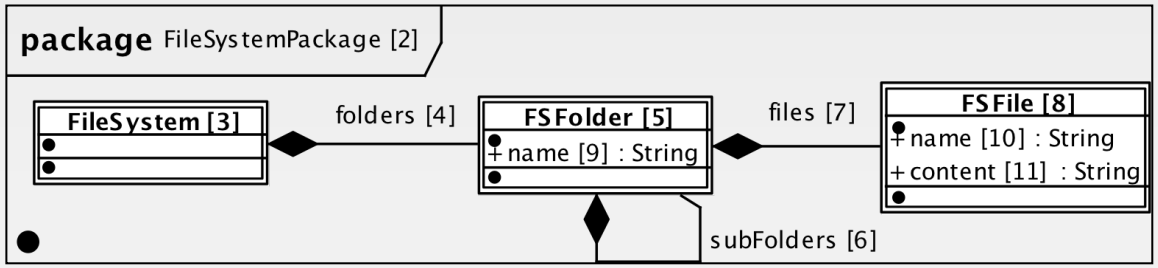


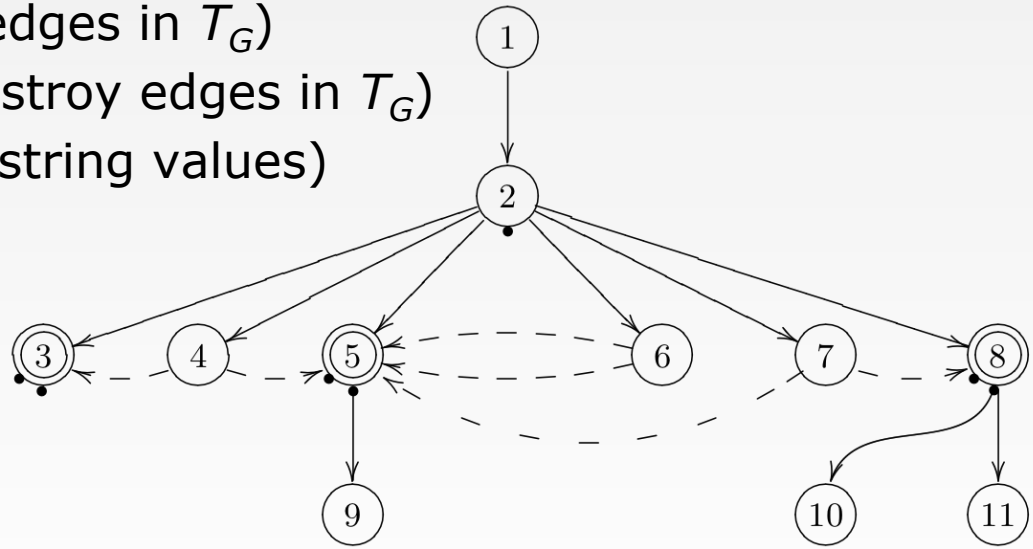


Courtesy J. Johannes, 2010

Variability Types of Model Components

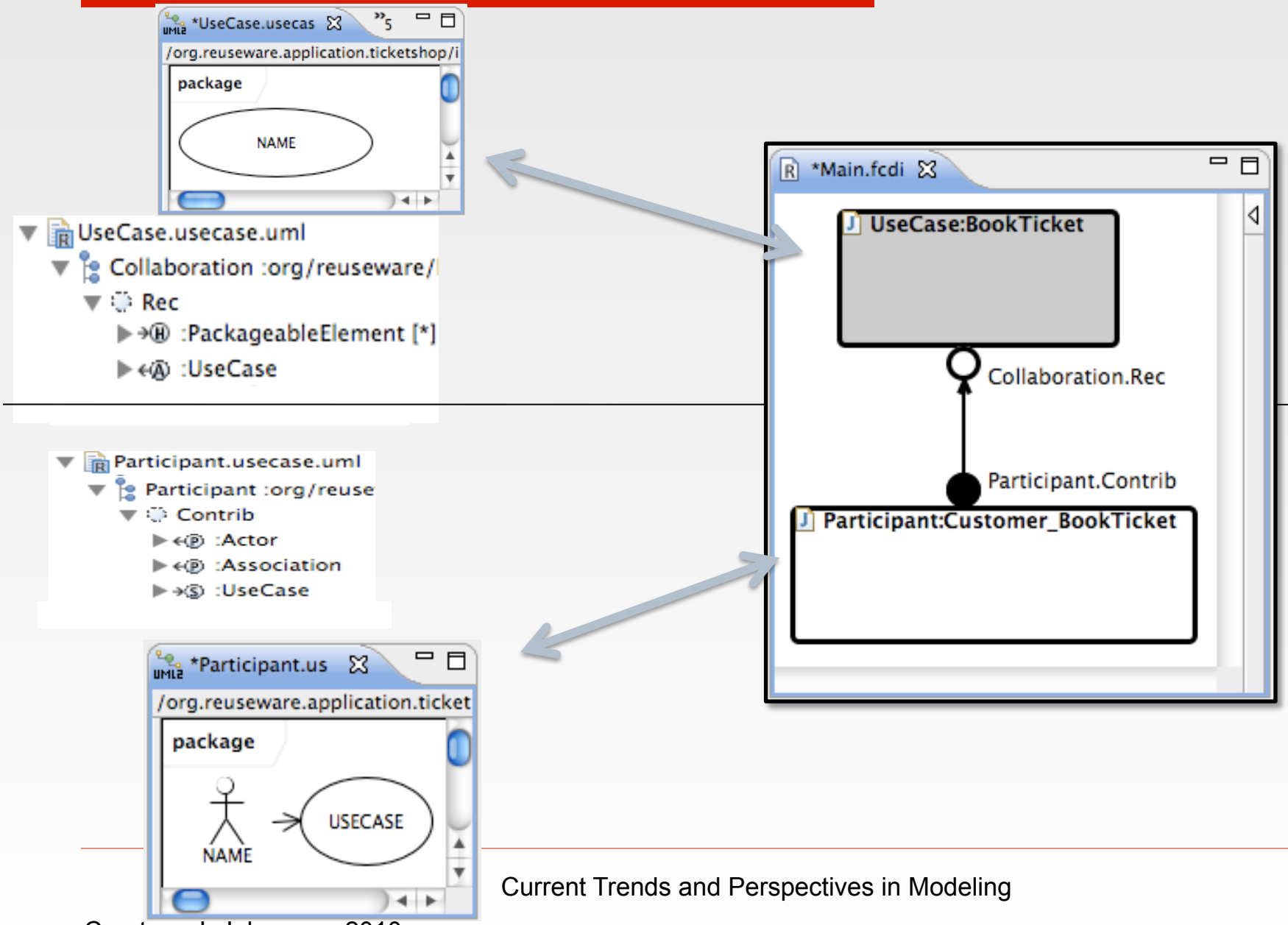
- Variability Types
  - Variation Points
    1. Hooks (nodes, edge lists for edges in T_G)
    2. Slots (nodes, edge lists for edges NOT in T_G)
    3. Value Hooks (attributes)
  - Reference Points
    1. Prototypes (nodes - destroys edges in T_G)
    2. Anchors (nodes - does NOT destroy edges in T_G)
    3. Value Prototypes (attributes - string values)





Courtesy J. Johannes, 2010

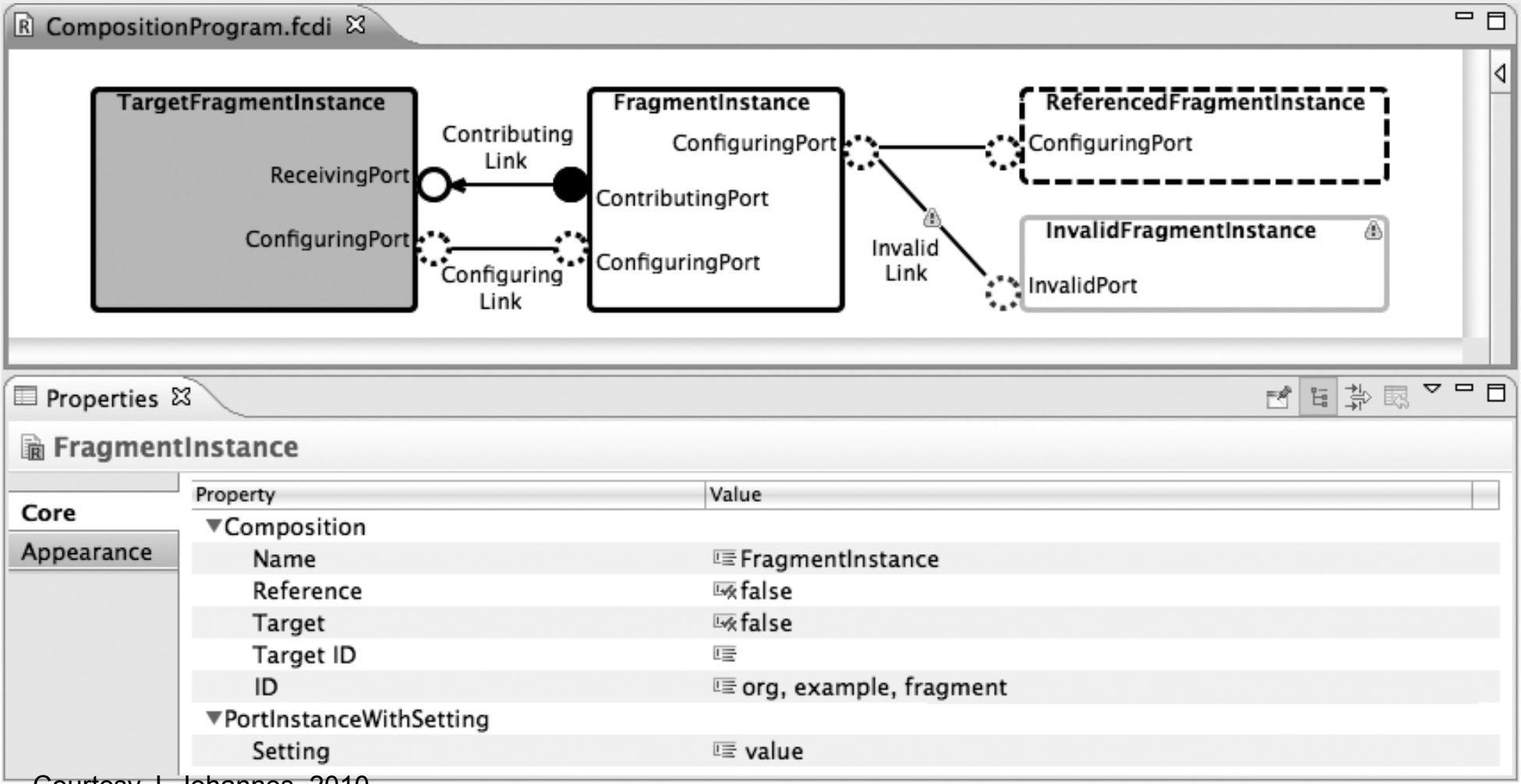
Model Components



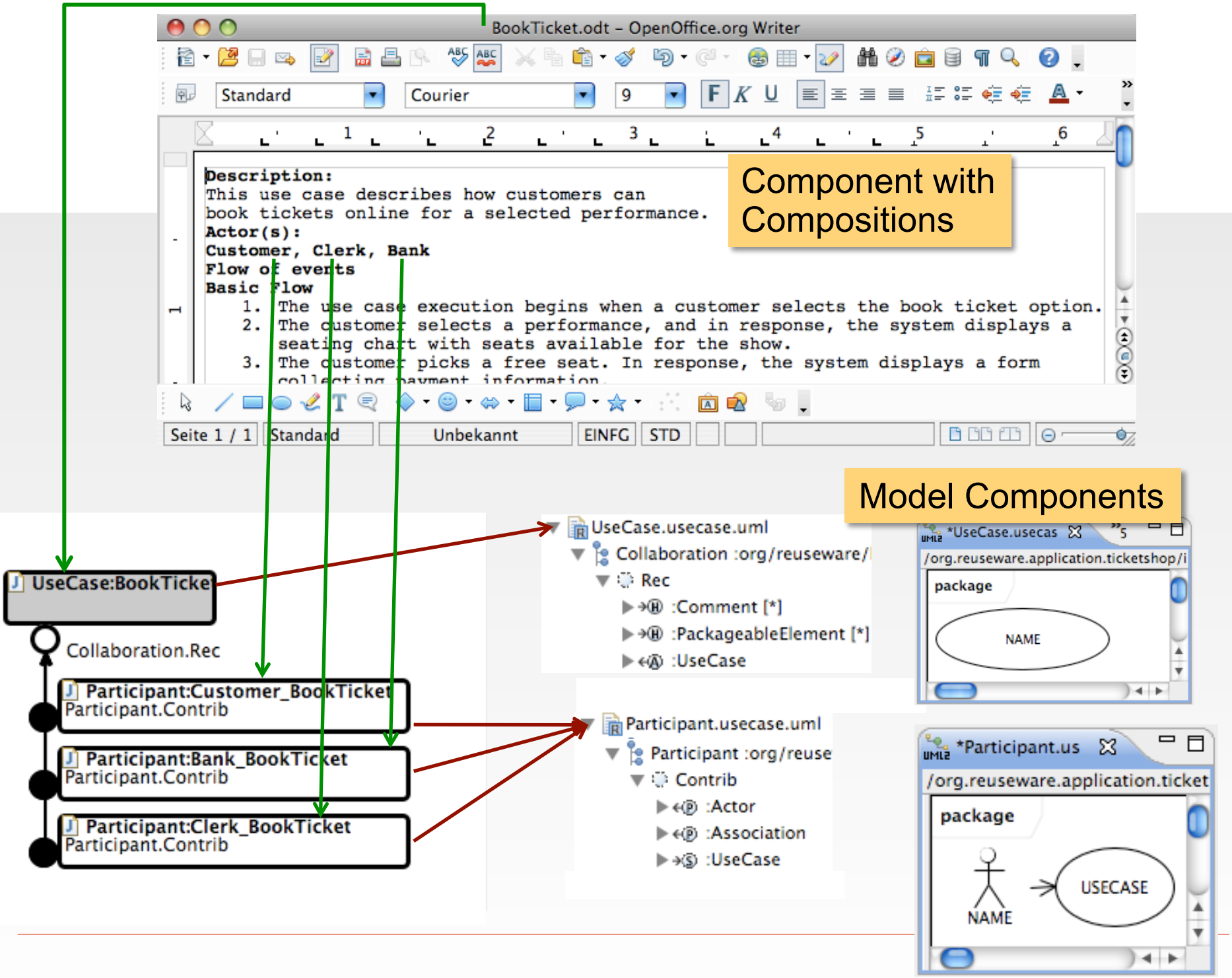
Courtesy J. Johannes, 2010

Model Component Composition

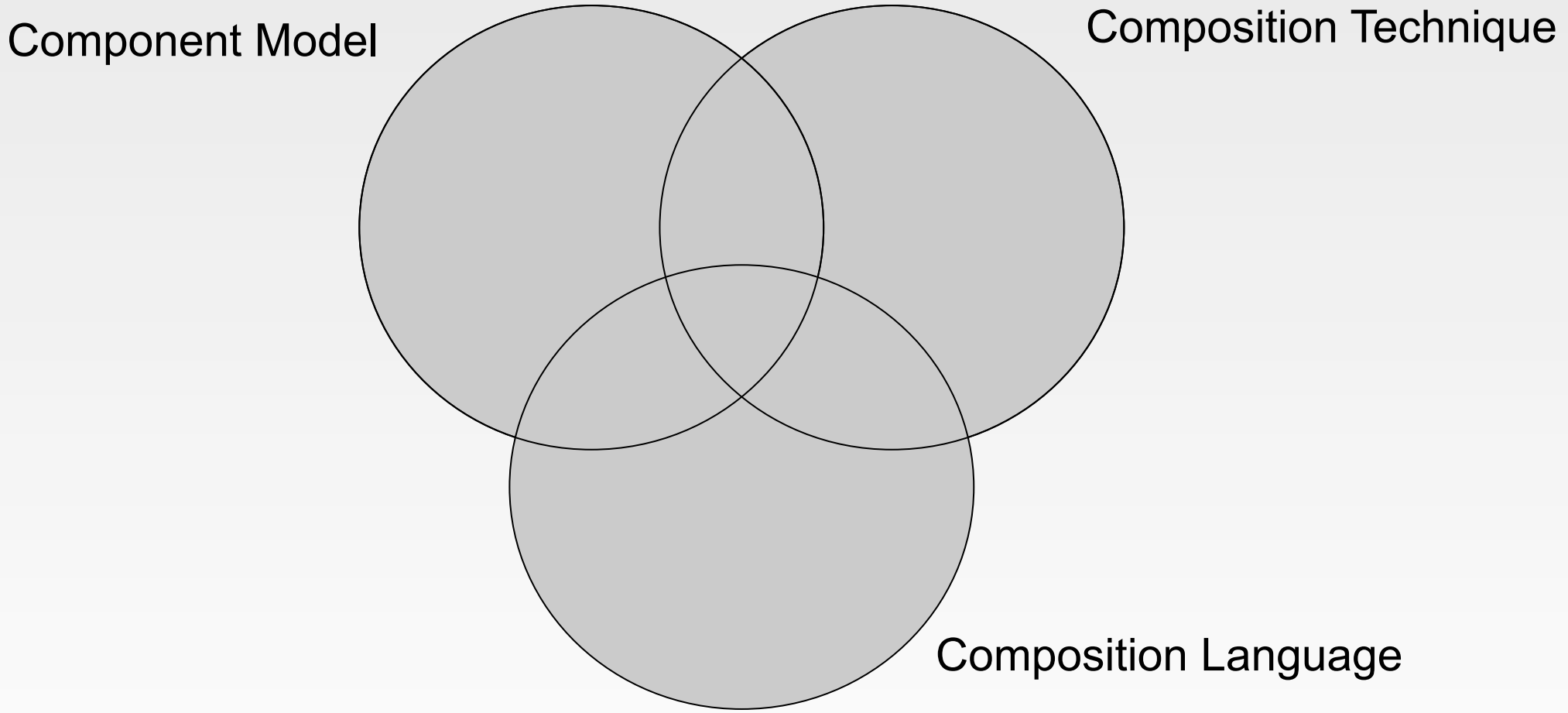
- Generic visual composition language



Courtesy J. Johannes, 2010

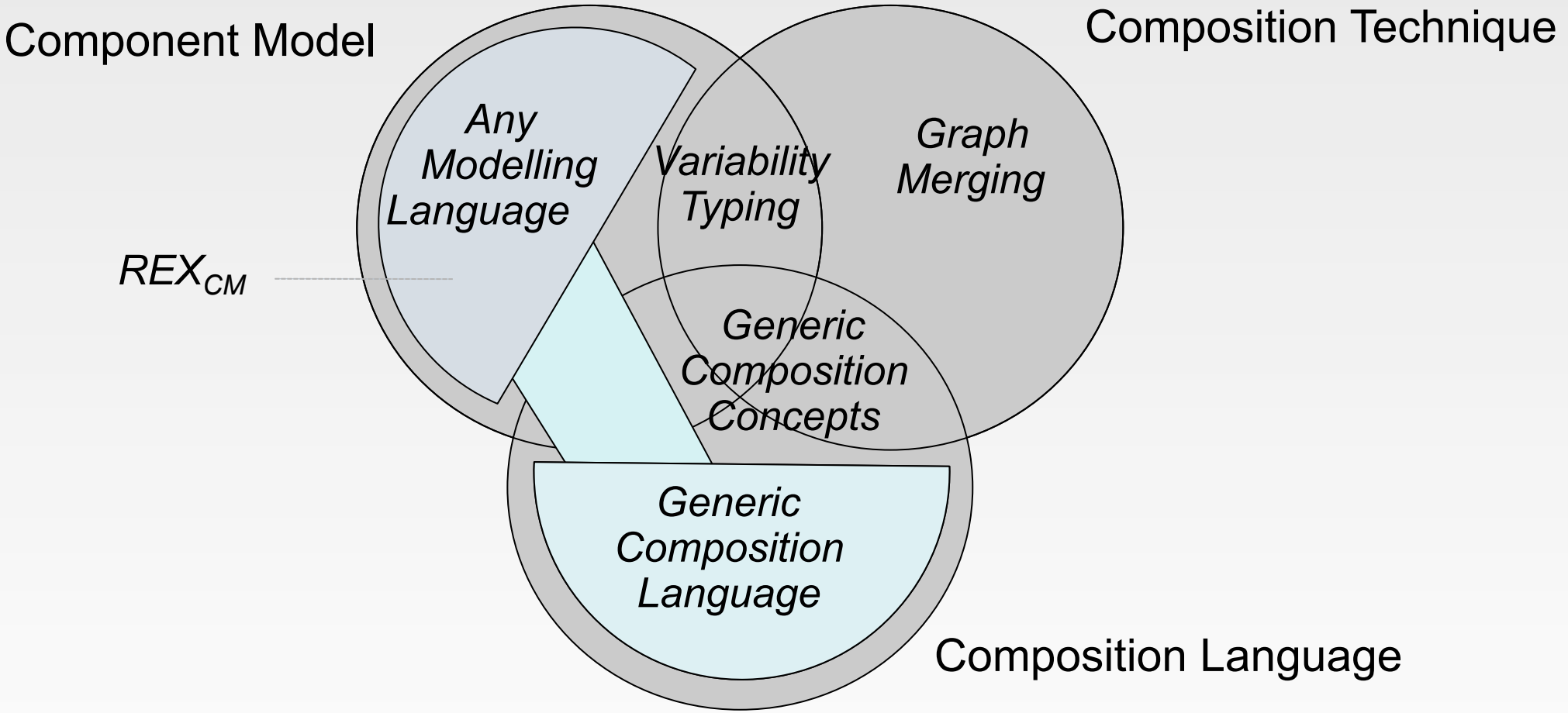

BookTicket.odt – OpenOffice.org Writer
Standard
Courier
9
Description:
This use case describes how customers can
book tickets online for a selected performance.
Actor(s):
Customer, Clerk, Bank
Flow of events
Basic Flow
1. The use case execution begins when a customer selects the book ticket option.
2. The customer selects a performance, and in response, the system displays a
seating chart with seats available for the show.
3. The customer picks a free seat. In response, the system displays a form
Seite 1 / 1
Standard
Unbekannt
EINFG
STD
Component with Compositions
Model Components
UseCase:BookTicket
Collaboration.Rec
Participant:Customer_BookTicket
Participant.Contrib
Participant:Bank_BookTicket
Participant.Contrib
Participant:Clerk_BookTicket
Participant.Contrib
UseCase.usecase.uml
Collaboration :org/reuseware/
Rec
:Comment [*]
:PackageableElement [*]
:UseCase
*UseCase.usecas
/org.reuseware.application.ticketshop/i
package
NAME
Participant.usecase.uml
Participant :org/reuse
Contrib
:Actor
:Association
:UseCase
*Participant.us
/org.reuseware.application.ticket
package
NAME
USECASE

Invasive Software Composition Systems



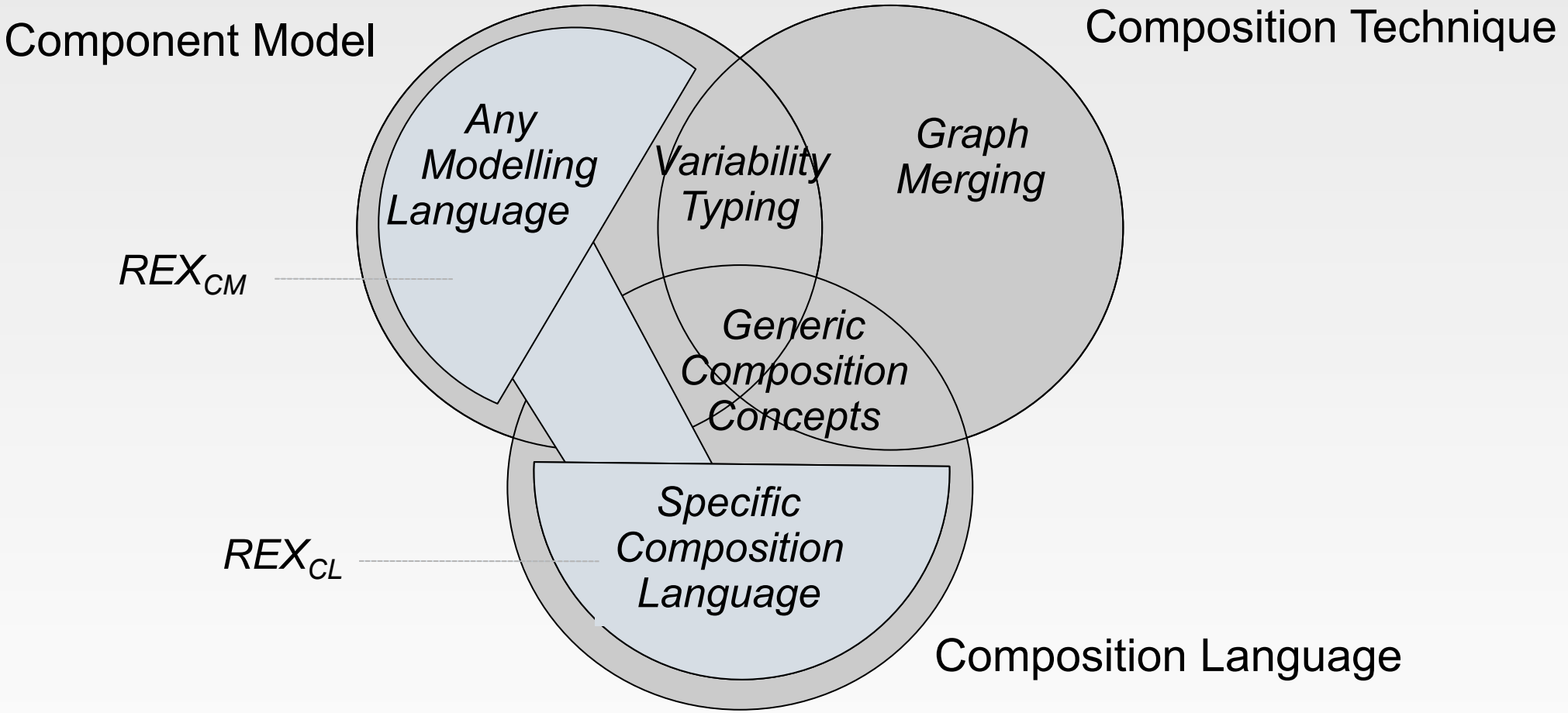
Courtesy J. Johannes, 2010

Invasive Model Composition Systems



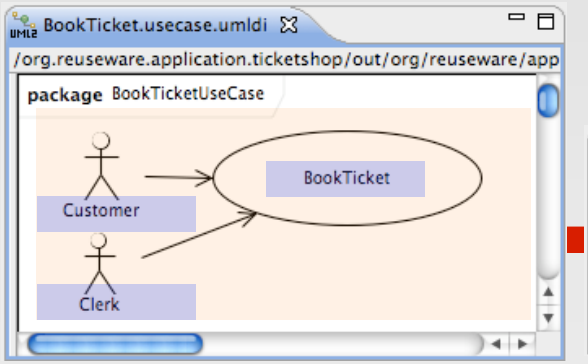
Courtesy J. Johannes, 2010

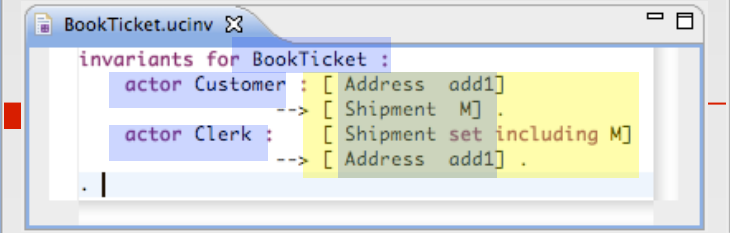
Invasive Model Composition Systems

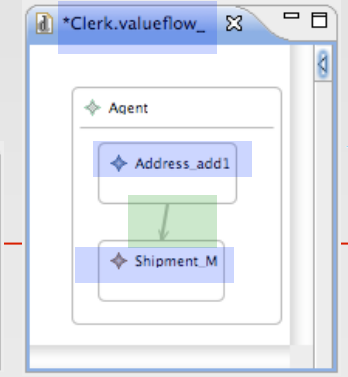


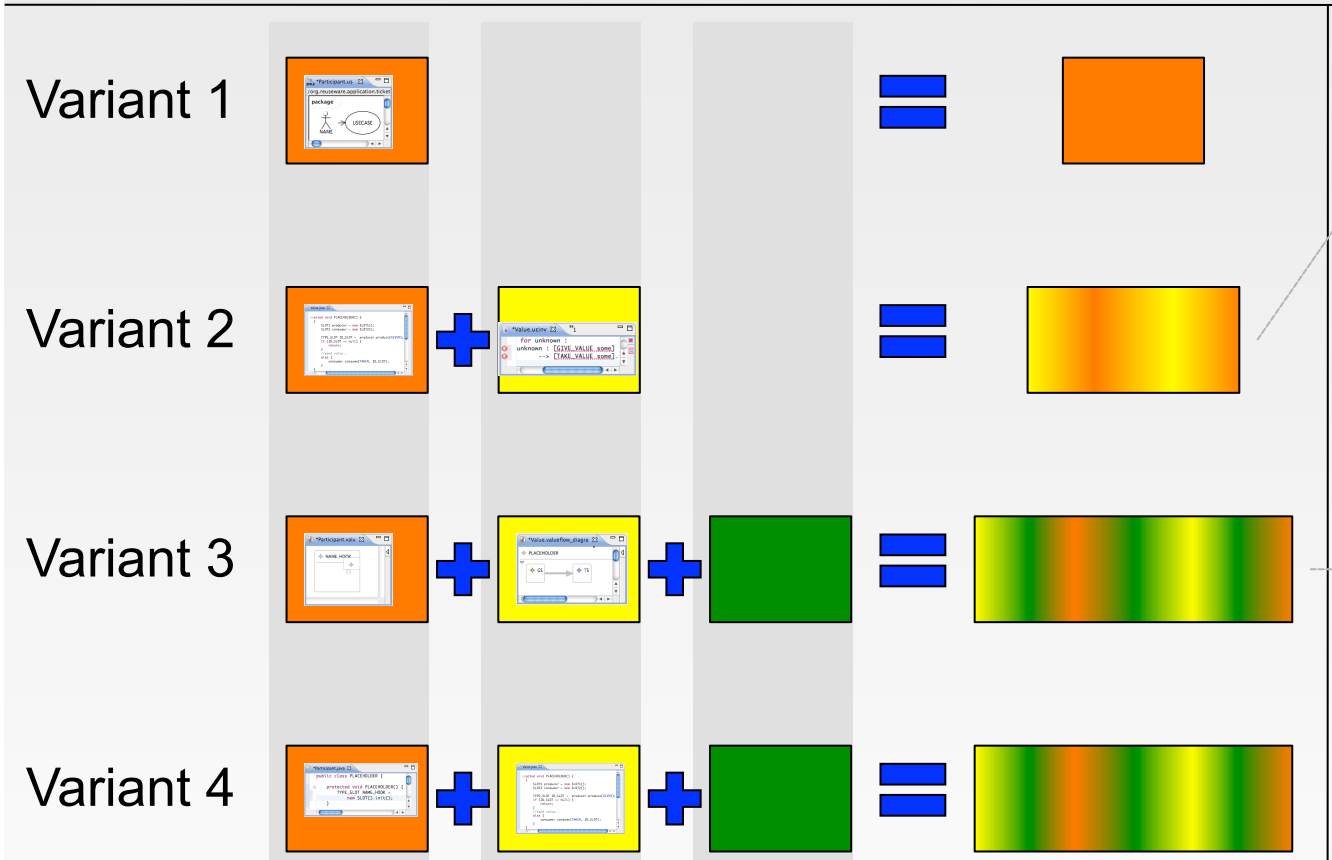
Courtesy J. Johannes, 2010

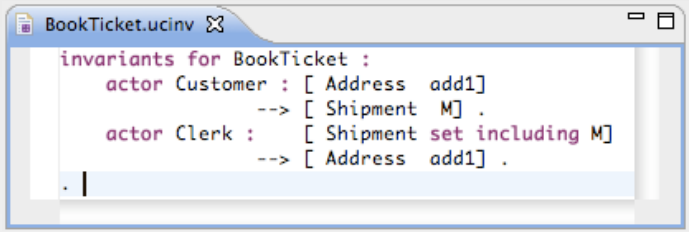
Case Study

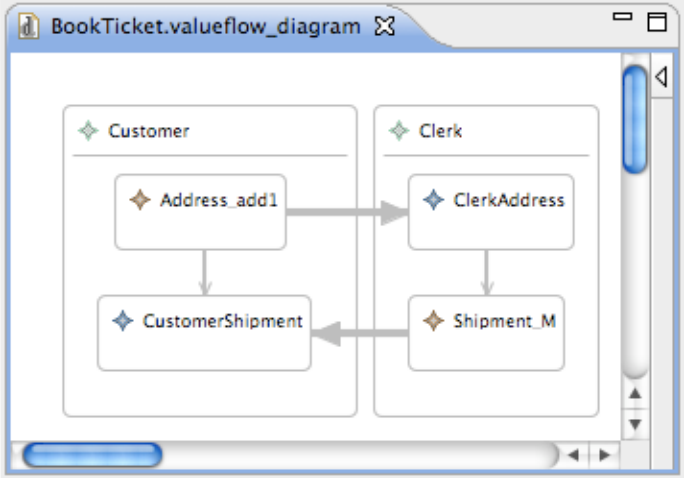

BookTicket.usecase.umldi
/org.reuseware.application.ticketshop/out/org/reuseware/app
package BookTicketUseCase
BookTicket
Customer
Clerk

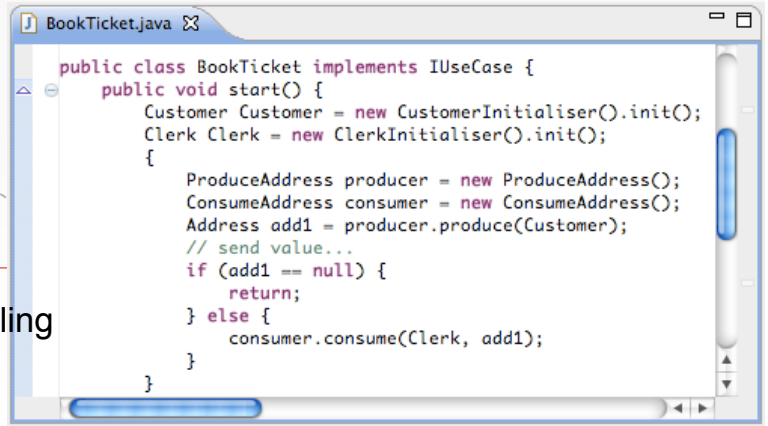

BookTicket.ucinv
invariants for BookTicket :
actor Customer : [Address add1]
--> [Shipment M] .
actor Clerk : [Shipment set including M]
--> [Address add1] .


*Clerk.valueflow_
Agent
Address_add1
Shipment_M


Variant 1
Variant 2
Variant 3
Variant 4
Modeling Language 1
Modeling Language 2
Modeling Language 3

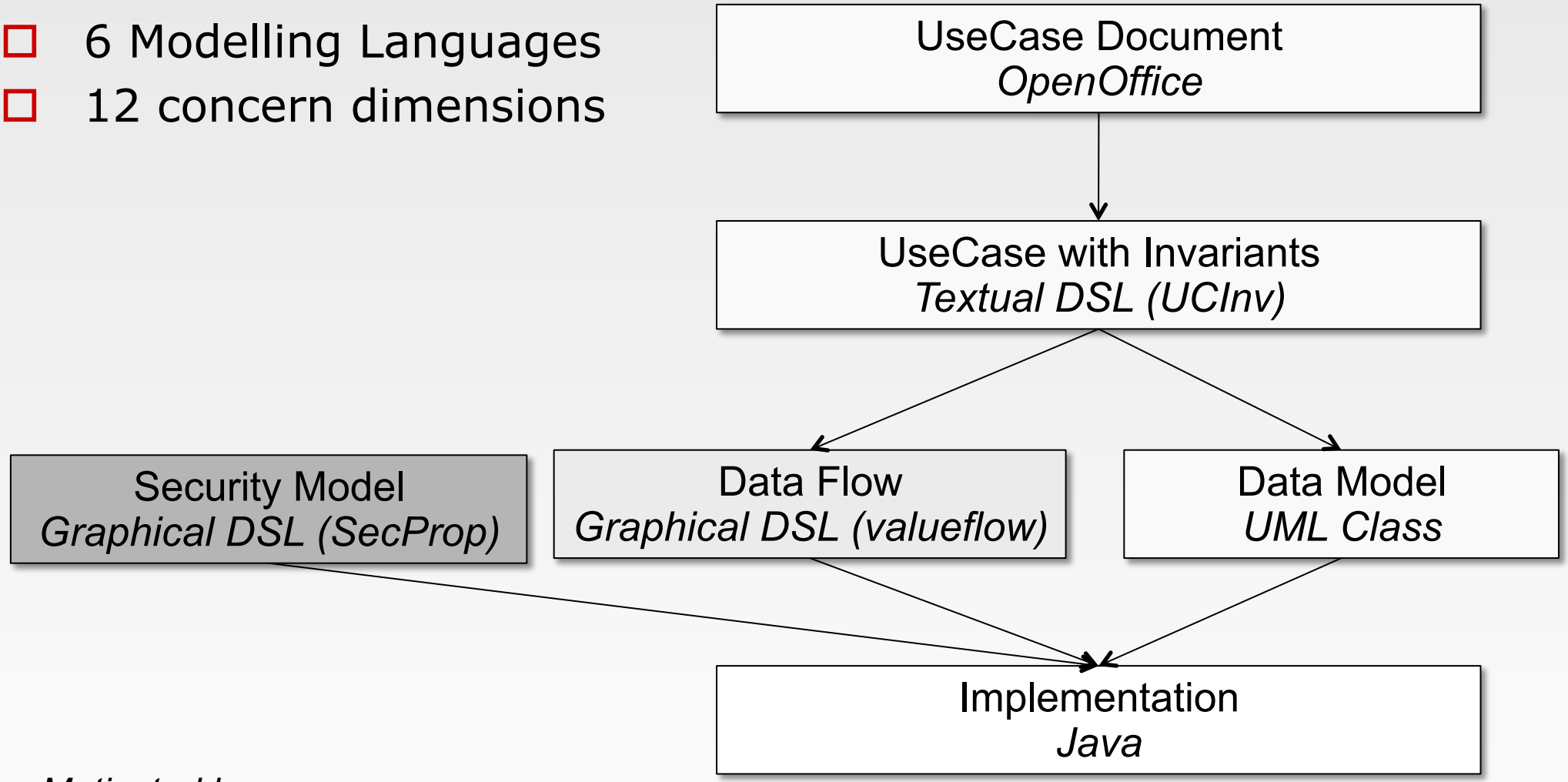

BookTicket.ucinv
invariants for BookTicket :
actor Customer : [Address add1]
--> [Shipment M] .
actor Clerk : [Shipment set including M]
--> [Address add1] .


BookTicket.valueflow_diagram
Customer
Clerk
Address_add1
ClerkAddress
CustomerShipment
Shipment_M


BookTicket.java
public class BookTicket implements IUseCase {
public void start() {
Customer Customer = new CustomerInitialiser().init();
Clerk Clerk = new ClerkInitialiser().init();
{
ProduceAddress producer = new ProduceAddress();
ConsumeAddress consumer = new ConsumeAddress();
Address add1 = producer.produce(Customer);
// send value...
if (add1 == null) {
return;
} else {
consumer.consume(Clerk, add1);
}
}

Case Study: Concern Separation

- 6 Modelling Languages
- 12 concern dimensions



Motivated by:

Roussev, B., Wu, J.: Transforming use case models to class models and ocl-specifications. (2007)

MODELPLEX IP Deliverable D1.1.a (v3): Case Study Scenario Definitions. (March 2008)

Case Study: Concern Separation

Concern Dimensions support by language

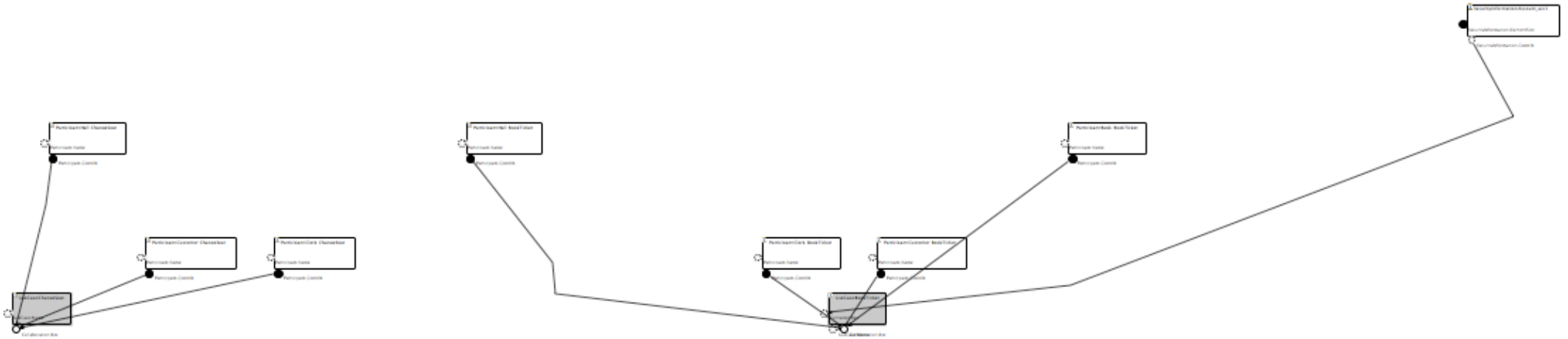
↑ *Fragment Languages*	usecase	participation	exchange	flow	trigger	factory	class	dataclass	associate	typebinding	app	security
OpenDocument	x	x										
UML Use Case	x	x										
Use Case Invariant	x	x	x									
Value Flow	x	x	x	x								
UML Class							x	x	x			
Java	x	x	x	x	x	x	x	x	x	x	x	x
SecProp	x	x	x									x

Case Study: Concern Separation

usecase
participation
exchange
flow
trigger
factory

class
dataclass
association
typebinding
app

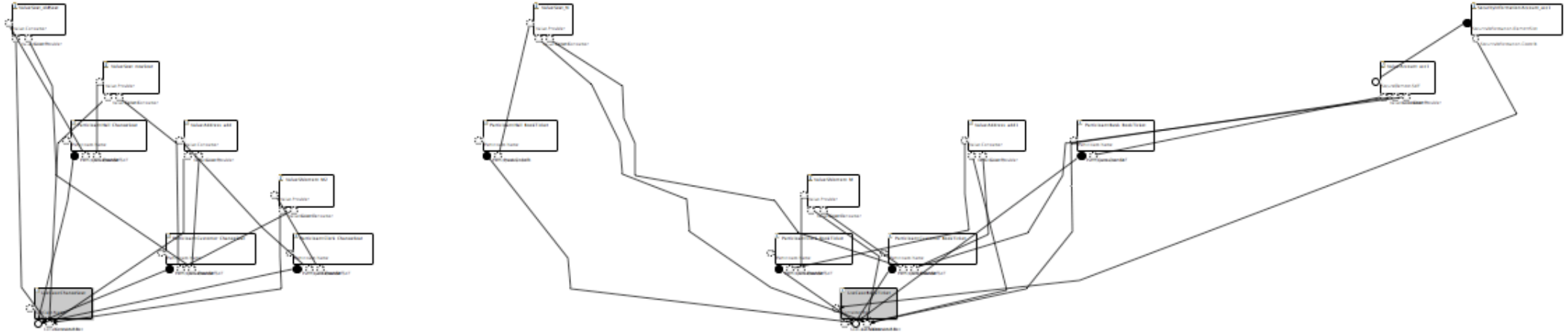
Case Study: Concern Separation



usecase
participation
exchange
flow
trigger
factory

class
dataclass
association
typebinding
app

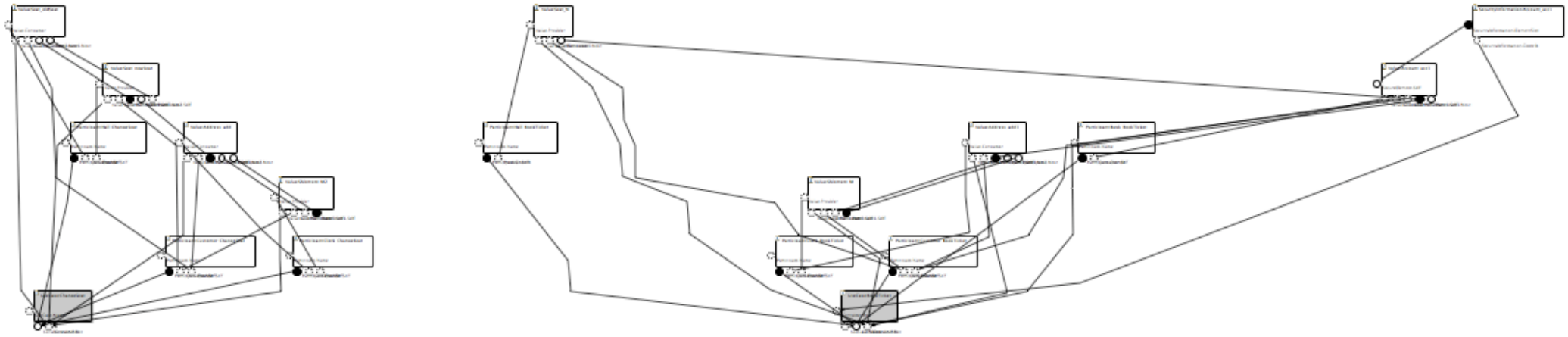
Case Study: Concern Separation



usecase
participation
exchange
flow
trigger
factory

class
dataclass
association
typebinding
app

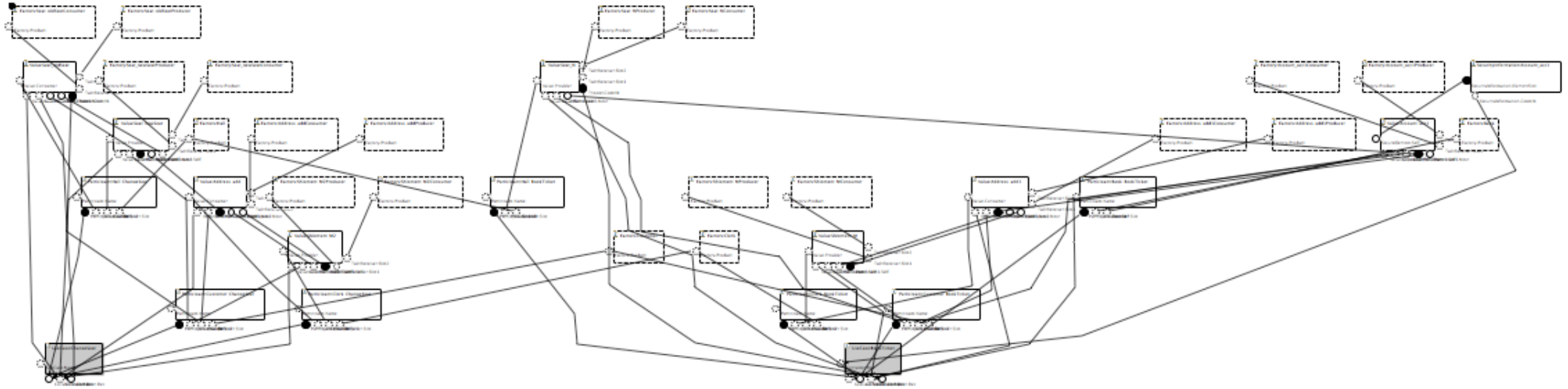
Case Study: Concern Separation



usecase
participation
exchange
flow
trigger
factory

class
dataclass
association
typebinding
app

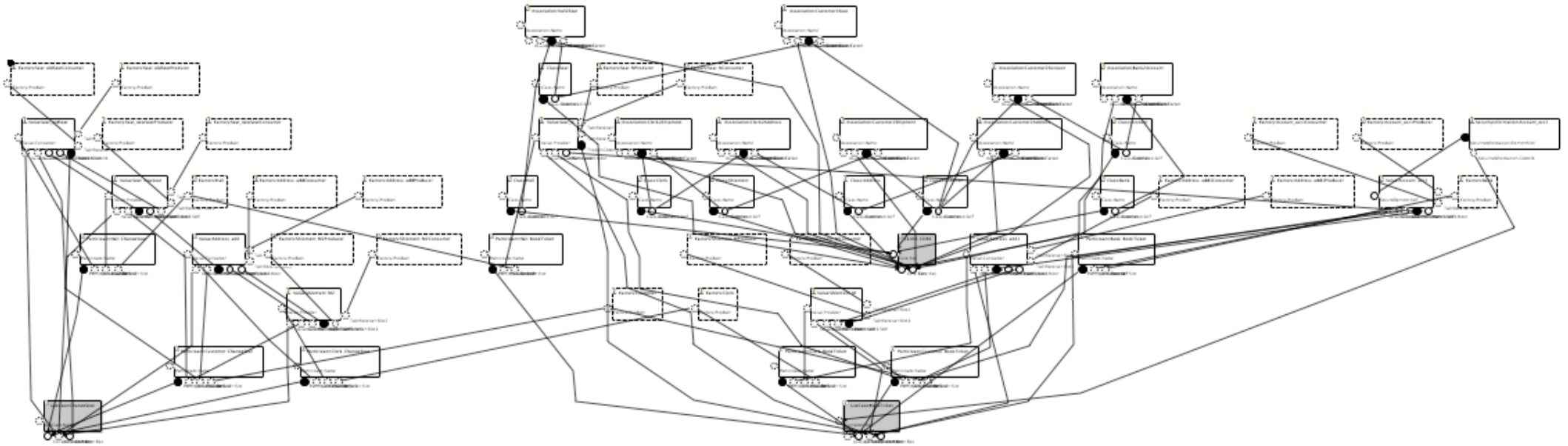
Case Study: Concern Separation



usecase
participation
exchange
flow
trigger
factory

class
dataclass
association
typebinding
app

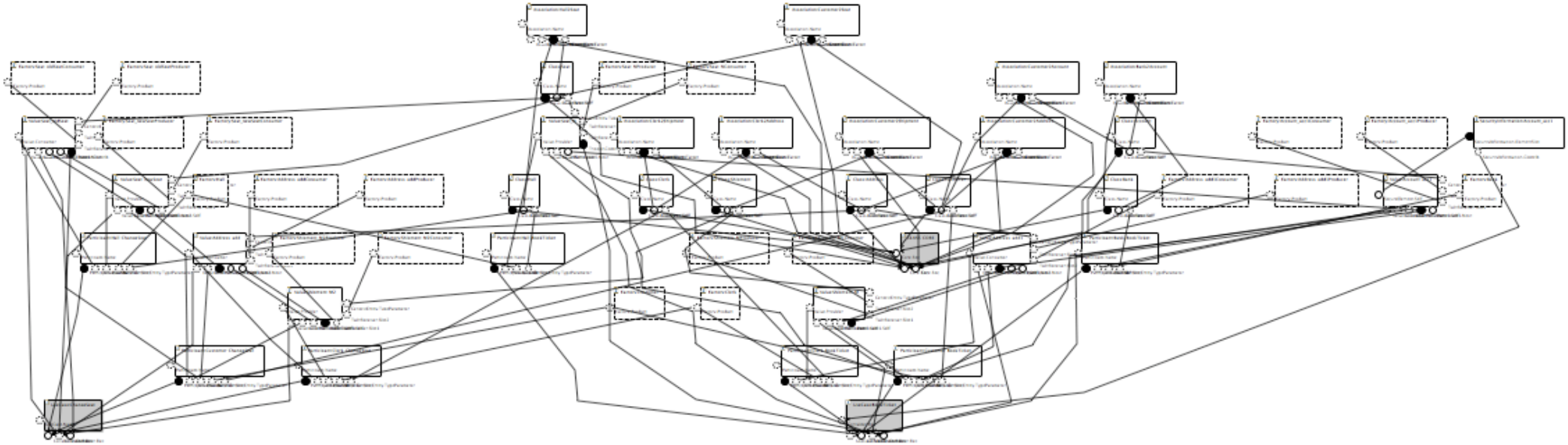
Case Study: Concern Separation



usecase
participation
exchange
flow
trigger
factory

class
dataclass
association
typebinding
app

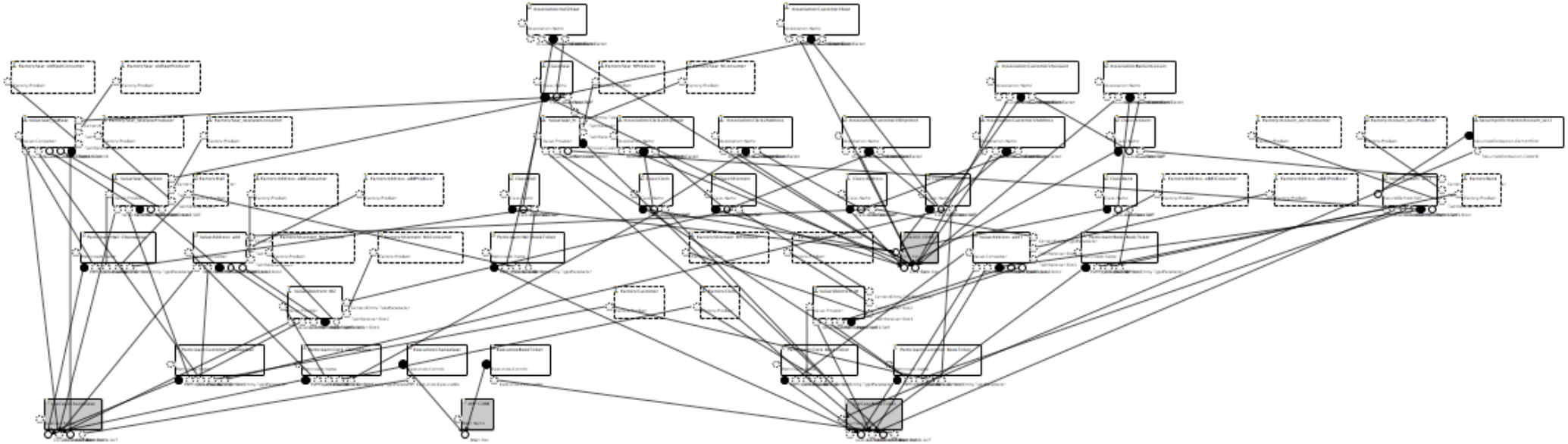
Case Study: Concern Separation



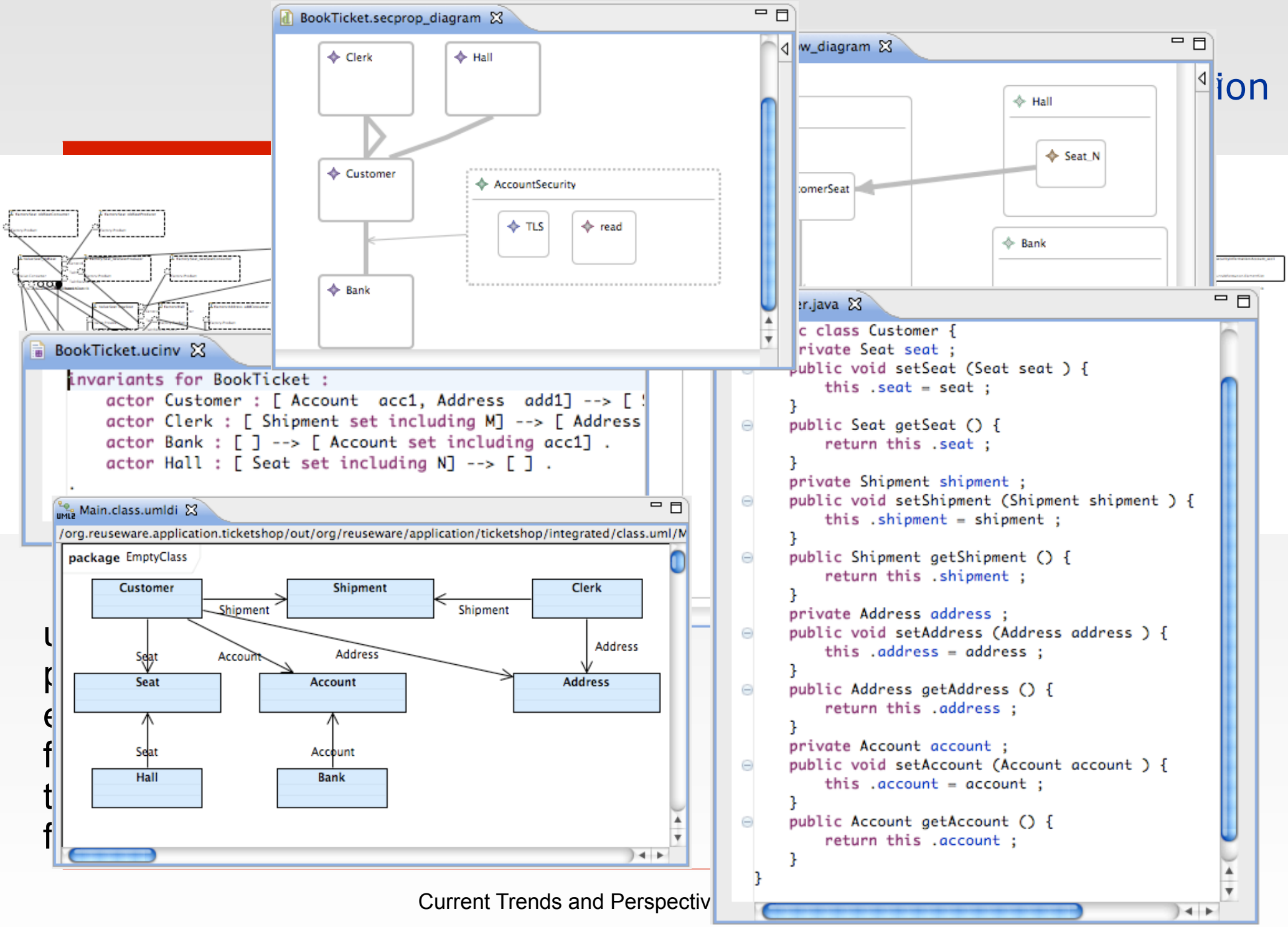
usecase
participation
exchange
flow
trigger
factory

class
dataclass
association
typebinding
app

Case Study: Concern Separation



usecase	class
participation	dataclass
exchange	association
flow	typebinding
trigger	app
factory	



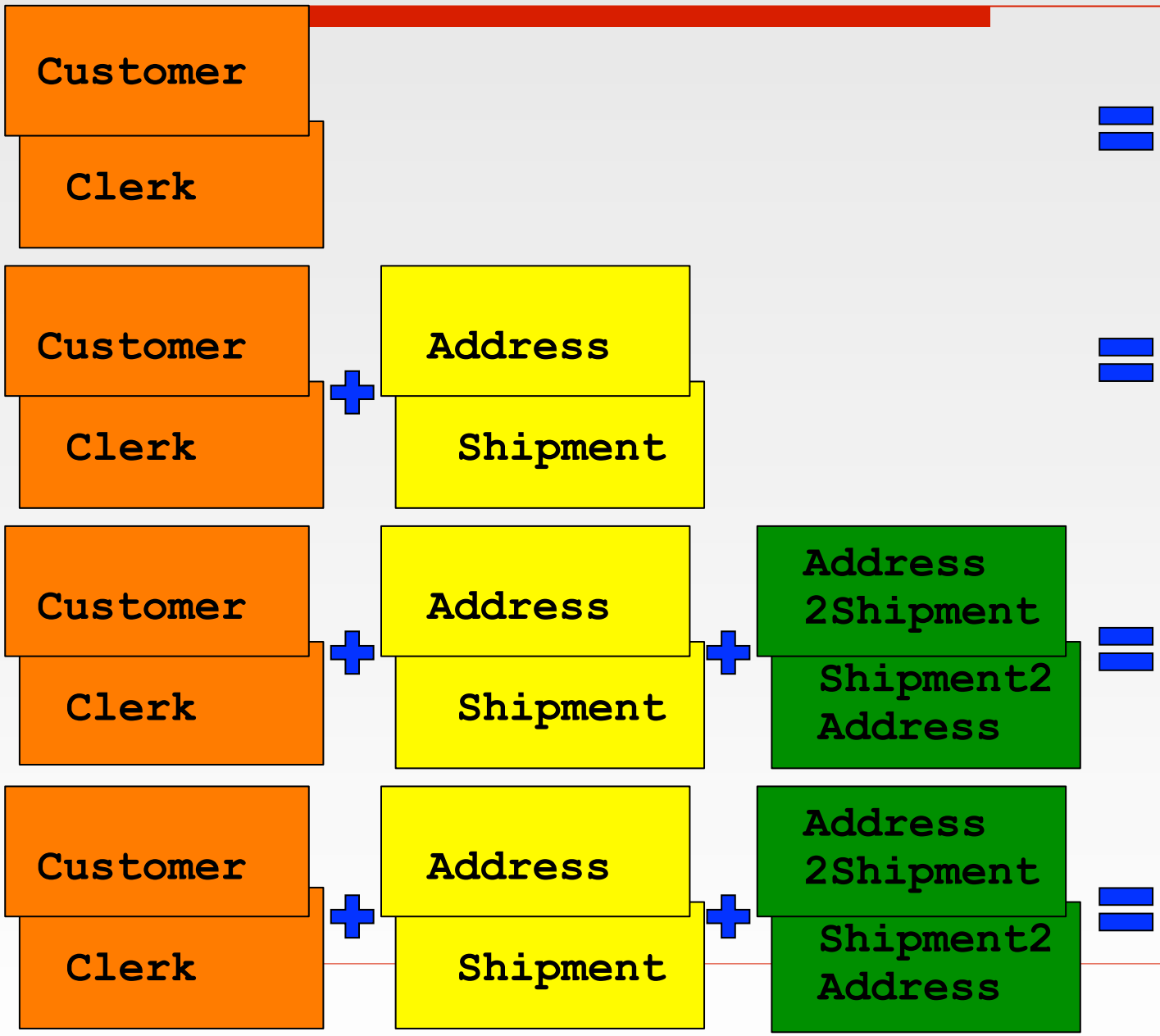
ion

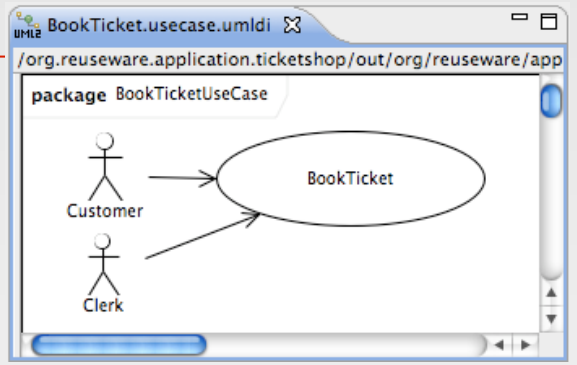
Current Trends and Perspectiv

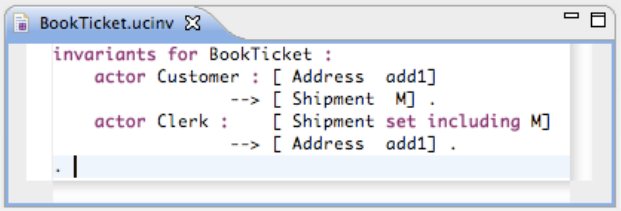
Connecting Abstraction Layers with Model-Based Separation of Concerns (ModelSOC)

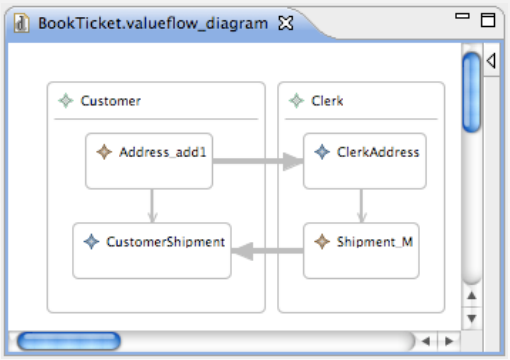
Courtesy J. Johannes, 2010

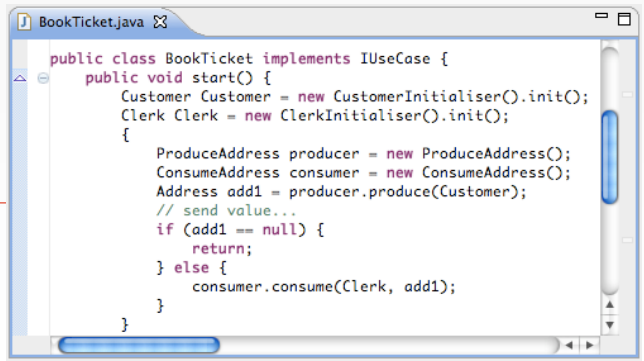
Multi-Dimensional SoC for MDSD



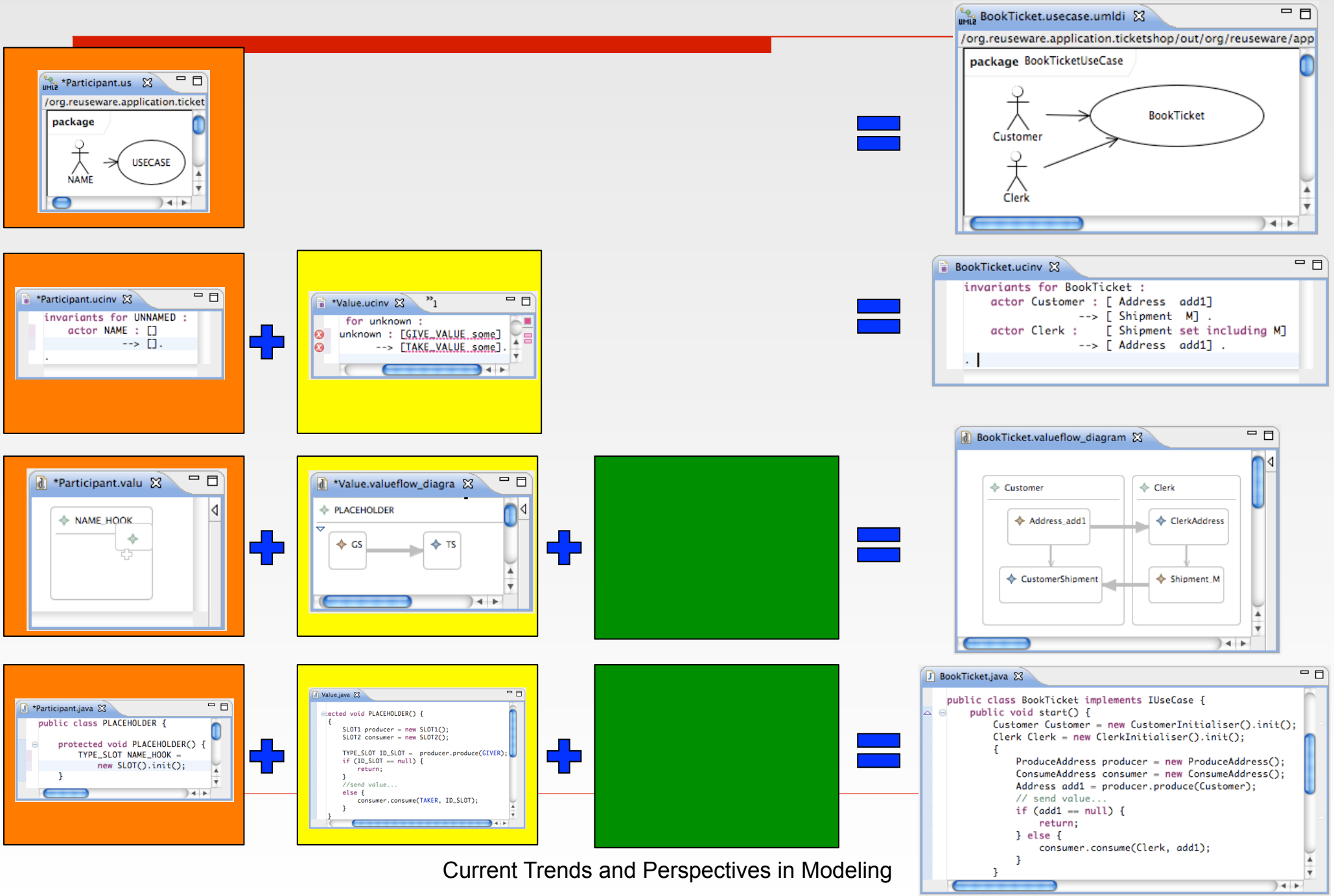




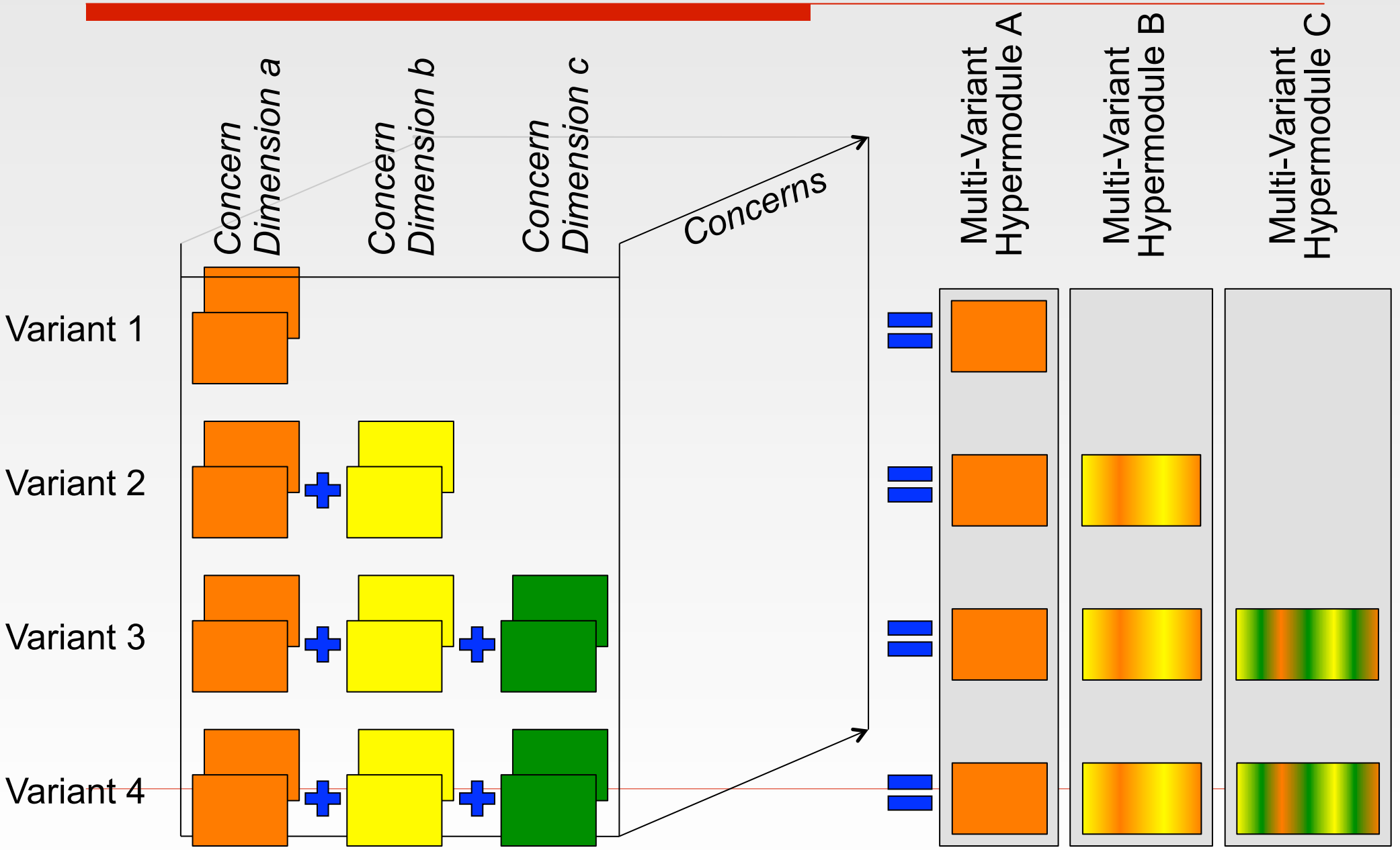




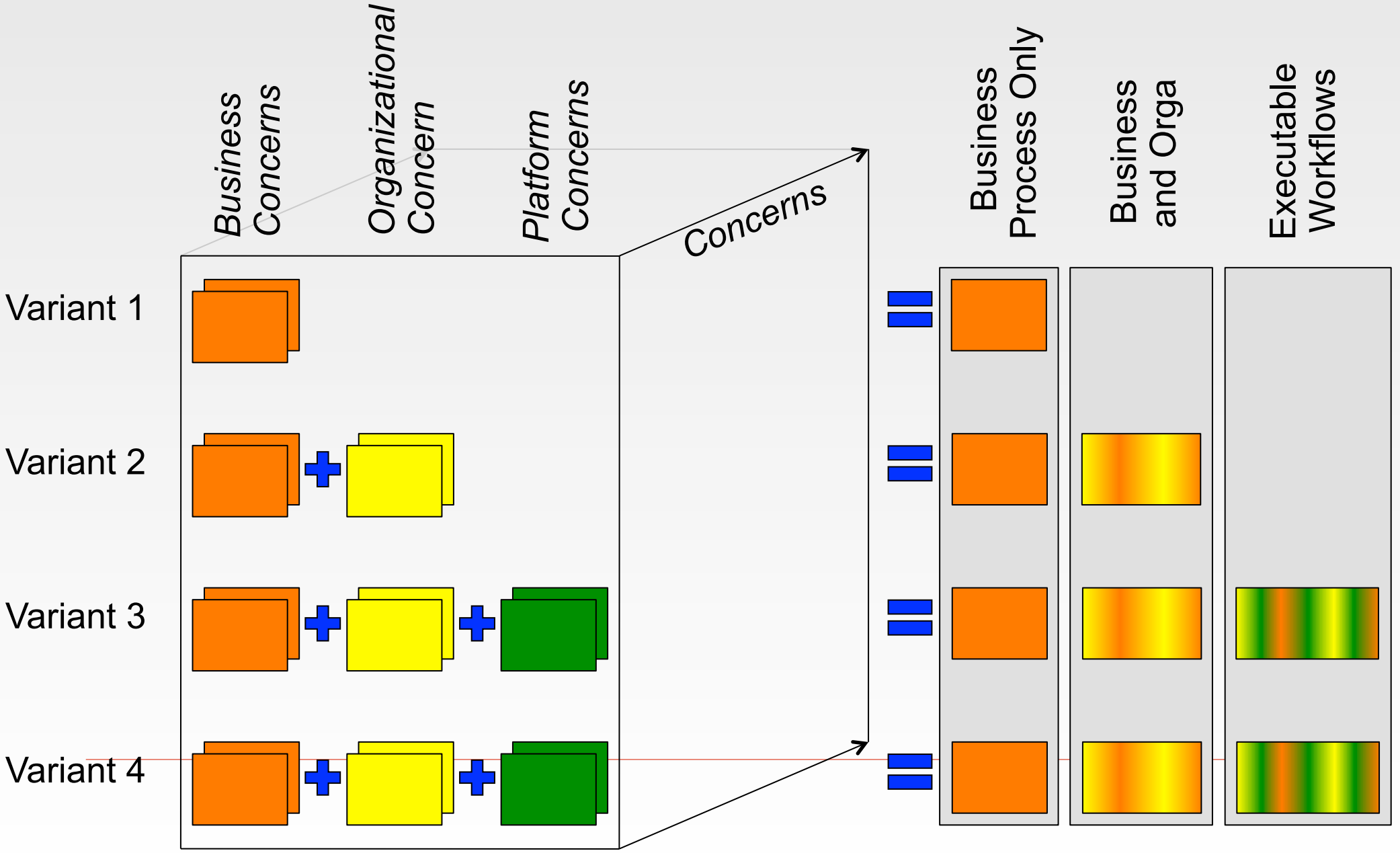
Multi-Dimensional SoC for MDSD



Multi-Dimensional SoC for MDSD



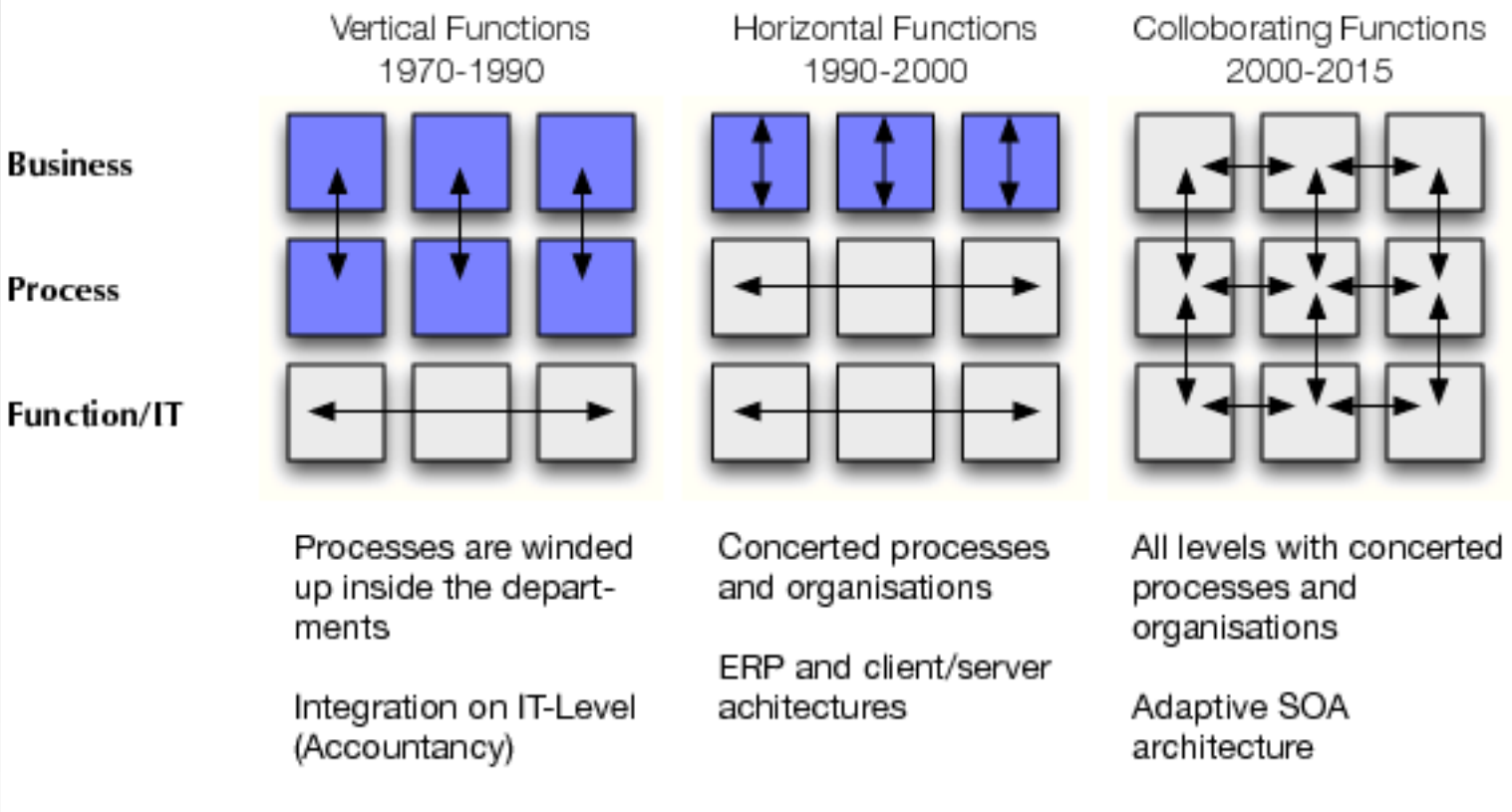
Challenge: Multi-Dimensional Viewpoint Frameworks for Connection of Business and IT-Level



3) Towards Model Synchronization

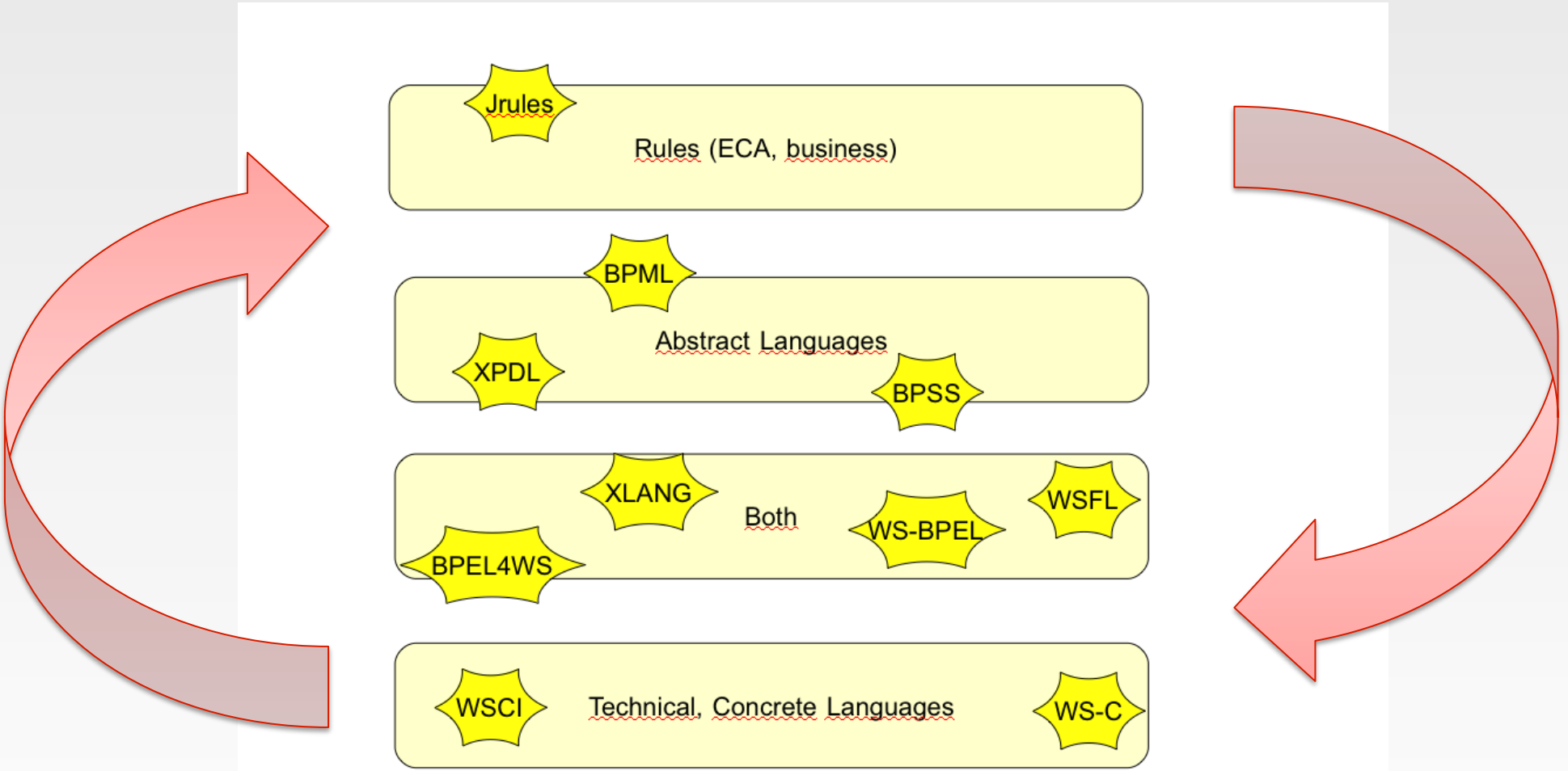
Connecting Business and IT-level

The Integration of Business and IT-level



Bitkom IT Society and Berger Consulting. Zukunft digitale Wirtschaft. study on growth fields in it, January 2007. www.BITKOM.com.

Integration on the Language Level…

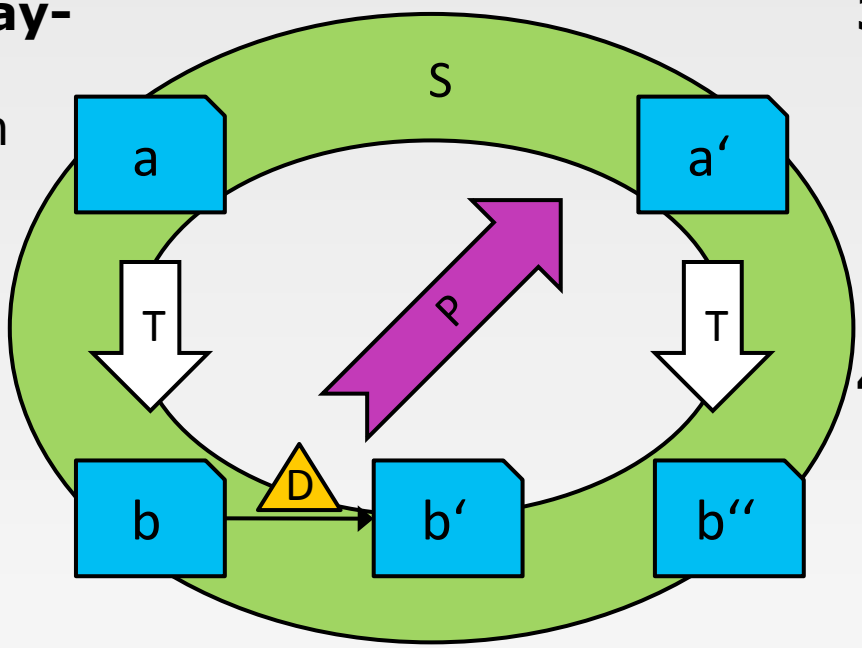


[Jaeger 2007]

Synchronization Predicates for Round-Trip Engineering

1. **Forward derivation (Play-Out)**
   A model **b** is derived from another model **a** using transformation T
   b = T(a)

2. **Delta change** in Forward space: Changes made to model **b** result in **b'**
   b' = D(b)



3. **Backpropagation**
   Find **a'** by propagating the changes
   a' = P(a,b,b',D,…)

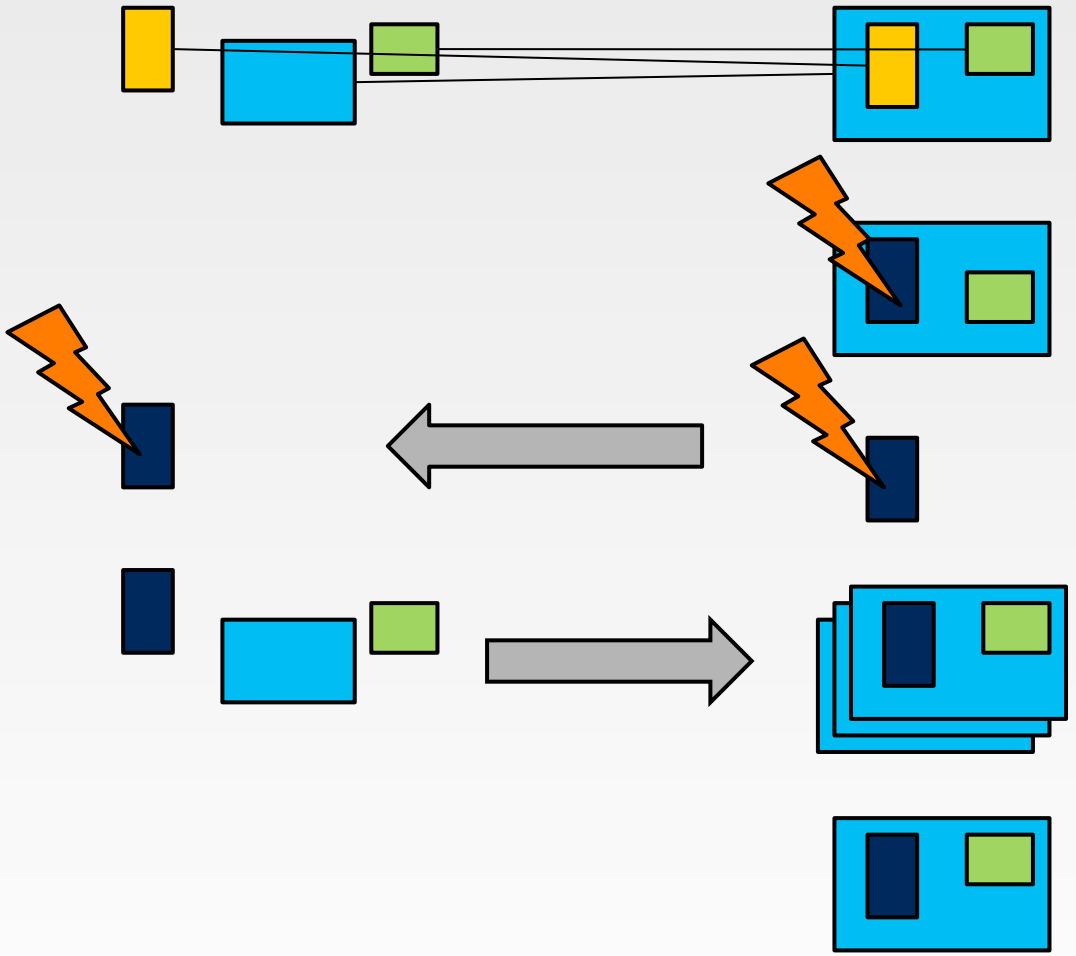
4. **Replay**
   Reexecute the transformation
   b'' = T(a')

5. **A synchronization predicate** must be preserved (strong) or granted (weak) or selected (nondeterministic)
   S(T, a, b, D, b', a', b'') is true

Courtesy M. Seifert, 2010

Replay Synchronization Support for Model Composition

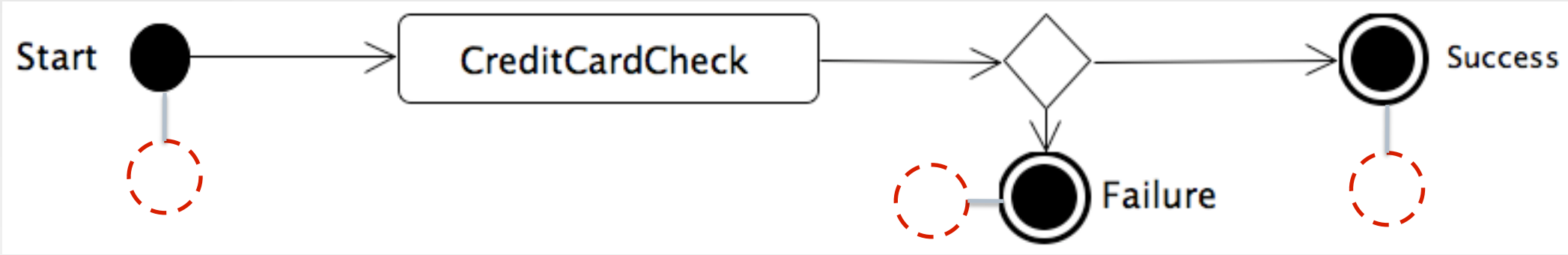
1. Trace fragment elements
2. Observe model changes
3. Propagate changes
4. Replay: Re-Compose



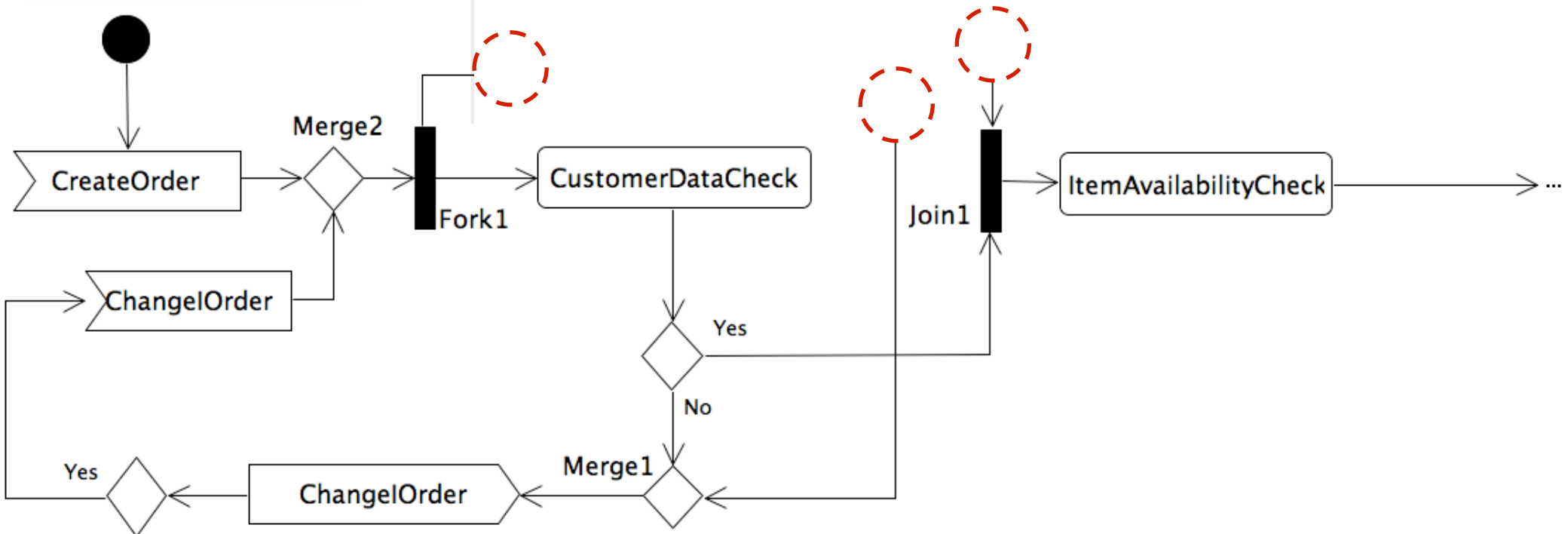
Models are synchronized!

Two Process Model Components

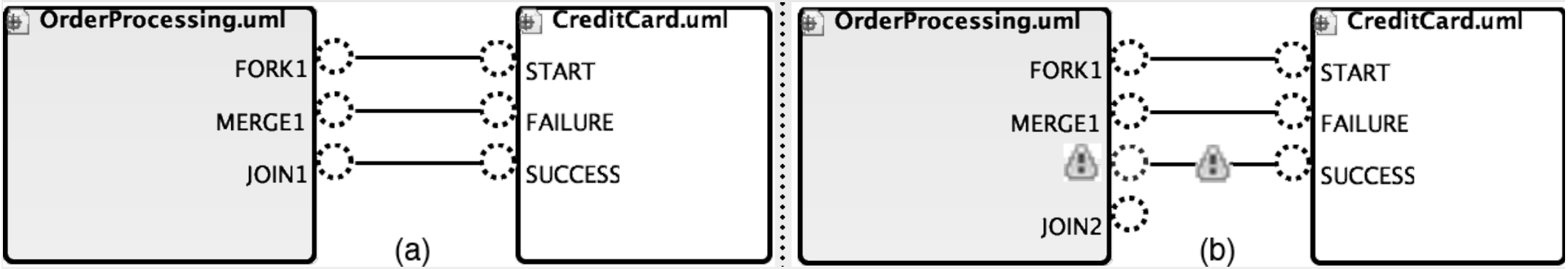
CreditCard

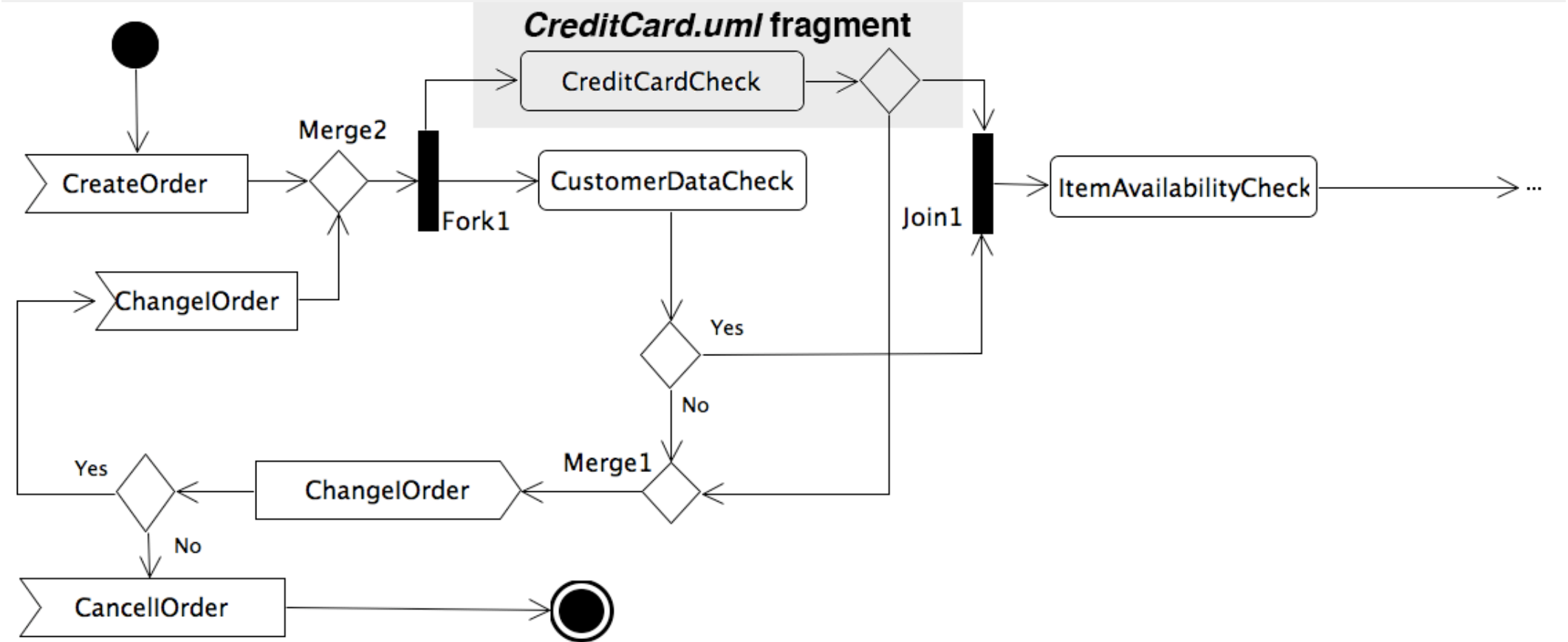


OrderProcessing



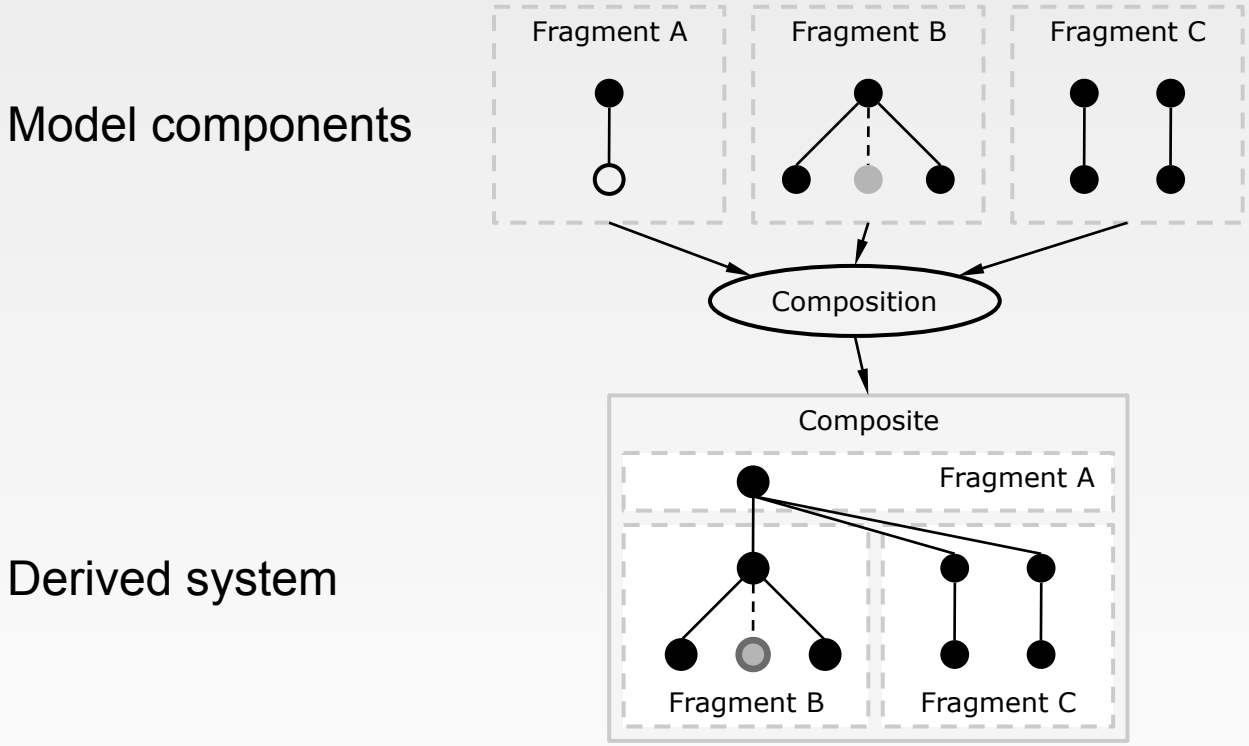
Composition of Processes





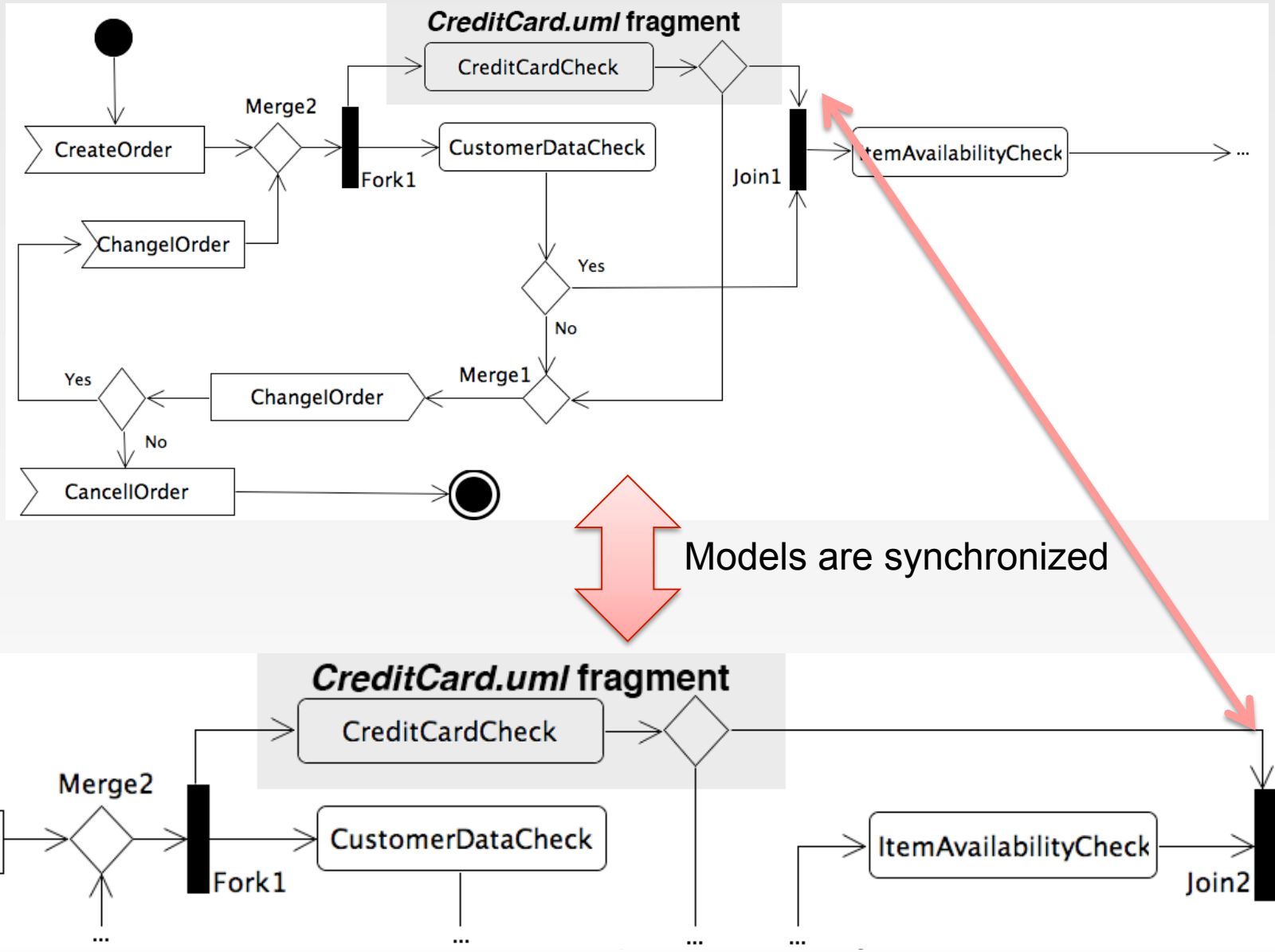
Tracing through Model Composition

- Use traces to determine source fragment of (existing) model elements in model components



Node | Variation Point | Reference | Added node | Potential source node

Manual Changes of Composed Models and their Round-Trip



Lessons learned

1. With Replay-Synchronization, it is possible to synchronize *abstract* and *executable* business process models
2. Necessity for a synchronization predicate
   - Cannot always be decided automatically, but always interactive
3. For a full synchronization of business and IT-level, other models must be synchronized, too

➔ Still effort necessary.

Conclusion

Let's Build Bridges

- .. Between technological spaces
  - By model bridges
- .. Between business and IT-level
  - By model composition
  - By model synchronization

Mhare CCBYSA-2.5 https://en.wikipedia.org/wiki/File:Stari_Most22.jpg

Thank you!
Questions?

www.most-project.eu
www.reuseware.org
www.emftext.org
st.inf.tu-dresden.de

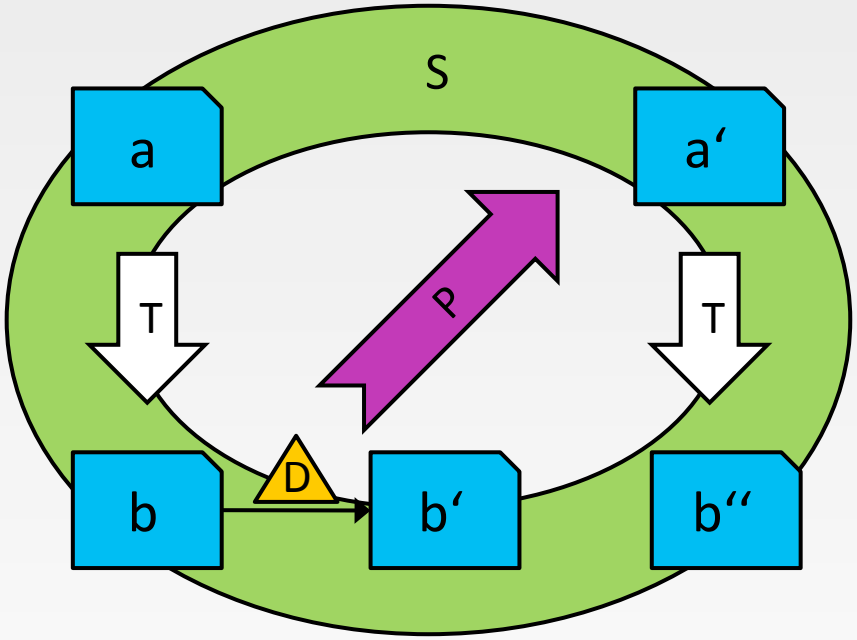
References

- [Aßm03] Aßmann, U.: Invasive Software Composition. 1 edn. Springer Verlag (April 2003)
- [Hen08] Jakob Henriksson. A Lightweight Framework for Universal Fragment Composition. PhD thesis, Technische Universität Dresden. http://nbn-resolving.de/urn:nbn:de:bsz:14-ds-1231251831567-11763
- [JA10] Jendrik Johannes, Uwe Aßmann. Concern-based (de) composition of model-driven software development processes. MoDELS 2010.
- [JSS09] Jendrik Johannes, Roland Samlaus, Mirko Seifert. Round-trip Support for Invasive Software Composition Systems. International Conference on Software Composition (SC) 2009
- [ABW10] U. Aßmann, A. Bartho, C. Wende. Reasoning Web. Semantic Technologies for Software Engineering, 6th International Summer School 2010, Dresden, Germany, August 30 - September 3, 2010. Tutorial Lectures, LNCS 6325, Springer

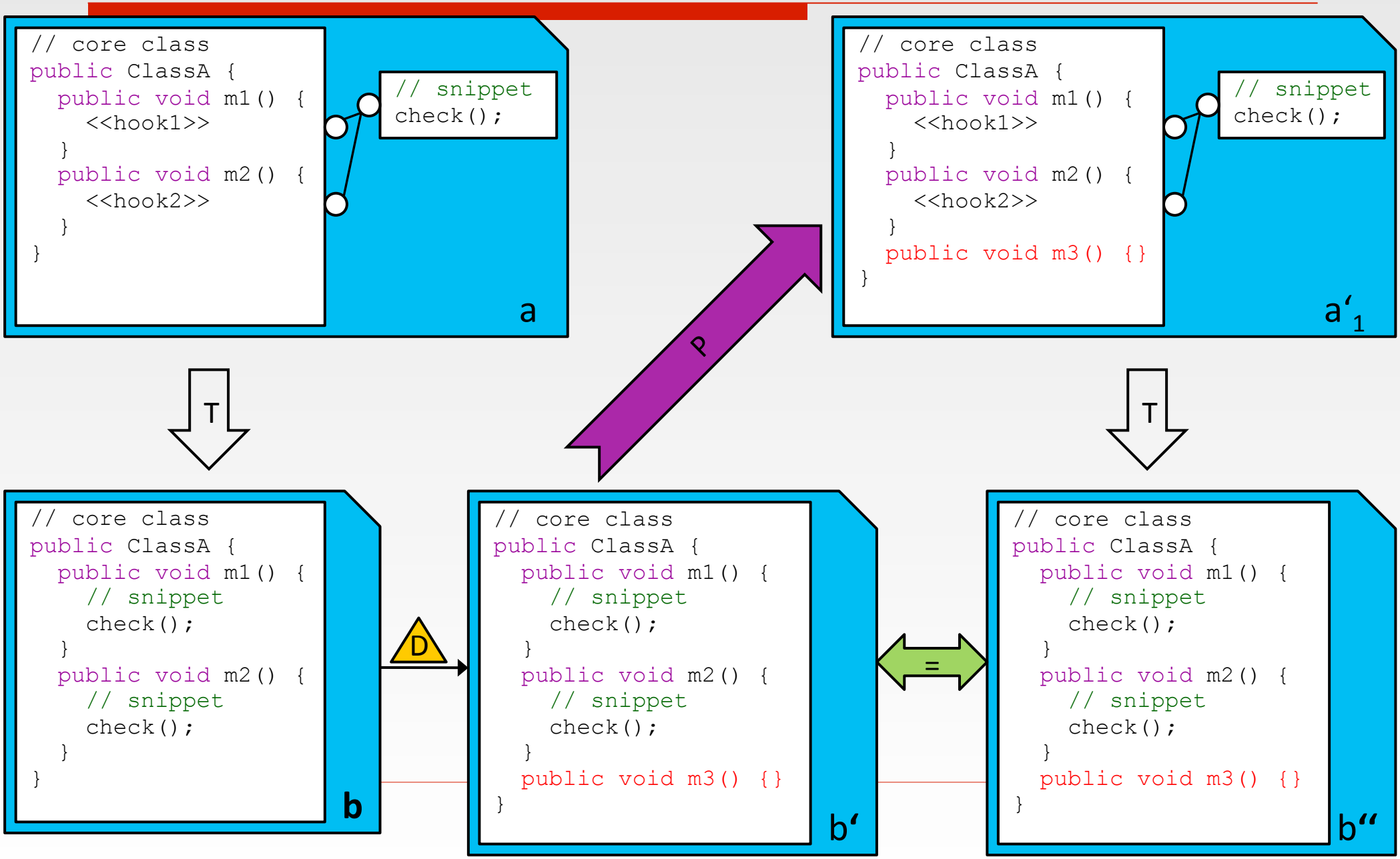
Backup Slides

Special Cases

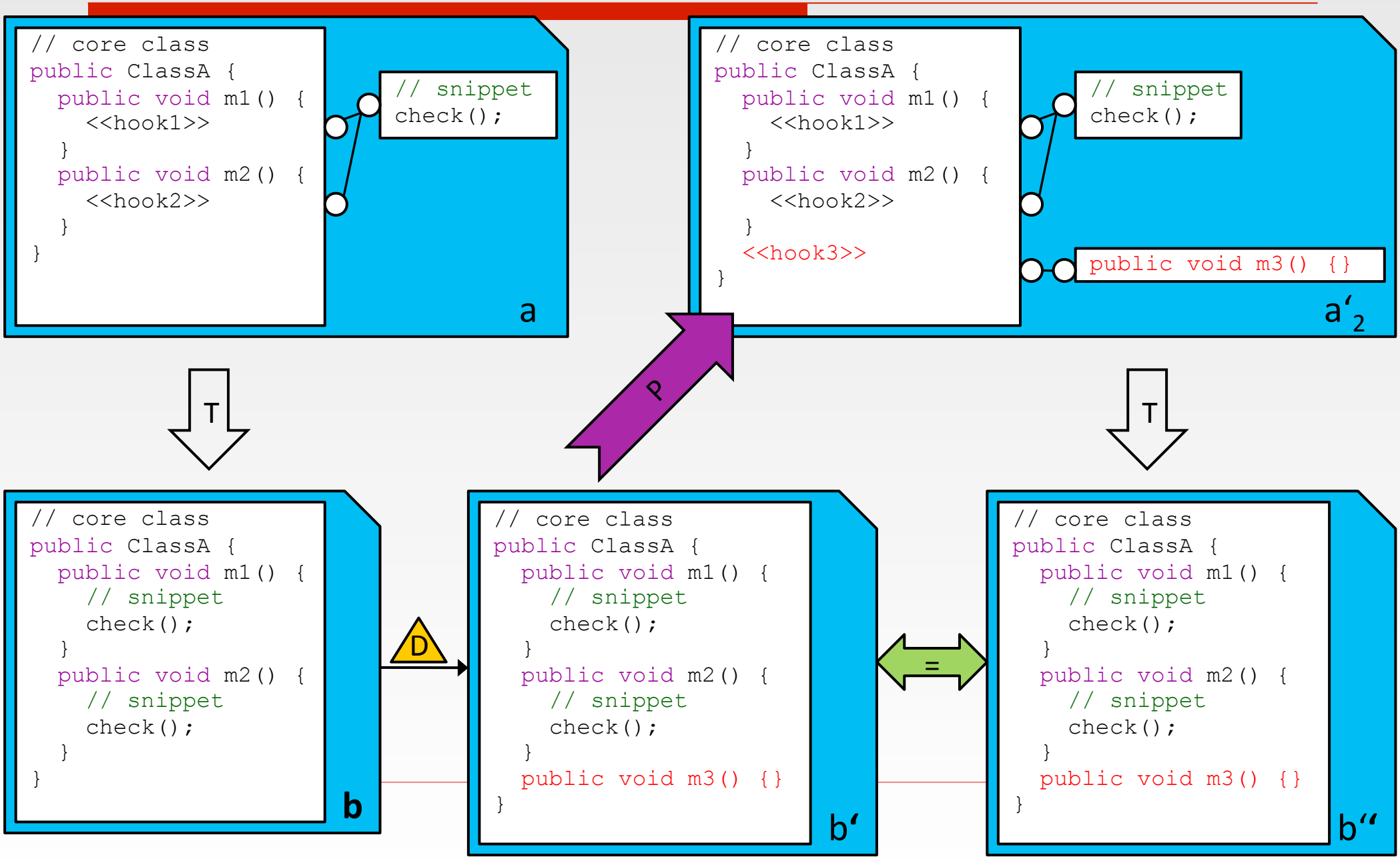
- Re-create **a** from **b** (i.e., invert T)
  - implies information preserving transformations
  - or bidirectional model transformations
- Find **a'** such that **b'** = **b''**
  (often there is no such a')
- Replay round-trip



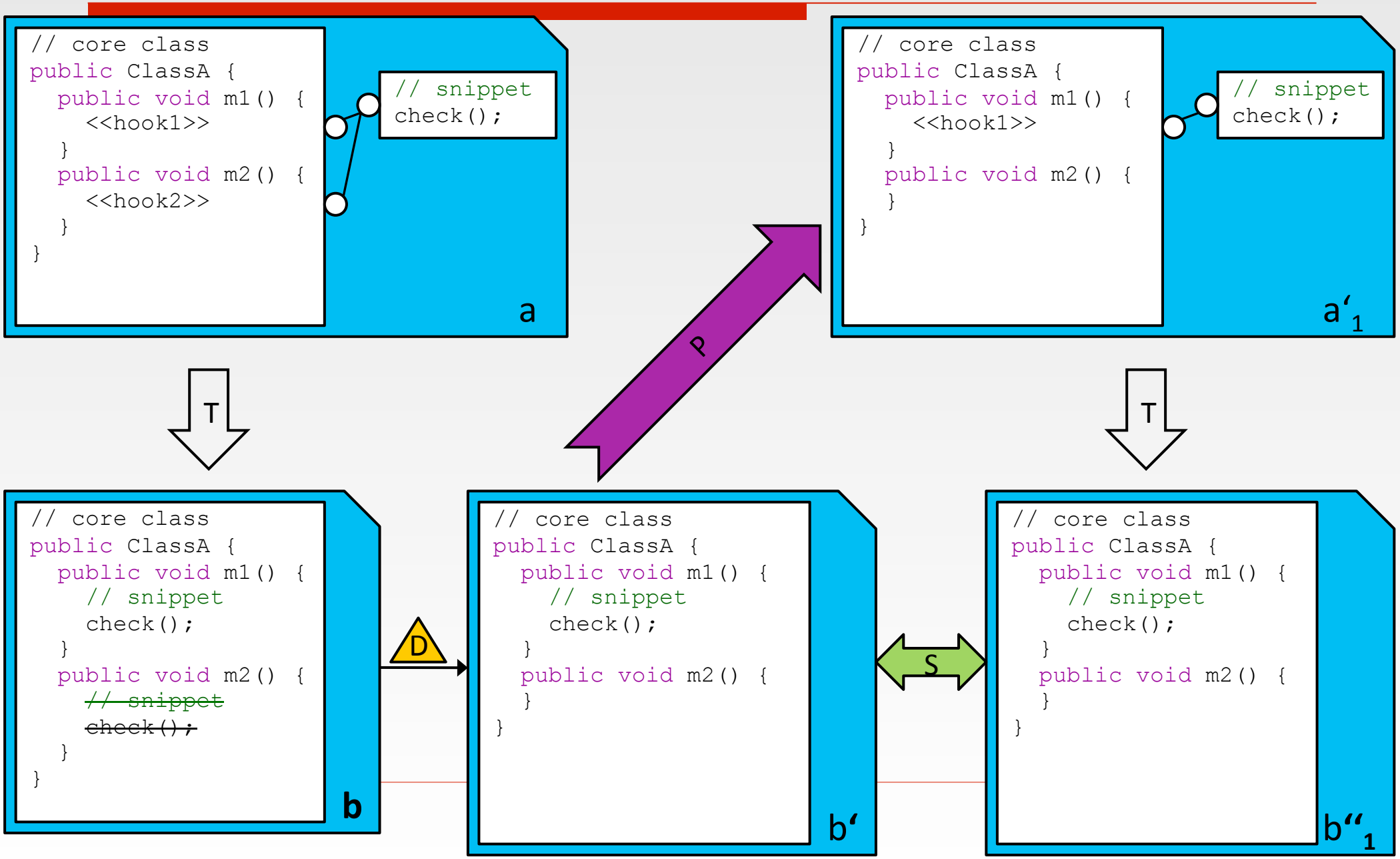
Synchronization – Example ISC Java
Multiple valid a'



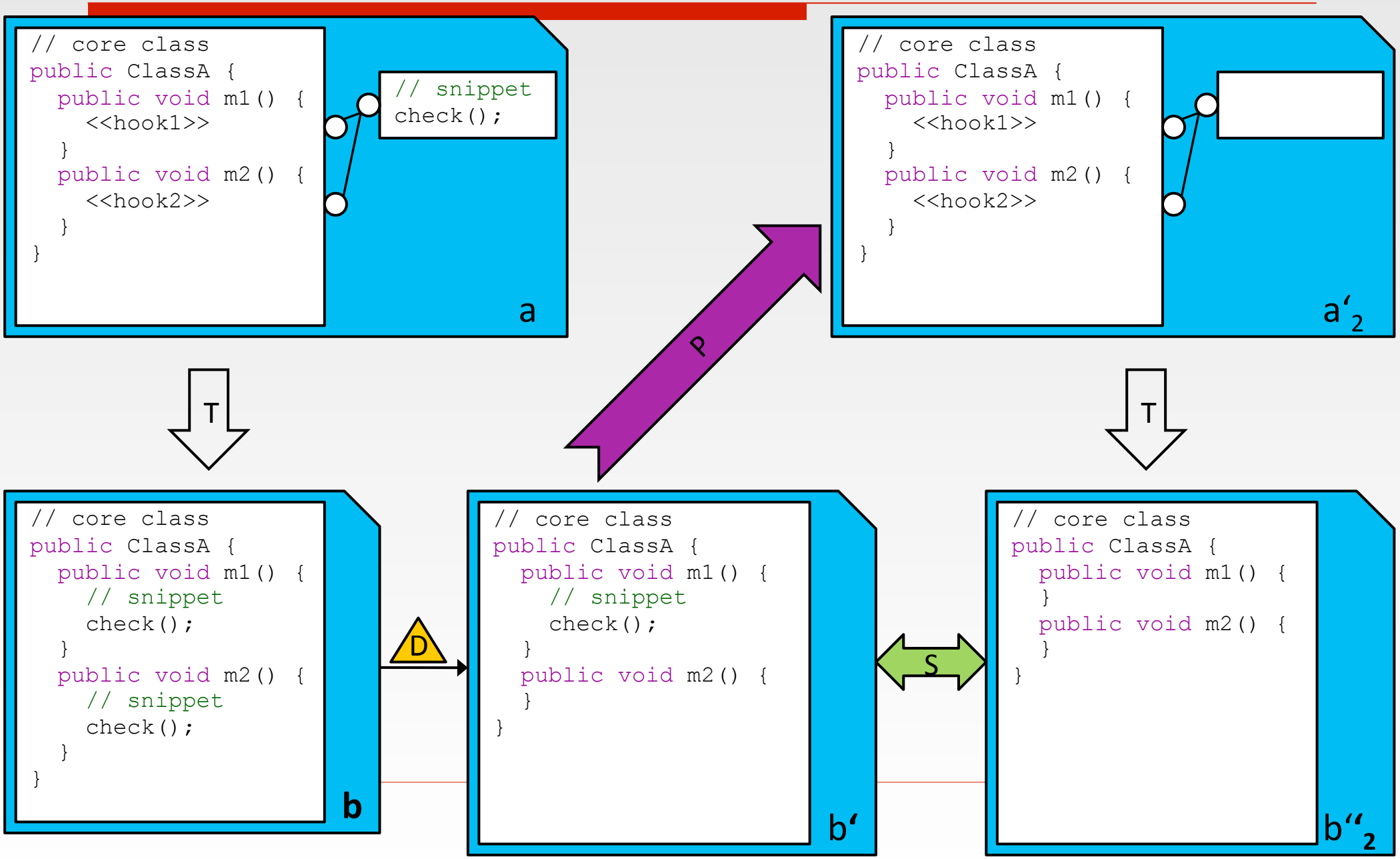
Synchronization – Example ISC Java
Multiple valid a'



Synchronization – Example ISC Java
Multiple valid b''



Synchronization – Example ISC Java Multiple valid b''



Other Backup

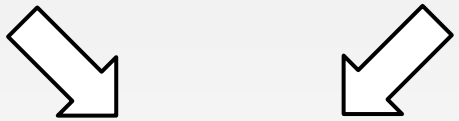
Tracing fragments

Fragment A:

```
public class
Customer {
   <<HOOK>>
   String name;
   public String
getName(){
      <<HOOK>>
      return name;
   }
}
```

Fragment B:

```
public class C {
   SecurityManager sm = new
SecurityManager();

   public void M(){
      sm.check(READ);
   }
}
```



Result:

```
public class Customer {
   SecurityManager sm = new
SecurityManager();
   String name;
   public String getName(){
      sm.check(READ);
      return name;
   }
}
```

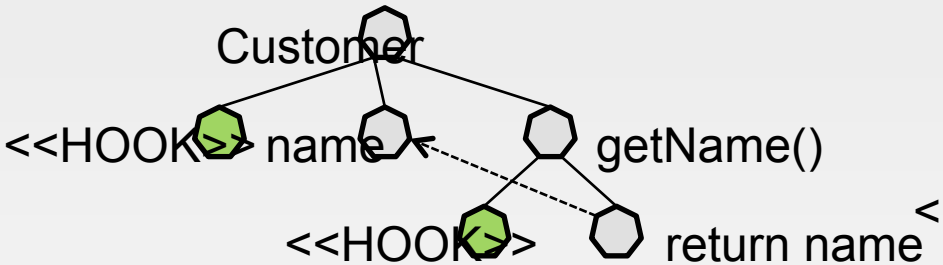
Change Propagation - Summary

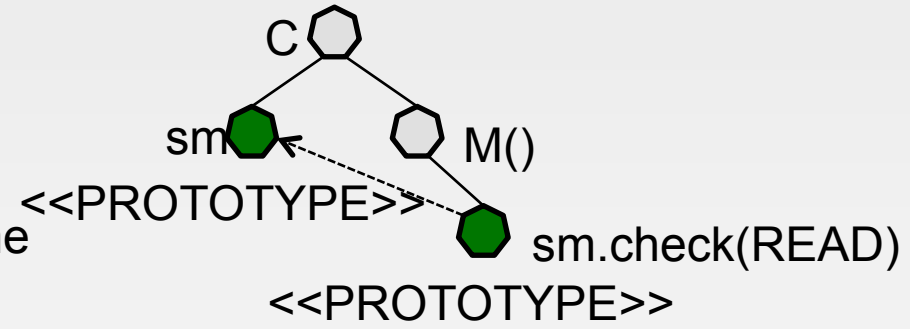
- Use traces to determine source fragment of model elements

Change vs. Place	„Inside" Fragment	„Between" Fragments
Insertion	Add in original fragment	(1) Add to hook fragment (2) Add to prototype (3) Add new fragment and compostion link
Deletion	Delete in original fragment	(1) Delete content of prototype (2) Delete composition link (3) Delete hook
Modification	Modify original model element	n/a

Invasive Software Composition (ISC) with Reuseware in a nutshell

- Fragments





Fragment A:

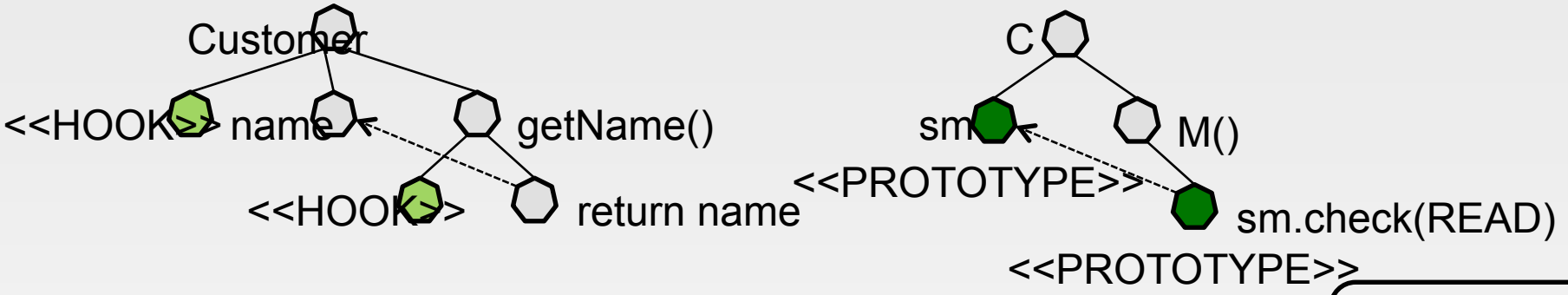
```
public class
Customer {
  <<HOOK>>
  String name;
  public String
getName(){
    <<HOOK>>
    return name;
  }
}
```

Fragment B:

```
public class C {

SecurityManager
sm =
    new
SecurityManager(
);

  public void
M(){

sm.check(READ);
```

Invasive Software Composition (ISC) with Reuseware in a nutshell

- Composition Interface/Composition Program



Fragment A:

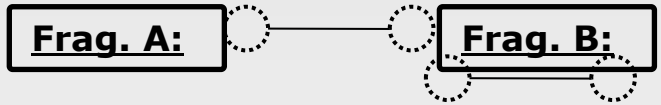
```
public class
Customer {
  <<HOOK>>
  String name;
  public String
getName(){
    <<HOOK>>
    return name;
  }
}
```

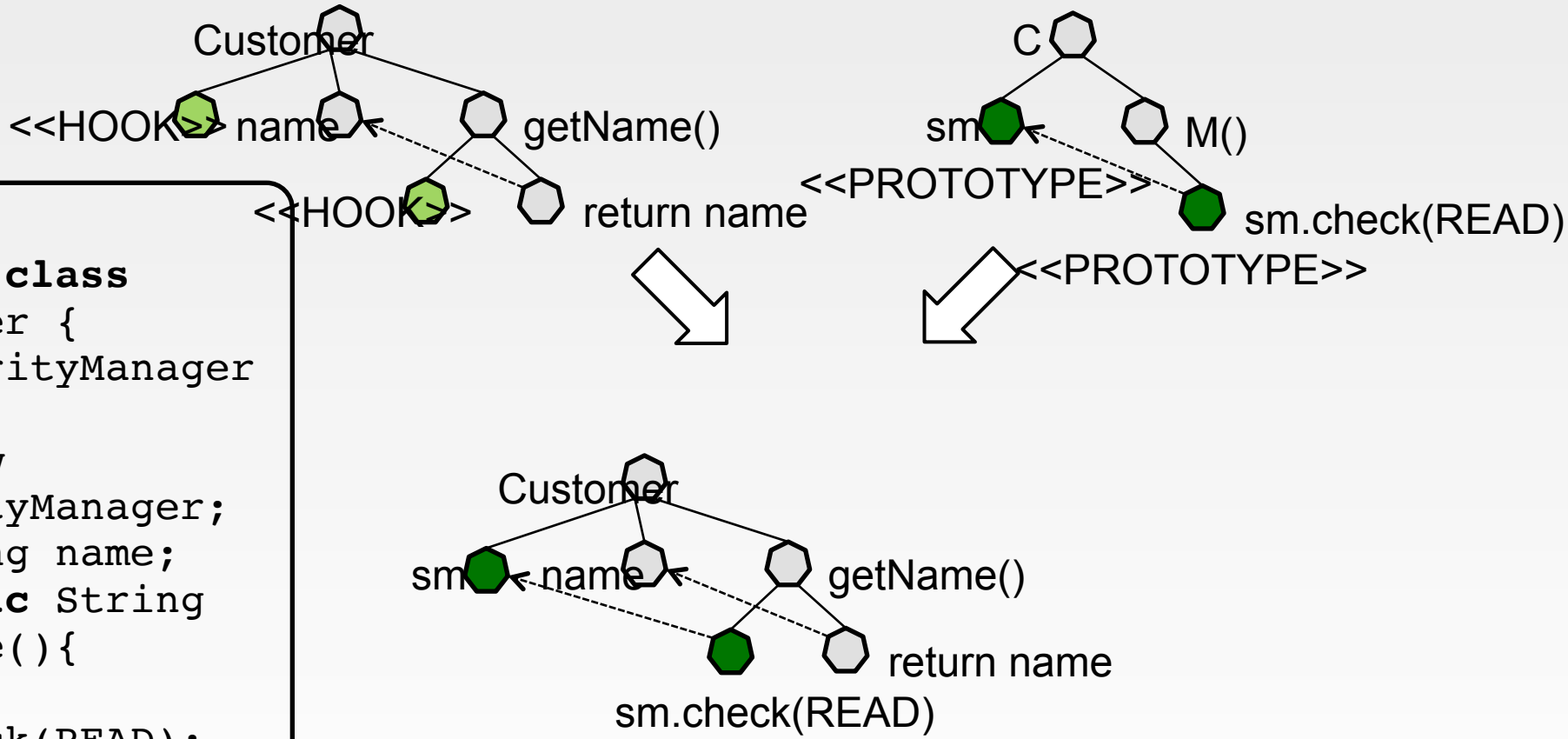
Fragment B:

```
public class C {

SecurityManager
sm =
    new
SecurityManager(
);

  public void
M(){

sm.check(READ);
  }
}
```

Invasive Software Composition (ICS) with Reuseware in a nutshell

- Composition





Result:

```
public class
Customer {
  SecurityManager
sm =
    new
SecurityManager;
  String name;
  public String
getName(){

sm.check(READ);
    return name;
  }
}
```