Practice&Innovation Track

Solving the Instance Model View-Update Problem in AADL

Rakshit Mittal[1,2], Dominique Blouin[1], Anish Bhobe[1], and Soumyadip Bandyopadhyay[2]

Tool Demonstration Track

OSATE-DIM Solves the Instance Model View-Update Problem in AADL

Rakshit Mittal and Dominique Blouin

[1]Telecom Paris, Institut Polytechnique de Paris, Palaiseau, France
[2]Birla Institute of Technology and Science Pilani, Goa, India

AADL

- Architecture Analysis and Design Language
  - SAE Standard AS5506D

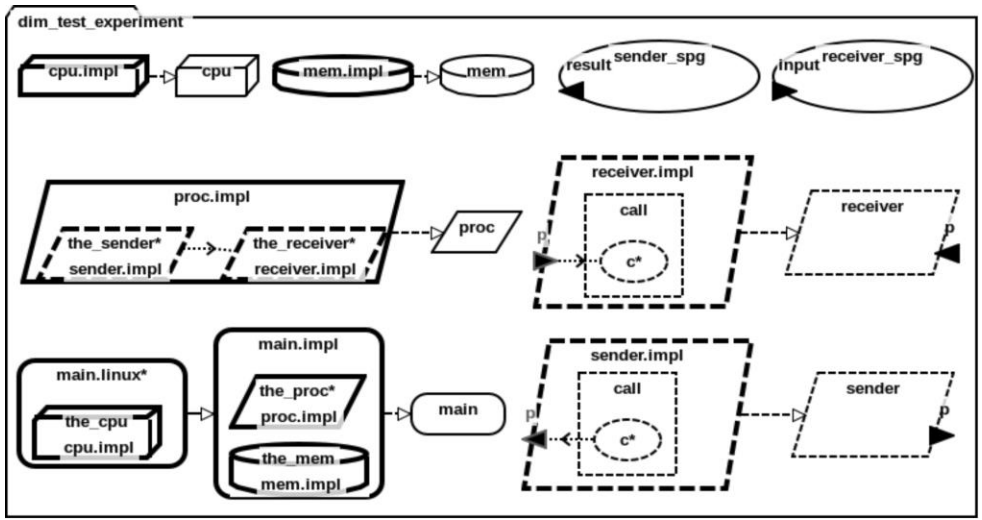
- to model real-time embedded systems composed of
  - software and
  - physical execution platform components
  - tightly coupled with actuators and sensors
  - to interact with their environments (Cyber-Physical Systems!)

- scheduling/flow-control analyses
- code generation for embedded platforms

- Adopted as the core of the US DoD Digital Engineering Strategy
  - entering production within the Future Vertical Lift program of US Army
  - this project funded by US Army CCDC-DevCom ATLANTIC

AADL Constructs

Blended syntax




```
1  package dim_test_experiment
2  public
3  with Base_Types; renames Base_Types::all;
4  with RAMSES_Properties;
5
6    system main
7    end main;
8
9    system implementation main.impl
10   subcomponents
11     the_proc: process proc.impl;
12     the_mem: memory mem.impl;
13   properties
14     actual_memory_binding => (reference (the_mem)) applies to the_proc;
15   end main.impl;
16
17   system implementation main.linux extends main.impl
18     subcomponents
19         the_cpu: processor cpu.impl {RAMSES_Properties::Target => "linux";};
20     properties
21         actual_processor_binding => (reference (the_cpu)) applies to the_proc;
22   end main.linux;
23
24   processor cpu
25   end cpu;
26
27   processor implementation cpu.impl
28   properties
29       Scheduling_Protocol => (RMS) ;
30   end cpu.impl;
31
32   process proc
33   end proc;
```

- Components
  - Classifiers
    - Type
    - Implementation
  - Extensions
- Features
  - Refinements
- Connections
  - Refinements

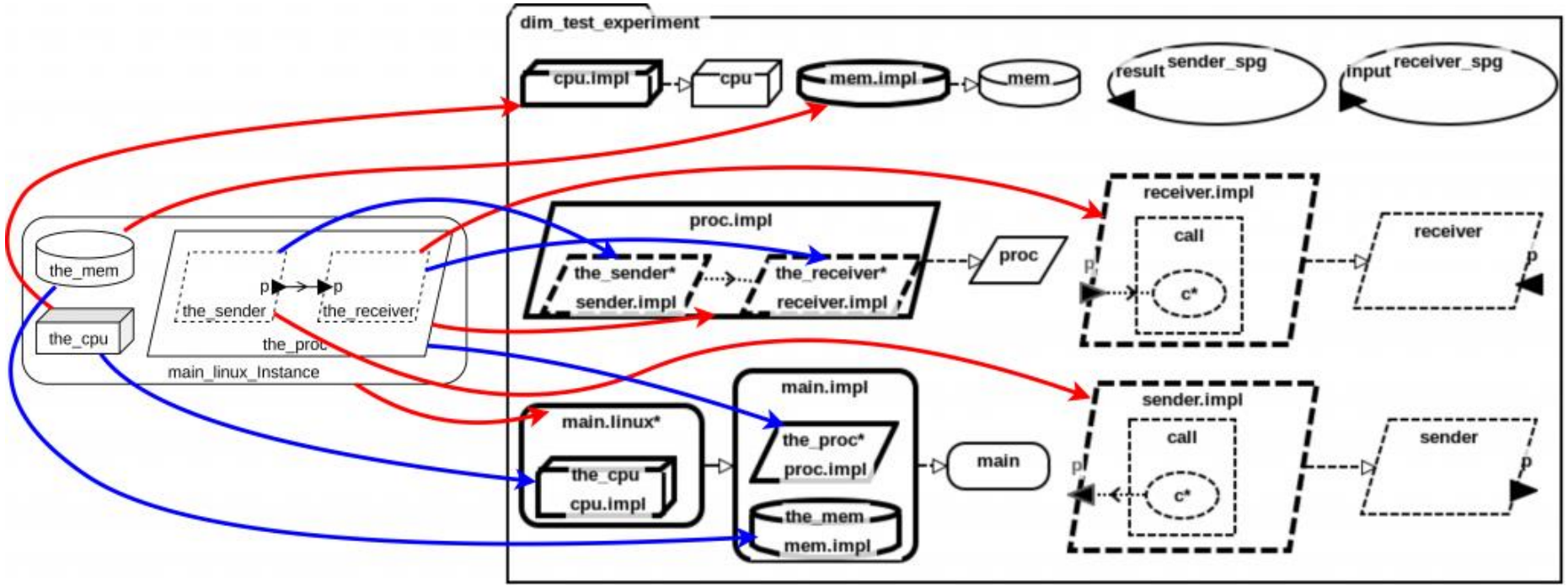
OSATE

- Open-Source AADL Tool Environment
- Eclipse IDE plugin
- Latest version (OSATE 2.11) compatible with Eclipse 2022-03
- Primarily maintained by SEI-CMU
- Safety-critical analyses and verification with languages like Resolute and AGREE
- Code generation capabilities with RAMSES, OCARINA
- https://osate.org
- Many industrial projects > many projects to make OSATE more robust

-- OSATE demo --

OSATE Instance Model

blue: subcomponent reference
red: classifier reference



Instance Model

Declarative Model

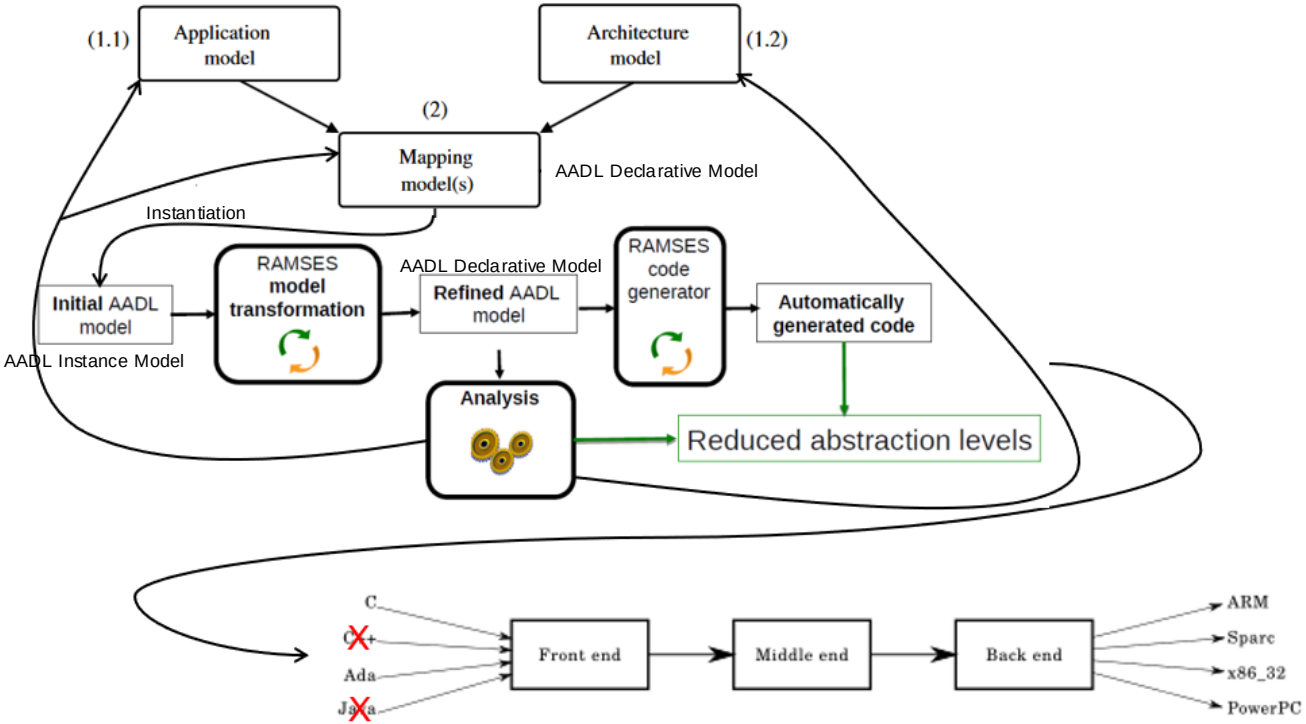
Why Instance model?

- Simplicity
- Tree-graph structure of containment
- No complex cross-tree relations (except property references)
    - No references up the branch as well
- All information stored locally within each component/element
- Hence, most tools use Instance models for analysis.

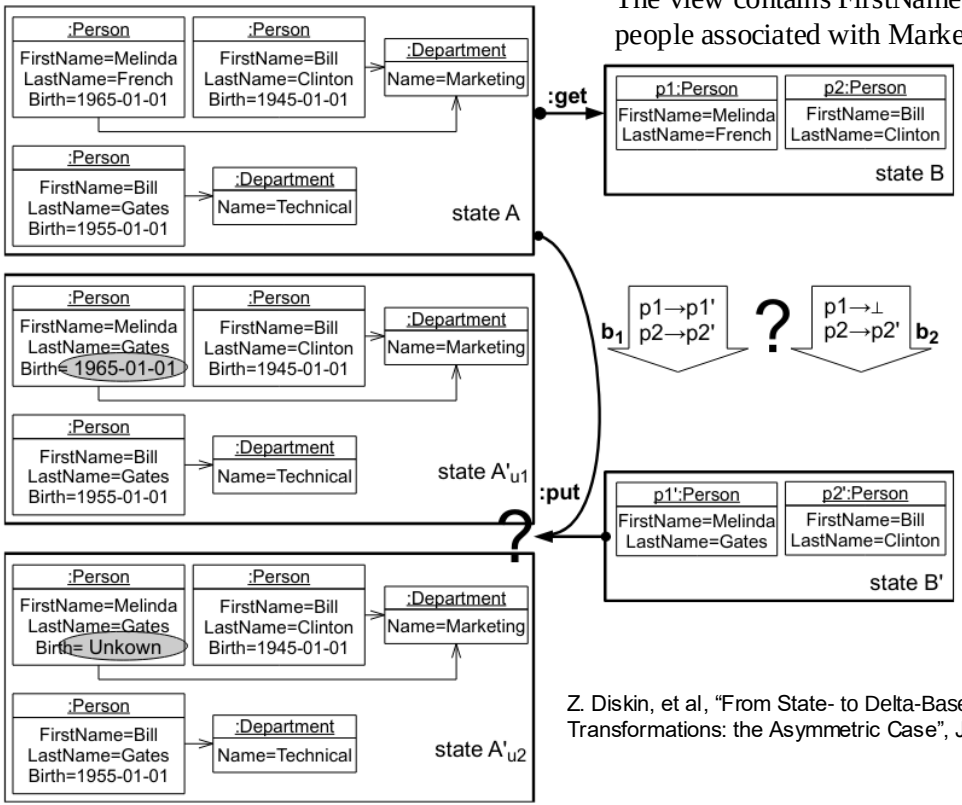
-- Instantiation demo --

RAMSES Workflow

(Refinement of AADL Models for the Synthesis of Embedded Systems)



View-Update Problem



The view contains FirstName and LastName only, of all people associated with Marketing department

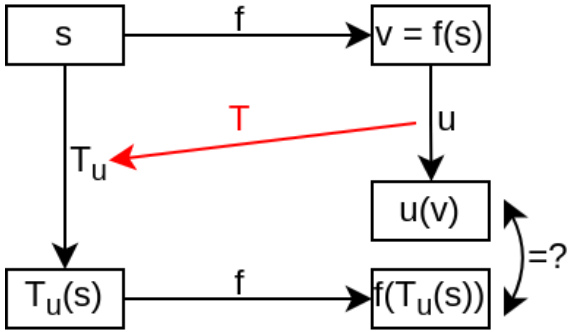
Q. Change name of Melinda French to Melinda Gates
b1 > Change the LastName property of p1
b2 > Delete p1and add p1' with FirstName, LastName

b2 causes loss of Birth data of Melinda.

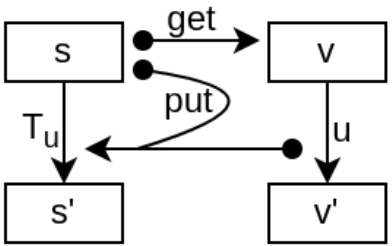
Even though the state B' is the same for b1 and b2.
How to capture such a difference in the base model?

Z. Diskin, et al, "From State- to Delta-Based Bidirectional Model Transformations: the Asymmetric Case", Journal of Object Technology, 2011

Solution: Delta-based Lens



(a)



(GetPut) s = put(Φ,s)(s)
(PutGet) get(put(u,s)(s)) = v'
(PutPut) put(u',put(u,s)(s)) = put(u∘u',s)

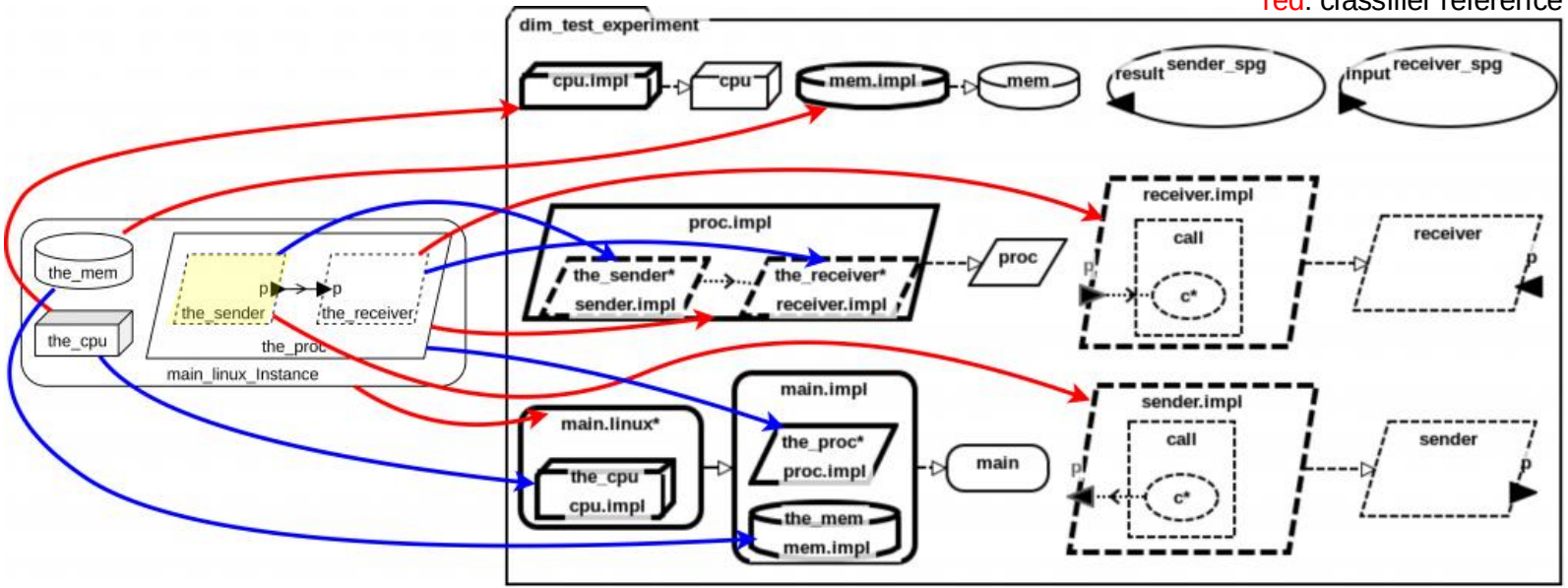
(b)

s : Model State => Declarative Model
f : View-Generating Function => Instantiation
v : View-State => Instance Model
u : View-Update => Updates of Instance Model
T : Translation => Deinstantiation

f is not a bijection!

Complications in AADL View-Updates

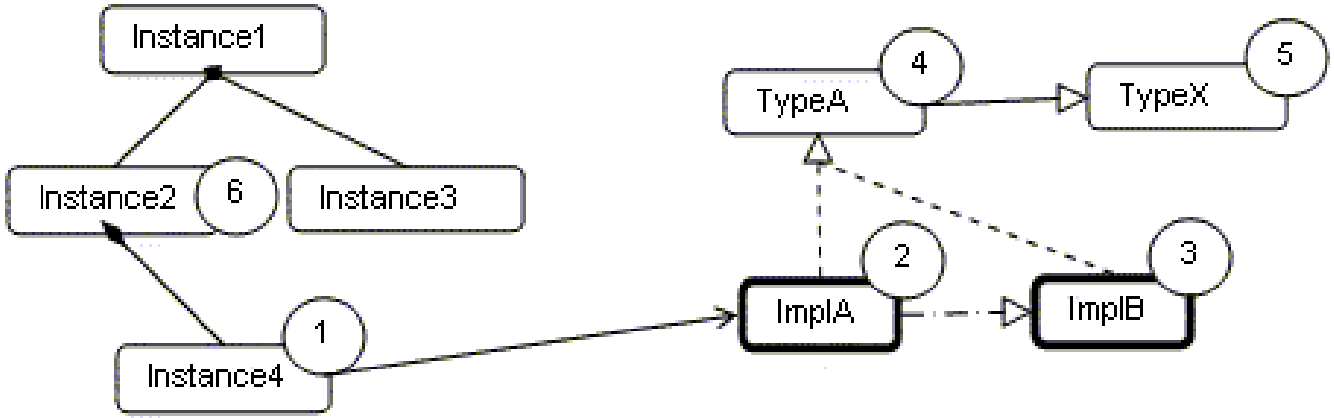
blue: subcomponent reference
red: classifier reference



Q. Add a property to *the_sender*

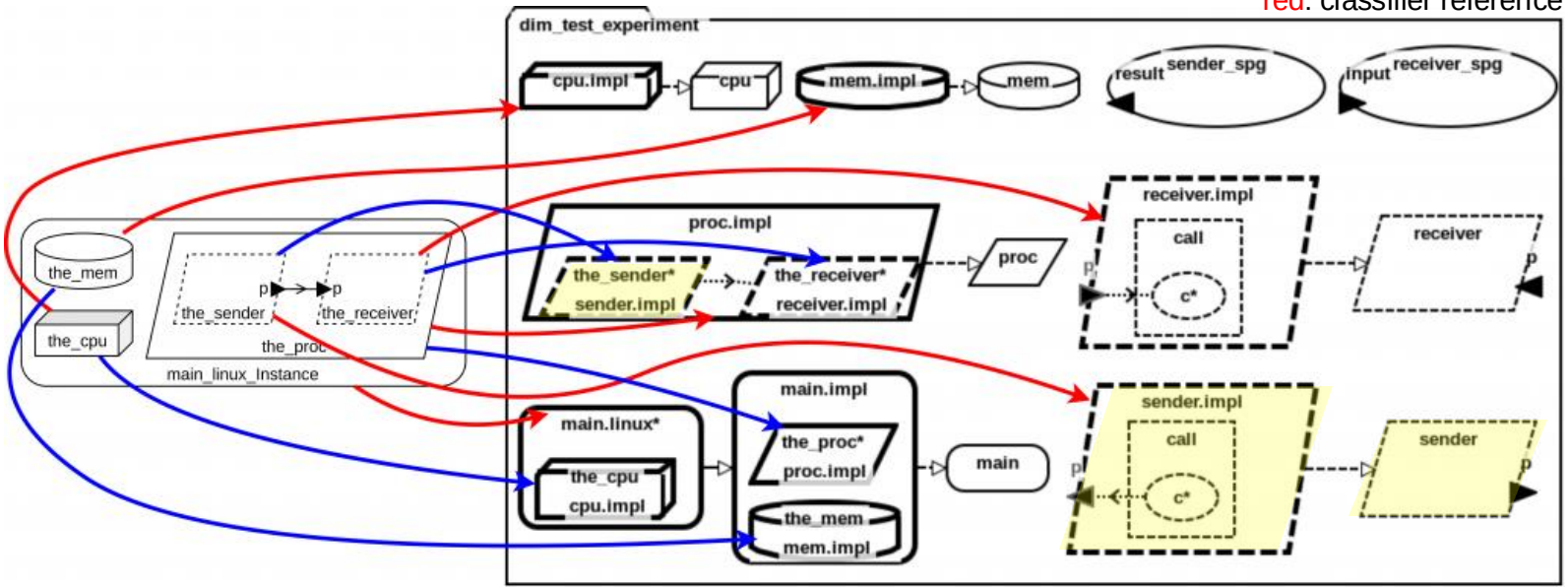
Complications in AADL View-Updates

AADL Property Value Determination



Complications in AADL View-Updates

blue: subcomponent reference
red: classifier reference

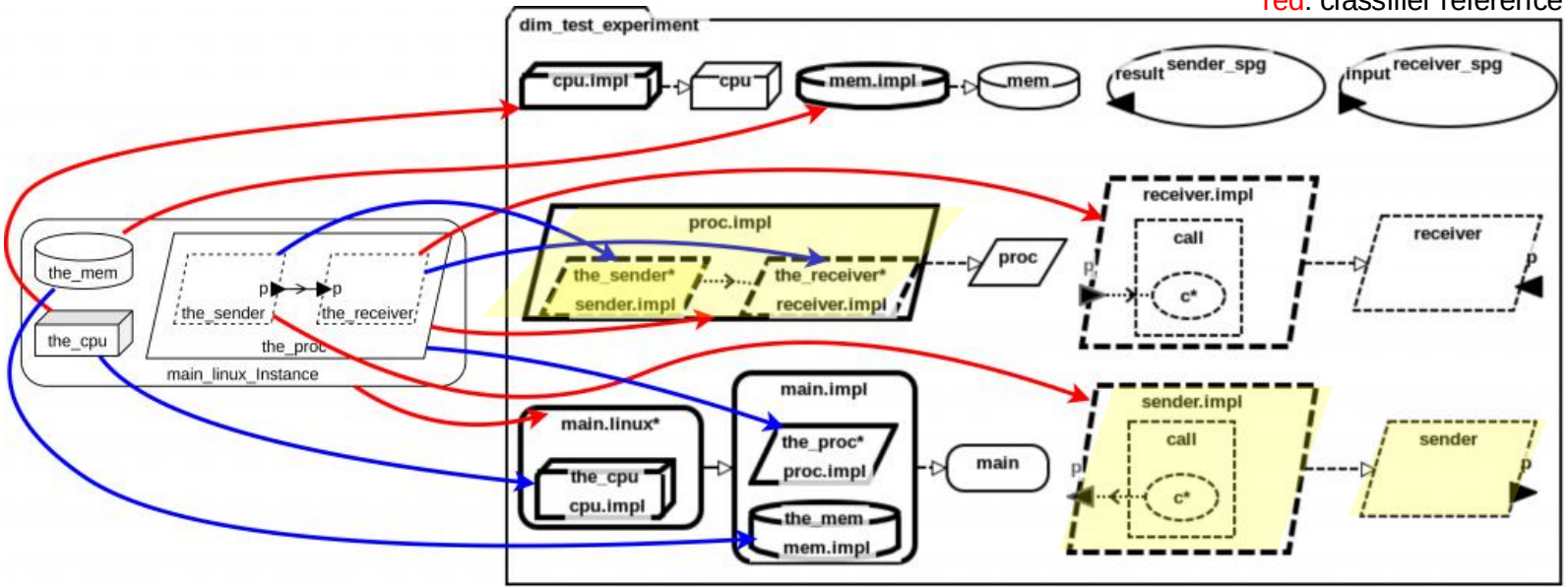


Q. Add a property to *the_sender*

Soln. Add property to ***the_sender**** or ***sender.impl*** or ***sender*** ??

Complications in AADL View-Updates

blue: subcomponent reference
red: classifier reference



Q. Add a property to *the_sender*

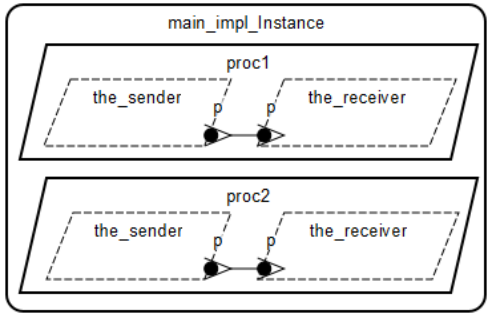
Soln. Add property to ***the_sender**** or ***sender.impl*** or ***sender***

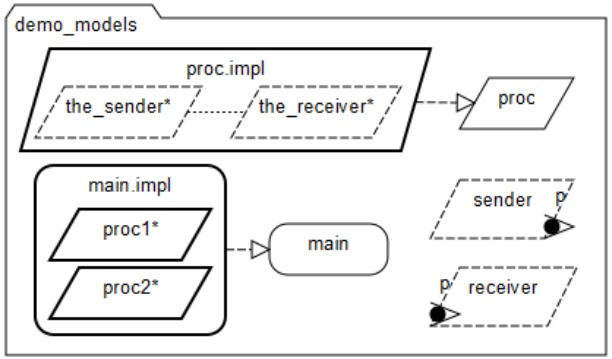
or to ***proc.impl*** *(applies to the_sender*)* ??

Too many choices for de-instantiation in AADL!

Another Example: Need for automated de-instantiation

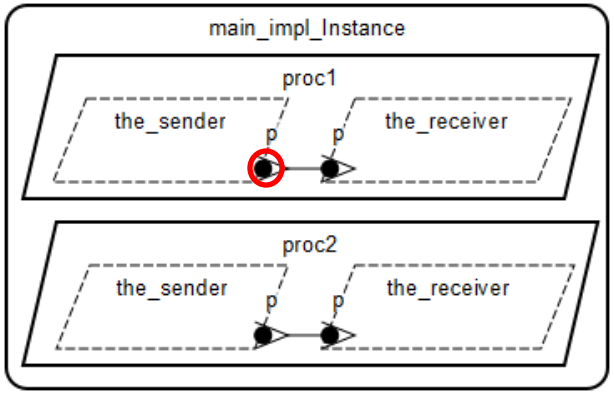
```
package demo_models
public
    system main
    end main;

    system implementation main.impl
        subcomponents
            proc1: process proc.impl;
            proc2: process proc.impl;
    end main.impl;

    process proc
    end proc;

    process implementation proc.impl
        subcomponents
            the_sender: thread sender;
            the_receiver: thread receiver;
        connections
            cnx: feature the_sender.p -> the_receiver.p;
    end proc.impl;

    thread sender
        features
            p: out feature;
    end sender;

    thread receiver
        features
            p: in feature;
    end receiver;
end demo_models;
```

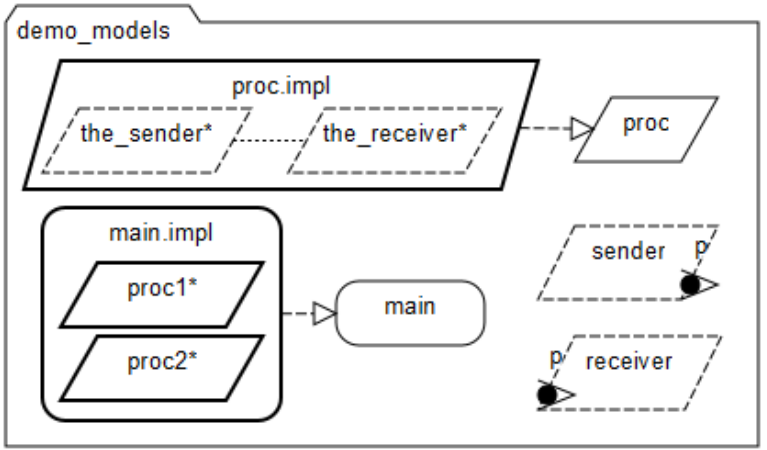




proc1 and *proc2* have the same classifier *proc.impl*

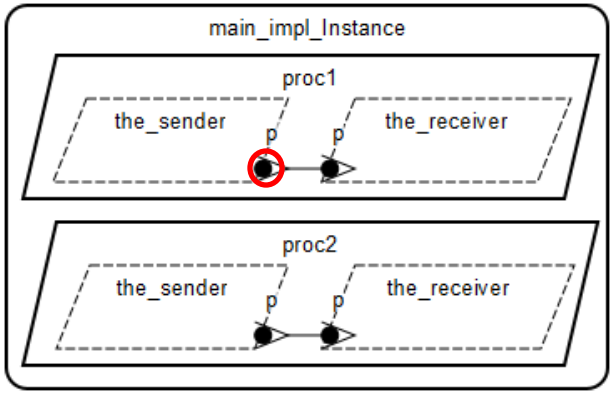
Another Example: Need for automated de-instantiation

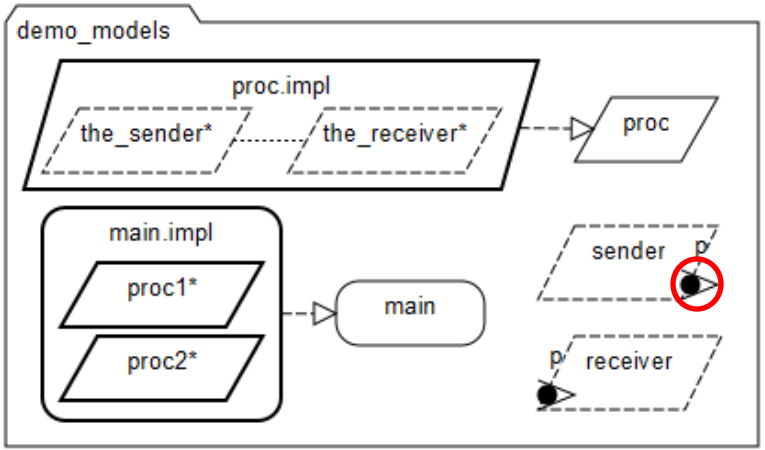




We want to change the abstract feature *p* in *proc1.the_sender* to data port

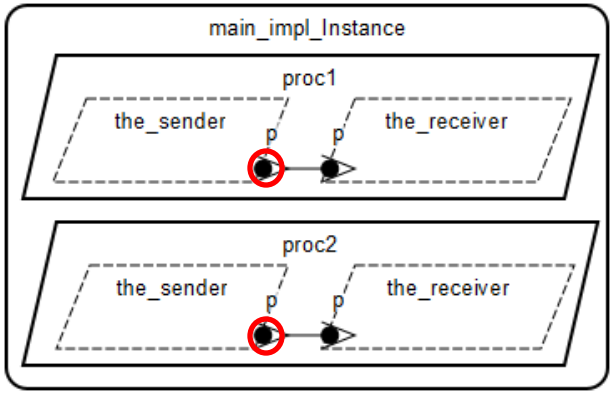
Example: Need for automated de-instantiation

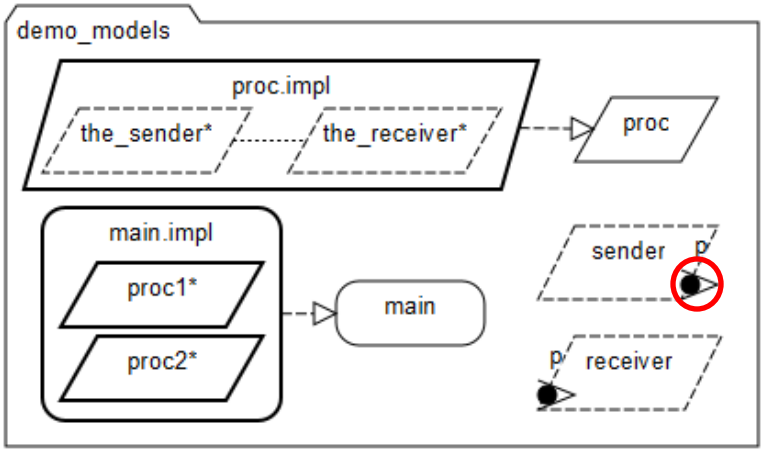




Simple! Change the corresponding feature.

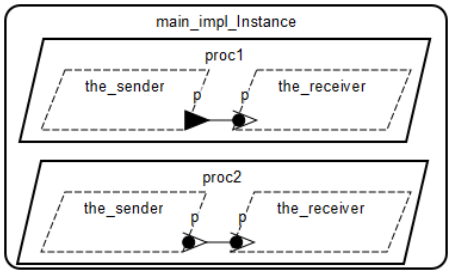
Example: Need for automated de-instantiation

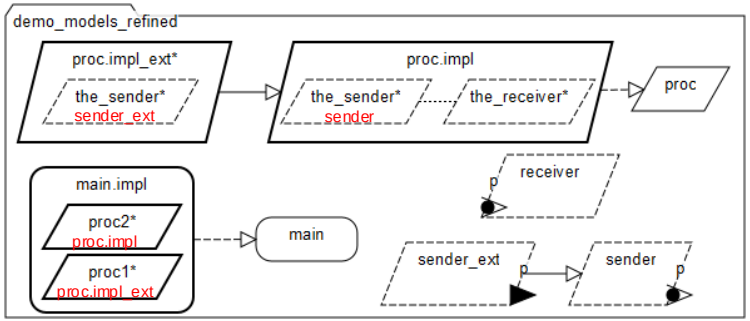




But this also changes *p* in *proc2.the_sender !!*

Example: Need for automated de-instantiation

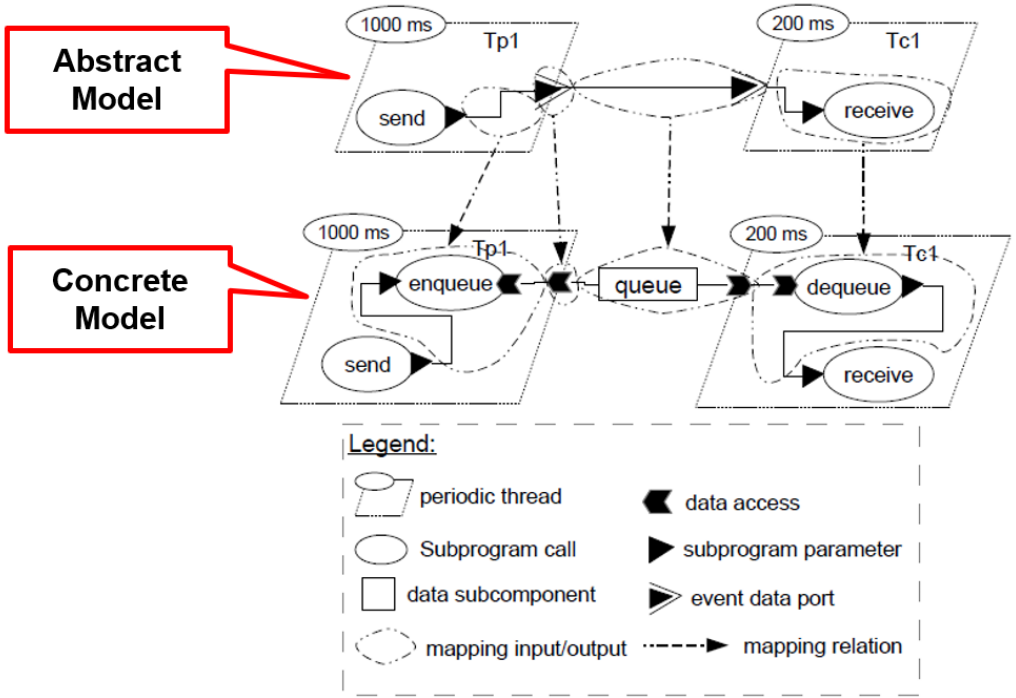




Solution with preservation of information is very complicated with many extensions and refinements, even for a simple instance update!

```
package demo_models_refined
public
    system main
    end main;

    system implementation main.impl
        subcomponents
            proc1: process proc.impl_ext;
            proc2: process proc.impl;
    end main.impl;

    process proc
    end proc;

    process implementation proc.impl
        subcomponents
            the_sender: thread sender;
            the_receiver: thread receiver;
        connections
            cnx: feature the_sender.p -> the_receiver.p;
    end proc.impl;

    process implementation proc.impl_ext extends proc.impl
        subcomponents
            the_sender: refined to thread sender_ext;
    end proc.impl_ext;

    thread sender
        features
            p: out feature;
    end sender;

    thread sender_ext extends sender
        features
            p: refined to out data port;
    end sender_ext;

    thread receiver
        features
            p: in feature;
    end receiver;

end demo_models_refined;
```

Real-life Scenario: RAMSES



Simplest refinement pattern in RAMSES:

Changing data port connection between two threads by replacing with a shared data component.

The data port features are changed to data access features.

Too many choices for de-instantiation in AADL!

+

Complications for information preservation due to many dependencies between elements and modularity.

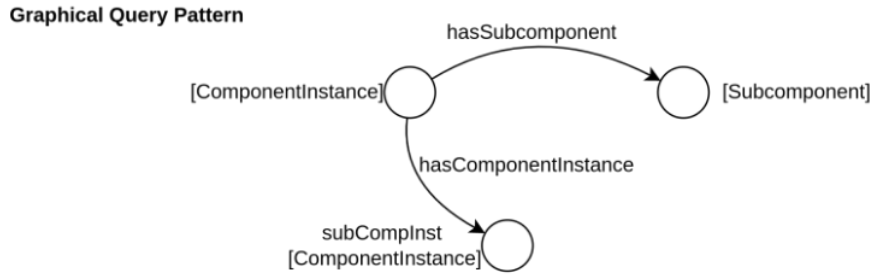
=

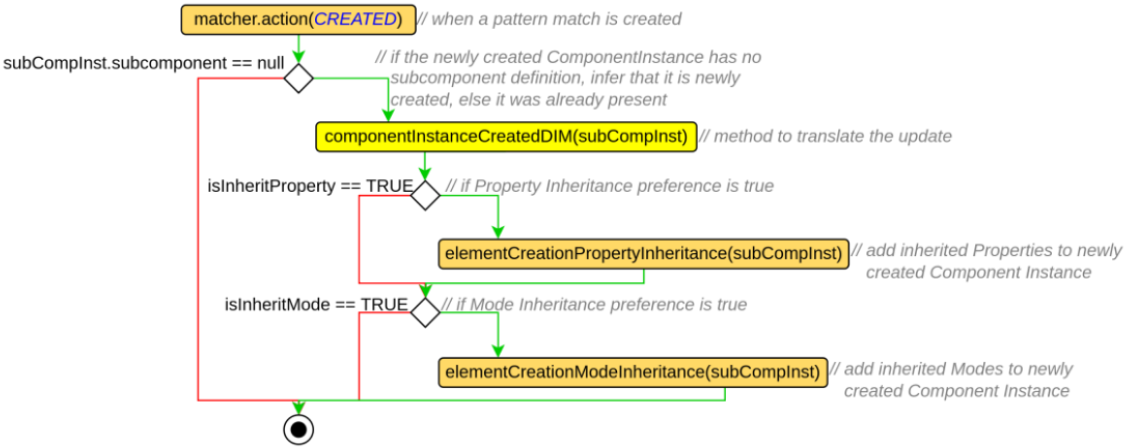
Makes de-instantiation of updates highly complex; requiring automation

OSATE-DIM

- OSATE Declarative-Instance Mapping
- Eclipse/OSATE-based plugin

- Graph Transformations
  - VIATRA
    - Graphical Queries
    - Model Transformation Rules





OSATE-DIM Values/Aims

- Maximum Information Preservation
- Least/Minimal Change

- Very-well behaved lens (3 Lens laws)
  - No extraneous model updates.
  - Equality of updated-model state with updated view-state.
  - Composability of updates.

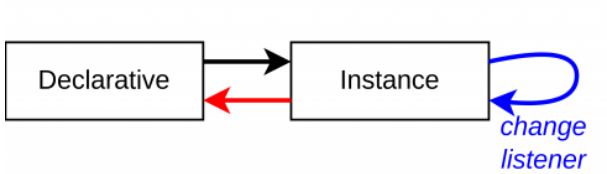
- Flexibility
  - Scenarios
  - User preferences

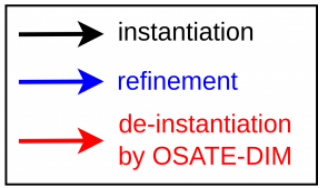
Transformation Scenarios

State-based

-- State-based Demo --

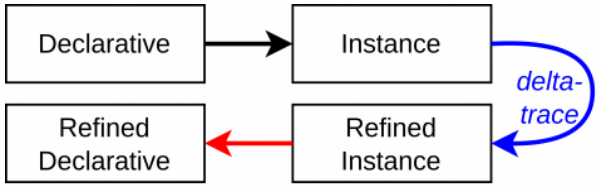
Delta-based with In-place refinement



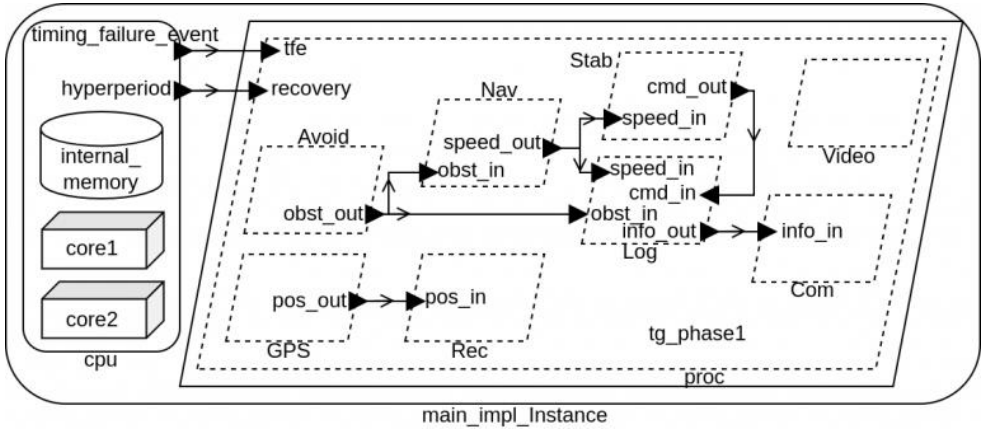


-- Delta in-place Demo --

Delta-based with Out-of-place refinement

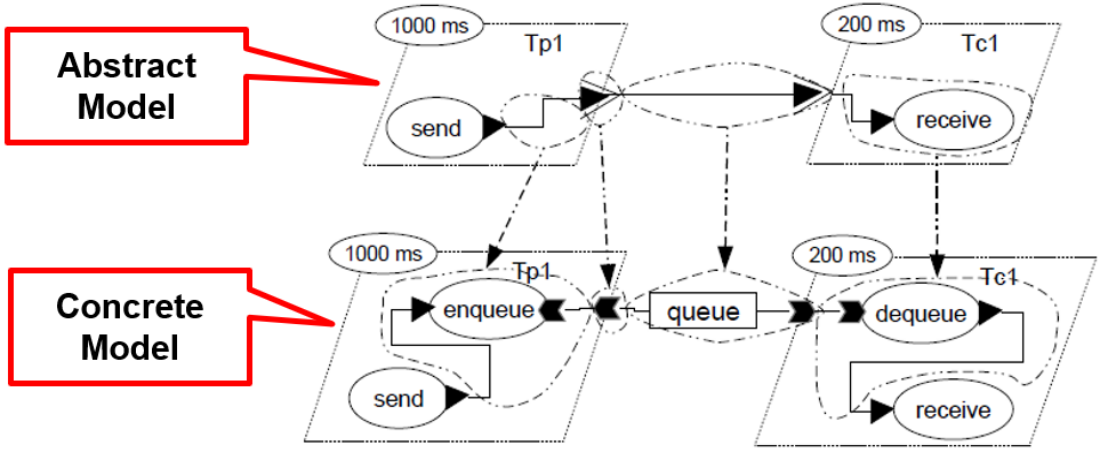


Case Study: MC-DAG



- Addition of *Property Associations* (RAMSES::Execution_Slots) for each *Thread*.
- Contain static scheduling tables for each *Thread* in different *Modes* LO and HI.
- Properties also reference the core and memory binding model elements, not just static data.

Case Study: RAMSES



- Addition of 1 (+40) *Data Components* to a *Process Component*, which are shared by two threads.
- The *Port Features* interfacing the two threads with each other are changed to *Data Access* kinds.
- New *Data Access Connections* are also added between the shared *Data Components* and the *Threads*.
- The added *Data Components* have varying numbers of *Properties*, and the total number of newly added properties is 122.

Conclusion

Introduced View-Update Problem in AADL-OSATE

OSATE-DIM is an automated solution for synchronizing Instance and Declarative models: 'de-instantiating' the Instance model

Three different scenarios

Wide range of view-updates supported

Tested on a preliminary test-bench

Simplifies the development of AADL model refinement tools

Future Work

Complete Implementation of Delta Out-of-place scenario

Further validation

Dissemination of OSATE-DIM to the AADL community

Concepts have potential to be used for "transpilation"

Thank you for your attention
Questions?: Dominique

Tool
Webpage: mem4csd.telecom-paristech.fr
Zenodo artifact DOI: 10.5281/zenodo.6971720

Contact:
Dominique: dominique.blouin@telecom-paris.fr
Rakshit: rakshit.mittal@uantwerpen.be