Model Synchronization: Theory and Practice

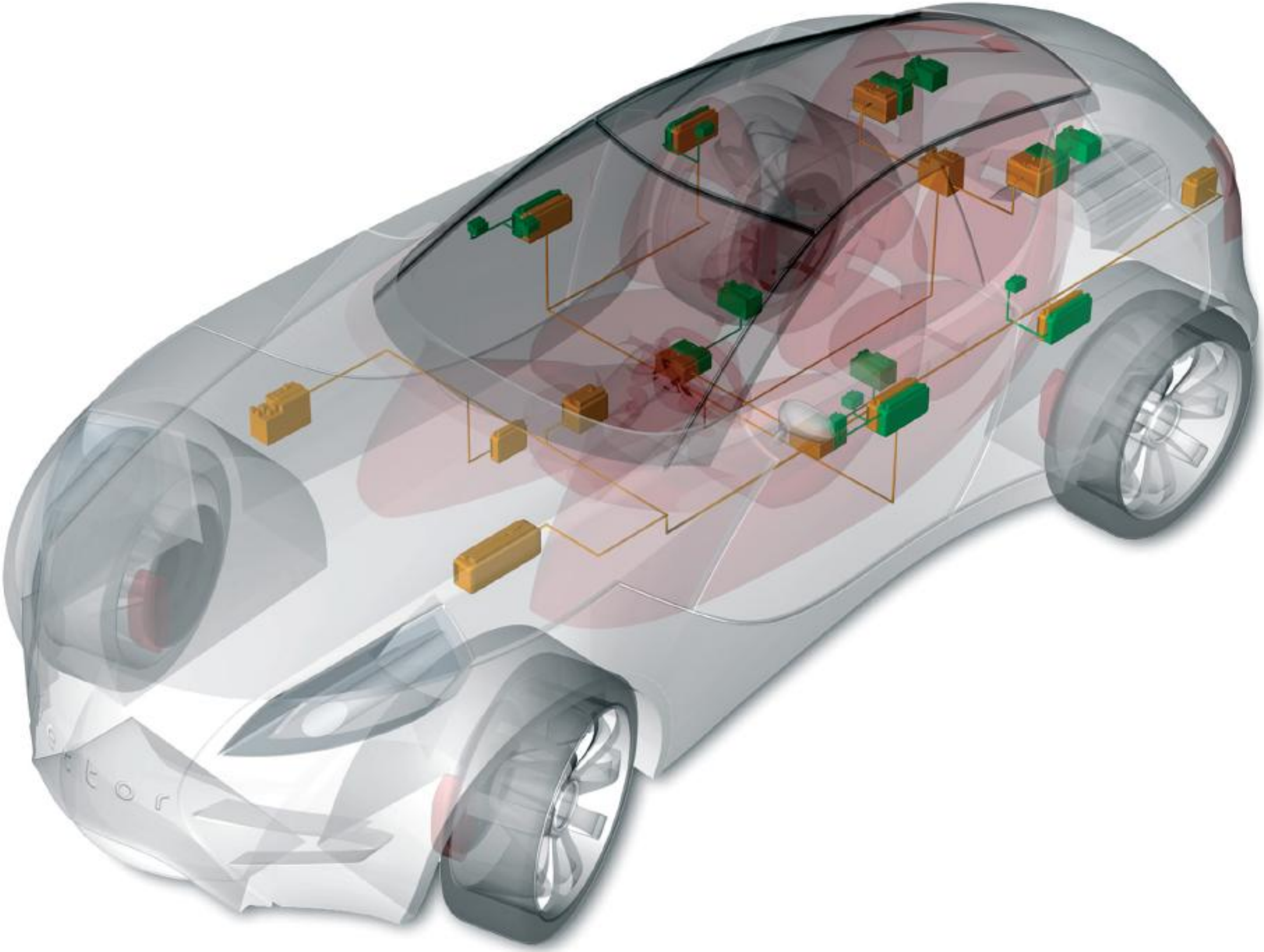
Krzysztof Czarnecki

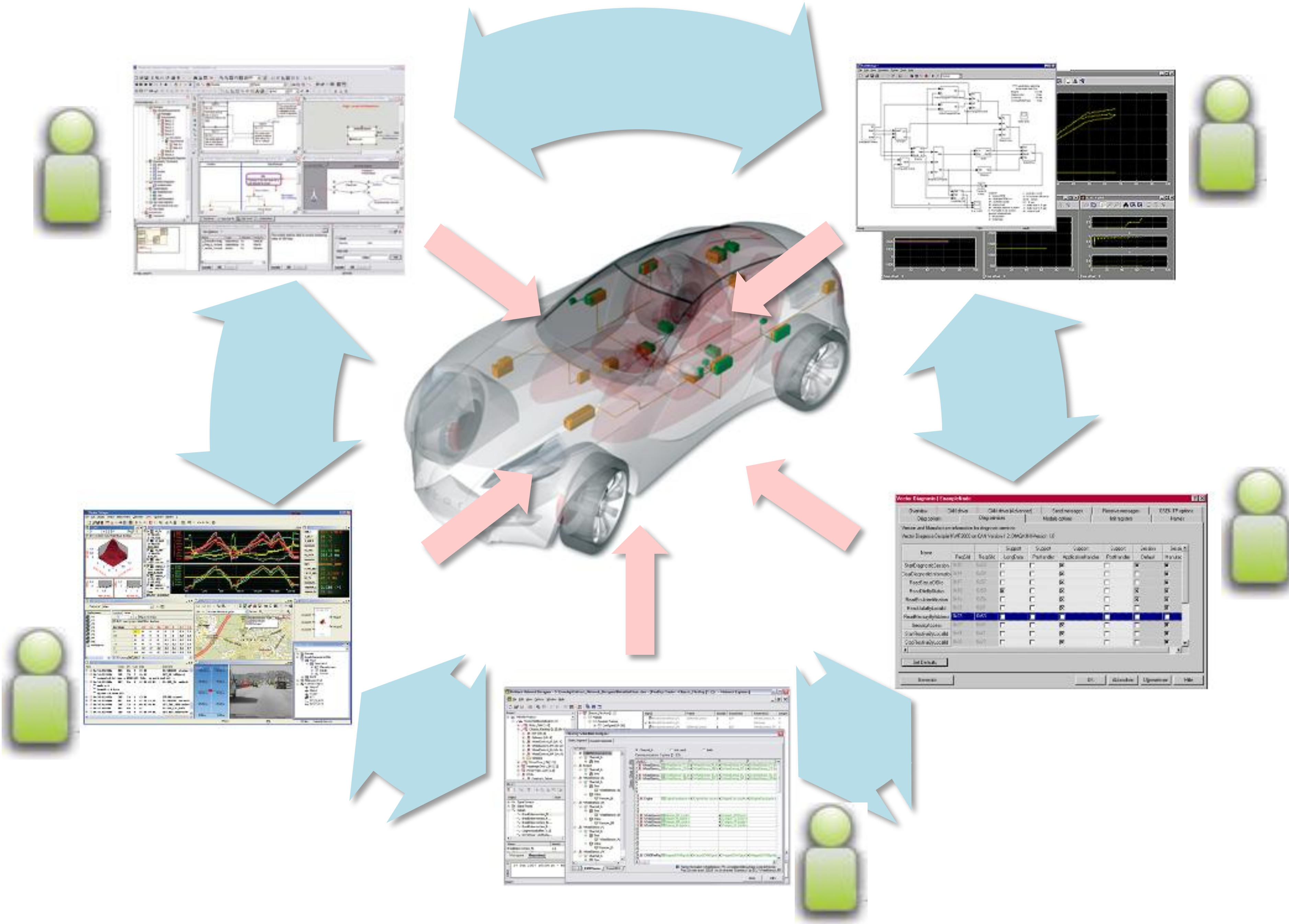
University of Waterloo

Canada

Acknowledgements

Michal Antkiewicz
Zinovy Diskin
Yingfei Xiong





Goals

Map out the problem space

Theory

- Sketch elements of an algebraic framework to model sync
- See Zinovy Diskin’s tutorial paper for more precise account [GTTSE’11]; also [JOT’11], [MODELS’11]

Practice

- Focus on practical examples
- Solutions to various problems in model sync

Roadmap

Single model consistency management

Multi-model consistency management

Examples

Replica synchronization

View synchronization

General overlap

Roadmap

Single model consistency management

Multi-model consistency management

Examples

Replica synchronization

View synchronization

General overlap

Single-Model Consistency Management

Consistency: Model satisfies some constraint

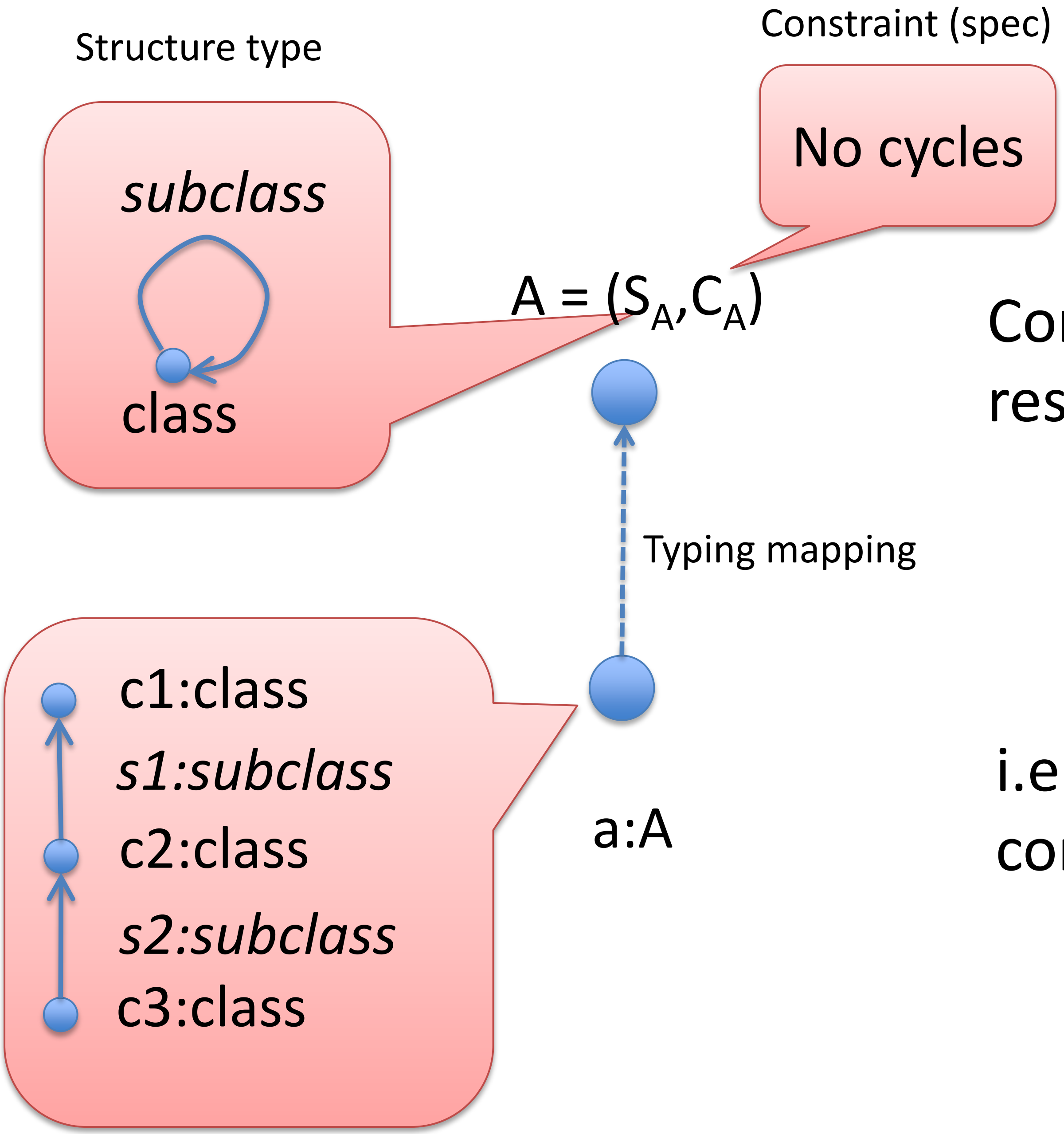
- E.g., well-formedness, instance space properties, behavioral correctness

Consistency management

- Check for constraint satisfaction
- Identify and explain sources of inconsistency
- Generate fix proposals

Examples

- Java type checking and quick fixes in Eclipse
- Alloy instance generation
- Behavioral model checking



Structure type

subclass

class

Constraint (spec)

No cycles

$A = (S_A, C_A)$

Typing mapping

c1:class

s1:subclass

c2:class

s2:subclass

c3:class

a:A

Consistency of **a** with respect to C_A:

$$a \models C_A$$

i.e., model **a** satisfies constraint C_A

Roadmap

Single model consistency management

Multi-model consistency management

Examples

Replica synchronization

View synchronization

General overlap

Multi-Model Consistency (aka model sync)

Complex notion

- Model overlaps, often implicit
- Global consistency of N models means consistency of any subset of them

Its management is complex, too

- Discover correspondences among models
- Update of a multi-model is multi-update
- Updates potentially done by different people

Consistency Management Operations

Matching

- Produces model correspondences
- Heuristic vs. precise matching

Consistency check wrt. correspondences

Resolution of conflicting updates

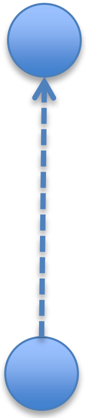
Update propagation

- May involve update translation

Consider Two Models

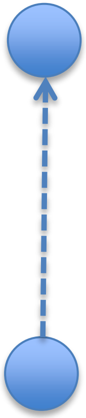
a
b

$A = (S_A,C_A)$

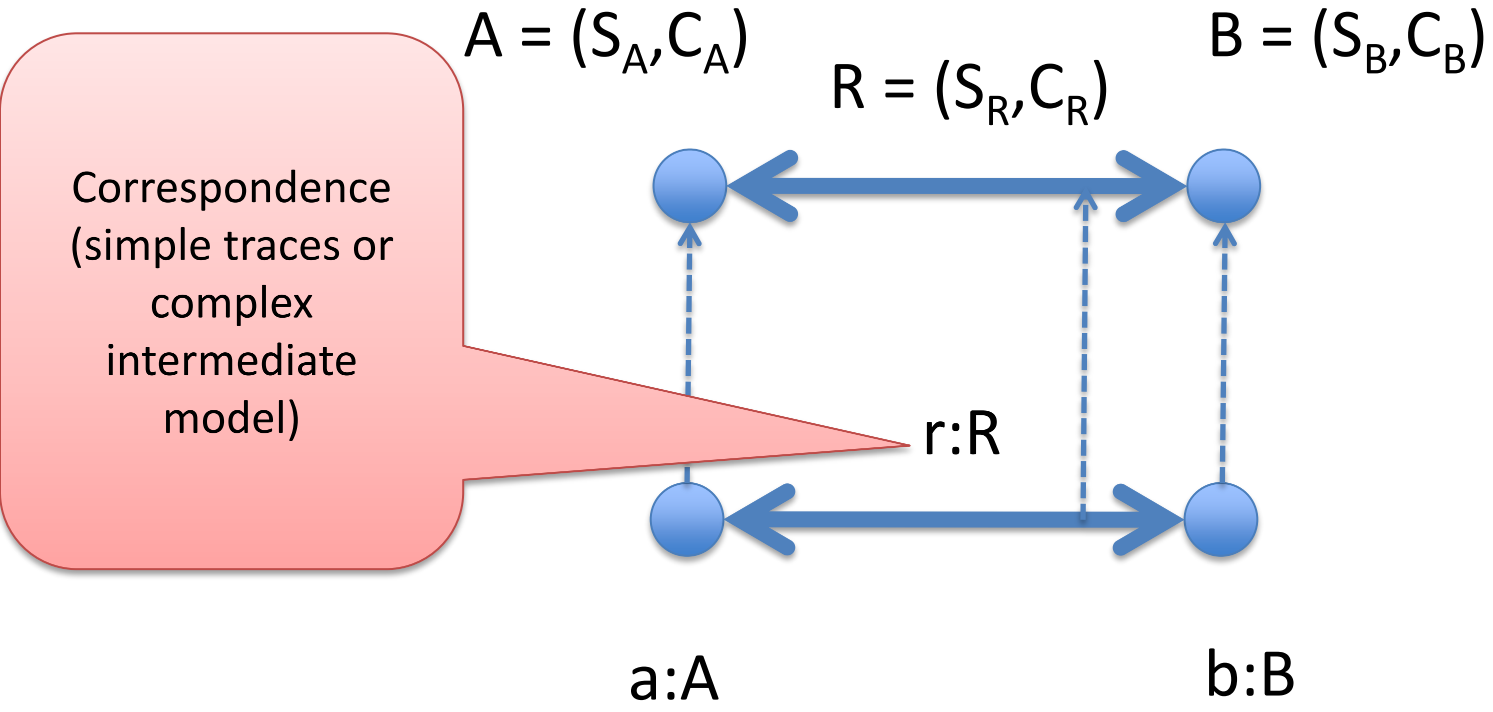


$a{:}A$

$B = (S_B,C_B)$



$b{:}B$



Correspondence (simple traces or complex intermediate model)

Saying **a** consistent with **b** means

a,b consistent and $\left\{ \begin{array}{l} a \models C_A \\ b \models C_B \end{array} \right.$

their correspondence consistent $\{ \quad \mathbf{r \models C_R}$

Model Correspondence

Different types

Set of element-to-element and link-to-link correspondences (e.g., replica sync)

Complex intermediate model (e.g., across languages)

How to obtain

Could be produced by a matching procedure, e.g., match(a,b) = r

May need to be constructed manually

And consistency…

Given an automatic match procedure, the consistency relation becomes binary

$(a,b) \models_{match(a,b):R} C_R$

Two Dimensions

Modeling languages

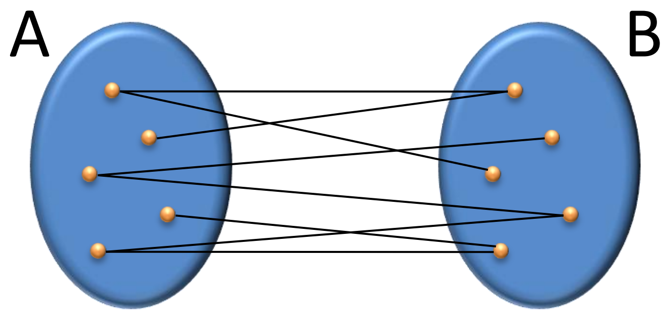
- Homogenous: both models in same language
- Heterogeneous: both models in different languages

Consistency relation (modulo matching)

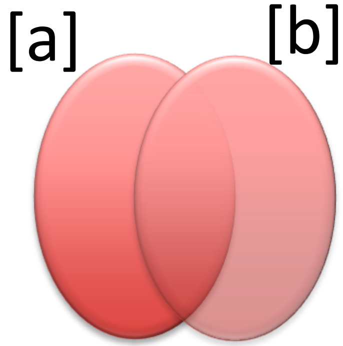
- Relational
- Functional
- Bijective

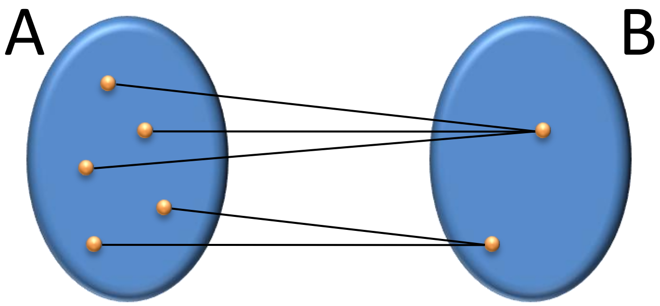
Model mappings

Info in models

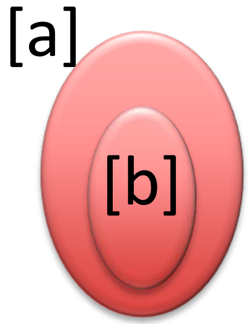


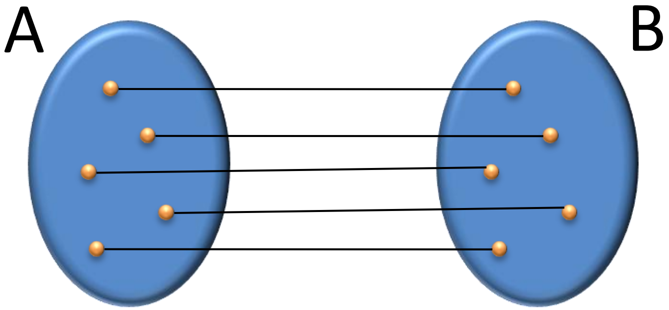
relational



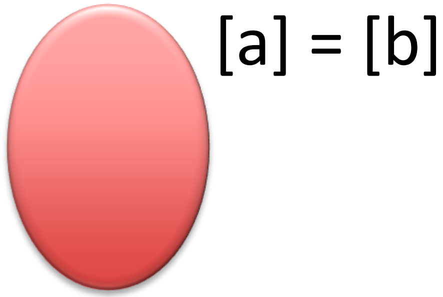


functional

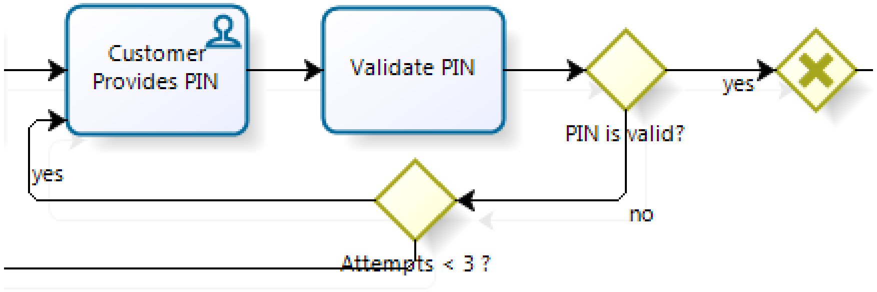


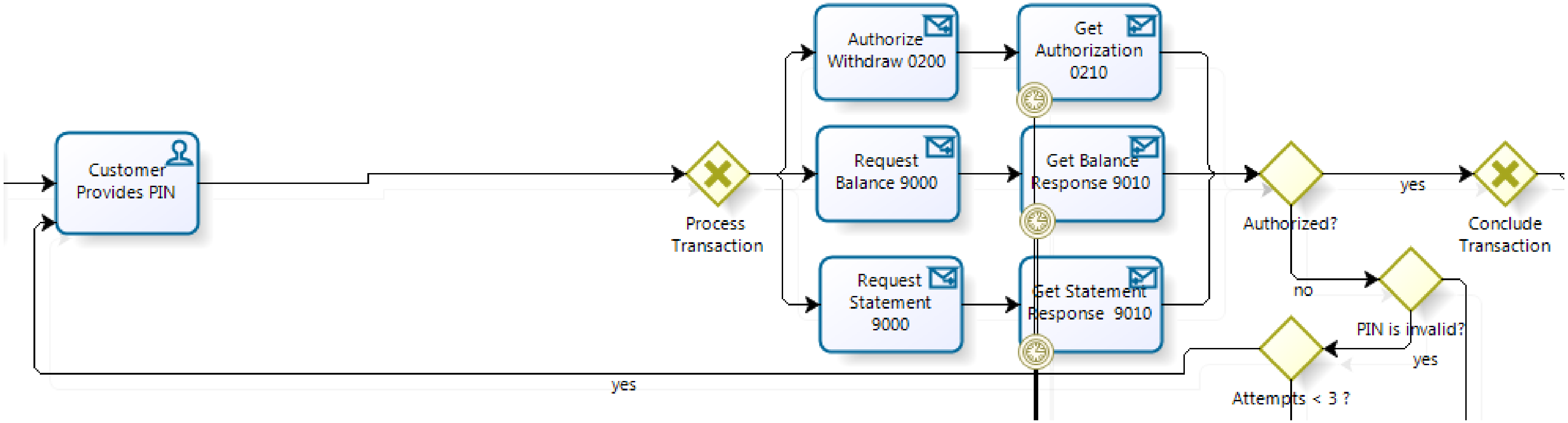


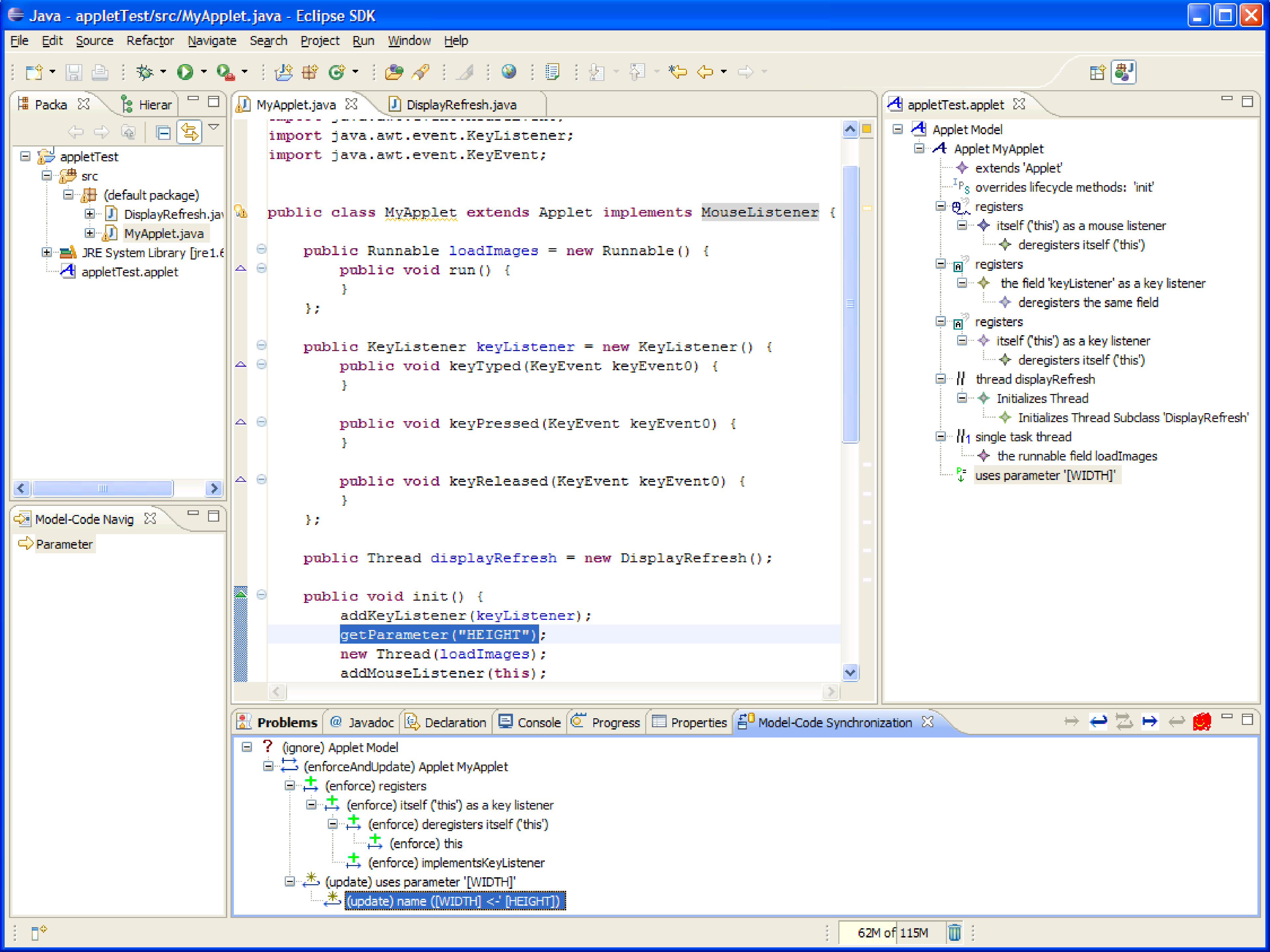
bijective



Manual Refinement Example






Java - appletTest/src/MyApplet.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Packa
Hierar
MyApplet.java
DisplayRefresh.java
appletTest.applet
Model-Code Navig
Problems
Javadoc
Declaration
Console
Progress
Properties
Model-Code Synchronization
62M of 115M

appletTest
src
(default package)
DisplayRefresh.ja
MyApplet.java
JRE System Library [jre1.6
appletTest.applet

Parameter

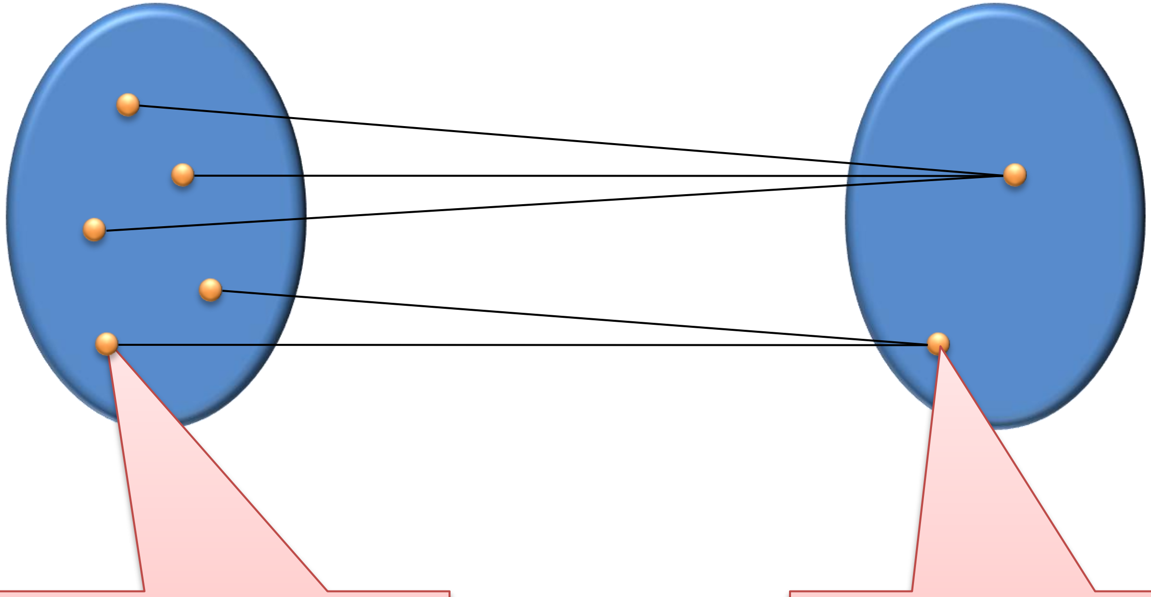
```
import java.awt.event.KeyListener;
import java.awt.event.KeyEvent;


public class MyApplet extends Applet implements MouseListener {

    public Runnable loadImages = new Runnable() {
        public void run() {
        }
    };

    public KeyListener keyListener = new KeyListener() {
        public void keyTyped(KeyEvent keyEvent0) {
        }

        public void keyPressed(KeyEvent keyEvent0) {
        }

        public void keyReleased(KeyEvent keyEvent0) {
        }
    };

    public Thread displayRefresh = new DisplayRefresh();

    public void init() {
        addKeyListener(keyListener);
        getParameter("HEIGHT");
        new Thread(loadImages);
        addMouseListener(this);
```

```
Applet Model
  Applet MyApplet
    extends 'Applet'
    overrides lifecycle methods: 'init'
    registers
      itself ('this') as a mouse listener
        deregisters itself ('this')
    registers
      the field 'keyListener' as a key listener
        deregisters the same field
    registers
      itself ('this') as a key listener
        deregisters itself ('this')
    thread displayRefresh
      Initializes Thread
        Initializes Thread Subclass 'DisplayRefresh'
    single task thread
      the runnable field loadImages
    uses parameter '[WIDTH]'
```

```
(ignore) Applet Model
  (enforceAndUpdate) Applet MyApplet
    (enforce) registers
      (enforce) itself ('this') as a key listener
        (enforce) deregisters itself ('this')
          (enforce) this
        (enforce) implementsKeyListener
    (update) uses parameter '[WIDTH]'
      (update) name ([WIDTH] <-' [HEIGHT])
```

Java

Applet Modeling Language



Java code using Applet framework

Applet model

Examples

Homogenous

- Relation: workflow refinement (heuristic match)
- Function: projection of a product-line variant (automatic match)
- Bijection: replica synchronization

Heterogeneous

- Relation: BPMN-to-BPEL
- Function: FSMLs
- Bijection: KM3-to-UML class models

Updates

State-based

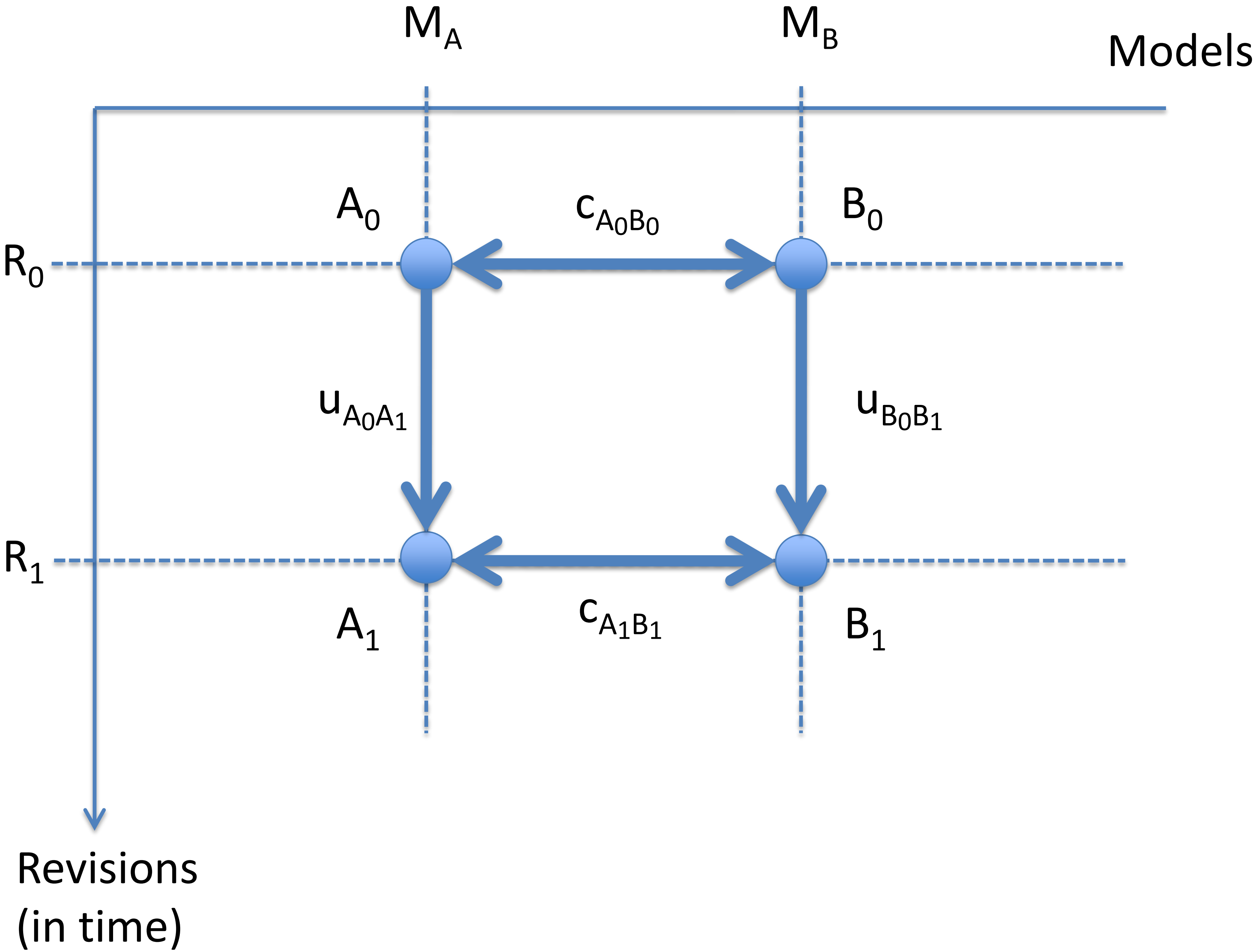
- Two revisions of a model + element-wise correspondence
- Reduces to a pair of revisions if correspondence automatic

Operation-based

- Edits logs
- Element correspondence automatic or in the log

Composition

- Correspondence composition (state-based)
- Type-aware (updates typed by their operations)


M_A
M_B
Models
A_0
$c_{A_0B_0}$
B_0
R_0
$u_{A_0A_1}$
$u_{B_0B_1}$
R_1
A_1
$c_{A_1B_1}$
B_1
Revisions
(in time)

Roadmap

Single model consistency management

Multi-model consistency management

Examples

Replica synchronization

View synchronization

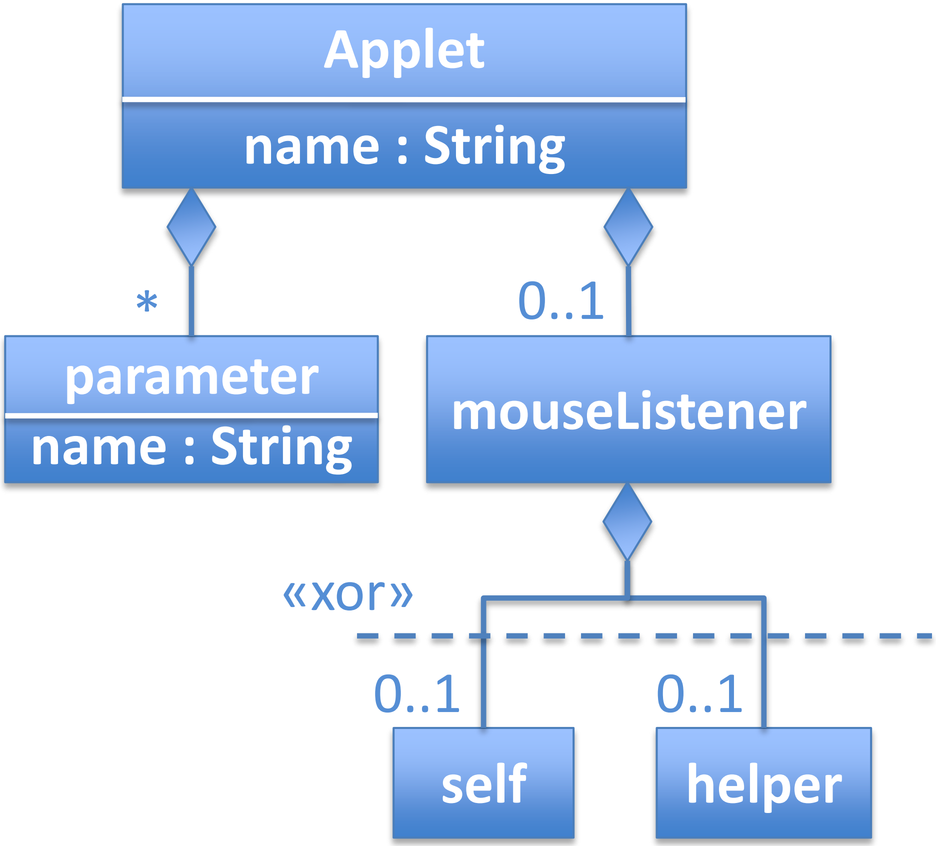
General overlap

Applet design language

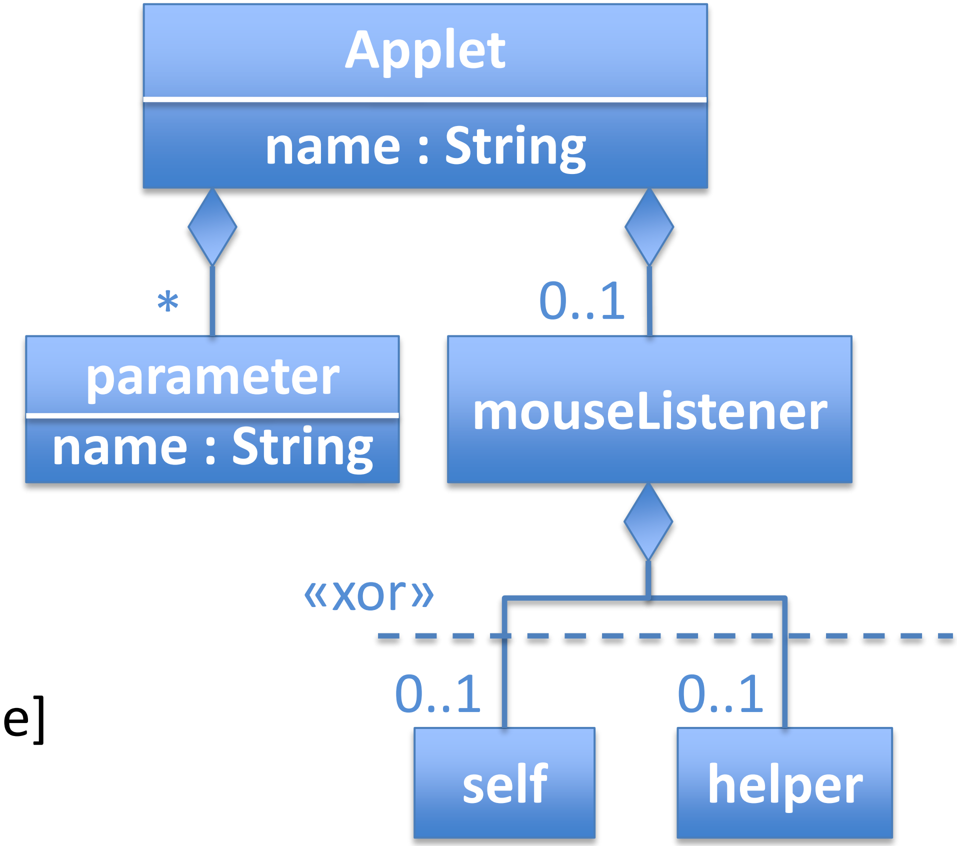
Metamodel in Clafer

[SLE’10]

```
Applet *
  name : String
  parameter *
    name : String
  xor mouseListener ?
    self
    helper
```



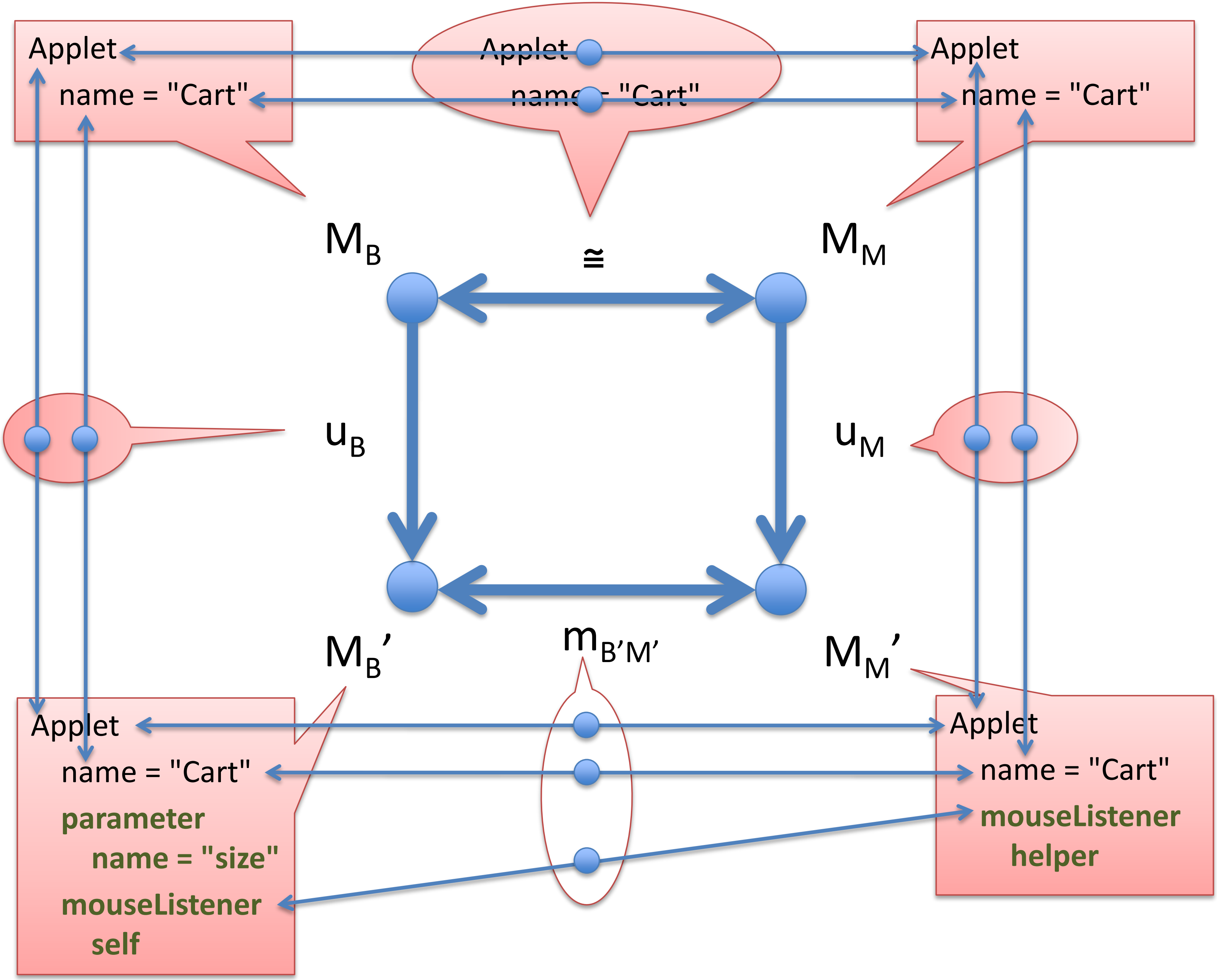
```
Applet *
  name : String <key>
  parameter *
    name : String <key>
  **xor** mouseListener ?
    self
    helper
  [mouseListener =>
    "dblClick" in parameter.name]
```



Replica synchronization

homogenous, bijective

(consistent when isomorphic, ≅)



$$\begin{tikzcd}
M_B \arrow[r, leftrightarrow, "\cong"] \arrow[d, "u_B"'] & M_M \arrow[d, "u_M"] \\
M_B' \arrow[r, leftrightarrow, "m_{B'M'}"'] & M_M'
\end{tikzcd}$$

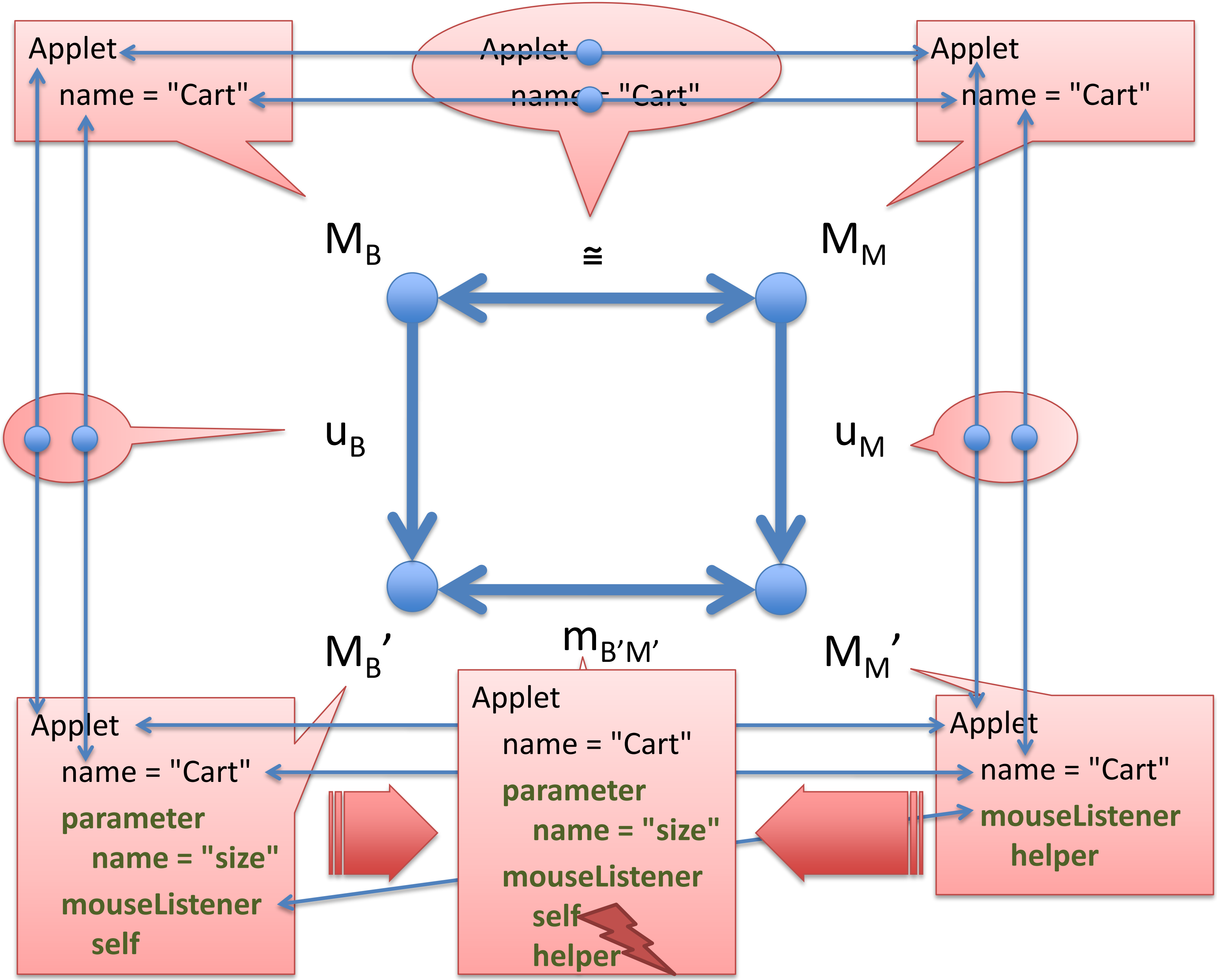
Applet
name = "Cart"

Applet
name = "Cart"

Applet
name = "Cart"

Applet
name = "Cart"
parameter
name = "size"
mouseListener
self

Applet
name = "Cart"
mouseListener
helper



Applet
name = "Cart"

Applet
name = "Cart"

Applet
name = "Cart"

M_B

$\cong$

M_M

u_B

u_M

M_B’

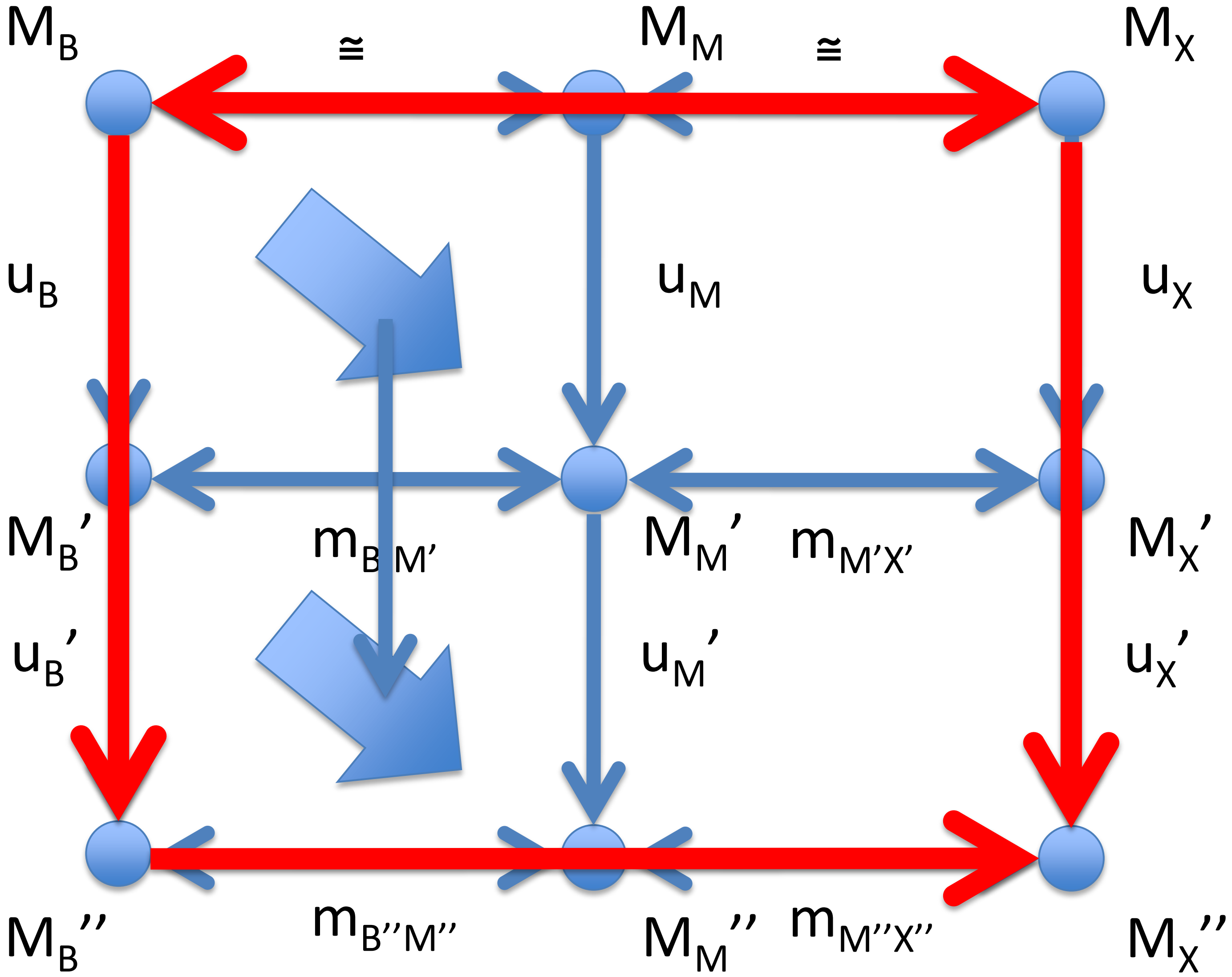
$m_{B’M’}$

M_M’

Applet
name = "Cart"
parameter
name = "size"
mouseListener
self

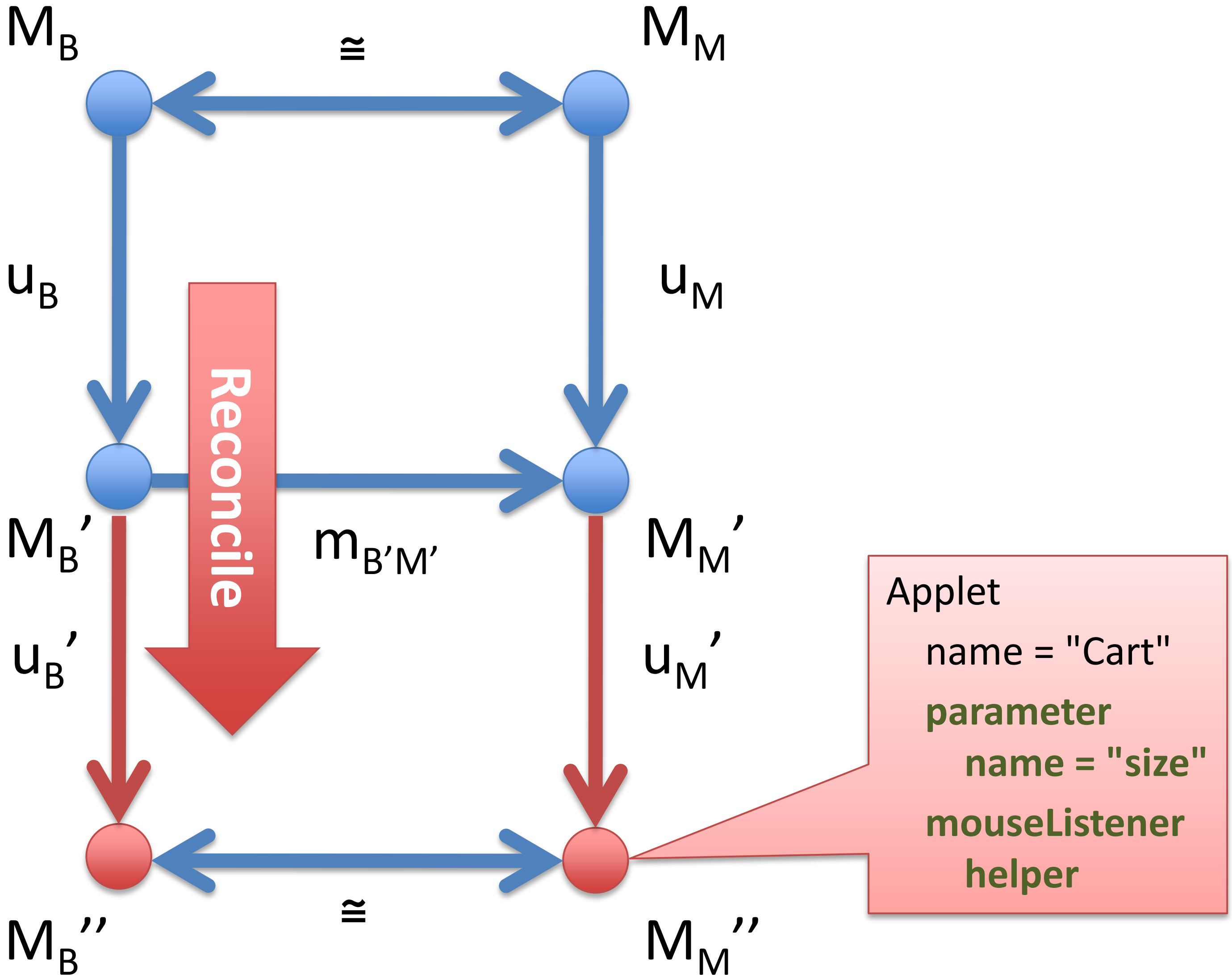
Applet
name = "Cart"
parameter
name = "size"
mouseListener
self
helper

Applet
name = "Cart"
mouseListener
helper


M_B
$\cong$
M_M
$\cong$
M_X
u_B
u_M
u_X
M_B'
$m_{B'M'}$
M_M'
$m_{M'X'}$
M_X'
u_B'
u_M'
u_X'
M_B''
$m_{B''M''}$
M_M''
$m_{M''X''}$
M_X''

$$\begin{tikzcd}
M_B \arrow[rr, leftrightarrow, "\cong"] \arrow[d, "u_B"'] & & M_M \arrow[d, "u_M"] \\
M_B' \arrow[rr, "m_{B'M'}"'] \arrow[d, "u_B'"'] & & M_M' \arrow[d, "u_M'"] \\
M_B'' \arrow[rr, leftrightarrow, "\cong"'] & & M_M''
\end{tikzcd}$$

Reconcile



Applet
    name = "Cart"
    parameter
        name = "size"
    mouseListener
        helper

Replica synch – Summary

Homogenous consistency check

- Match as span
- Merge via co-limit
  - result over same metamodel
- Constraint check on merge result

- [Sabetzadeh, Easterbrook 2006]

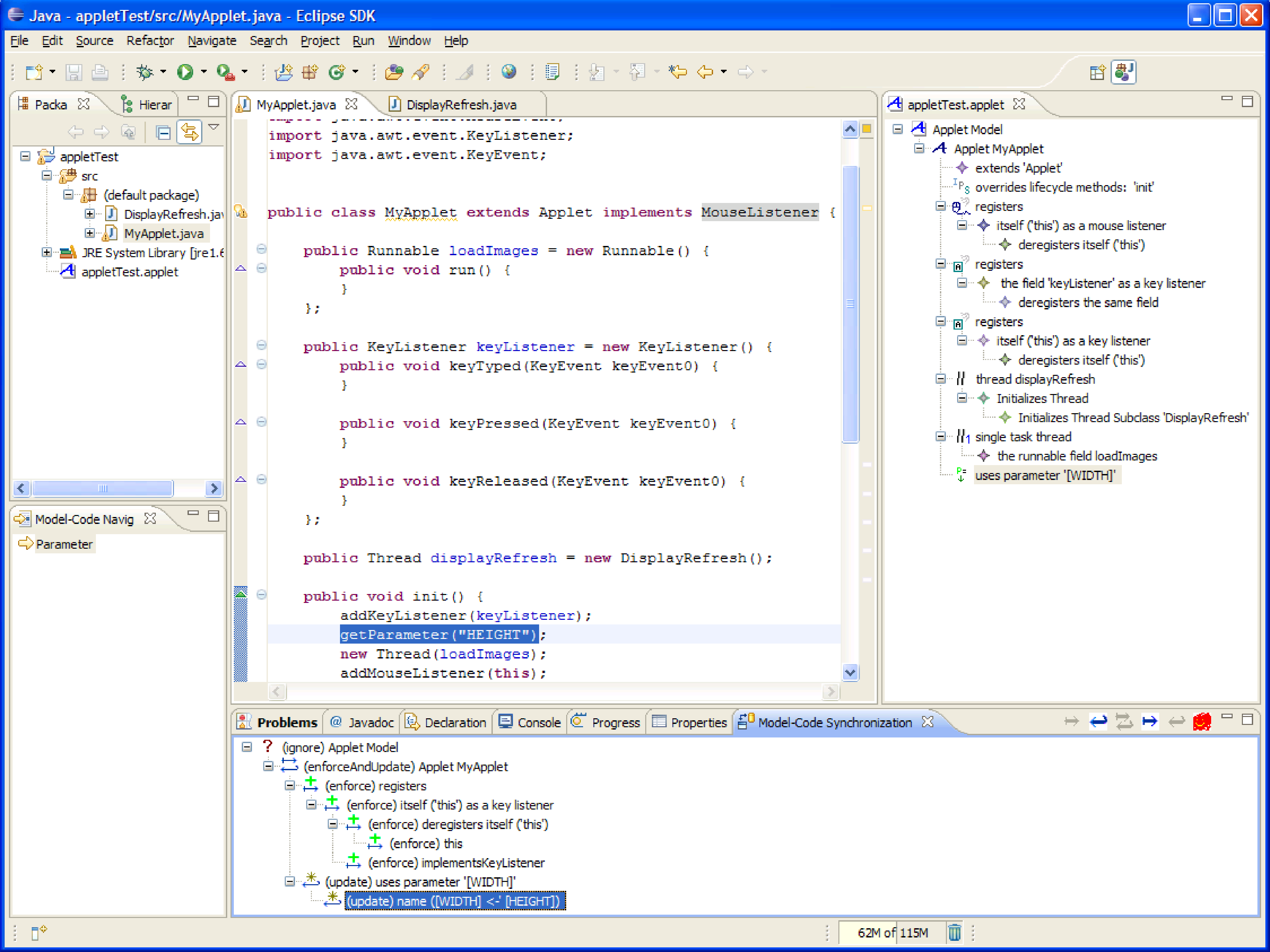
Tile composition and operations

- 2D deltas in space of replicas and versions
- Rephrased as double categories
  - With horizontal and vertical composition
- Reconciliation as a **tile** operation

- [CVSM’09]

Heterogeneous view synchronization

Back to our Applet example…



- appletTest
  - src
    - (default package)
      - DisplayRefresh.ja
      - MyApplet.java
  - JRE System Library [jre1.6
  - appletTest.applet

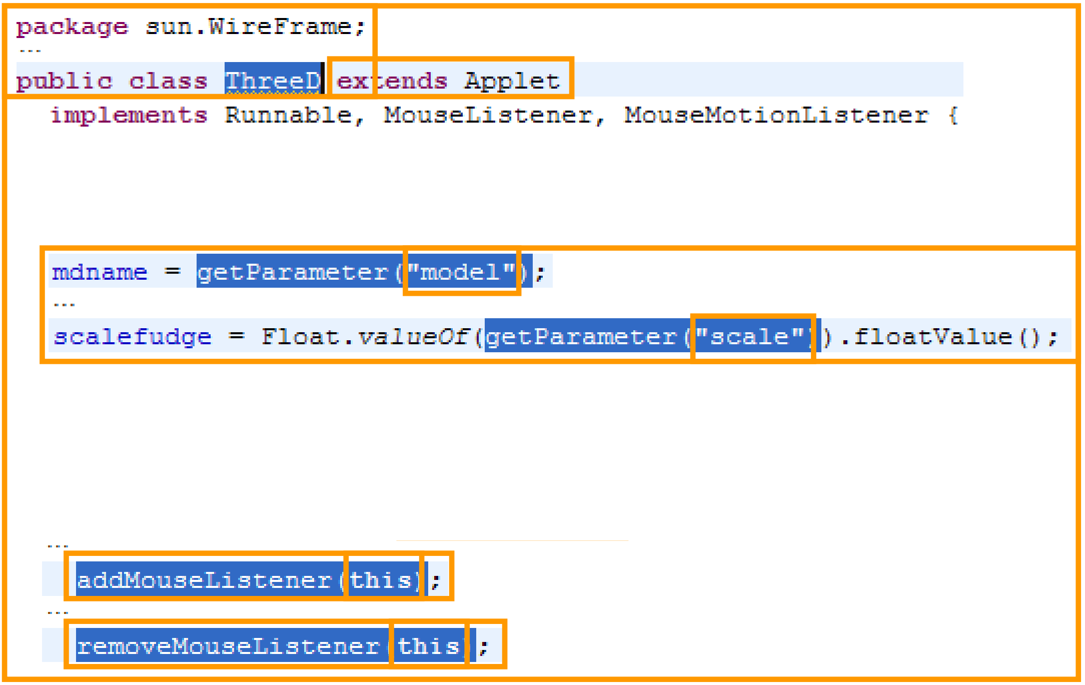
Parameter

```java
import java.awt.event.KeyListener;
import java.awt.event.KeyEvent;


public class MyApplet extends Applet implements MouseListener {

    public Runnable loadImages = new Runnable() {
        public void run() {
        }
    };

    public KeyListener keyListener = new KeyListener() {
        public void keyTyped(KeyEvent keyEvent0) {
        }

        public void keyPressed(KeyEvent keyEvent0) {
        }

        public void keyReleased(KeyEvent keyEvent0) {
        }
    };

    public Thread displayRefresh = new DisplayRefresh();

    public void init() {
        addKeyListener(keyListener);
        getParameter("HEIGHT");
        new Thread(loadImages);
        addMouseListener(this);
```

- Applet Model
  - Applet MyApplet
    - extends 'Applet'
    - overrides lifecycle methods: 'init'
    - registers
      - itself ('this') as a mouse listener
        - deregisters itself ('this')
    - registers
      - the field 'keyListener' as a key listener
        - deregisters the same field
    - registers
      - itself ('this') as a key listener
        - deregisters itself ('this')
    - thread displayRefresh
      - Initializes Thread
        - Initializes Thread Subclass 'DisplayRefresh'
    - single task thread
      - the runnable field loadImages
    - uses parameter '[WIDTH]'

- (ignore) Applet Model
  - (enforceAndUpdate) Applet MyApplet
    - (enforce) registers
      - (enforce) itself ('this') as a key listener
        - (enforce) deregisters itself ('this')
          - (enforce) this
        - (enforce) implementsKeyListener
    - (update) uses parameter '[WIDTH]'
      - (update) name ([WIDTH] <-' [HEIGHT])

Applet code



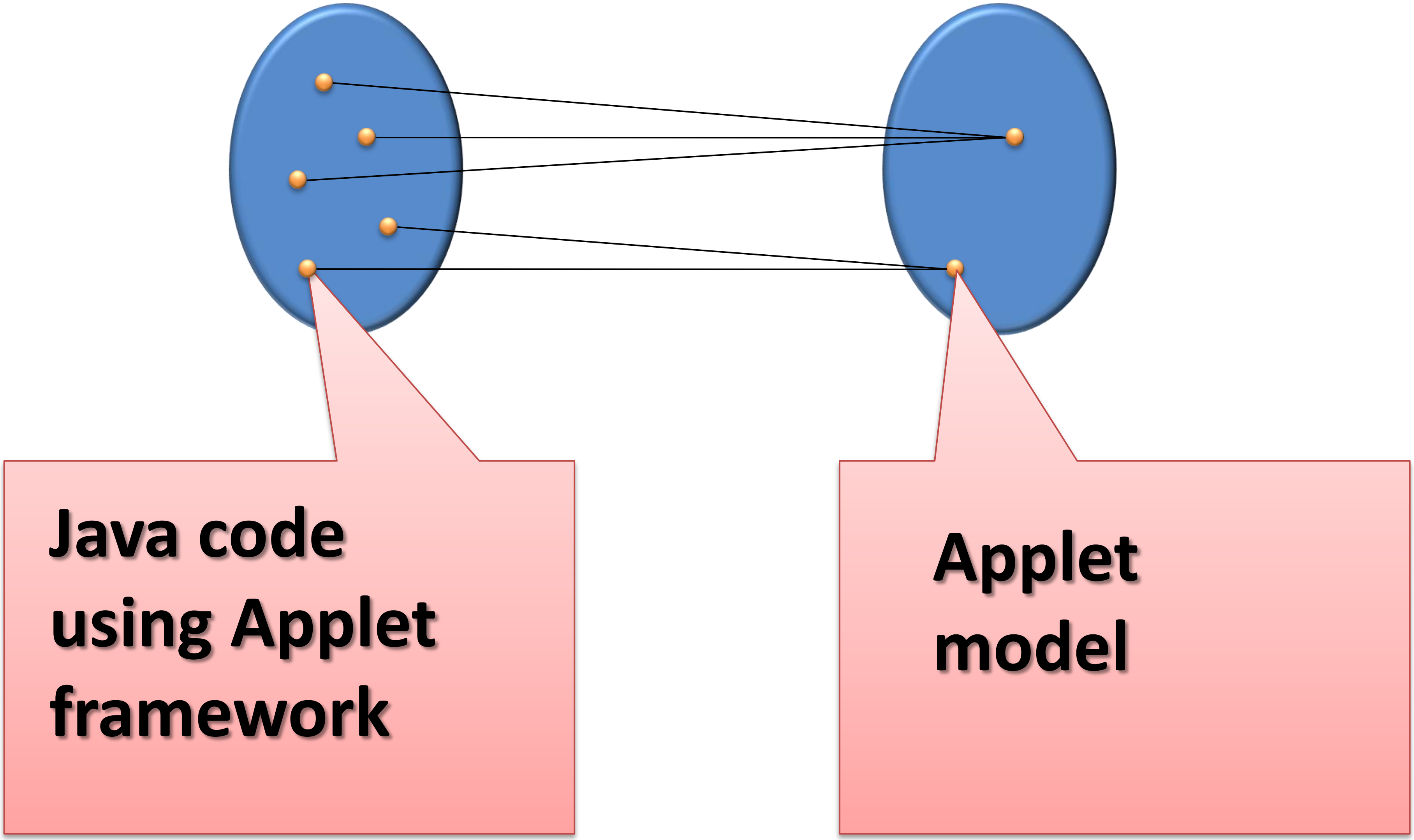
```
package sun.WireFrame;
...
public class ThreeD extends Applet
  implements Runnable, MouseListener, MouseMotionListener {

  mdname = getParameter("model");
  ...
  scalefudge = Float.valueOf(getParameter("scale")).floatValue();

  ...
  addMouseListener(this);
  ...
  removeMouseListener(this);
```

Applet model

```
Applet
   name = "sun.WireFrame.ThreeD"
   !extendsApplet

   parameter
      name = "model”
   parameter
      name = "scale”
   listensToMouse
      !implementsMouseListener
      !registers
       deregisters
            deregistersSameObject
               registersBeforeDeregisters
```

Java

Applet Modeling Language



Java code using Applet framework

Applet model

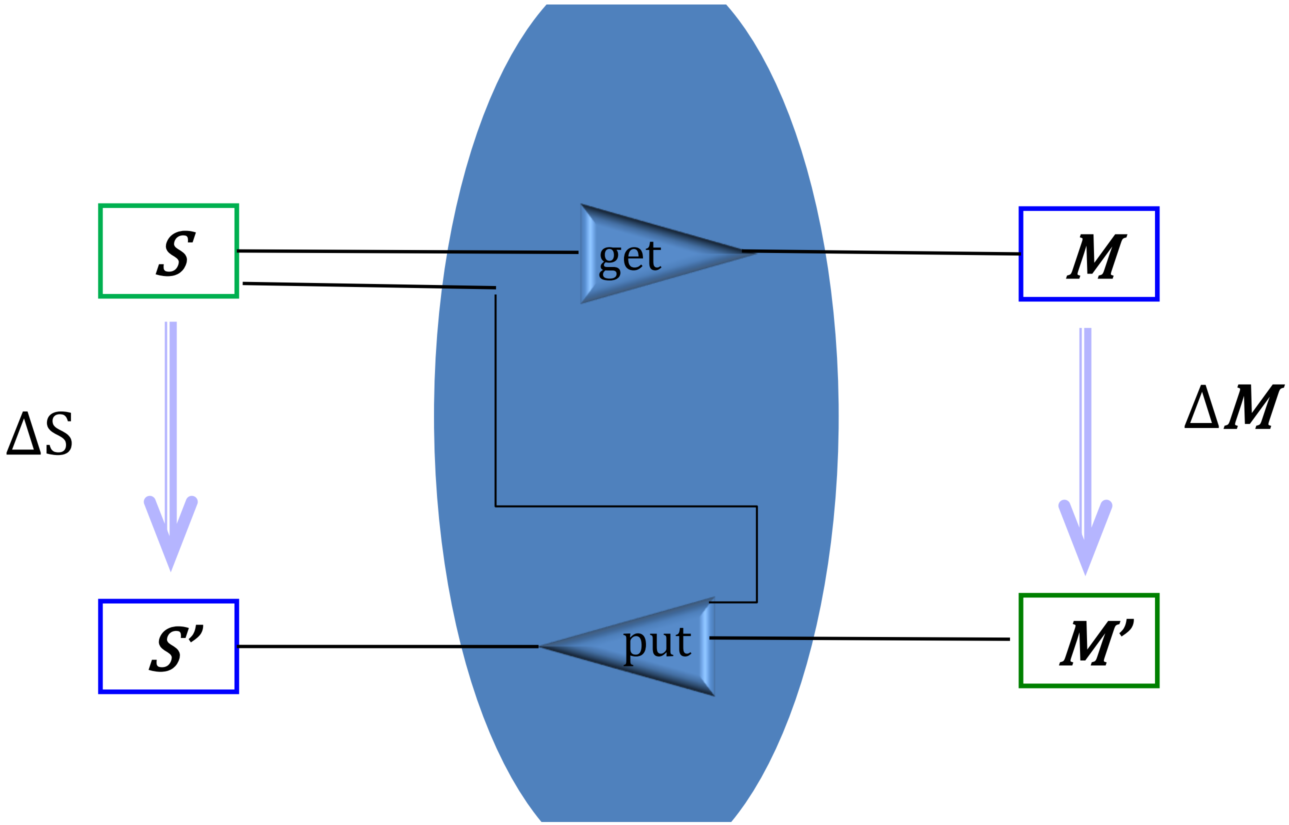
Applet modeling language syntax

Mapping to code

```
Applet *                            <class>
  name : String                     <fullyQualifiedName>
 !extendsApplet                     <assignableTo: ‘Applet’>
  parameter *                       <callsReceived: ‘getParameter(String)’>
    name : String ?                 <valueOfArg: 1>
  listensToMouse ?
   !implementsMouseListener
   !registers                       <callsReceived: ‘addMouseListener(Mous[…])’
    deregisters                     <callsReceived: ‘removeMouseListener(M[…])’

      deregistersSameObject         <argument:1 of call: ../../registers
                                       sameAsArg: 1 of call: ../../deregisters>

        registersBeforeDeregisters  <methodCall: ../../../registers before:
                                       ../..>
```

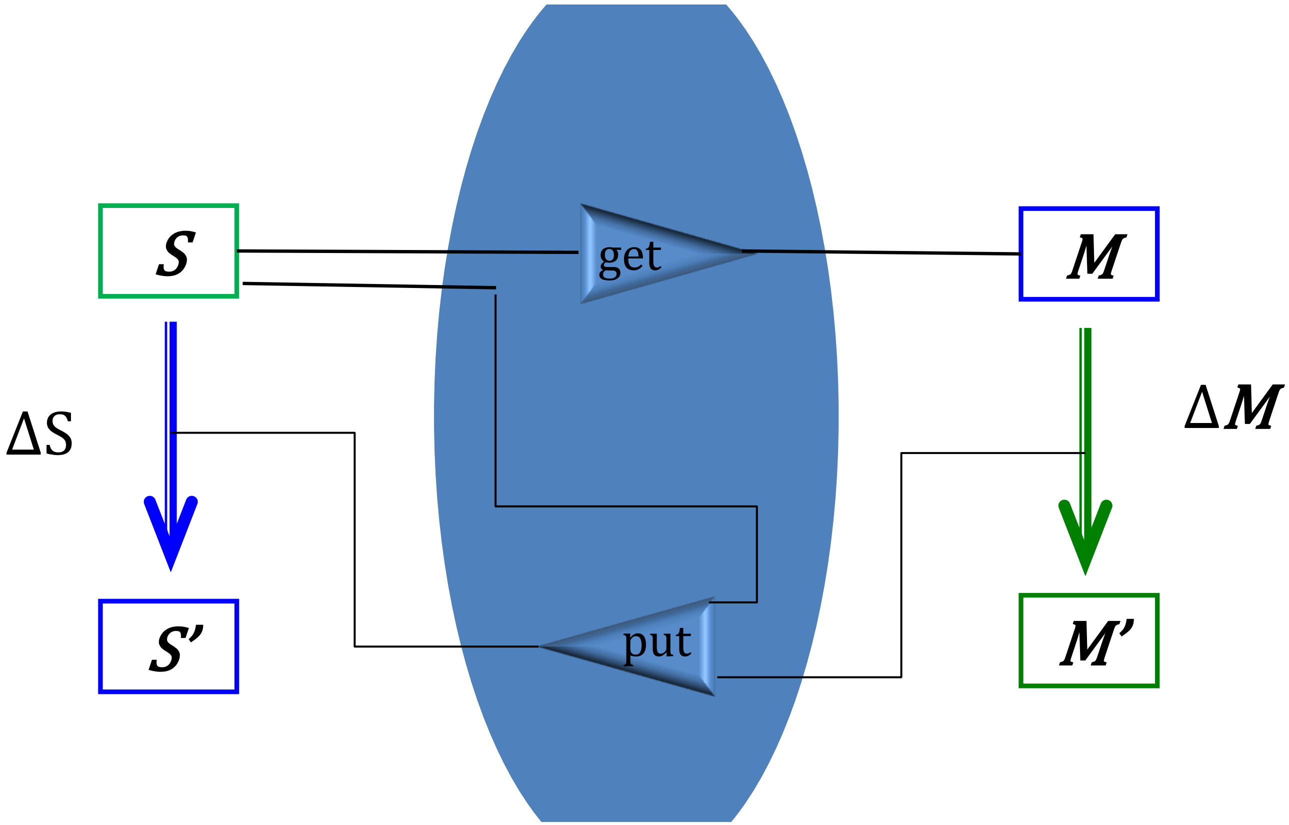
Bidirectional transformation via Lenses

State-based lens



[Pierce et al. 2003-2010]

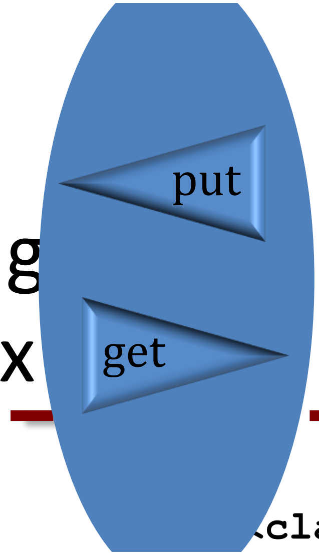
Delta-based lens

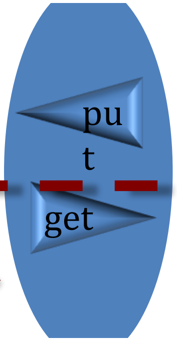


[ICMT’10]

Applet modeling language syntax

Mapping to code





```
Applet *                            <class>
  name : String                       llyQualifiedName>
 !extendsApplet                       signableTo: 'Applet'>
  parameter *                       <callsReceived: 'getParameter(String)'>
    name : String ?                 <valueOfArg: 1>
  listensToMouse ?
   !implementsMouseListener
   !registers                       <callsReceived: 'addMouseListener(Mous[…])'
    deregisters                     <callsReceived: 'removeMouseListener(M[…])'

      deregistersSameObject         <argument:1 of call: ../../registers
                                       sameAsArg: 1 of call: ../../deregisters>

        registersBeforeDeregisters  <methodCall: ../../../registers before:
                                       ../..>
```

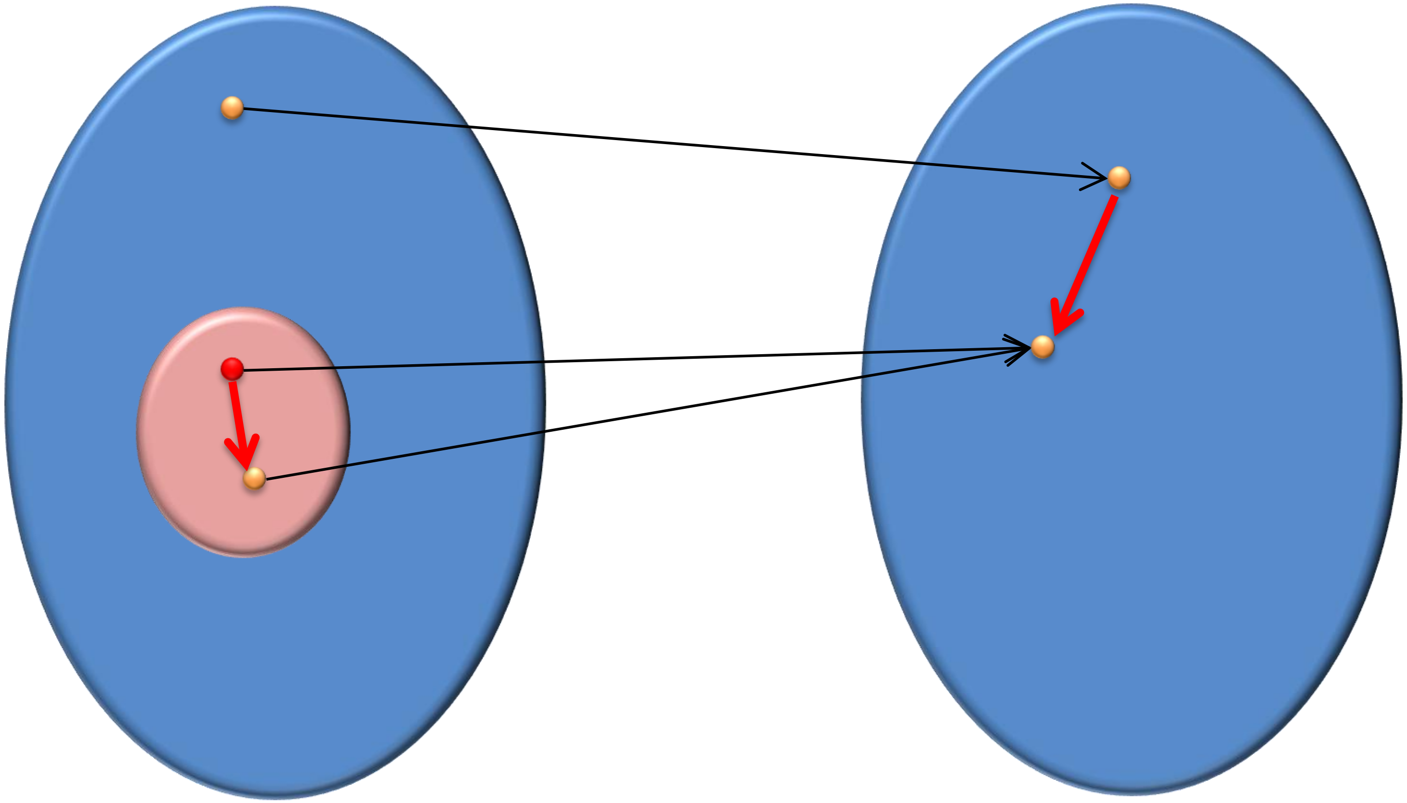
Queries and update trafos

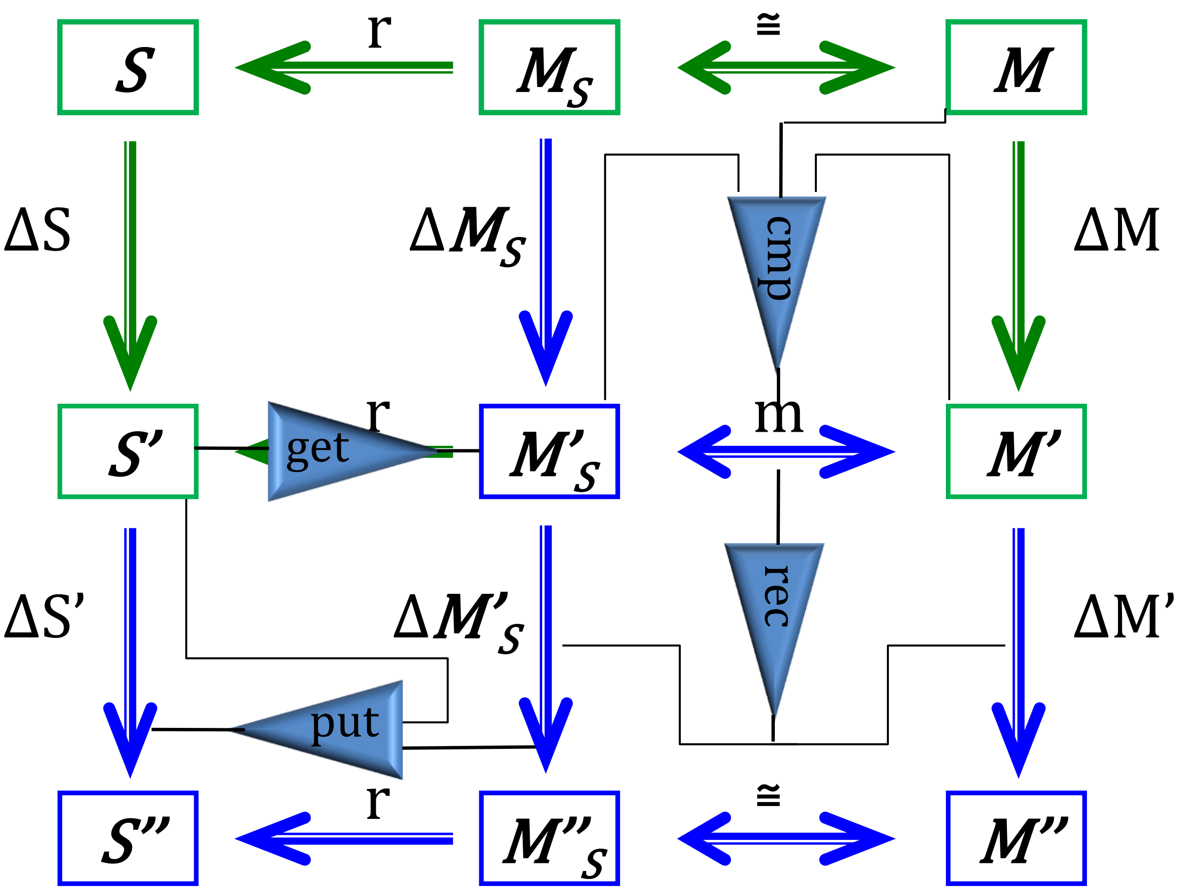
- Approximations of behavioural mapping types
  - Precision and recall for queries
  - Potentially partial implementation by transformations
- Refinements through additional parameters for queries and transformations
  - Query (get) – different precision
  - Trafo (put) – e.g., additional control over location of additions

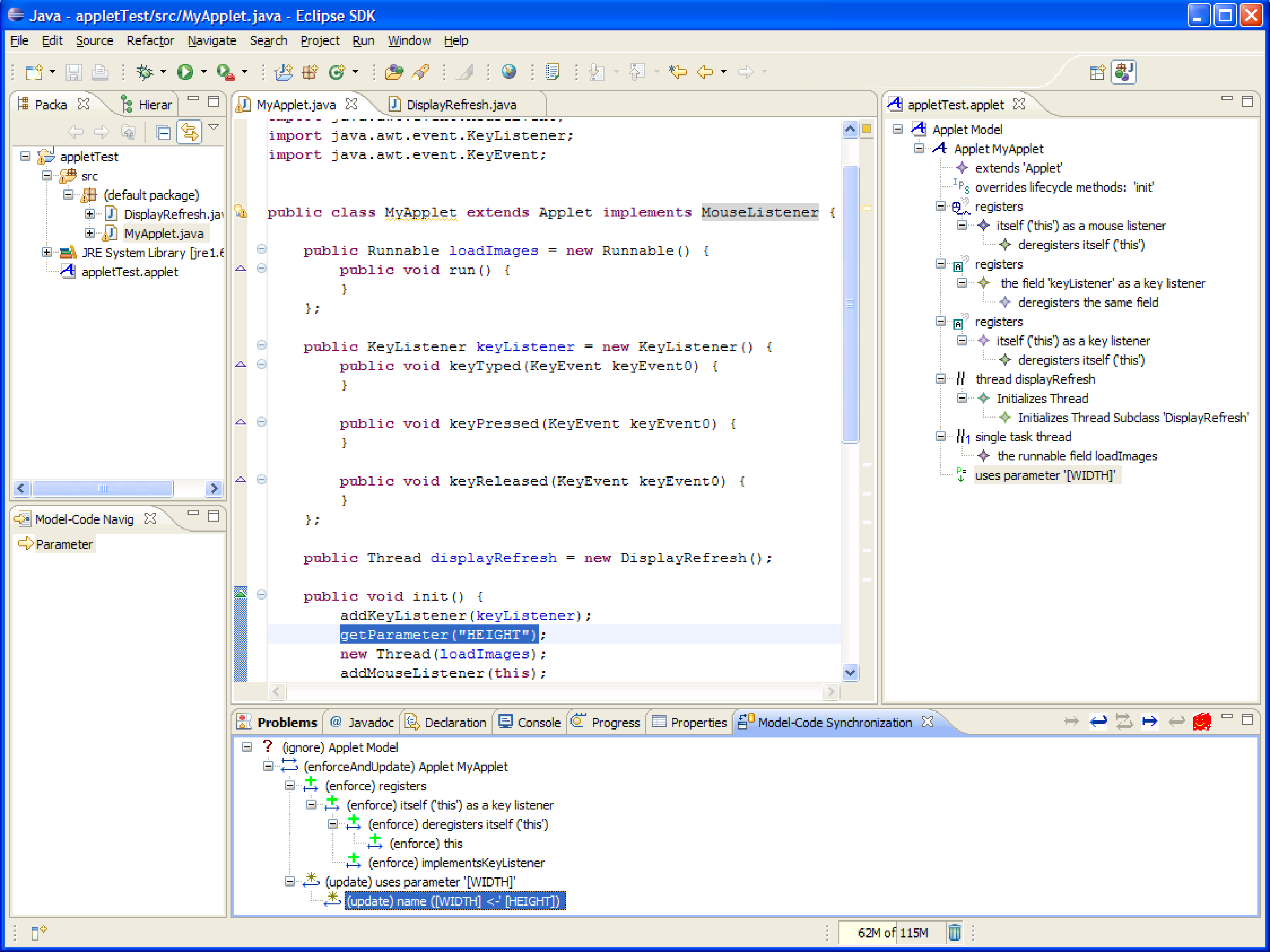
[TSE’09, ASEJ’09]

Applet implementations In Java

Applet models




S
r
M_S
≅
M
ΔS
ΔM_S
cmp
ΔM
S'
get
r
M'_S
m
M'
rec
ΔS'
ΔM'_S
ΔM'
put
S''
r
M''_S
≅
M''



- appletTest
  - src
    - (default package)
      - DisplayRefresh.ja
      - MyApplet.java
  - JRE System Library [jre1.6
  - appletTest.applet

Parameter

```
import java.awt.event.KeyListener;
import java.awt.event.KeyEvent;


public class MyApplet extends Applet implements MouseListener {

    public Runnable loadImages = new Runnable() {
        public void run() {
        }
    };

    public KeyListener keyListener = new KeyListener() {
        public void keyTyped(KeyEvent keyEvent0) {
        }

        public void keyPressed(KeyEvent keyEvent0) {
        }

        public void keyReleased(KeyEvent keyEvent0) {
        }
    };

    public Thread displayRefresh = new DisplayRefresh();

    public void init() {
        addKeyListener(keyListener);
        getParameter("HEIGHT");
        new Thread(loadImages);
        addMouseListener(this);
```

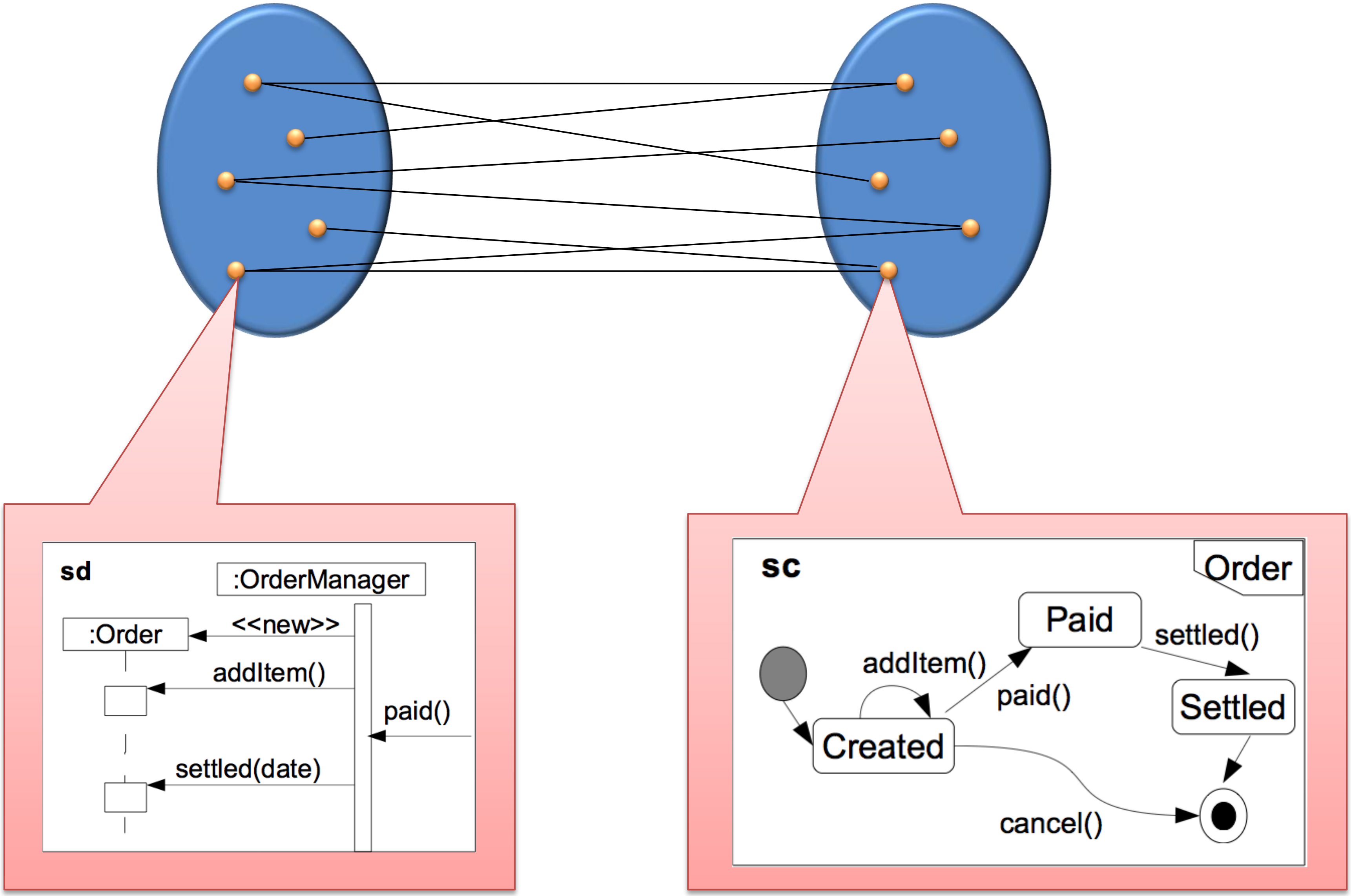
- Applet Model
  - Applet MyApplet
    - extends 'Applet'
    - overrides lifecycle methods: 'init'
    - registers
      - itself ('this') as a mouse listener
        - deregisters itself ('this')
    - registers
      - the field 'keyListener' as a key listener
        - deregisters the same field
    - registers
      - itself ('this') as a key listener
        - deregisters itself ('this')
    - thread displayRefresh
      - Initializes Thread
        - Initializes Thread Subclass 'DisplayRefresh'
    - single task thread
      - the runnable field loadImages
    - uses parameter '[WIDTH]'

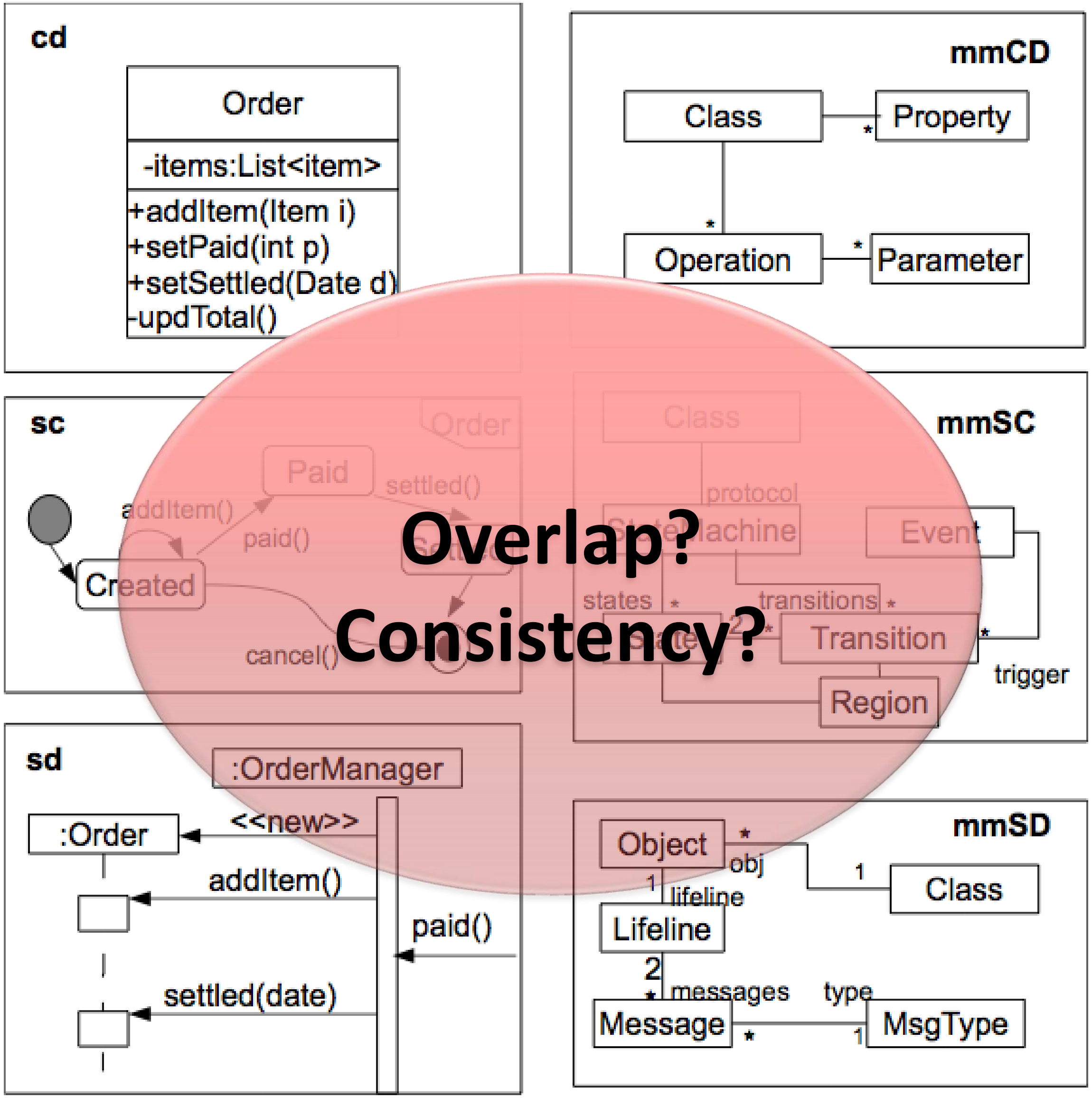
- (ignore) Applet Model
  - (enforceAndUpdate) Applet MyApplet
    - (enforce) registers
      - (enforce) itself ('this') as a key listener
        - (enforce) deregisters itself ('this')
          - (enforce) this
        - (enforce) implementsKeyListener
    - (update) uses parameter '[WIDTH]'
      - (update) name ([WIDTH] <-' [HEIGHT])

General overlap

UML sequence diagram

UML state chart



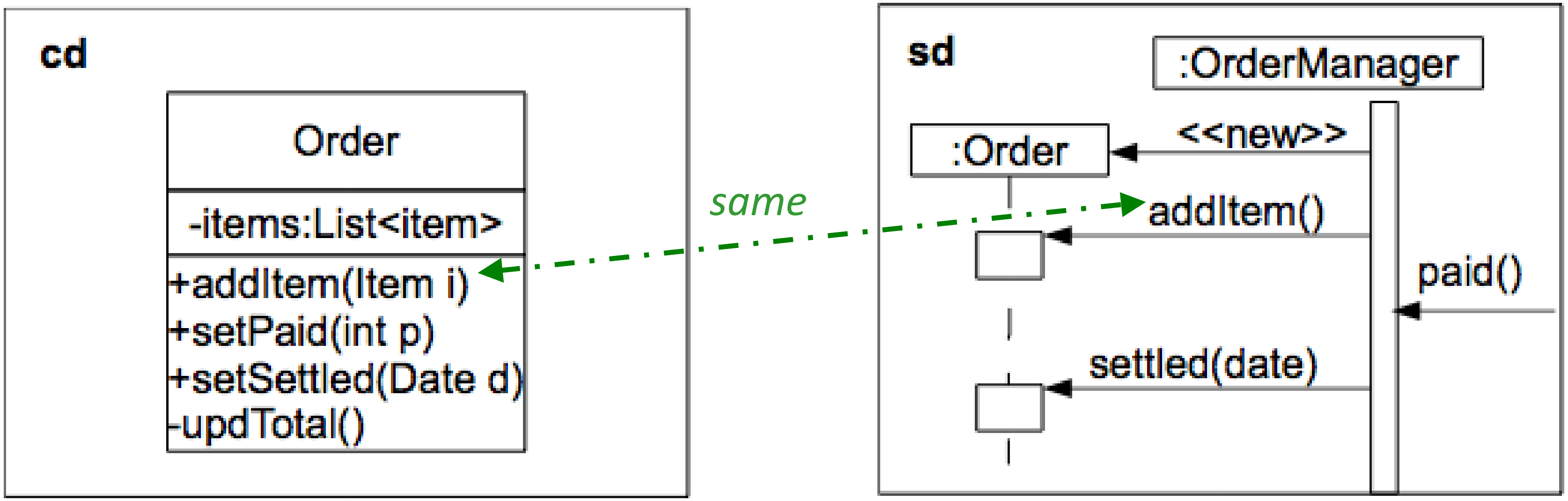


Overlap?

Consistency?

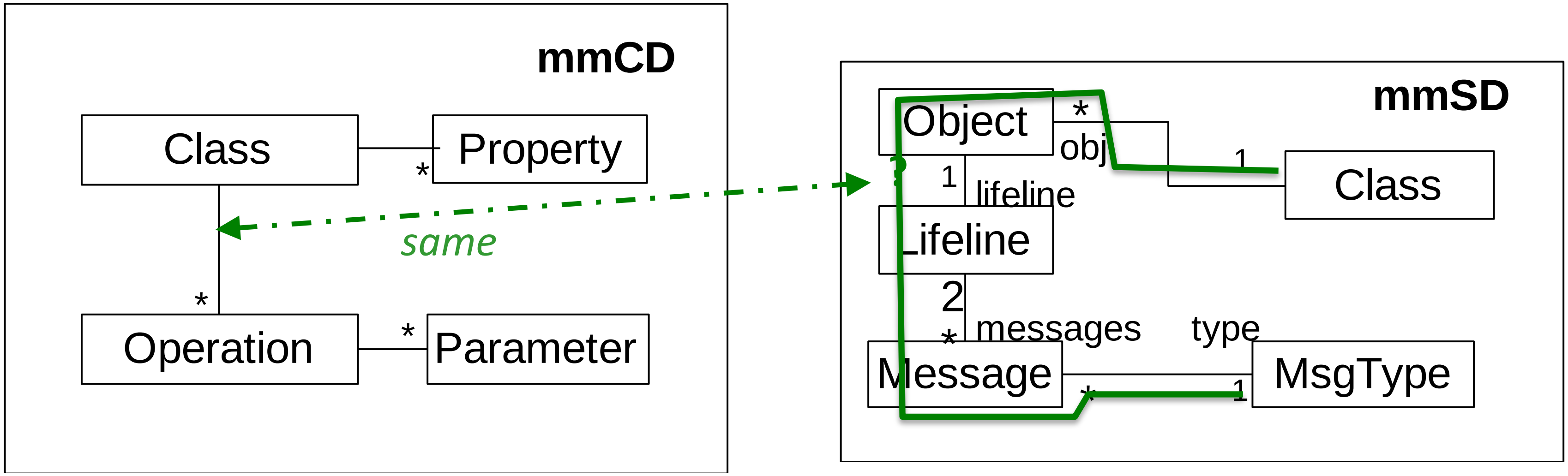
Four problems

Problems 1: Type Safety



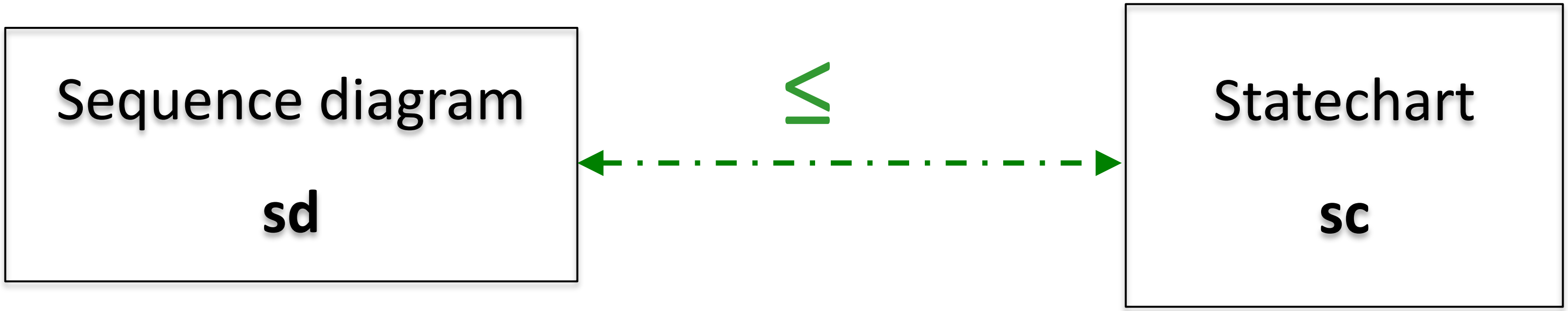
Incompatible types: Operation vs. MessageType !

Problem 2: Indirect correspondence



No explicit target in mmSD (and sd)!

Problem 3: Inter-Model Constraints



Sequence diagram

sd

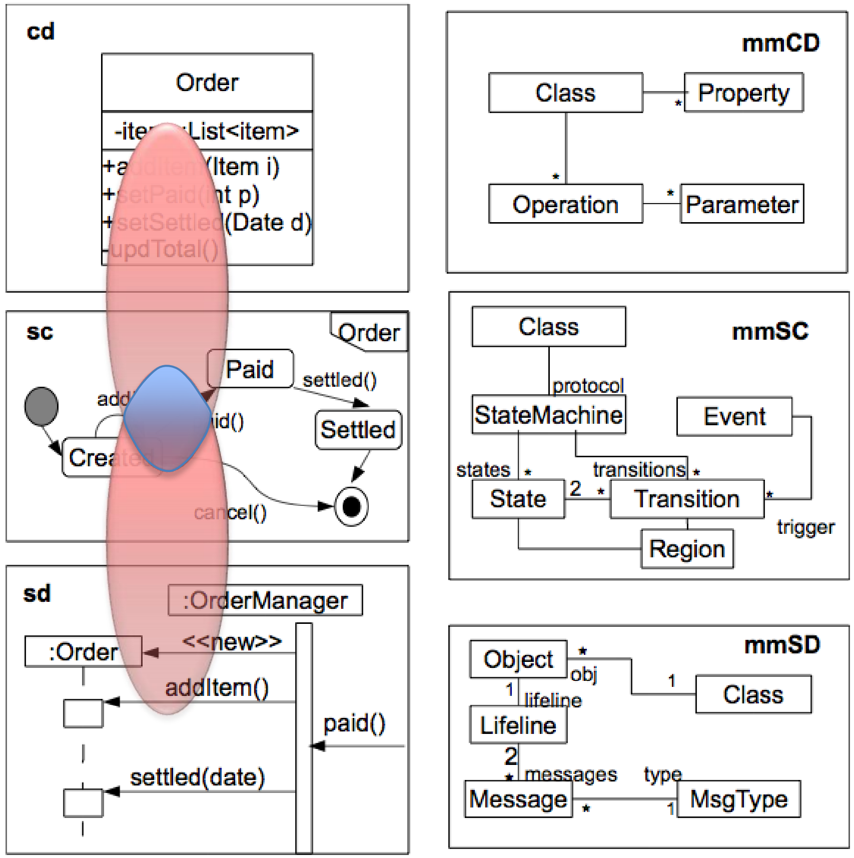
$\leq$

Statechart

sc

The inter-model constraint is neither in mmSD nor mmSC!

Problem 4: N-ary Metamodel Relations

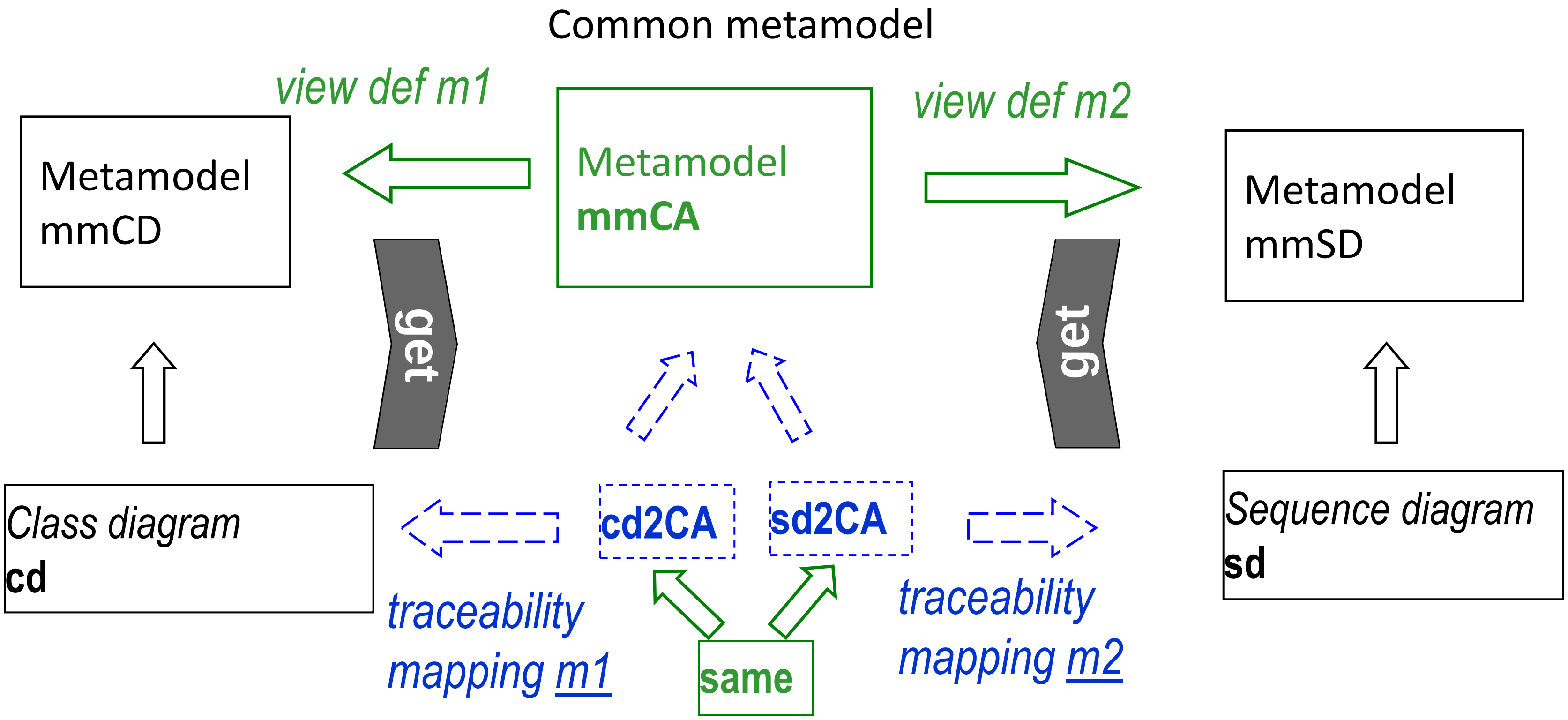


Pairwise, ternary, … overlaps! ●

Overlaps between overlaps! ◆

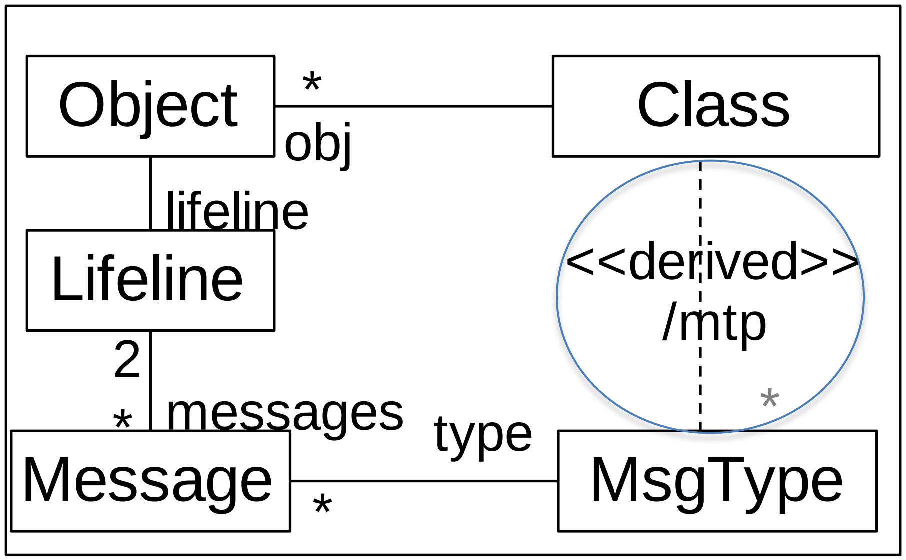
Solutions

Problem 1: Type Correspondence



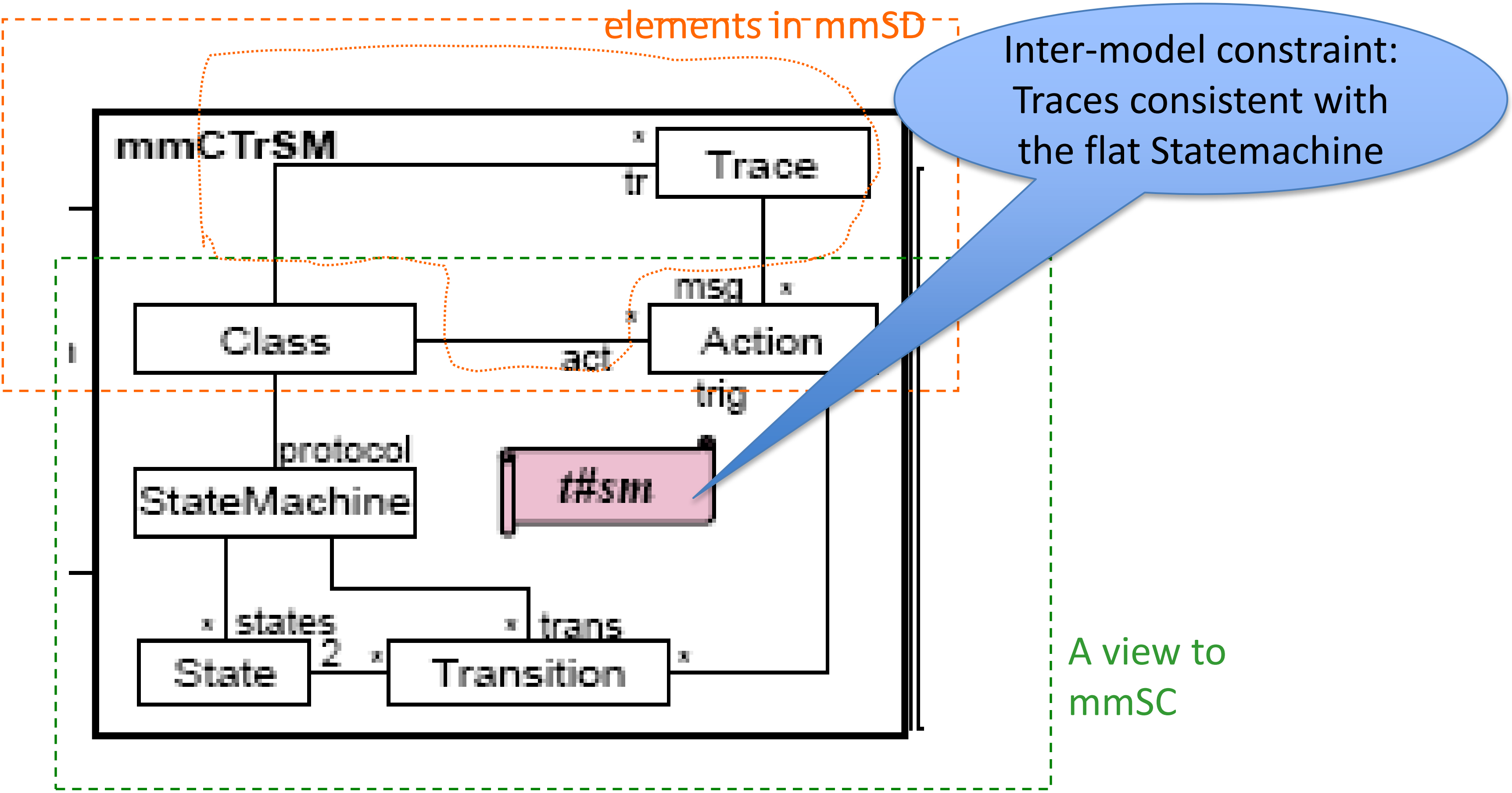
*Operation '**get**' models view execution mechanism*

Problem 2: Indirect Overlap



mmSD⁺

Problem 3: Inter-Model Constraints



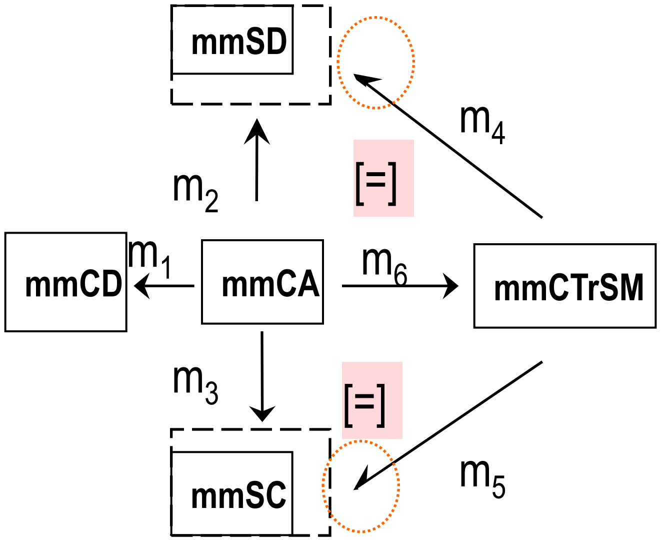
Mapped to derived elements in mmSD

A view to mmSD

Inter-model constraint: Traces consistent with the flat Statemachine

A view to mmSC

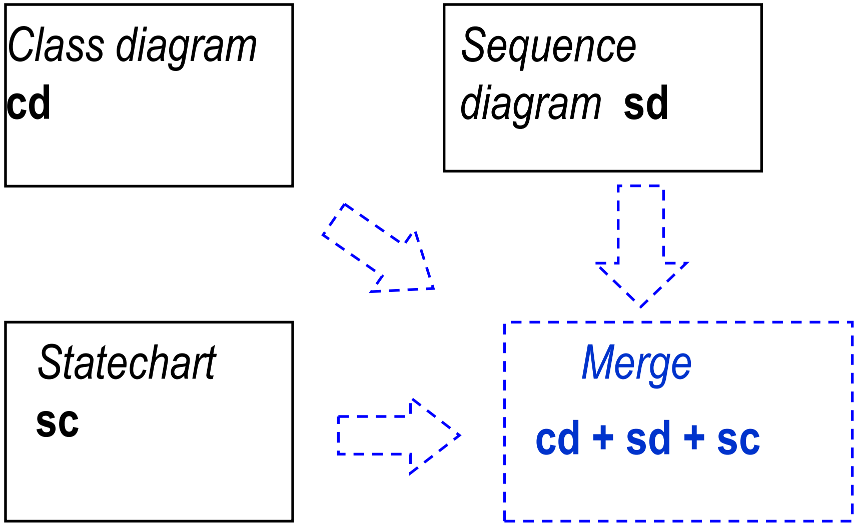
Problem 4: N-ary Metamodel Interrelations

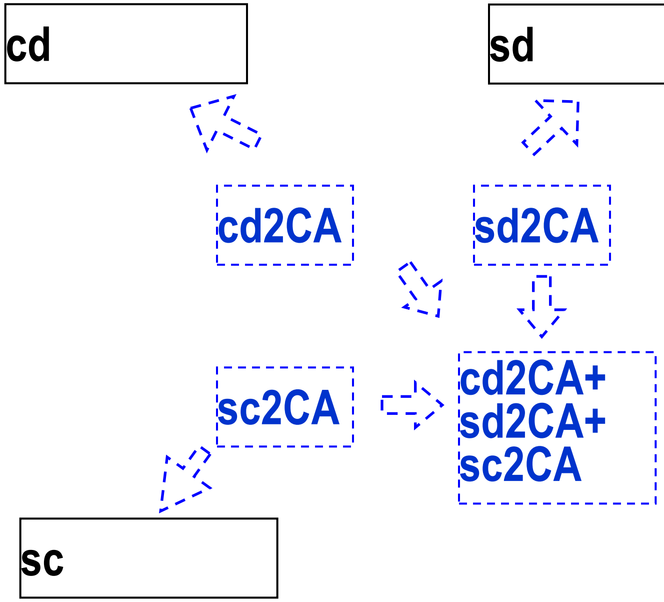


Summary – Heterogeneous Case

- Heterogeneous consistency check is reduced to the homogeneous one but metamodel merging is minimal
  - only to manage inter-metamodel constraints, working as locally as possible
- Despite heterogeneity, matching is type safe
- Applicability to a wide class of metamodeling techniques (based on graph-like structures)
- Formal foundations based on the well-established *institution theory*

Local vs. total consistency checking





Two approaches:

(a) Total direct merge: cd, sd, sc are considered instances of the same global metamodel M.
M can be derived from the metamodel mappings.

(b) Local merge: we first specify an overlap metamodel CA = a common view to CD, SD, SC. Then project the three models to the overlap and apply Consistency Checking by Merge.

At least two approaches sync in the relational case

Symmetric Lenses

- Complement-based
  - [Hofmann, Pierce, Wagner 2011]
  - Two functions
    - putr : X × C -> Y × C
    - putl : Y × C -> X × C
  - Can be built from two asymmetric ones
    - X ← (X × Y) → Y
- Delta-based
  - [MODELS'11]
  - Generalization of asymmetric delta lenses

Overlap-based approach

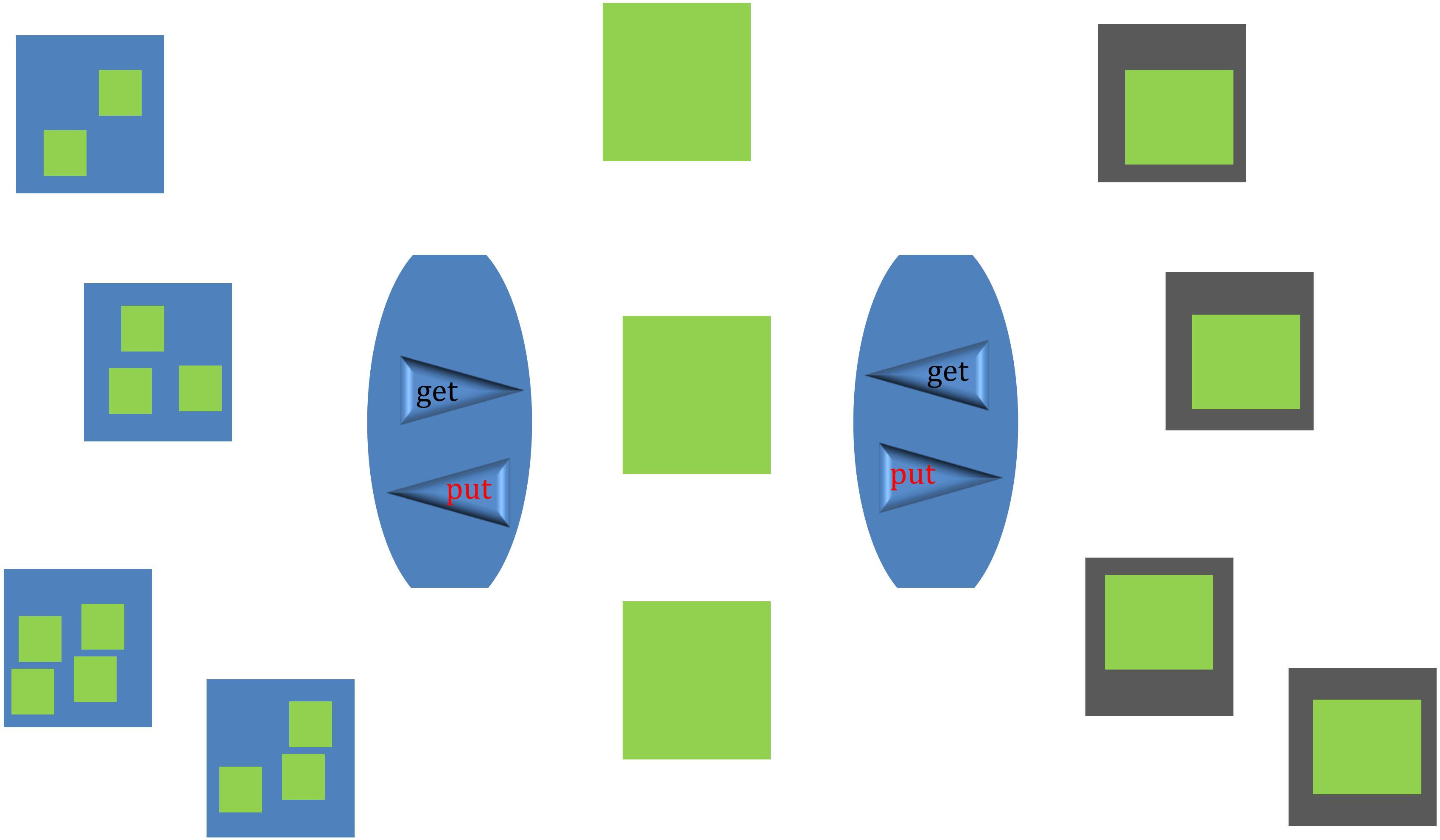
- Identify overlap metamodel
- Project both domains into the overlap
- Use two lenses into the overlap

- See [GTTSE’11]

Sequence diagrams

Overlap

State charts



Summary

Sketched an algebraic model-sync framework

Instantiated for design views on code

Advanced roundtrip engineering

Showed how to deal with general overlap of multiple heterogeneous models

Thanks for listening!

Questions?

References

[Sabetzadeh, Easterbrook 2006] M. Sabetzadeh and S. M. Easterbrook. View Merging in the presence of incompleteness and inconsistency. Requirements Engineering Journal, vol 11, pp174-193. 2006.

[Pierce et al] Foster,J.N., Greenwald, M.B.,Moore, J.T.,Pierce, B.C., Schmitt,A.. Combinators for bidirectional tree transformations: A linguistic approach to the view-update. problem. ACM Trans. Program. Lang. Syst. 29 (3) (2007)

[SLE’10] Bąk, K., K. Czarnecki, and A. Wąsowski, "Feature and Meta-Models in Clafer: Mixed, Specialized, and Coupled", 3rd International Conference on Software Language Engineering, Eindhoven, The Netherlands, 10/2010

[CVSM’09] Diskin, Z., K. Czarnecki, and M. Antkiewicz, "Model-versioning-in-the-large: Algebraic foundations and the tile notation", 2009 ICSE Workshop on Comparison and Versioning of Software Models (CVSM), Vancouver, BC, Canada, IEEE, pp. 7 - 12, 2009

[ICMT’10] Diskin, Z., Y. Xiong, and K. Czarnecki, "From State-Based to Delta-Based Bidirectional Model Transformation", 3rd International Conference on Model Transformation, Malaga, Spain, Springer, pp. 61-76, 06/2010

[TSE’09] M. Antkiewicz, K. Czarnecki, and M. Stephan, "Engineering of Framework-Specific Modeling Languages", IEEE Transactions on Software Engineering, vol. 35, issue 6, pp. 795 - 824, 11/2009

[ASEJ’09] M. Antkiewicz, T. Tonelli Bartolomei, and K. Czarnecki, "Fast Extraction of High-Quality Framework-Specific Models from Application Code", Automated Software Engineering, vol. 16, issue 1, pp. 101 - 144, 03/2009

[MDI’10] Diskin, Z., Y. Xiong, and K. Czarnecki, "Specifying Overlaps of Heterogeneous Models for Global Consistency Checking", 1st Workshop on Model Driven Interoperability, Co-located with MoDELS 2010, Oslo, Norway, ACM Press, pp. 42-51, 10/2010

[GTTSE’11] Diskin, Z. Model Synchronization: Mappings, Tiles, and Categories. GTTSE’09 Post-Proceedings, Springer, 2011

[MODELS’11] Diskin, Z., Y. Xiong, K. Czarnecki, H. Ehrig, F. Hermann, and F. Orejas, "From State- to Delta-based Bidirectional Model Transformations: the Symmetric Case", ACM/IEEE 14th International Conference on Model Driven Engineering Languages and Systems: Springer, 10/201

[JOT’11] Diskin, Z., Y. Xiong, and K. Czarnecki, "From State- to Delta-Based Bidirectional Model Transformations: the Asymmetric Case", Journal of Object Technology, vol. 10, 2011

See http://gsd.uwaterloo.ca/publications