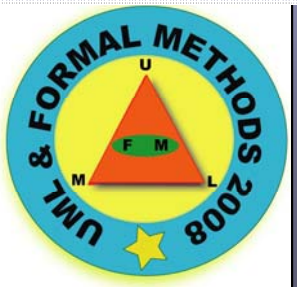
UML Behavioral Consistency Checking using Instantiable Petri Nets

Oct 27, 2008

Yann Thierry-Mieg, Lom-Messan Hillah
Laboratoire d'Informatique de Paris 6 (LIP6)

First IEEE International workshop UML and Formal Methods
In Conjunction with: 10th Int'l Conf. On Formal Engineering Methods



Context: ModelPlex project

- European IP: *MODELling solution for comPLEX software systems*
  - 2006-2010, 21 partners, 20M€
  - Model centric development
  - Reduce costs, increase quality/productivity
- WP 4: simulation, *verification* & testing (SVT)
  - Model checking is one of many services provided to models
- Develop "Behavioral Consistency Checker" a tool that transparently relies on Model-Checking to offer verification of UML behaviors

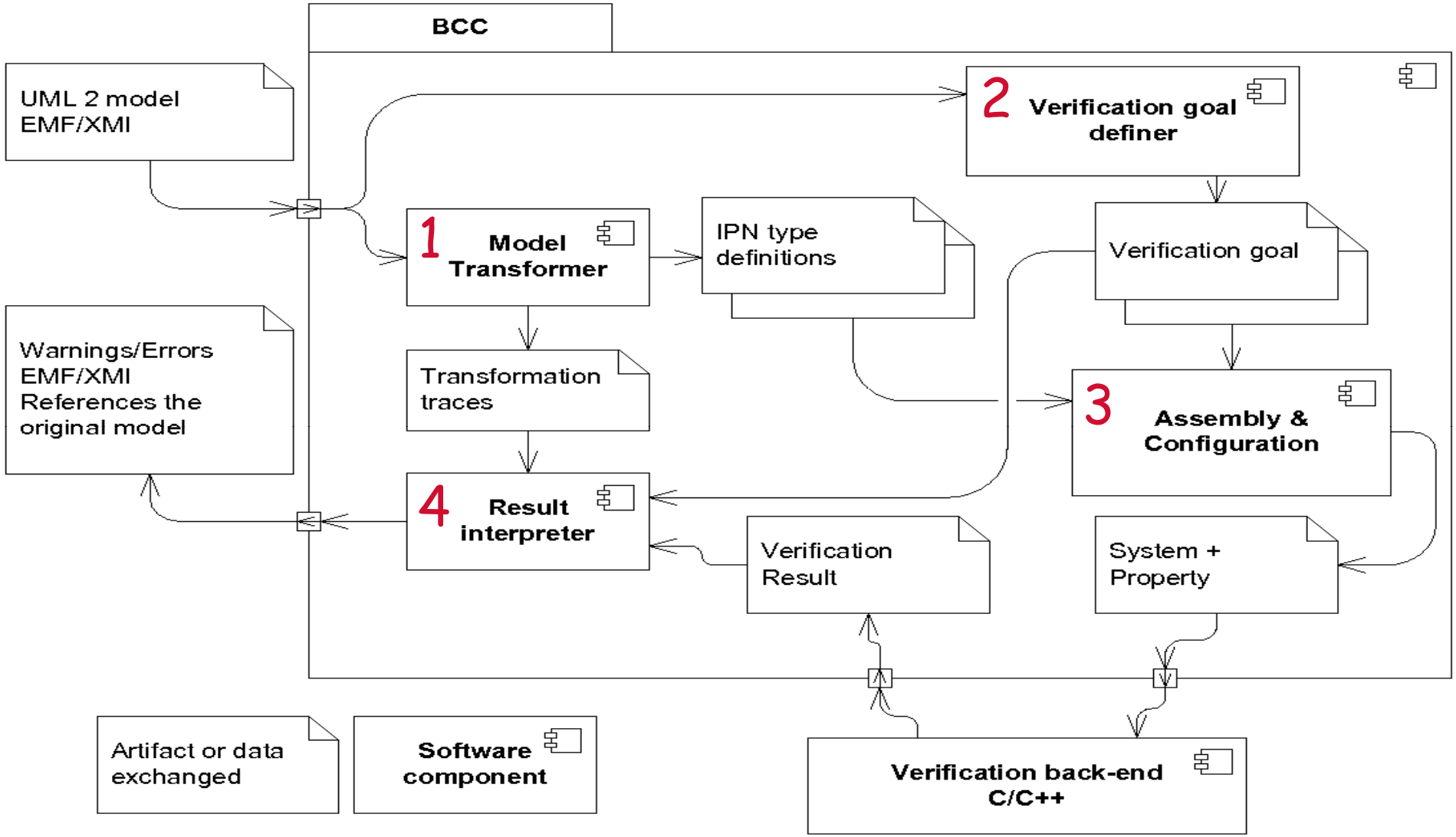
Goals of BCC

- Go beyond syntactic checks:
  - Offered by EVL (Epsilon), OCL
- Transparently use formal verification:
  - Minimal user input
  - Model-checking technology (e.g. Decision diagrams)
- Check that the *behavior* of a system specified in UML2 is *consistent*
  - Integrate execution semantics

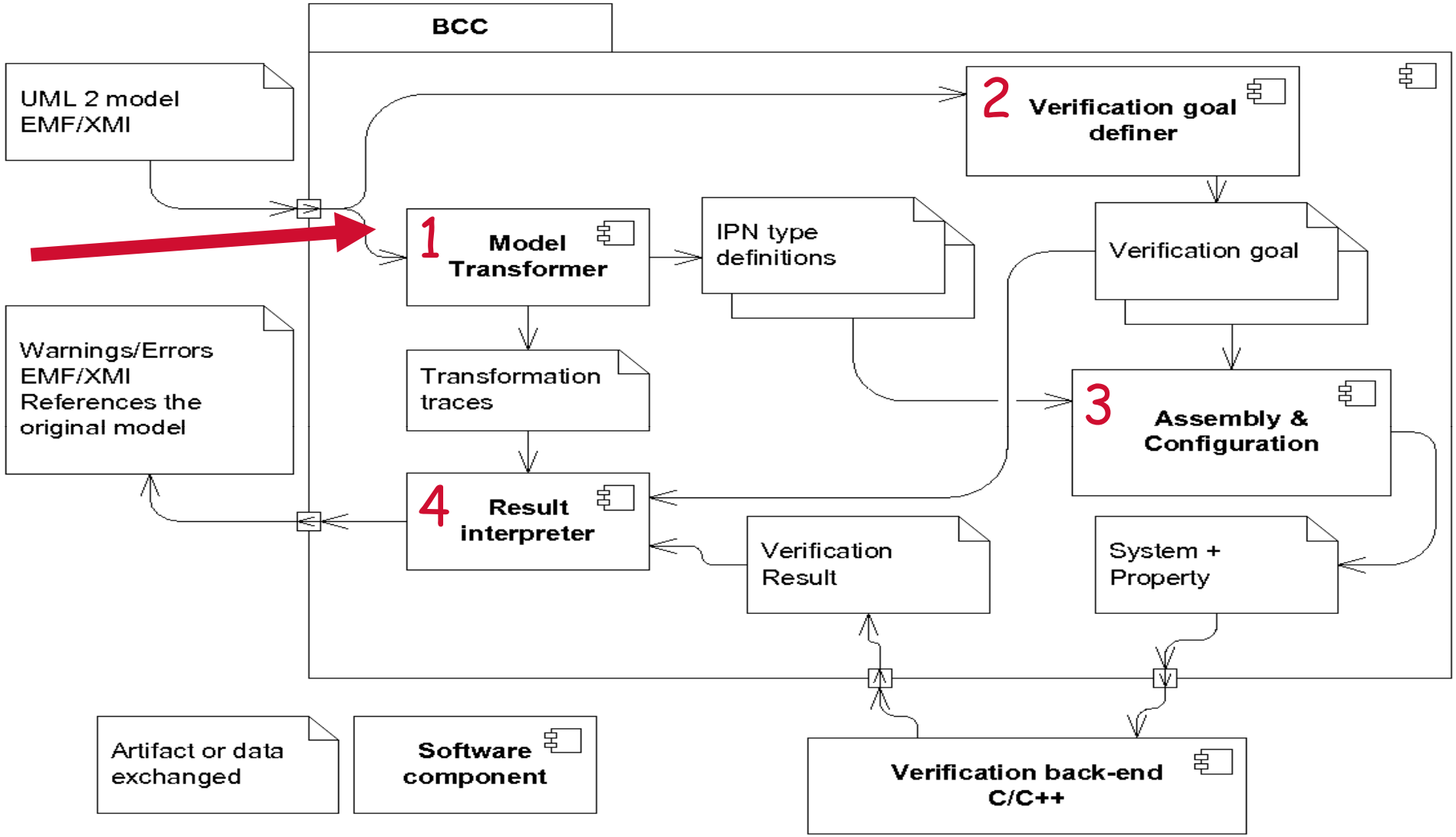
Means of BCC

- Translate UML to a formal notation
  - Consistent with MDA approach: a PSM for verification
- A generic description of "UML Behavior" semantics
  - Notion of type and instance
  - Use of an interface = public transitions
  - Composition through synchronizations:
    - *CallBehavior/EndBehavior*
    - *Send/Receive Event*
- Petri nets as an elementary component
  - Allow reuse of existing verification technology/tools

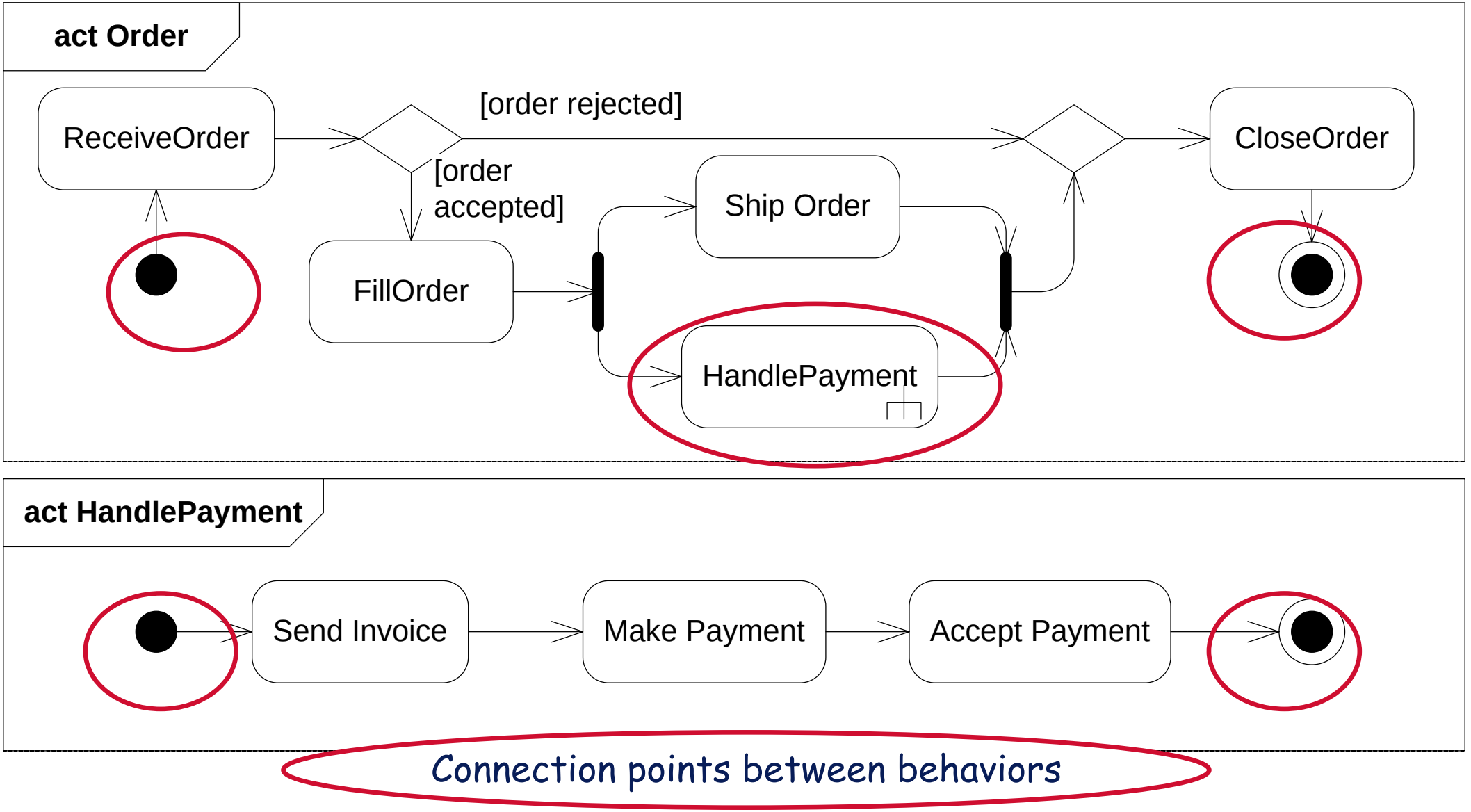
Behavioral Consistency Checker: Internal Architecture



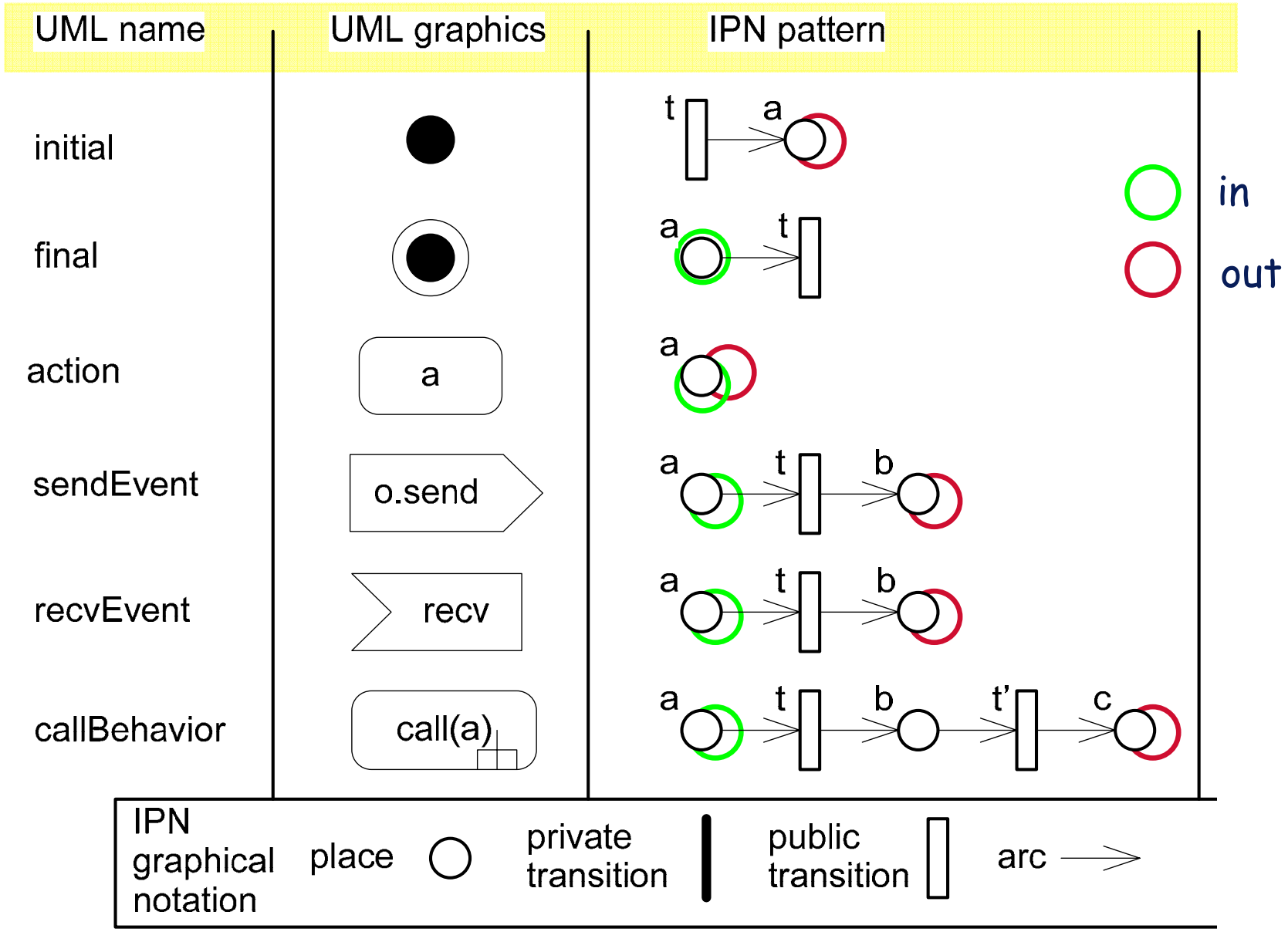
Step 1: Transformation



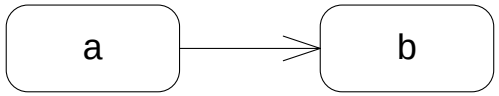
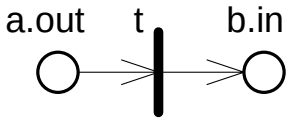
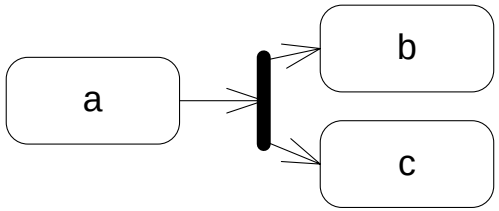
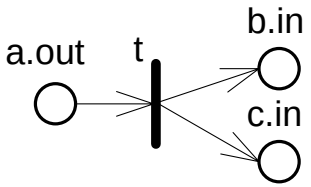
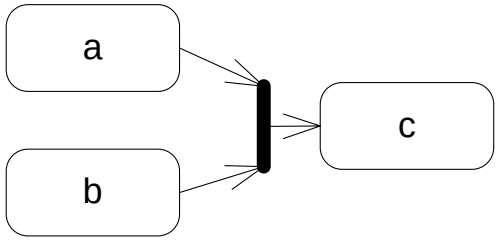
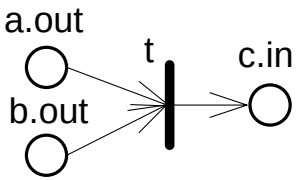
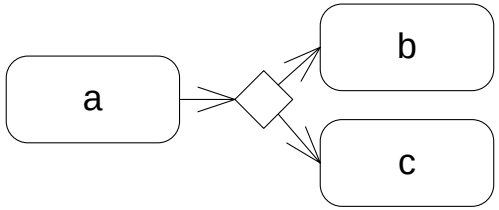
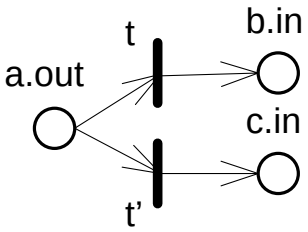
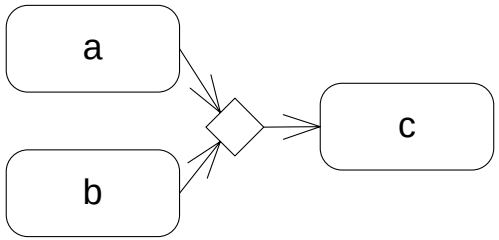
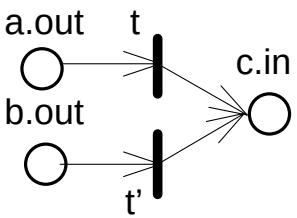
An example translation (Activity diagram)



Translation rules from UML-AD to IPN: nodes

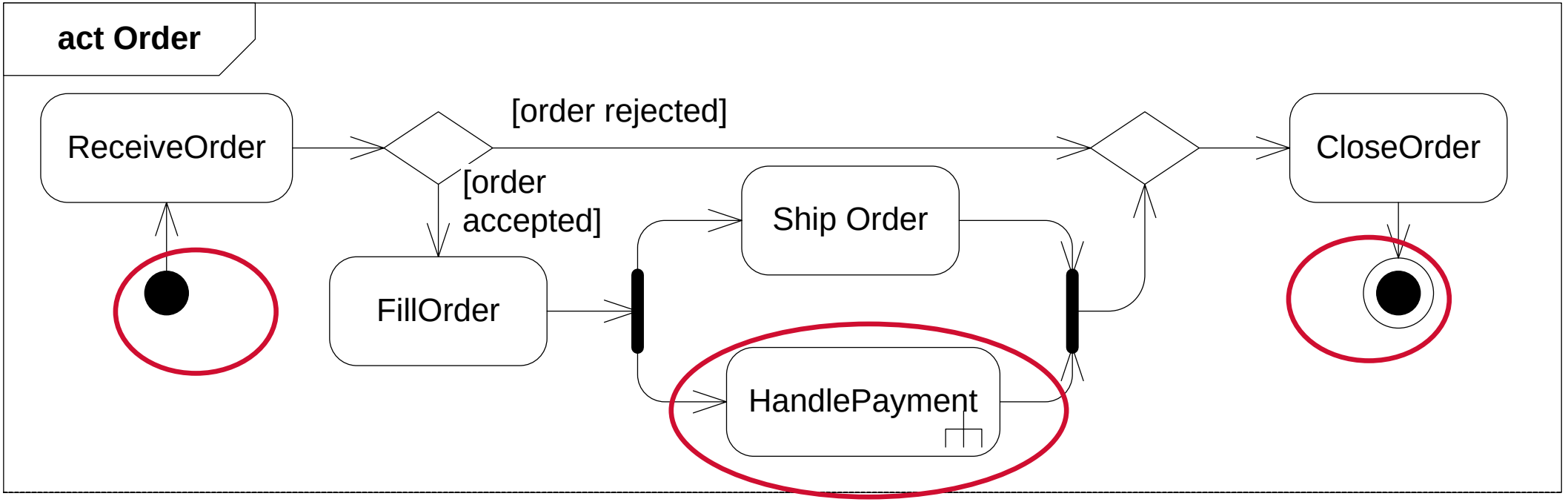


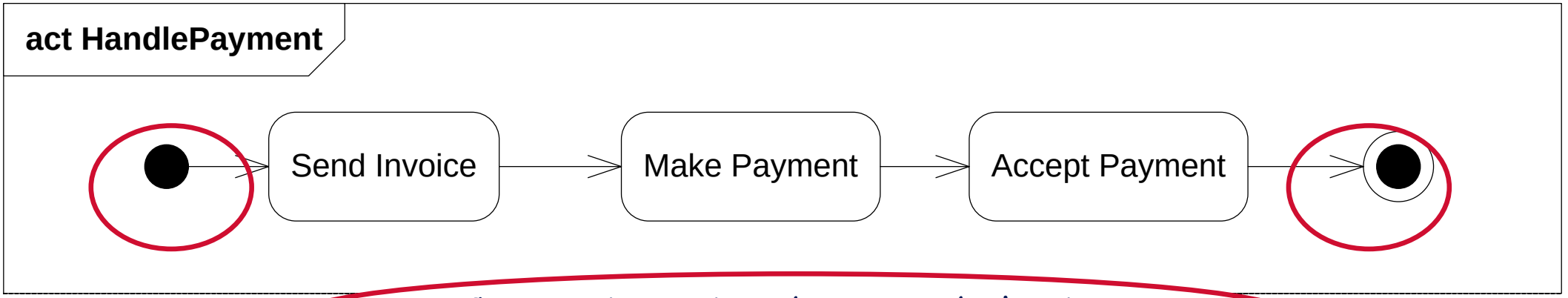
Translation rules from UML-AD to IPN: edges

UML name	UML graphics	IPN pattern
Control Flow		
Fork		
Join		
Decision		
Merge		

Only private transitions

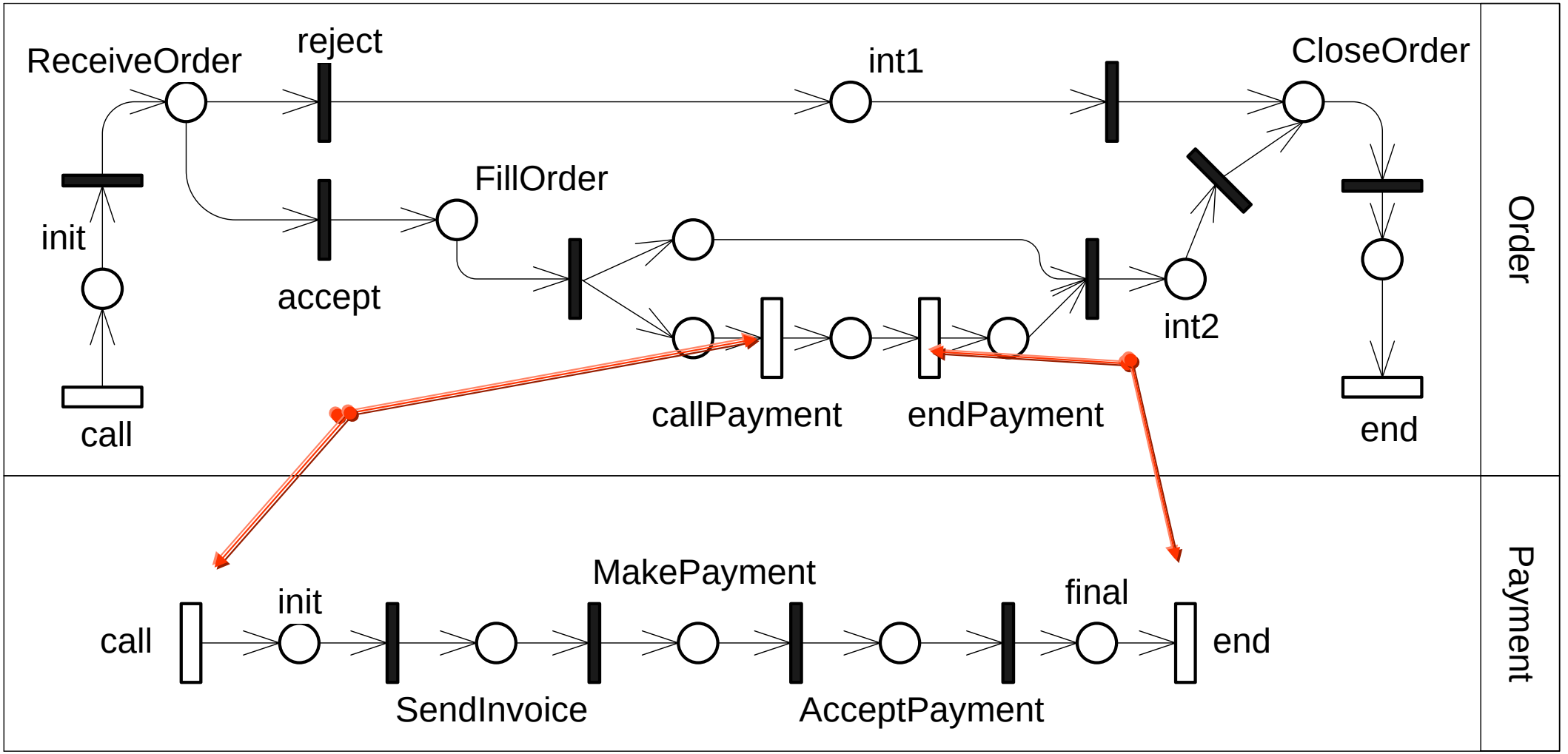
An example translation (Activity diagram)



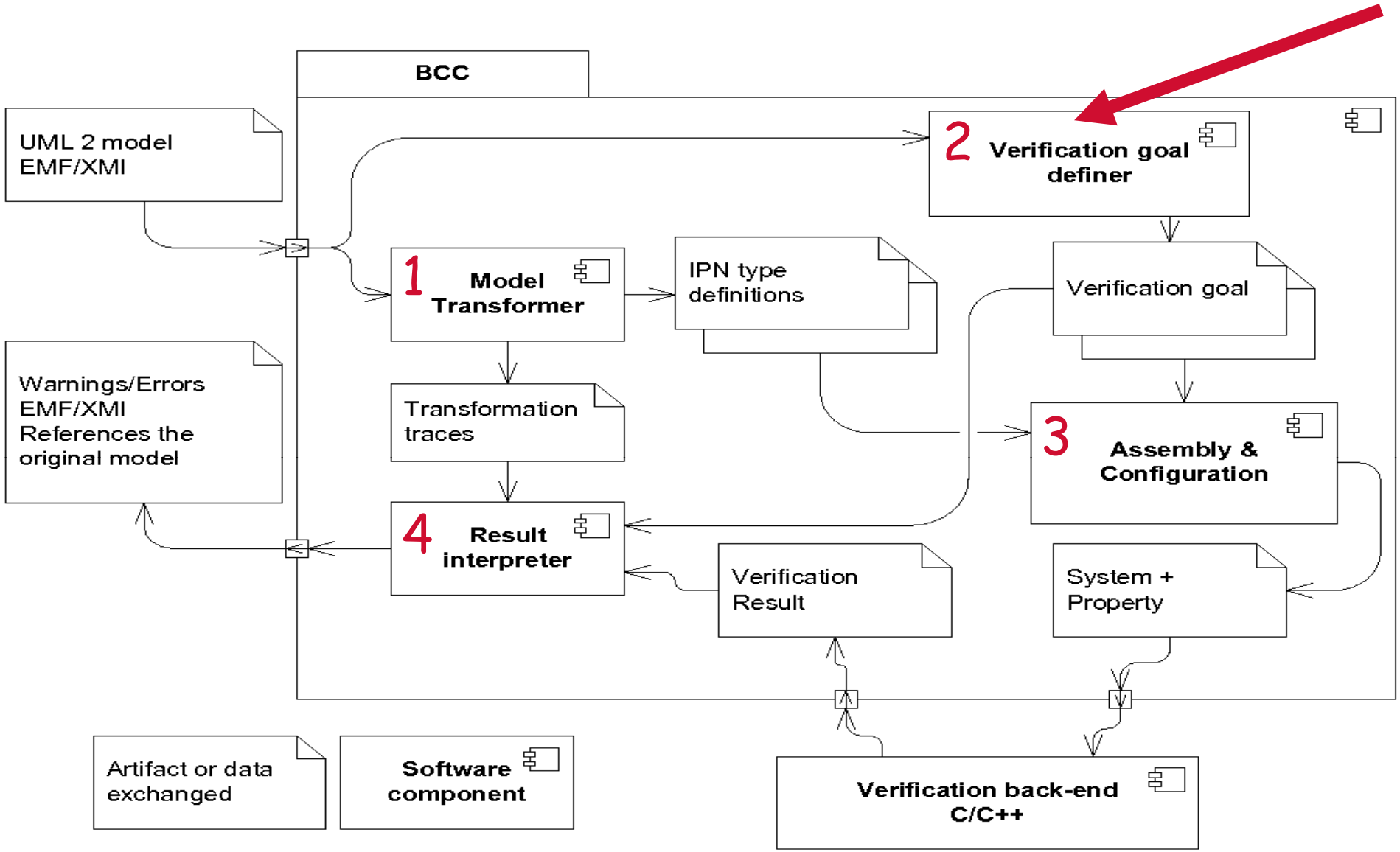


Connection points between behaviors

An example translation (IPN): 1 Diagram <-> 1 IPN Type



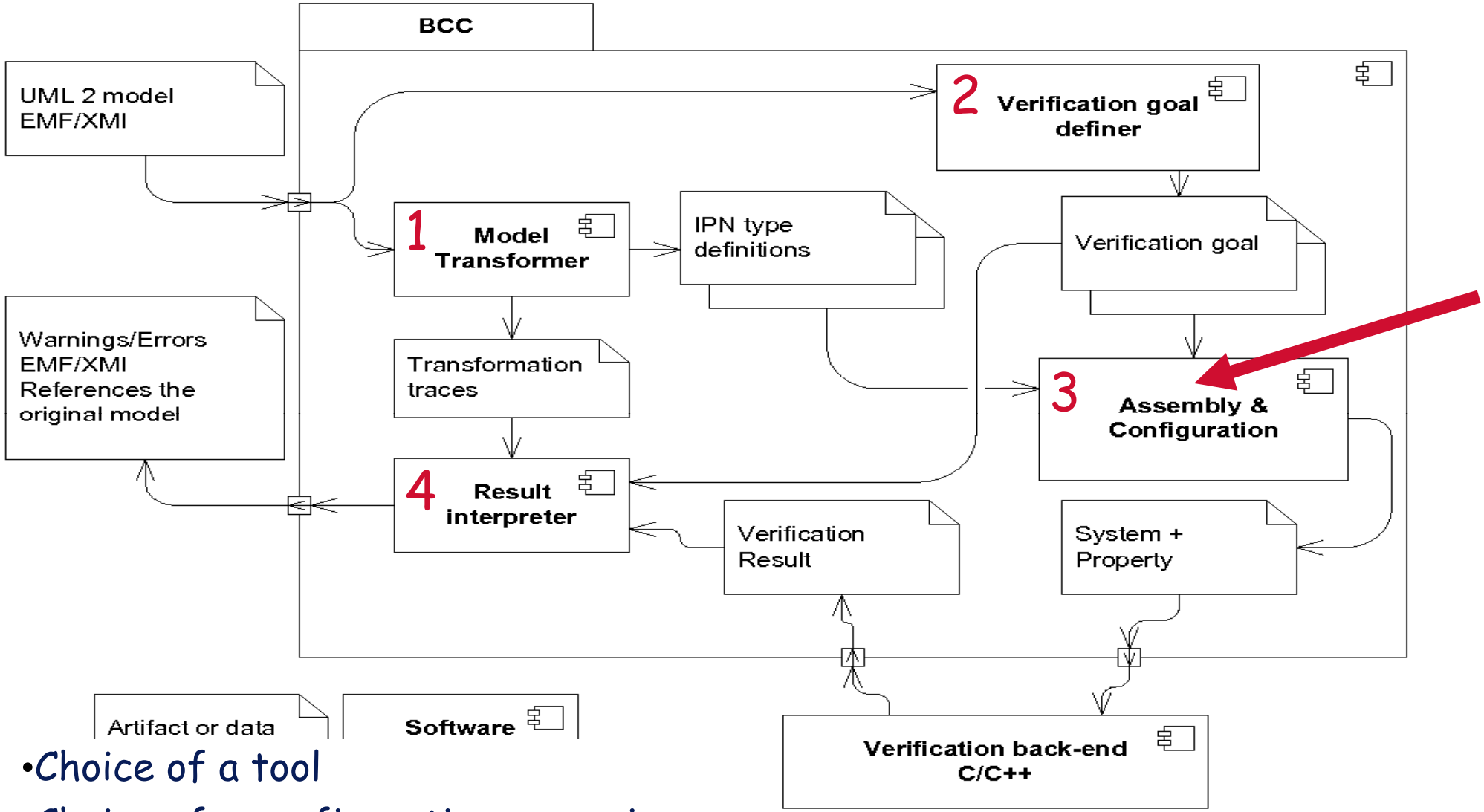
Step 2 : verification goals



Classification of properties

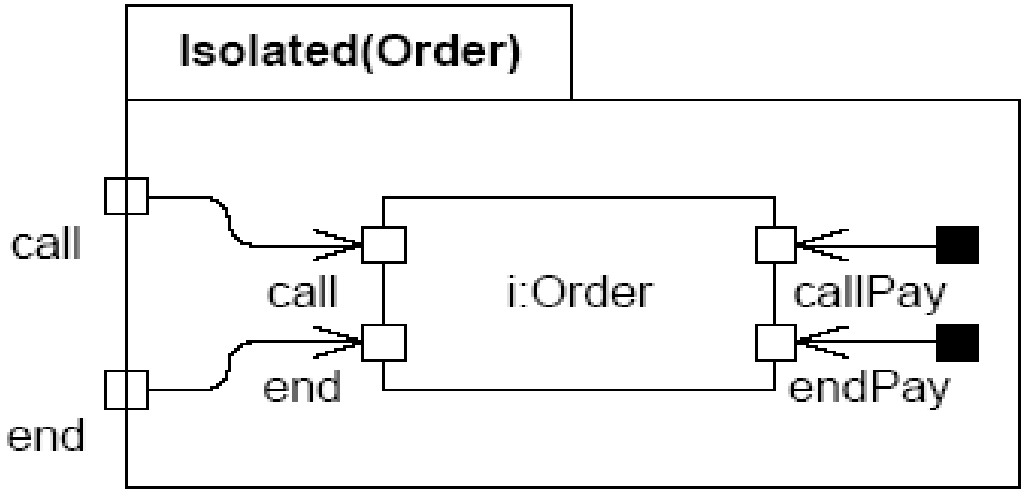
- Model independent: (No user input !!)
  - Deadlocks, Livelocks
  - Unbounded behavior (bad specification)
  - Unreachable behavior (dead code)
  - UML specific, model dependent: e.g. Reachability of final state from initial configuration
- Model instance dependent:
  - Reachability of a state: invariants, (un)desired states
    - *Would require a mapping from OCL to IPN*
  - Temporal logic (CTL, LTL…)
    - *Use stereotyped sequence diagrams ?*

Step 3 : Assembly and Verification Tool configuration

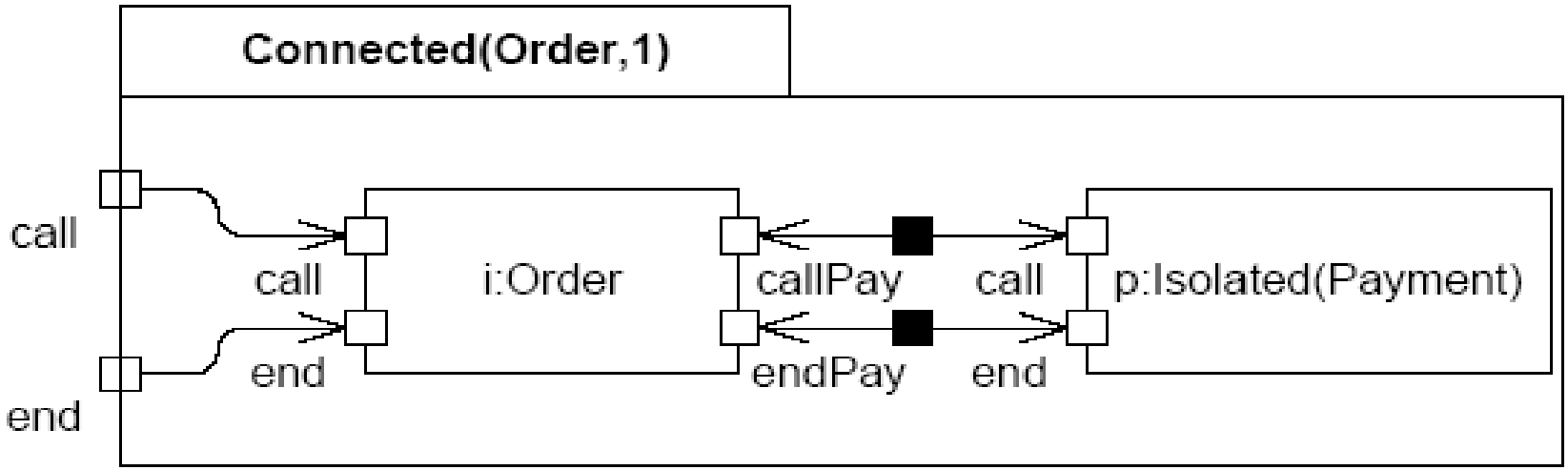


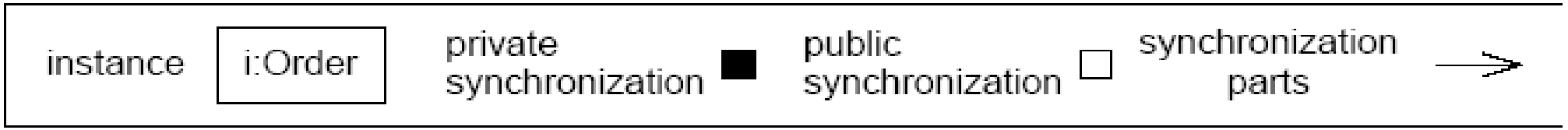
- Choice of a tool
- Choice of a configuration scenario

Incremental verification

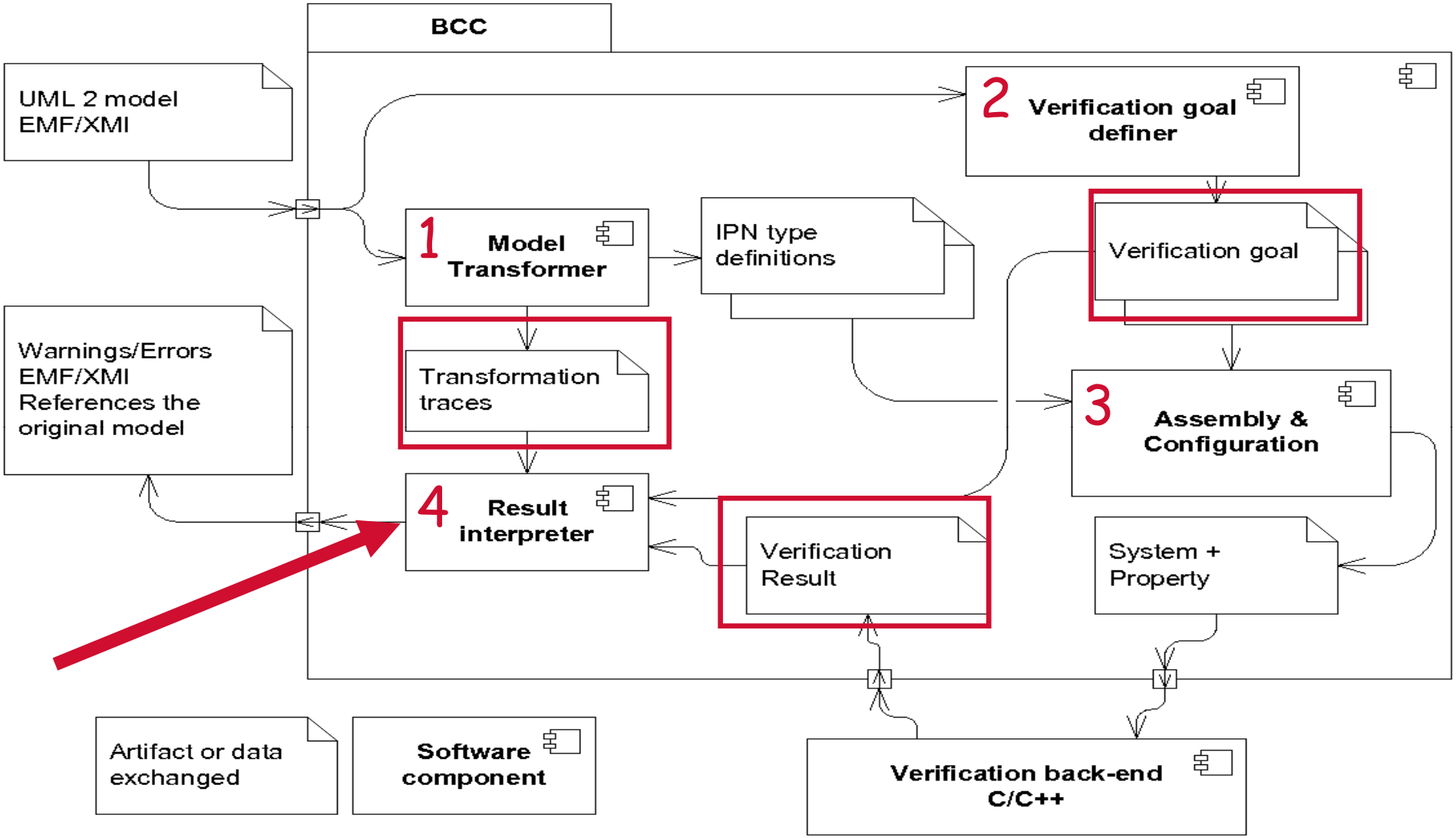


- Limit complexity of simple checks
- Optimize w.r.t. property.
- Reuse IPN type declarations





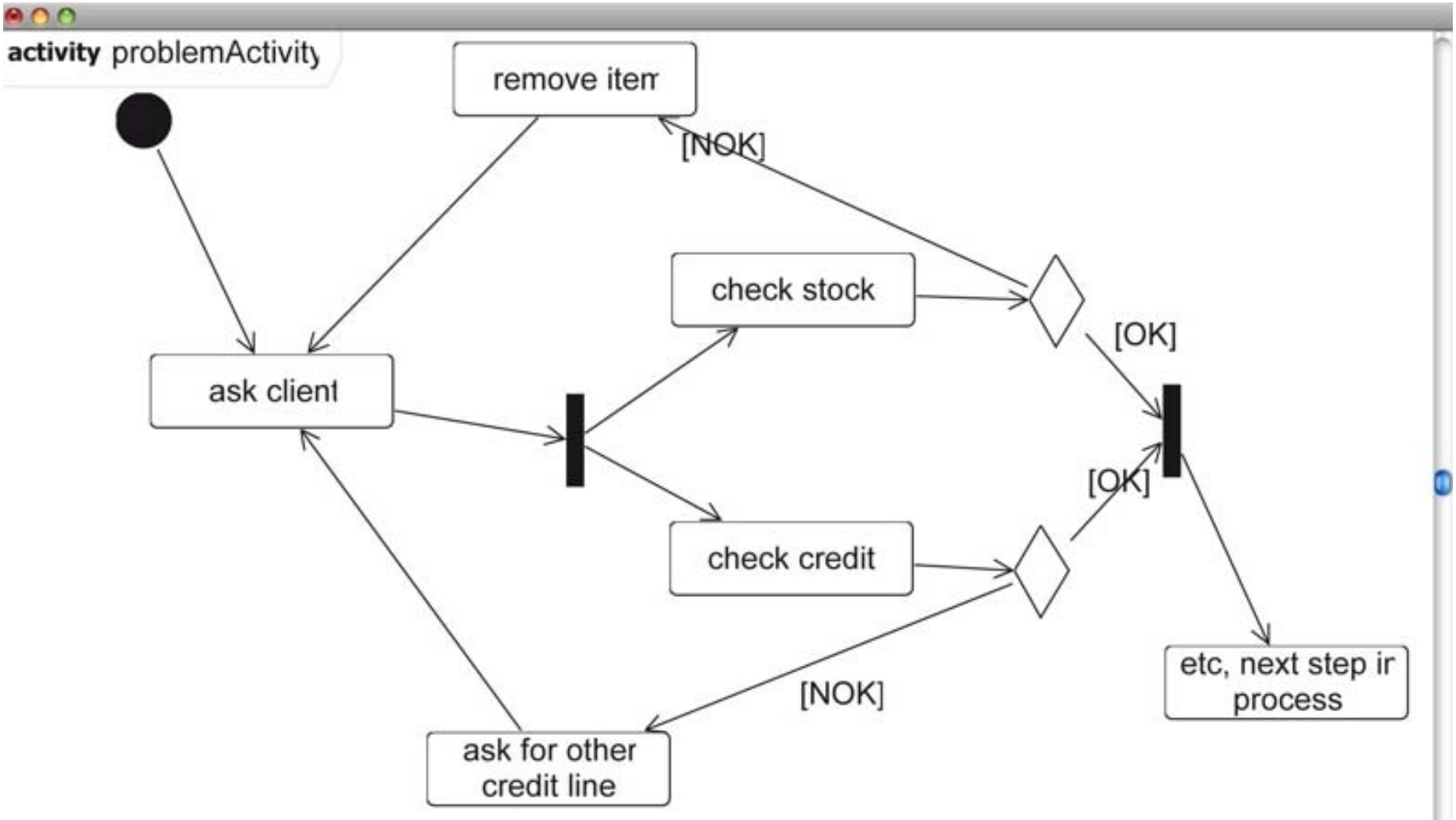
Step 4 : Result Interpretation
Keep it simple !



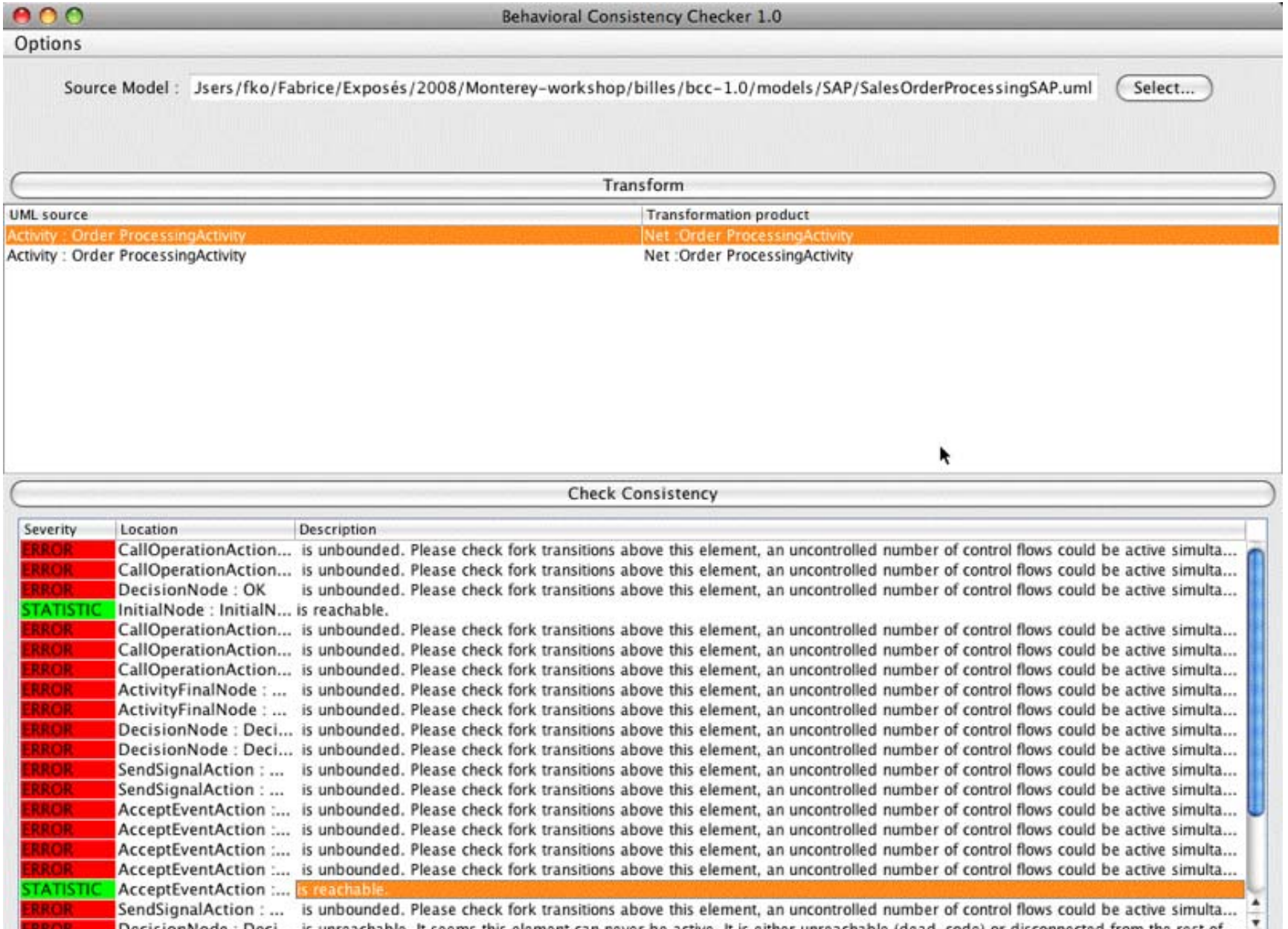
The SAP specification

Activity diagram from SAP

- Any of the two checks can fail and trigger the fork again…

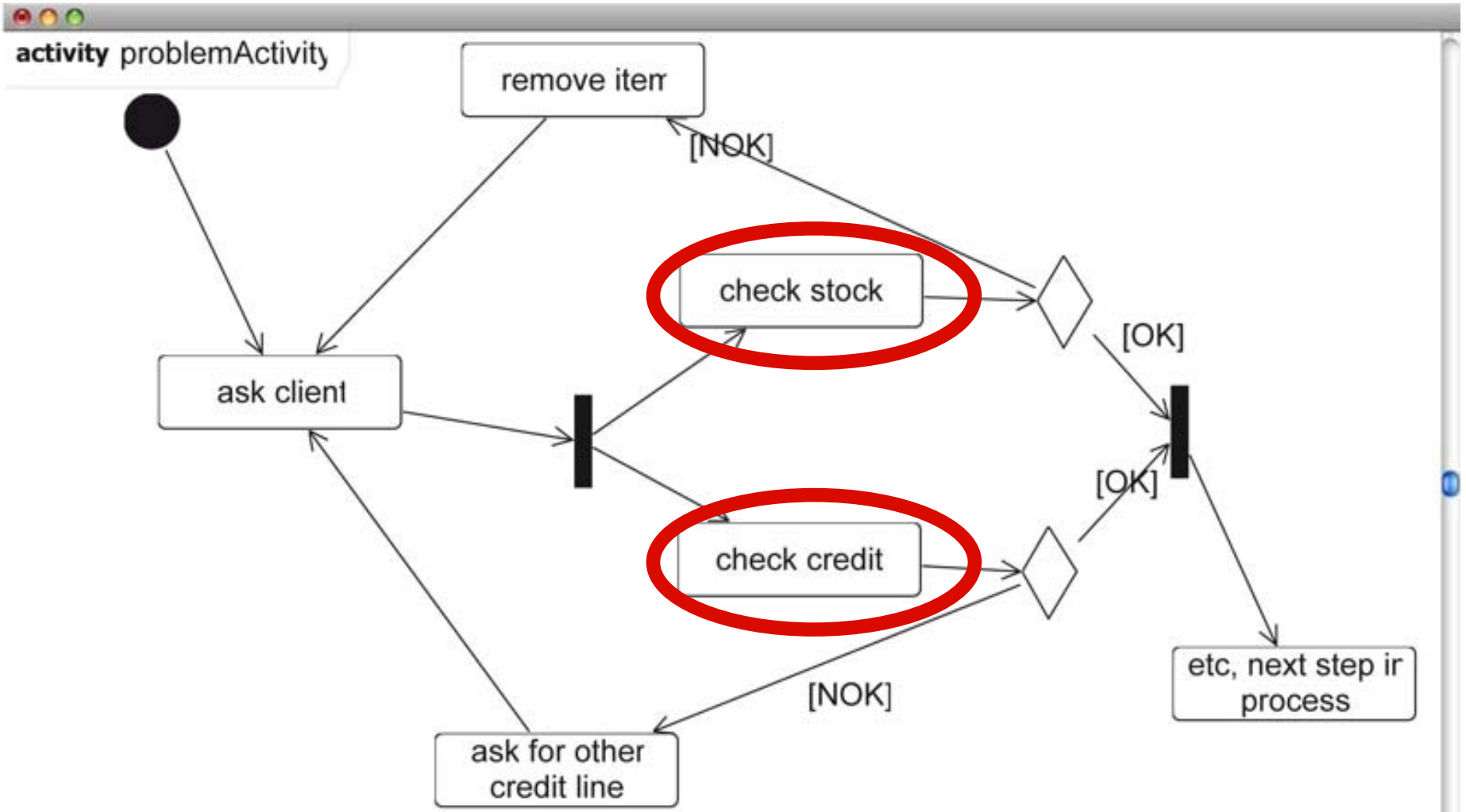


Working on the SAP model with BCC



Meanings of red errors

- From simulation, more CPU is needed due to important activity
- From verification: the spec's structure is unbounded by construction



Conclusions

- Still work in progress
  - Define additional consistency rules
  - Use profiled models for model specific properties
  - Eclipse problem view integration
- Experience learned:
  - Use of default properties already useful
  - Fully automated chain necessary
- Need for structure and hierarchy in target formalism
  - Applicability of compositional approach to other elementary bricks than Petri nets
- We also develop an efficient decision diagram based model-checking solution see ddd.lip6.fr
  - Hierarchy allows more efficient verification algorithms


Behavioral Consistency Checker - Home page - Windows Internet Explorer
http://move.lip6.fr/software/BCC/index.html
Behavioral Consistency Checker - Home p...
BCC
Home
Download
Documentation
About us
Contact
Releases
BCC 1.0 (current)
User guide 0.3 (current)
Behavioral Consistency Checker
BCC is a model checking based tool to check consistency of UML diagrams. It is developed within the ModelPlex project.
BCC is produced as LIP6 contribution within task 4.3, Model-Based Verification:
Software property description metamodel with domain specific property description dialects.
Techniques for hierarchical composition of verification models.
Verification engines that integrate multiple, concern-specific technologies with different applicability and scope, with heuristics for choosing the most appropriate technology depending on context.
You can download and run BCC on your project from the download page.
Move.lip6.fr/software/BCC/
démarrer