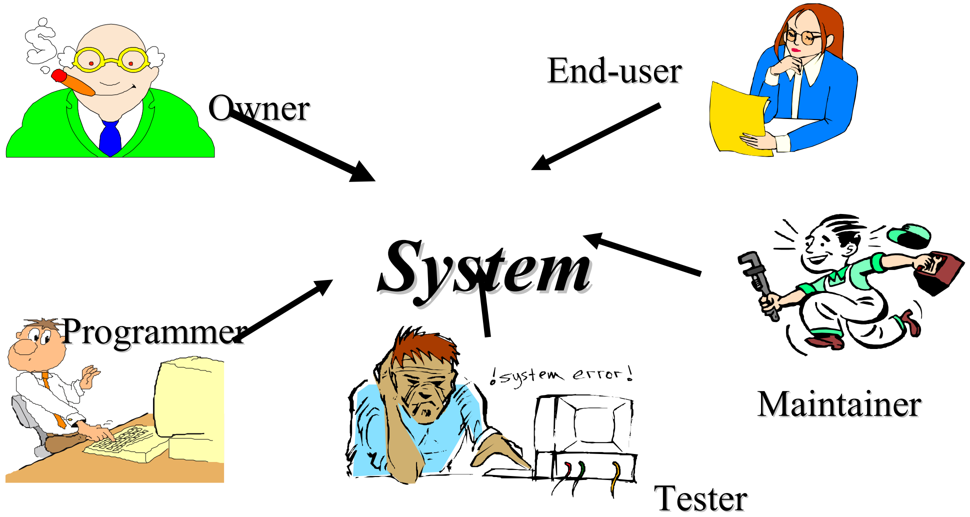
Realizing Correspondences in Multi-viewpoint Specifications

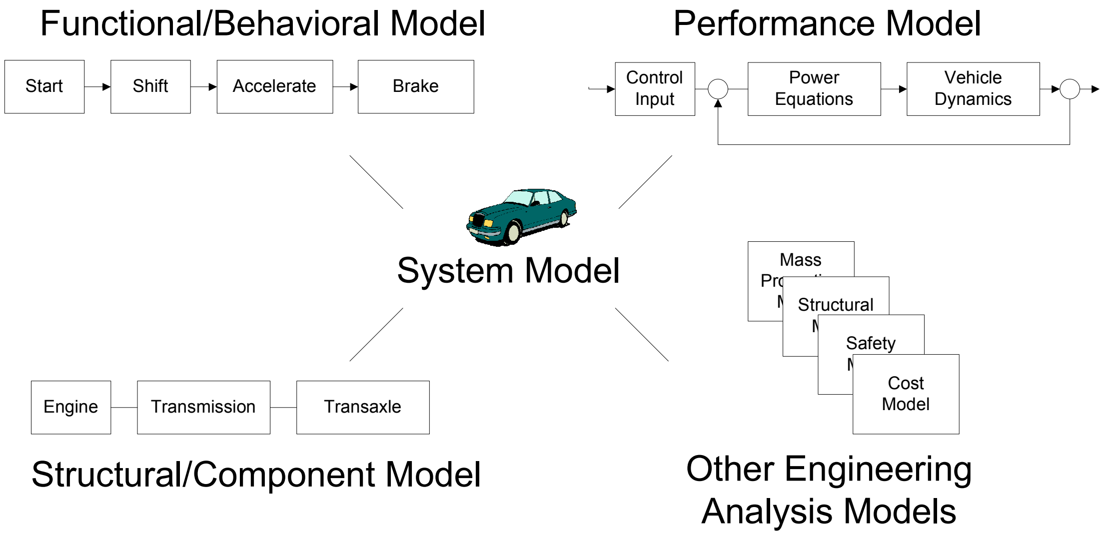
J.R. Romero, A. Vallecillo
Atenea Group
http://atenea.lcc.uma.es

GRACE Meeting on Bi-Directional Transformations
Japan, Dec 2008

Multiviewpoint specifications

Different stakeholders' views

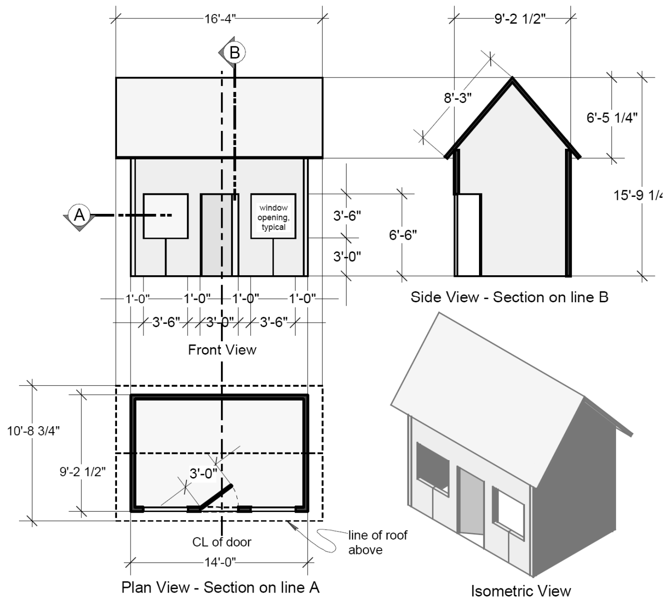




Multiple aspects of a system: Consistency

Multiviewpoint specifications

- Viewpoint modeling tackles complexity but introduces other problems
  - What is (in) a multiviewpoint specification?
  - Viewpoint integration?
  - Change propagation?
  - Viewpoint synchronization?
  - And many others...



What is a Multi-viewpoint Specification?

Definition 1 (Initial) *A* System Specification *consists of a set of views* $V = \{V_1, \ldots, V_n\}$. *Each view* V_i *is a model that conforms to a metamodel* $\mathcal{M}_i$ *(the viewpoint language).*

- This is the approach used by most EAFs
- No correspondences between the viewpoint elements… … or trivially based on name matching
- Others assume the existence of a global metamodel

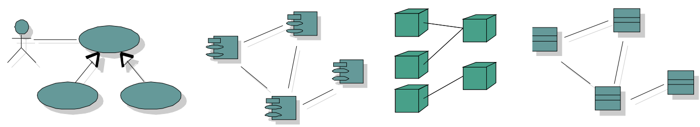
A global metamodel

- Easier to manipulate from a theoretical point
- Simplifies reasoning about consistency

BUT…

- The granularity and level of abstraction of the viewpoints can be **arbitrarily** different
- The viewpoints may have very **different formal semantics**
- Should it consist of the **intersection** or of the **union** of all viewpoints elements?
  - Both approaches have serious problems with extensibility and expressiveness (not to mention complexity of the second approach – think in the UML 2.0 metamodel)

- Only valid if viewpoints are **tightly coupled**!!! (semantically speaking)

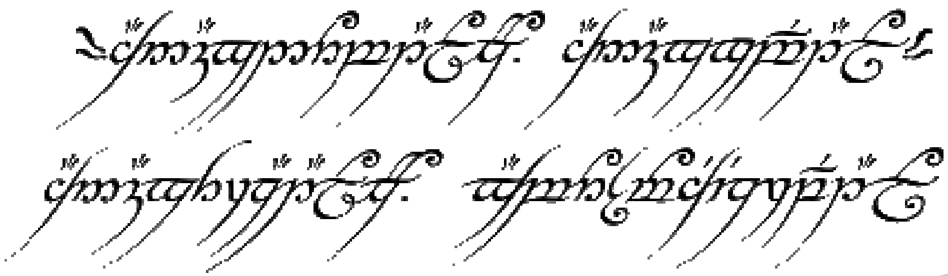


Sauron's approach to metamodeling
(e.g., OMG's UML metamodel)

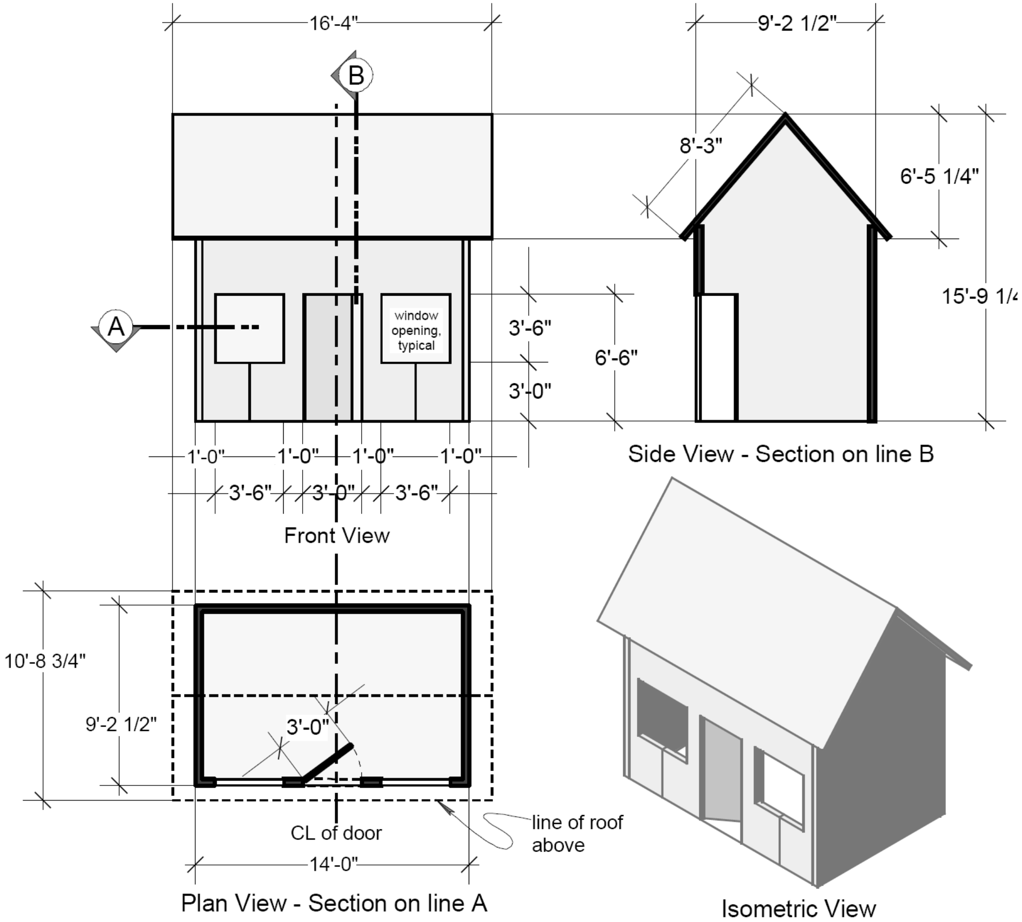
The lord of the Metamodels

(obviously, adapted)

Three notations for the Structure modelers under the sky,
Seven for the Behavior modelers in their halls of stone,
Tree for mortal Packagers doomed to die,
One for the Designer of the Whole System on his dark throne
In the Land of Mordor where the Shadows lie.
One Metamodel to rule them all, One Metamodel to find them,
One Metamodel to bring them all and in the darkness bind them
In the Land of Mordor where the Shadows lie.



Correspondences: Orthographic projections

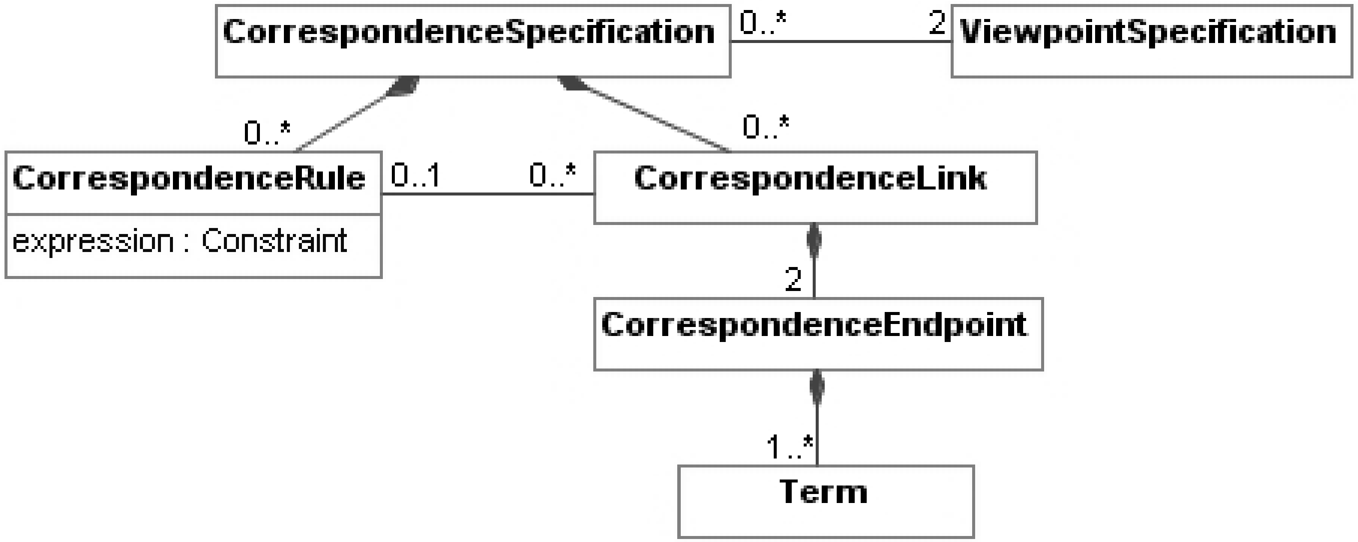


Multiviewpoint Specification

~~**Definition 1 (Initial)** *A* System Specification *consists of a set of views* $V = \{V_1, \ldots, V_n\}$. *Each view* V_i *is a model that conforms to a metamodel* $\mathcal{M}_i$ *(the viewpoint language).*~~

Definition 2 (With explicit correspondences) *A* System Specification *consists of a set of views* $V = \{V_1, \ldots, V_n\}$ *and a set of correspondences* $C = \{C_{(1,2)}, C_{(1,3)}, \ldots, C_{(n-1,n)}\}$ *between the views. Each view* V_i *is a model that conforms to a metamodel* $\mathcal{M}_i$ *(the viewpoint language). Correspondences are also models, and each* $C_{(i,j)}$ *conforms to a correspondence metamodel* $\mathcal{C}$. [1]

ODP Correspondence metamodel



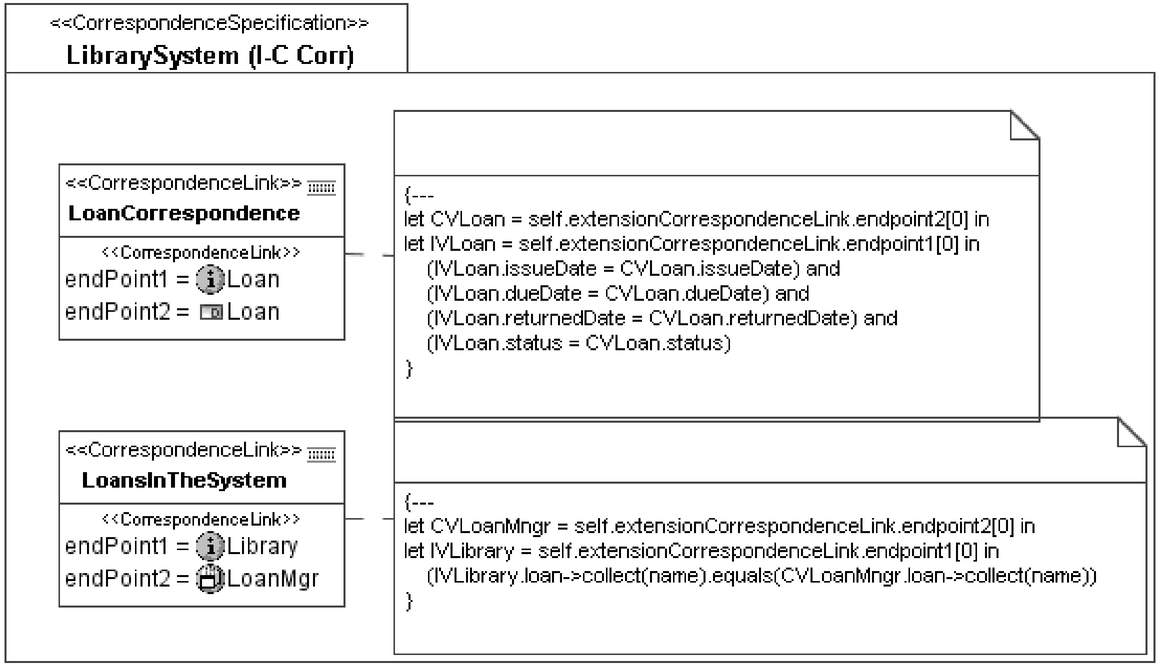
Correspondences

- Identify sets of related elements in each view
  - Defined in terms of ODP correspondenceSpecifications
  - Could be just UML traces or weaving models, too

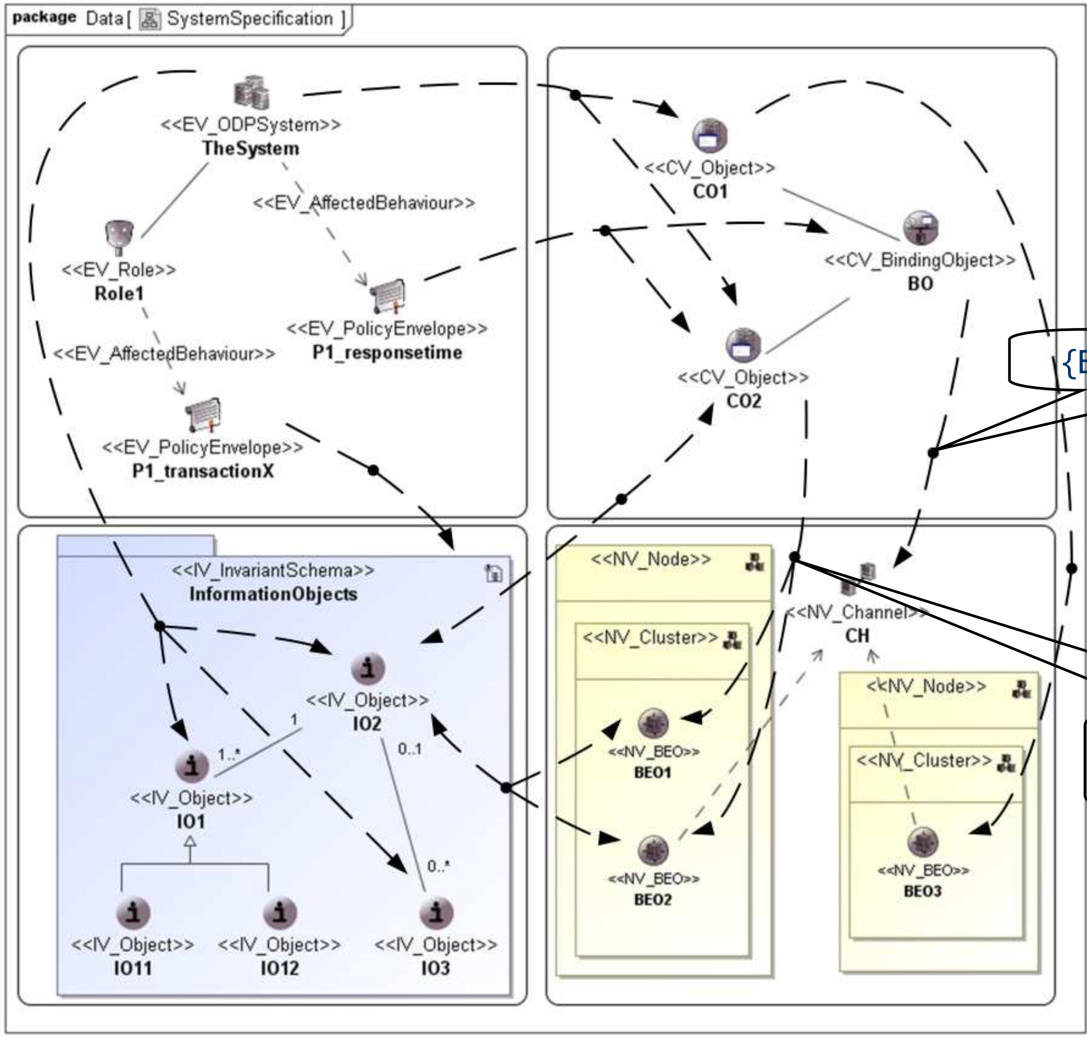
- Examples (from RM-ODP)

(A) Correspondence between **Loan** information and computational objects

(B) The sets of **Loan** instances in the information view should be consistent with the objects stored by the **LoanMgr** component of the computational view, which contains the loans stored in the application's database



Correspondences



{BO.name = CH.name}

{BEO1.x = CO1.x and
BEO2.x = CO1.x}

Required Correspondences

- Identify sets of related types (classes)
  - Defined by (directed) transformation functions; or
  - Defined by (bidirectional) transformations; or
  - Could be just mere traces…
- Examples (from RM-ODP)
  - "Each **computational object** that is not a binding object corresponds to a set of one or more **basic engineering objects** (and any **channels** which connect them). All the basic engineering objects in the set correspond only to that computational object"
  - "Except where transparencies which replicate objects are involved, each **computational interface** corresponds exactly to one **engineering interface**, and that engineering interface corresponds only to that computational interface"
  - "Where there is a correspondence between enterprise and information elements, the specifier has to provide…
    …for each **action** in the enterprise specification, the **information objects** (if any) subject to a **dynamic schema** constraining that action"

Expressing well-formed correspondences

Correspondences are not enough…

<u>Definition</u> 3 (With well-formed correspondences)
A System Specification *consists of a set of views* $V = \{V_1, \ldots, V_n\}$, *a set of correspondences* $C = \{C_{(1,2)}, C_{(1,3)}, \ldots, C_{(n-1,n)}\}$ *between the views, and a set of rules* $R = \{r_1, \ldots, r_k\}$ *that describe the constraints that the correspondences of* C *should fulfil in order for a specification to be well-formed. Each view* V_i *is a model that conforms to a metamodel* $\mathcal{M}_i$ *(the viewpoint language). Correspondences are also models, and* $C_{(i,j)}$ *conforms to a correspondence metamodel* $\mathcal{C}$*. Rules are expressed as constraints on the correspondence elements, using any constraint language (e.g., OCL).*

Well-formed rules for correspondences

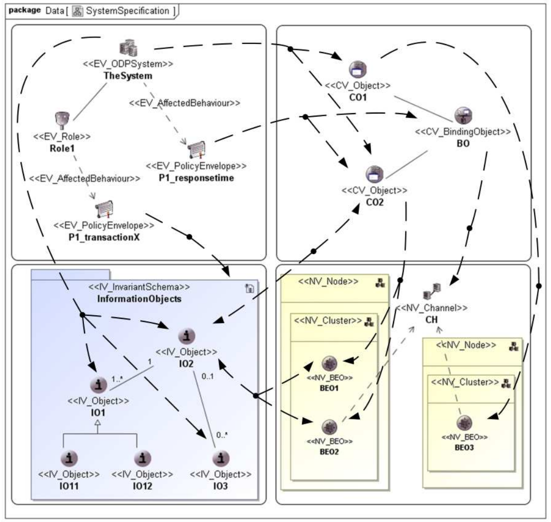
- Define constraints and invariants on the set of correspondences between the viewpoints
  - Check that the correspondences obey the ODP rules
  - Check that no correspondences are missing
- Examples (from RM-ODP)
  - "Each computational object that is not a binding object corresponds to a set of one or more basic engineering objects (and any channels which connect them)"

```
context CorrespondenceSpecification inv:
 let CVOBJECTS = self.viewpointSpecification->
       select(o:CV_Metamodel::CV_Object | not oclIsTypeOf(CV_Metamodel::Binding)) in
 let NVOBJECTS = self.viewpointSpecification->select(n : NV_Metamodel::BEO) in
 let CORRESPONDENCES = CorrespondenceLink->allInstances()->select(…) in

  (CVOBJECTS->size()) = (CORRESPONDENCES->size()) and
  NVOBJECTS->forAll(n | CVOBJECTS->exists(o | isRelated(o,n)) and
  CVOBJECTS->forAll(o1,o2 | isRelated(o1,n) and isRelated(o2,n) implies o1 = o2)))
```

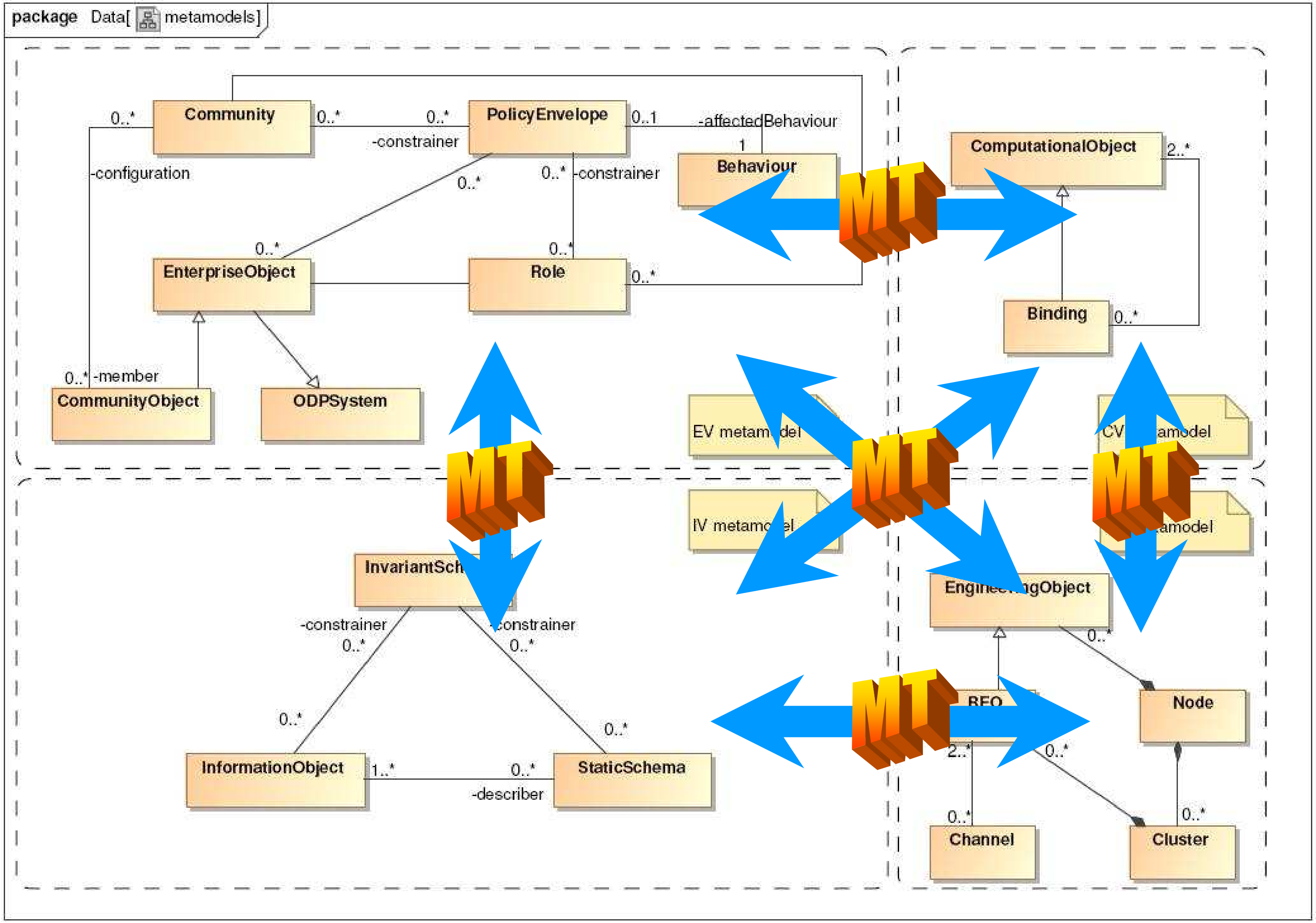
However…

- Scalability?
  - The number of correspondences does not scale at all!
  - How to define correspondences over complete sets of elements at once?
- Usability?
  - How to deal with correspondences without obtaining cluttered and unusable models?
  - How to visualize the models?
- Completeness
  - How do we check that all required correspondences are indeed specified?



- Expressiveness
  - How to describe the well-formed rules that the set of correspondences between views elements should obey
- We need better tool support for dealing with correspondences between the views
- Case studies:
  - RM-ODP; Model-Driven Web Engineering (WEI, UWE)

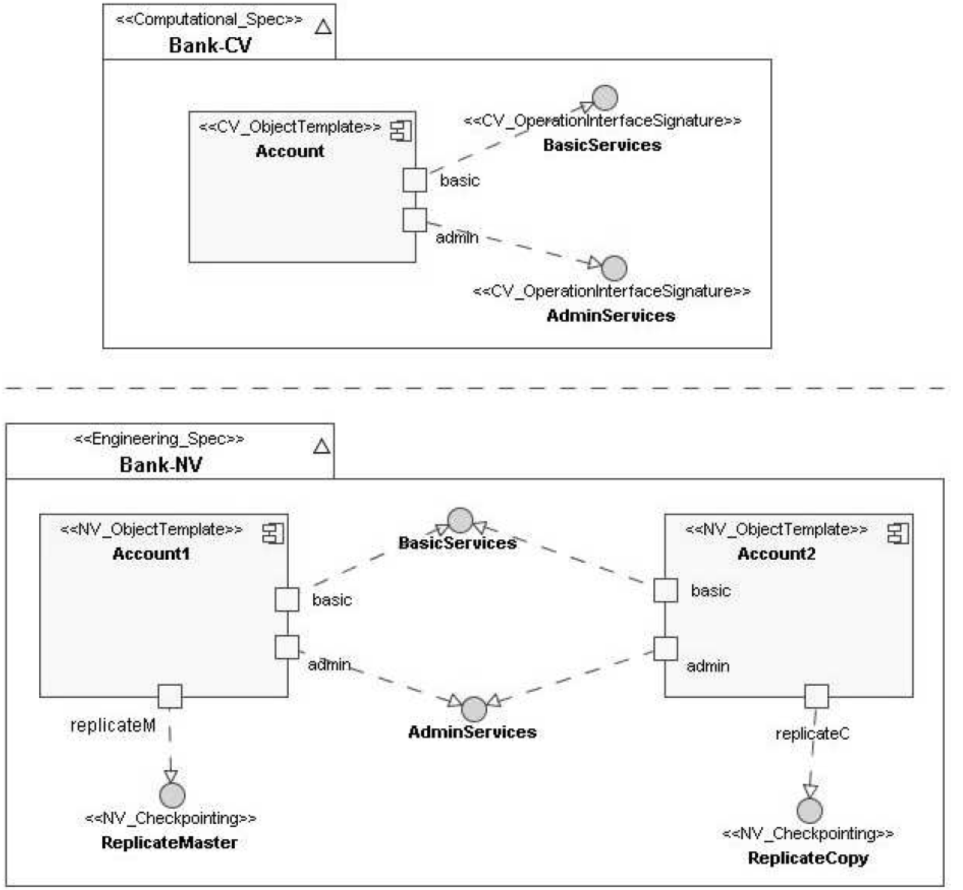
Correspondences at Metamodel level



An example

[MDEIS 2006]

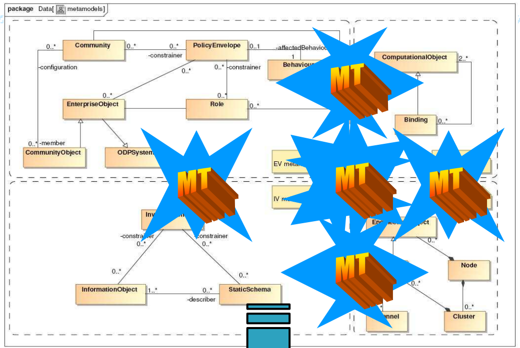
```
relation cv-account2twonv-accounts {
  domain cv a:Component {name=“Account”}
  domain nv a1:Component {name=“Account1”}
  domain nv a2:Component {name=“Account2”}
  when { a.stereotypedBy(”CV_Object”) }
  where {
     a.stereotypedBy(”CV_Object”) and
     a1.stereotypedBy(”NV_BEO”) and
     a2.stereotypedBy(”NV_BEO”) and
     sameODPInterfaces(a,a1) and
     sameODPInterfaces(a,a2)
  }
}
```



Our Approach

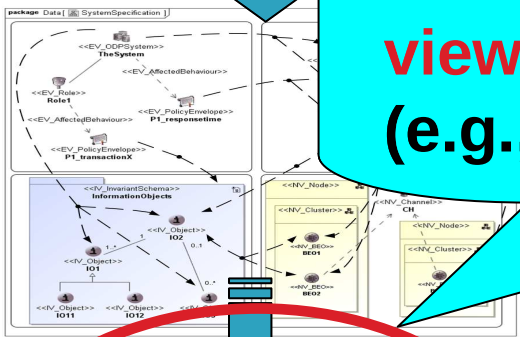
- Use QVT **relations** to define correspondences "intensionally"
- Generate the associated **trace instances** from QVT relations
- Trace instances can then be transformed to correspondenceSpecifications at model level (i.e., correspondences are given "extensionally")
- Well-formed rules are then checked against this full specification at model level
- The user normally works at the two levels!!!

Some issues

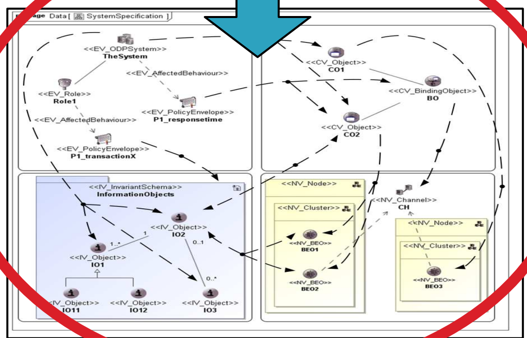


The user defines Relations at metamodel level

How to obtain different
views of the correspondences?
(e.g., per relation, user-defined, etc.)



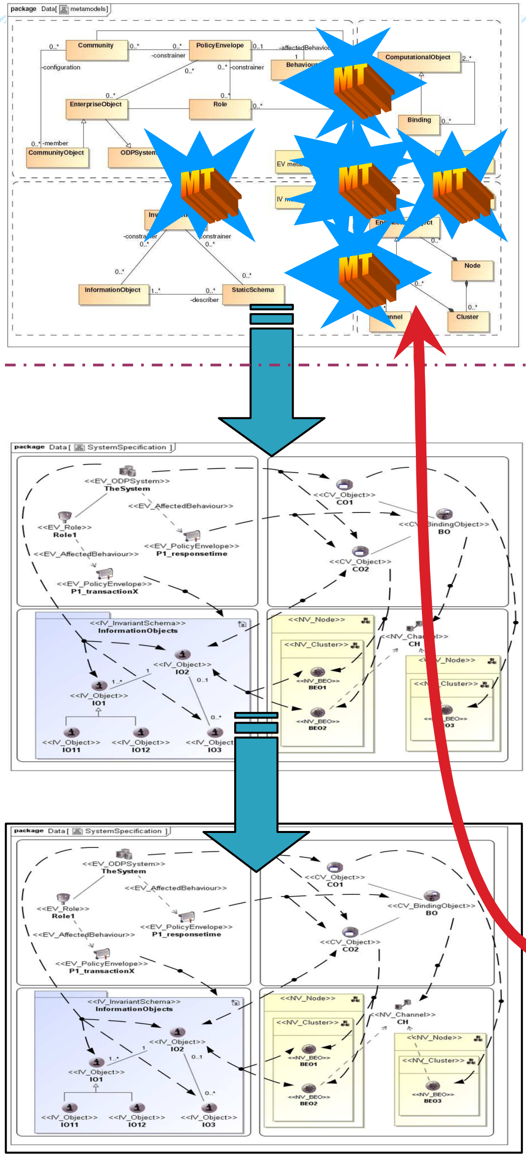
Transformation into correspondenceSpecifications



The final model with all correspondences!

Well-formed rules are then checked in the set of correspondences

Some issues



The user defines Relations at metamodel level

Generation of Trace instances

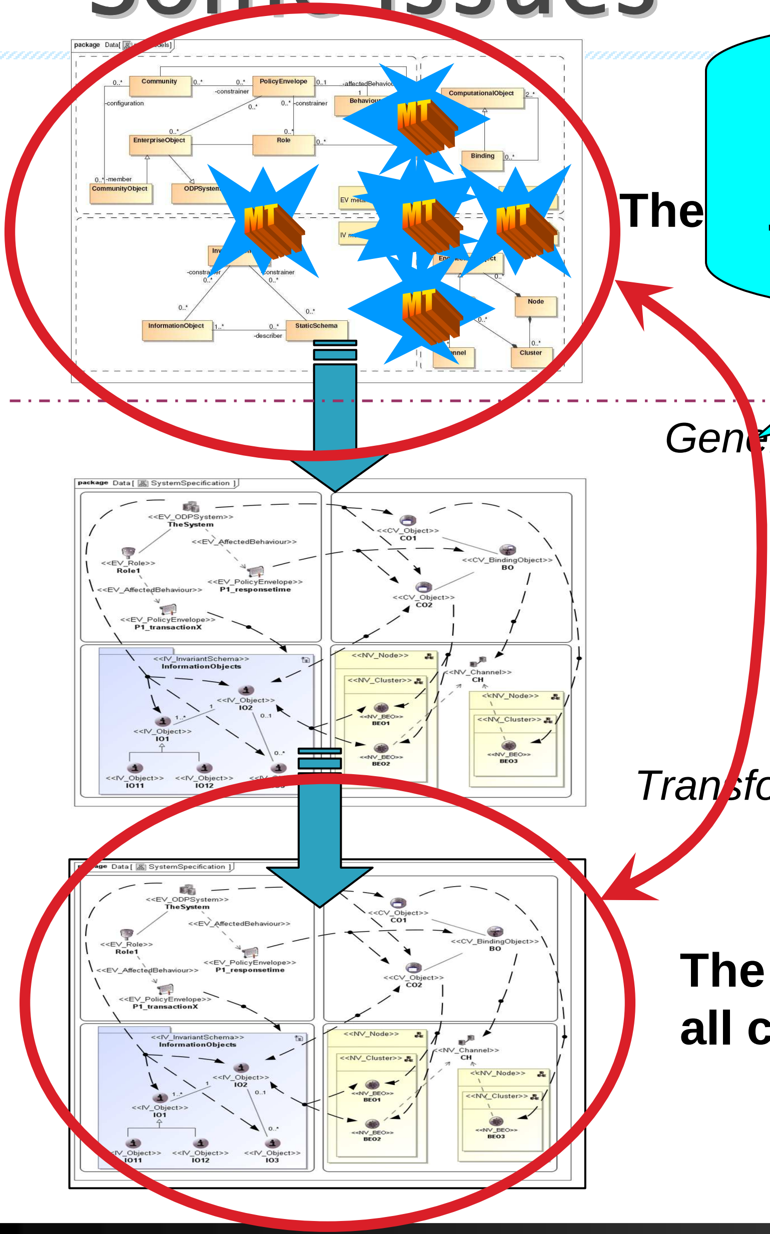
How to express the well-formed rules at the meta-model level?

Transformation into correspondenceSpecifications

The final model with all correspondences!

Well-formed rules are then checked in the set of correspondences

Some issues



The

How to maintain the consistency between the correspondences and the QVT transformations above?

Generation of Trace instances

Transformation into correspondenceSpecifications

The final model with all correspondences!

Well-formed rules are then checked in the set of correspondences

And now?

- Suppose that we already count on a tool for expressing correspondences between views…
- What can I use it for?

Viewpoint synchronization(*)

- During its life cycle, a software system evolves and its specification changes
  - The specification of a view should not conflict with the specification of another view
  - A modification in a view may induce a modification in another views to preserve consistency
- One solution is the adoption and implementation of synchronization mechanisms able to propagate the changes on the related views

(*) Joint work with Alfonso Pierantonio and Romina Eramo [WODPEC'08]

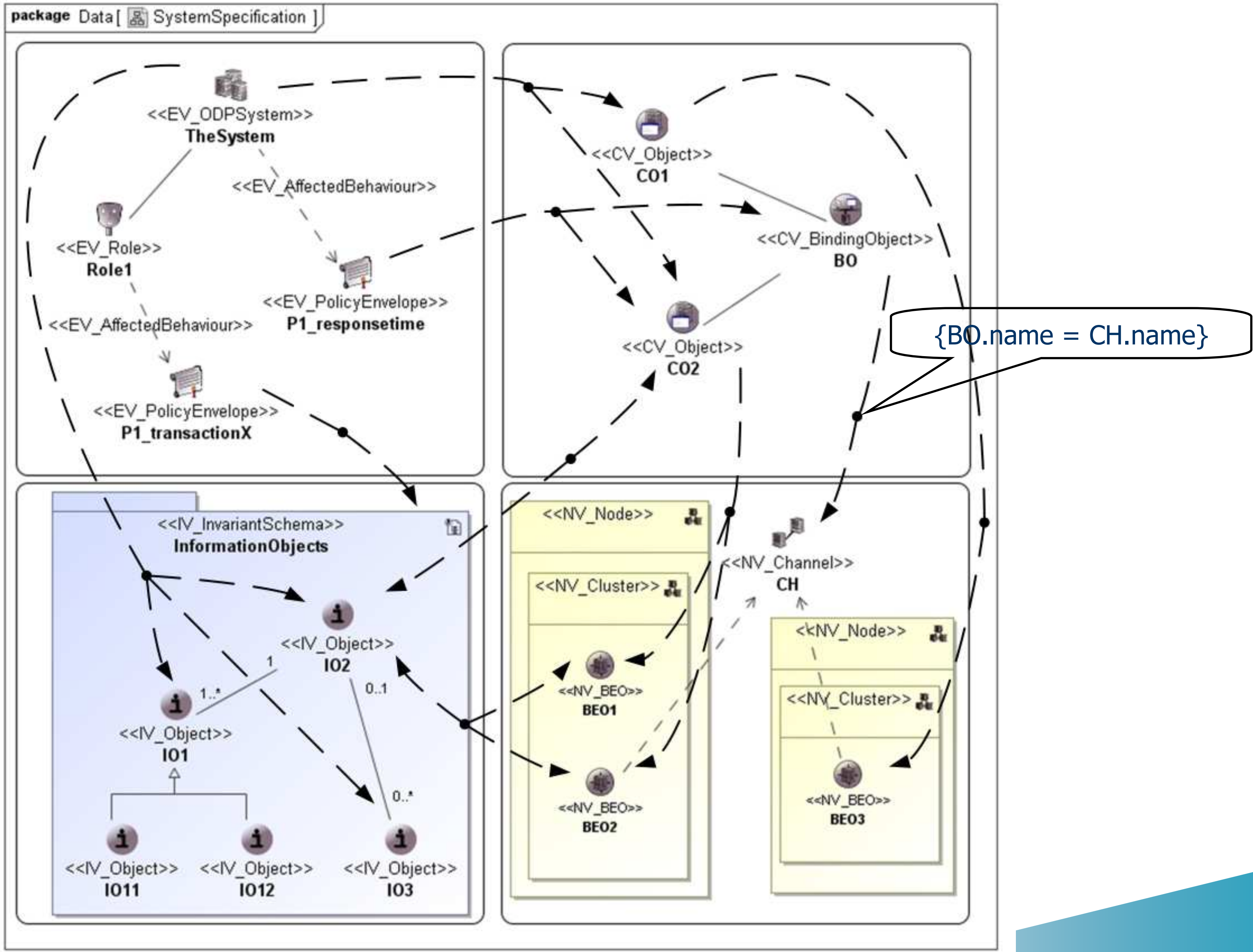
Viewpoint Evolution

- Systems are continuously changing
  - Changes may occur in the views by adding, modifying o deleting elements
  - Modifications are propagated through correspondences to elements in other views

- Propagated changes can introduce inconsistencies, which need to be found and solved
  - View synchronization mechanisms and tools are required

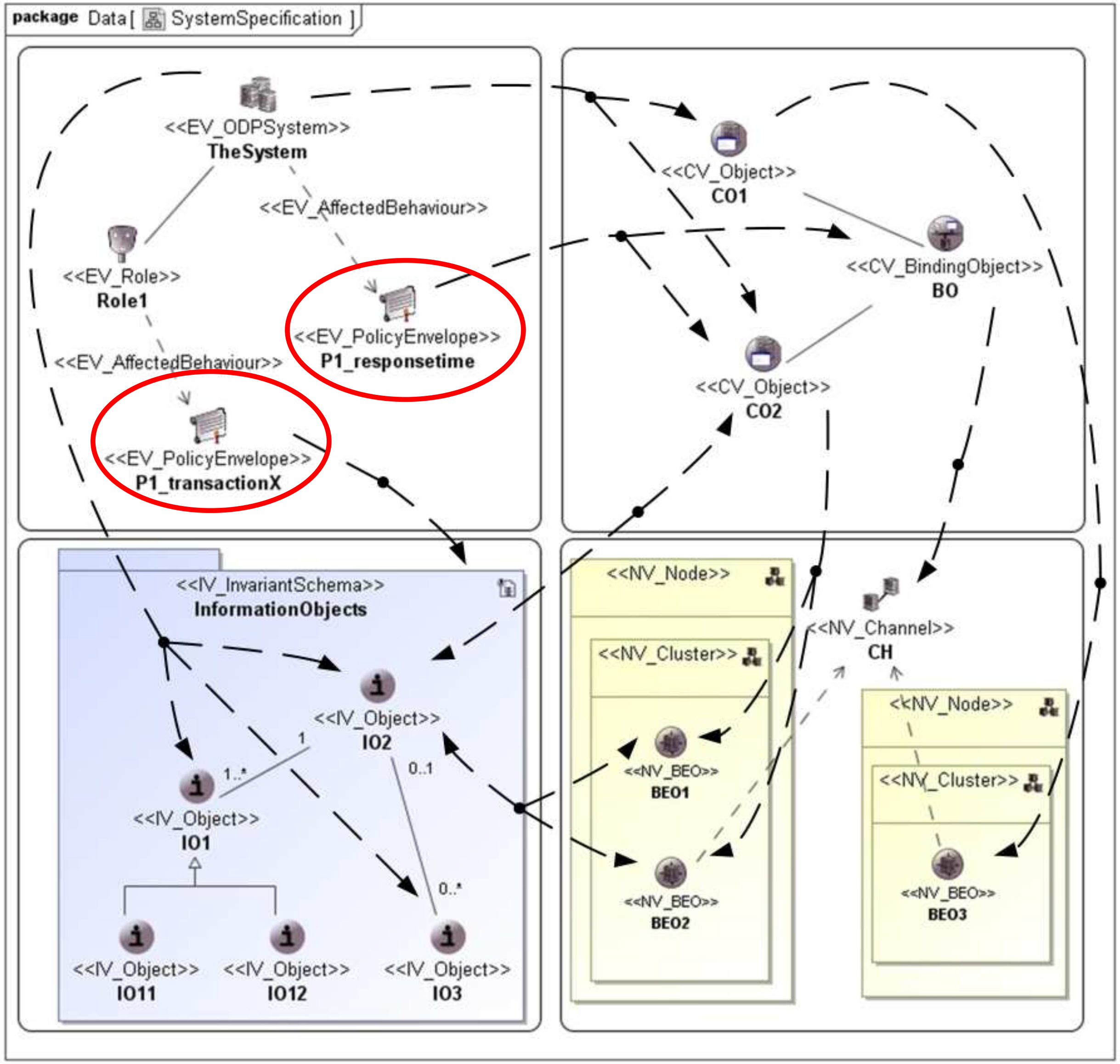
Problems

- Correspondences may not provide all information needed to perform automatic synchronization
  - Sometimes Correspondence rules help (e.g. { BO.name = CH.name })


package Data [SystemSpecification]
<<EV_ODPSystem>>
TheSystem
<<EV_AffectedBehaviour>>
<<EV_Role>>
Role1
<<EV_PolicyEnvelope>>
P1_responsetime
<<EV_AffectedBehaviour>>
<<EV_PolicyEnvelope>>
P1_transactionX
<<CV_Object>>
CO1
<<CV_BindingObject>>
BO
<<CV_Object>>
CO2
{BO.name = CH.name}
<<IV_InvariantSchema>>
InformationObjects
<<IV_Object>>
IO2
<<IV_Object>>
IO1
<<IV_Object>>
IO11
<<IV_Object>>
IO12
<<IV_Object>>
IO3
1
1..*
0..1
0..*
<<NV_Node>>
<<NV_Cluster>>
<<NV_BEO>>
BEO1
<<NV_BEO>>
BEO2
<<NV_Channel>>
CH
<<NV_Node>>
<<NV_Cluster>>
<<NV_BEO>>
BEO3

Problems

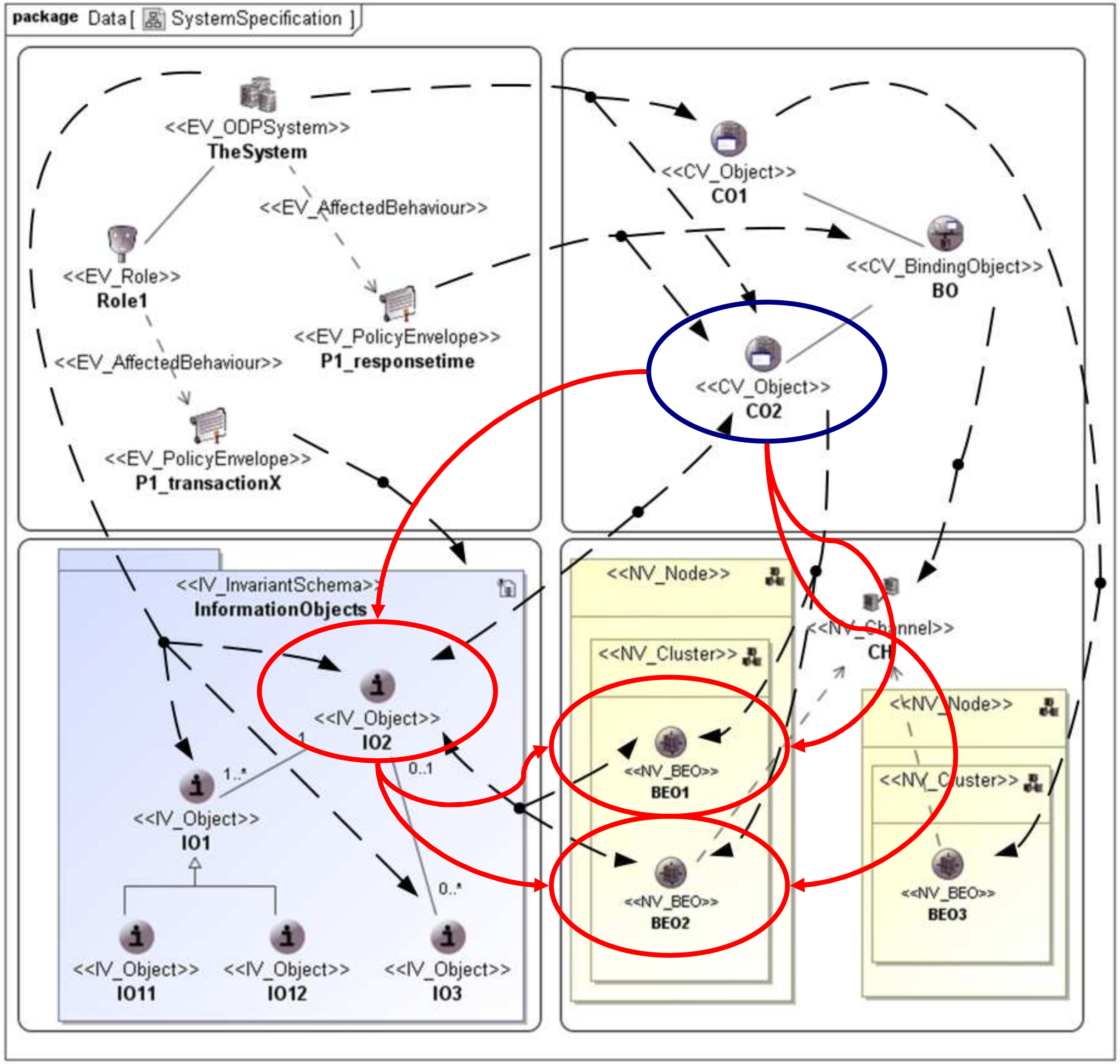
- Correspondences may not provide all information needed to perform automtic synchronization
  - Sometimes Correspondence rules help (e.g. { BO.name = CN.name })
  - Sometimes they are just "traces" (e.g. EV policies)


package Data [SystemSpecification]
<<EV_ODPSystem>>
TheSystem
<<EV_AffectedBehaviour>>
<<EV_Role>>
Role1
<<EV_PolicyEnvelope>>
P1_responsetime
<<EV_AffectedBehaviour>>
<<EV_PolicyEnvelope>>
P1_transactionX
<<CV_Object>>
CO1
<<CV_BindingObject>>
BO
<<CV_Object>>
CO2
<<IV_InvariantSchema>>
InformationObjects
<<IV_Object>>
IO2
<<IV_Object>>
IO1
<<IV_Object>>
IO11
<<IV_Object>>
IO12
<<IV_Object>>
IO3
1
1..*
0..1
0..*
<<NV_Node>>
<<NV_Cluster>>
<<NV_BEO>>
BEO1
<<NV_BEO>>
BEO2
<<NV_Channel>>
CH
<<NV_Node>>
<<NV_Cluster>>
<<NV_BEO>>
BEO3

Problems

- Correspondences may not provide all information needed to perform automtic synchronization
  - Sometimes Correspondence rules help (e.g. { BO.name = CN.name })
  - Sometimes they are just “traces” (e.g. EV policies)
- “Ripple” effect
  - Changes need to be propagated through correspondences.
  - Some correspondences may define “cycles”, which may introduce problems

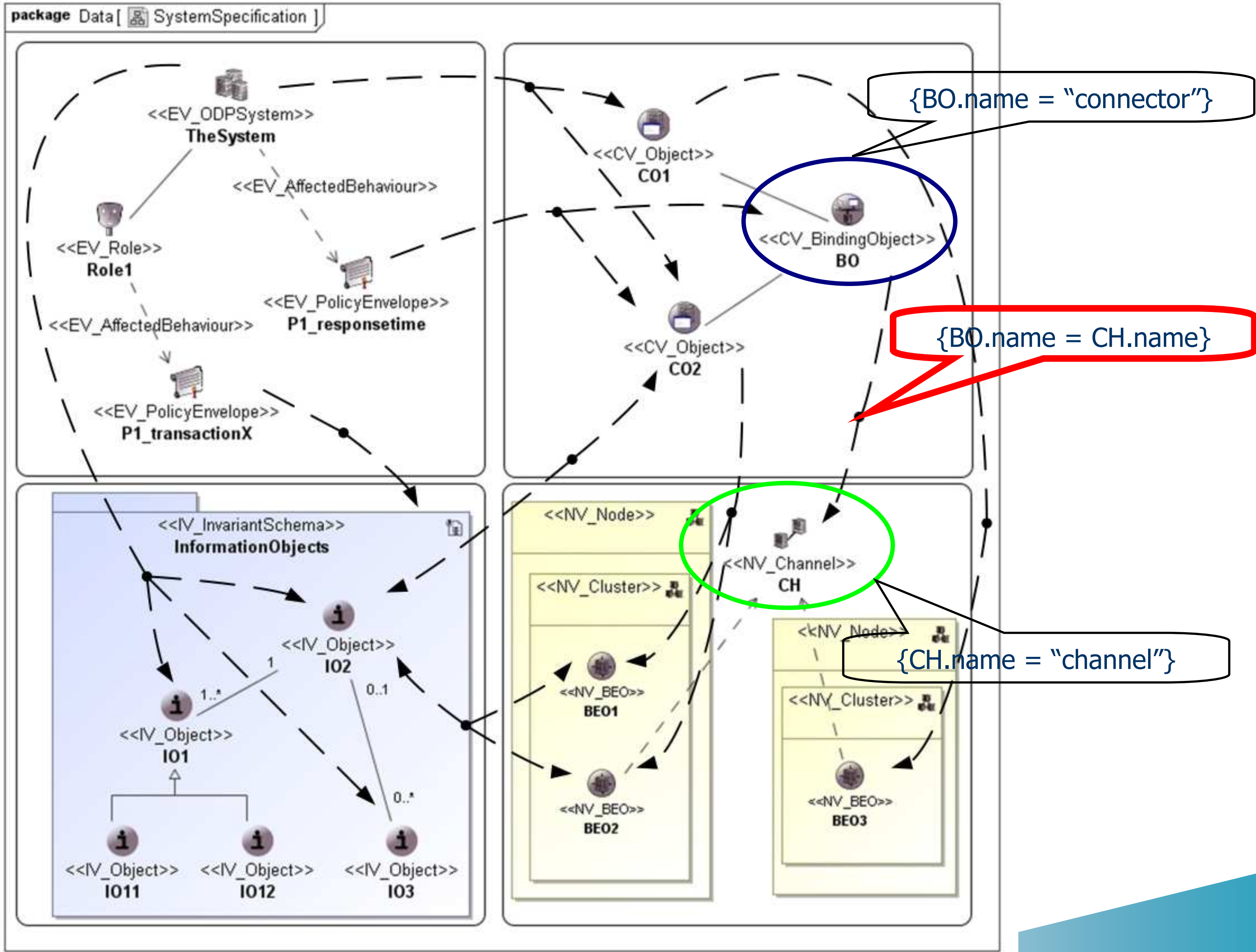
Viewpoint Modeling – Views



Problems

- Correspondences may not provide all information needed to perform automtic synchronization
  - Sometimes Correspondence rules help (e.g. { BO.name = CN.name })
  - Sometimes they are just "traces"
- "Ripple" effect
  - Changes need to be propagated through correspondences.
  - Some correspondences may define "cycles", which may introduce problems
- Distributed and independent changes
  - Changes independently introduced by different people may cause inconsistencies, too

Viewpoint Modeling – Views



Our goal

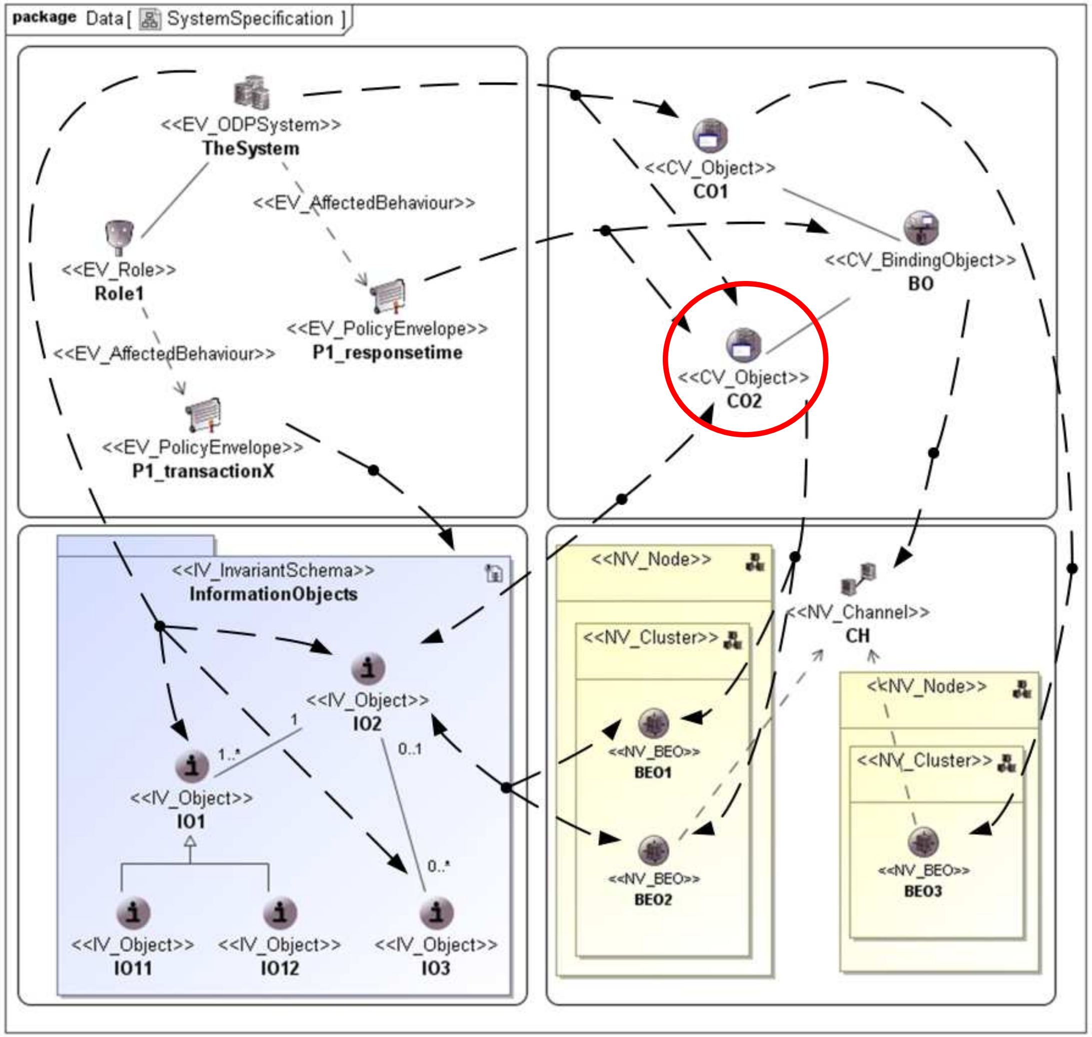
- An "engineering" approach to deal with the problem of viewpoint inconsistency management and synchronization
  - Semi-automated (user-guided)
  - Tool supported
- The "viewpoint synchronization" tool should be capable of helping the system designer:
  - identify the changes in the viewpoints,
  - propagate them to the rest of the viewpoints, and
  - (semi-automatically) manage and resolve inconsistencies

Change Management Approach

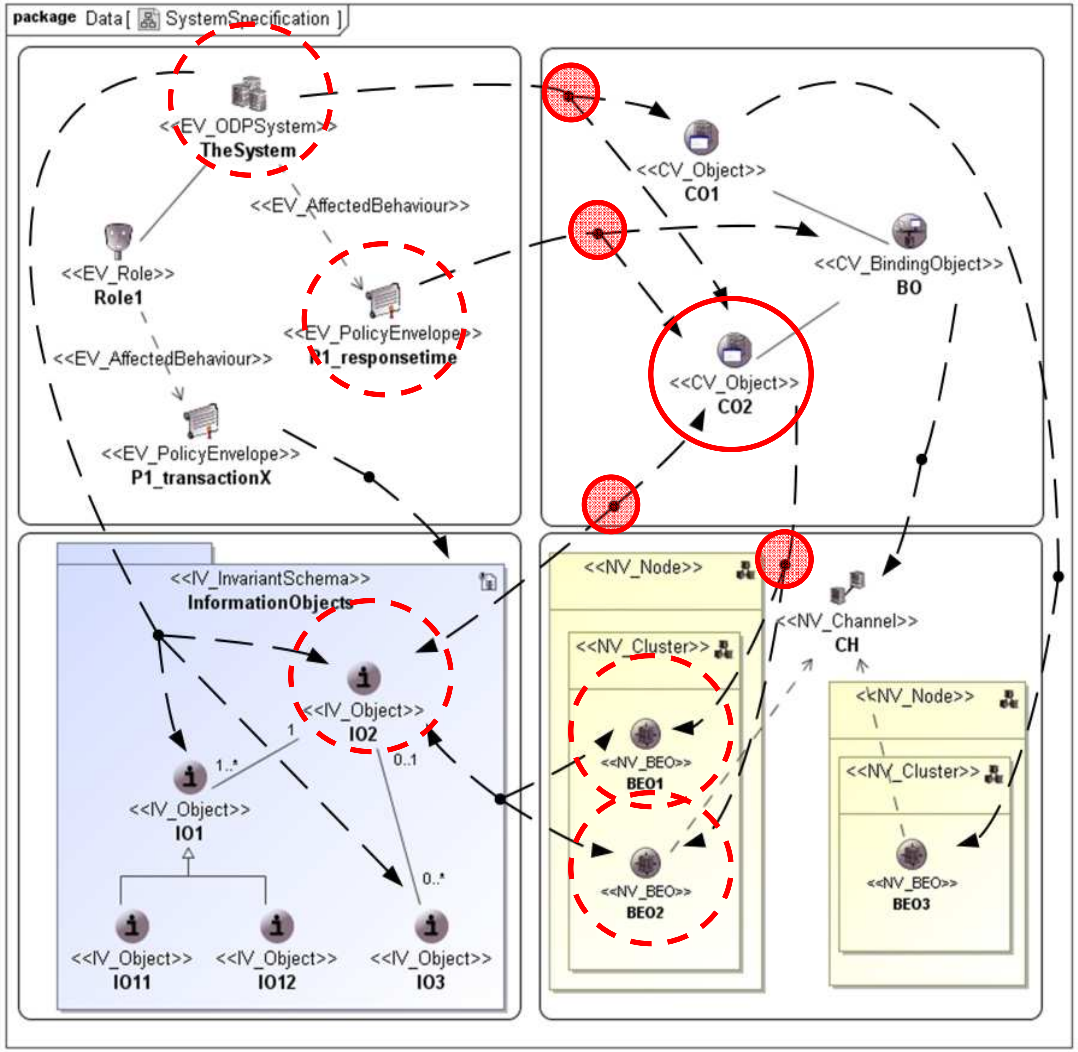
- The approach derives a set of models which represents all the possible consequences caused by the changes
  - Uses ASP to deal with non-deterministic derivations which represent alternative solutions to a given problem

- The approach consists of three (iterative) steps:
  1. Change identification
  2. Change classification and cascading
  3. Change commitment and propagation

1) Change identification (ModelDiff)



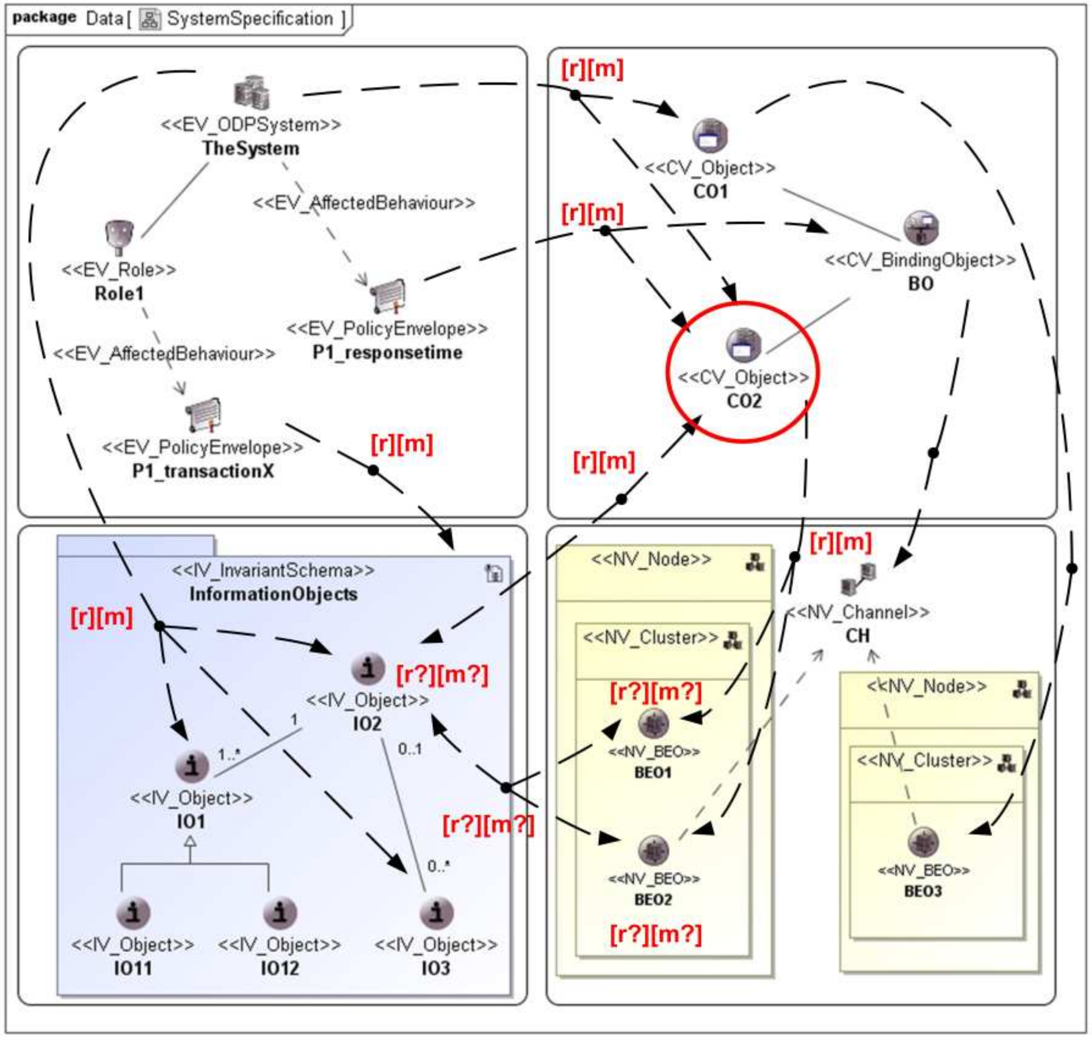
1’) Identification of related elements



2) Change classification and cascading

V_1 / V_2	Add_V	Rem_V	Mod_V	Add_C	Rem_C	Mod_C
Add_V				✓		✓
Rem_V		✓			✓	✓
Mod_V			✓		✓	✓
Add_C	✓		✓			
Rem_C		✓	✓			
Mod_C	✓		✓			

3) Proposal for change propagation



Tool support (ongoing)

- A visual tool for synchronizing the views and correspondences of a multi-view specification
- The goal is to guide the user in managing and browse the possible alternative adaptations
- The system designer can decide how to enforce changes in the related views in visual way
- It considers the potential effects on the rest of the system's views when a change in one element is recursively propagated to elements in other views through the correspondences (using ASP)

Realizing Correspondences in Multi-viewpoint Specifications

Thanks!

J.R. Romero, A. Vallecillo
Atenea Group
http://atenea.lcc.uma.es

GRACE Meeting on Bi-Directional Transformations
Japan, Dec 2008