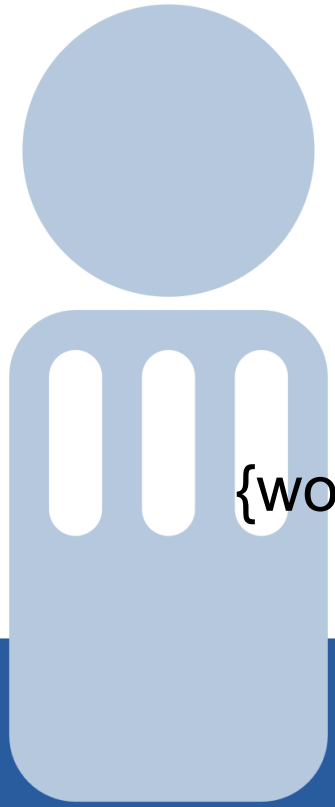
Process Model Editing Support Using Eclipse Modeling Project Tools

René Wörzberger
Thomas Heer

RWTH Aachen University
Department of Computer Science 3
(Software Engineering)
{woerzberger,heer}@i3.informatik.rwth-aachen.de



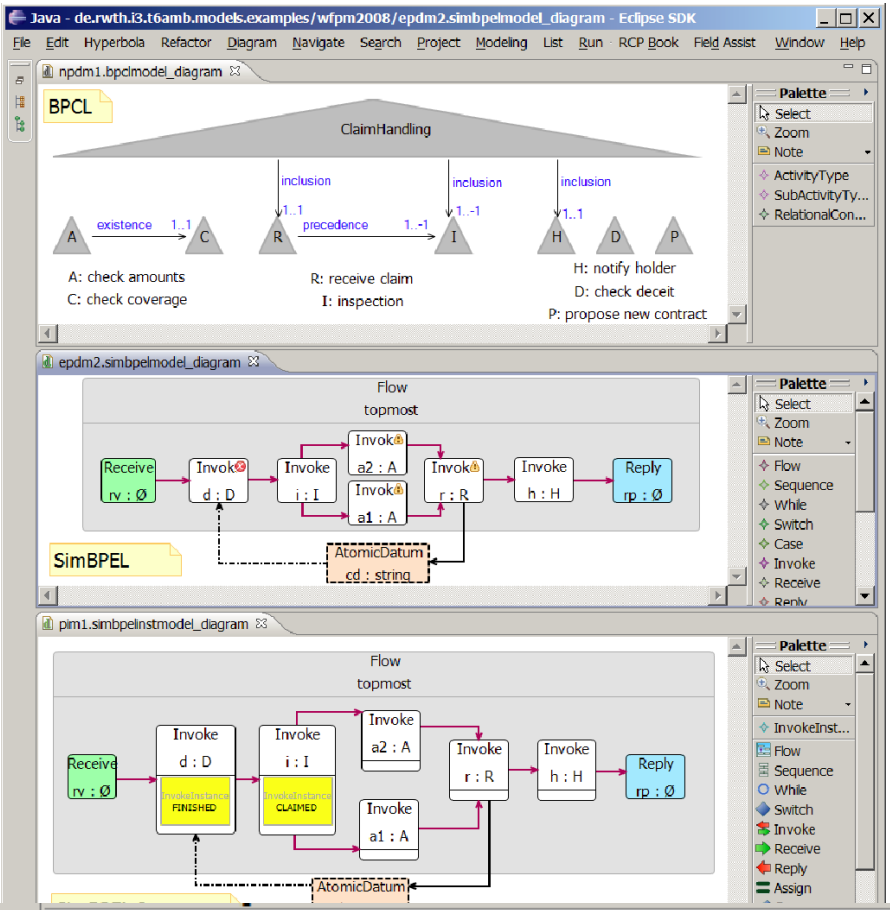
Outline

- Research **context** & **problem** statement
- **Interrelated** process models on different **layers** of abstraction
- **Constraints** on process models
- Application of **Eclipse GMF** and **OCL**

Research Context and Problem Statement

- **Research cooperation** with „AMB Generali Informatik Services GmbH“ (AMB-Informatik)
  - **IT service provider** for Generali Group (combine of insurance companies)
  - Management of **dynamic (insurance) processes** with extended IBM WebSphere tools
- **Problem statement**
  - **Related** models on different **abstraction layers**
  - Process **models** need to be **modified frequently**
  - Process models must or are supposed to adhere to **constraints** of different **types**
  - **Modeling tools** have to account for this

Process Model Editor

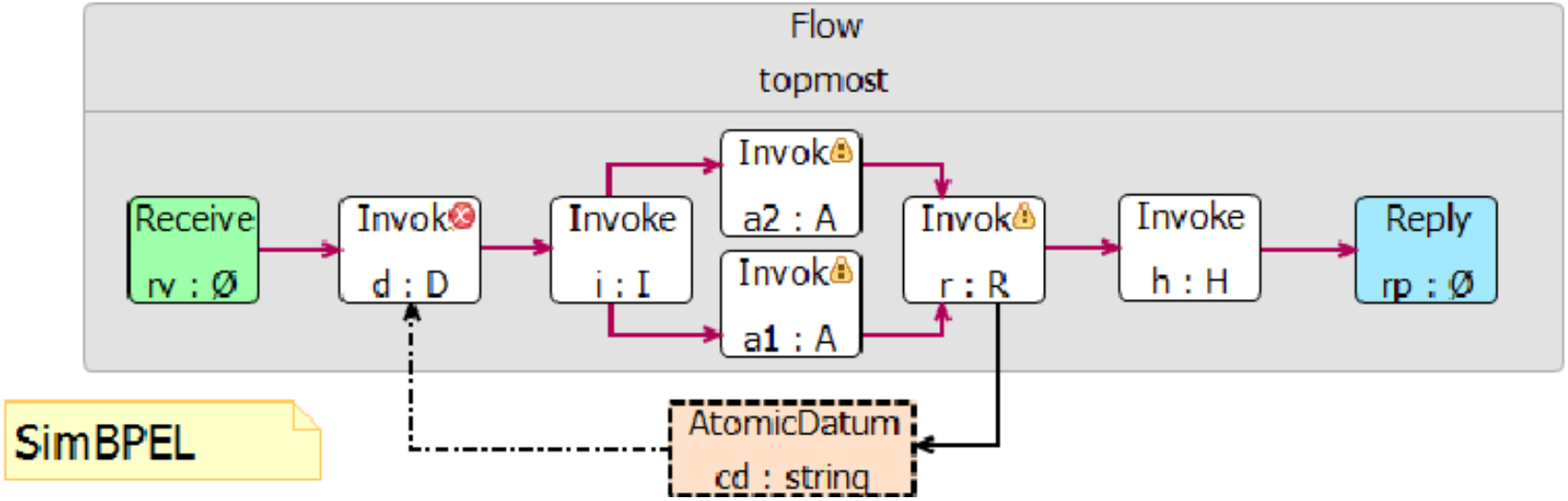


- Supports editing of process models on **three layers**
- Provides intra-model (correctness) and inter-model (compliance) **checks**
- Modeling **languages** are experimental, hence **prone to change**
- Rapidly implemented using **GMF** and **OCL**

Description	Resource	Location
Errors (1 item)		
Atomic Activity reads from uninitialized variable	epdm2.simbpelmodel_diagram	<Process>::topmost::d
Warnings (3 items)		
Activity violates existence compliance rules	epdm2.simbpelmodel_diagram	<Process>::topmost::a1
Activity violates existence compliance rules	epdm2.simbpelmodel_diagram	<Process>::topmost::a2
Activity violates precedence compliance rules	epdm2.simbpelmodel_diagram	<Process>::topmost::r

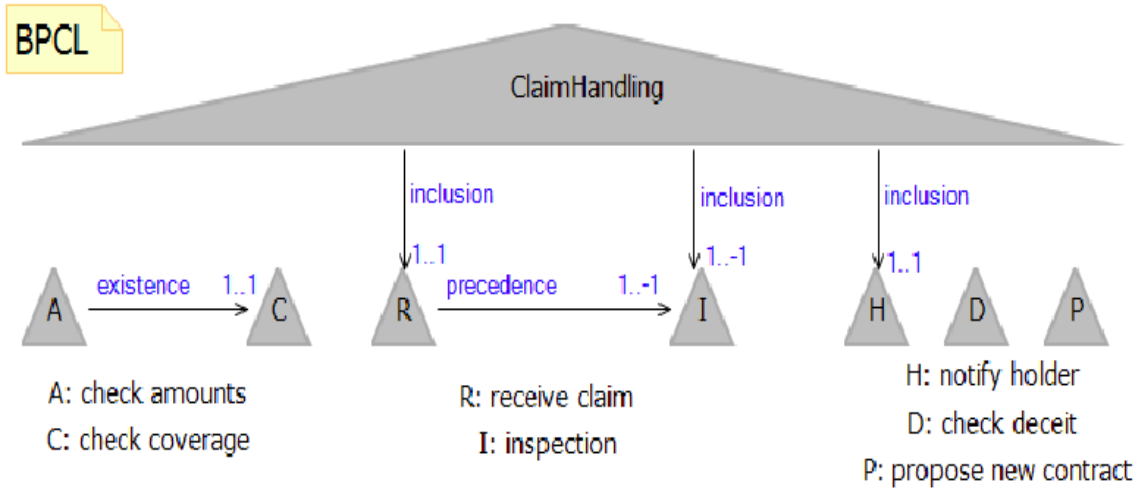
Ctrl Contrib (Bottom)

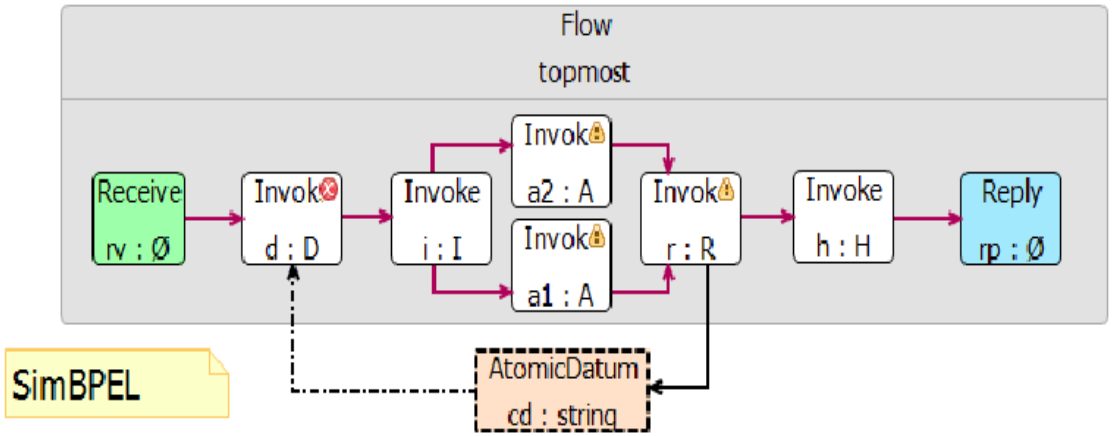
Correctness



- Adherence to **common constraints**, e.g.,
  - Links have to be acyclic (in WS-BPEL)
  - Variables have to be initialized before being read
- Violation normally leads to **technical problems**, e.g.,
  - abnormal termination, lost updates, deadlocks...
- Example
  - **d** (check deceit) *reads* from datum **cd** (claim description) and **r** (receive claim) *writes* to variable **cd**
  - but: **d** precedes **r** in the definition of the control flow
- Some problems are well known from Compiler Theory

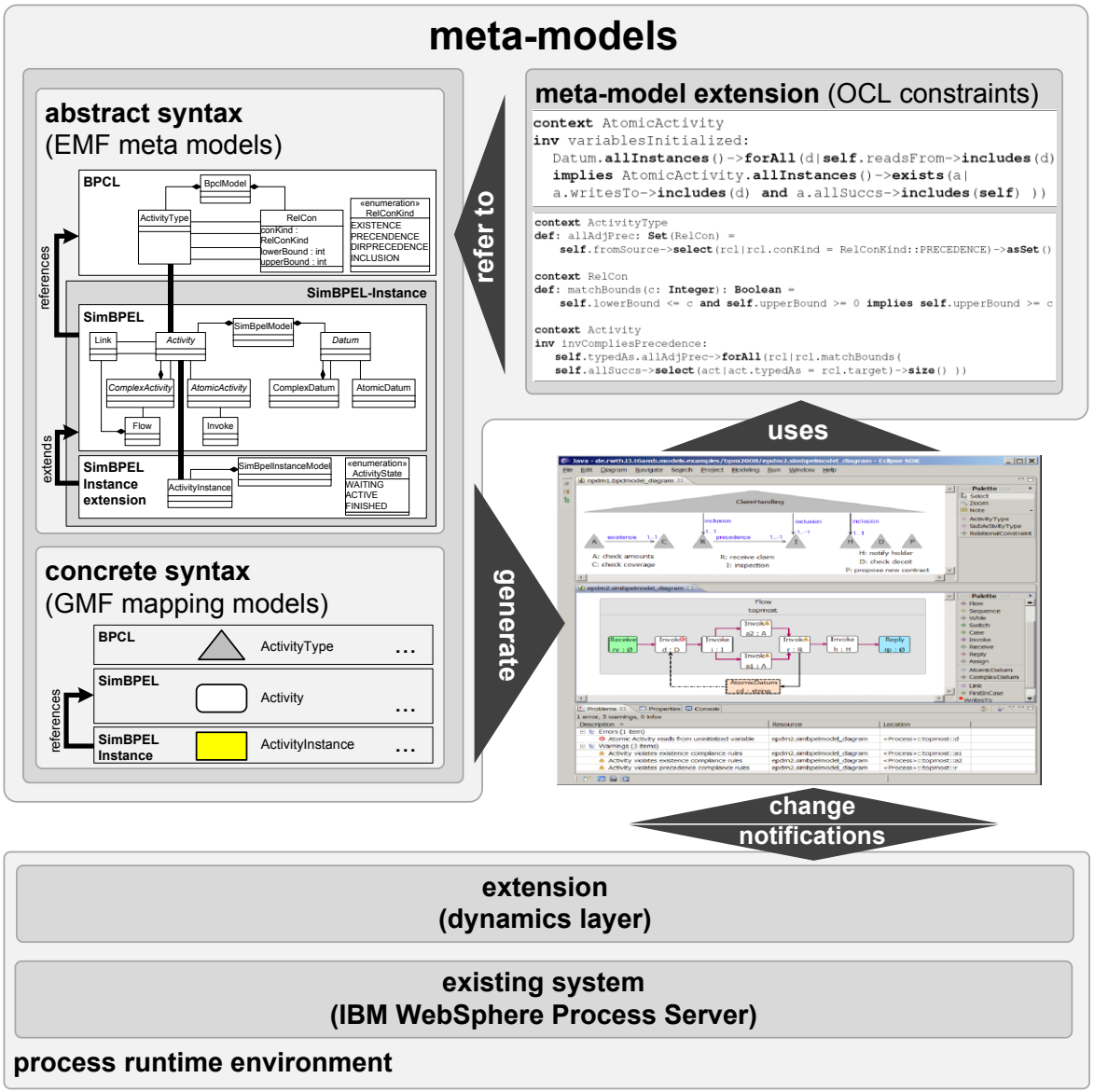
Compliance





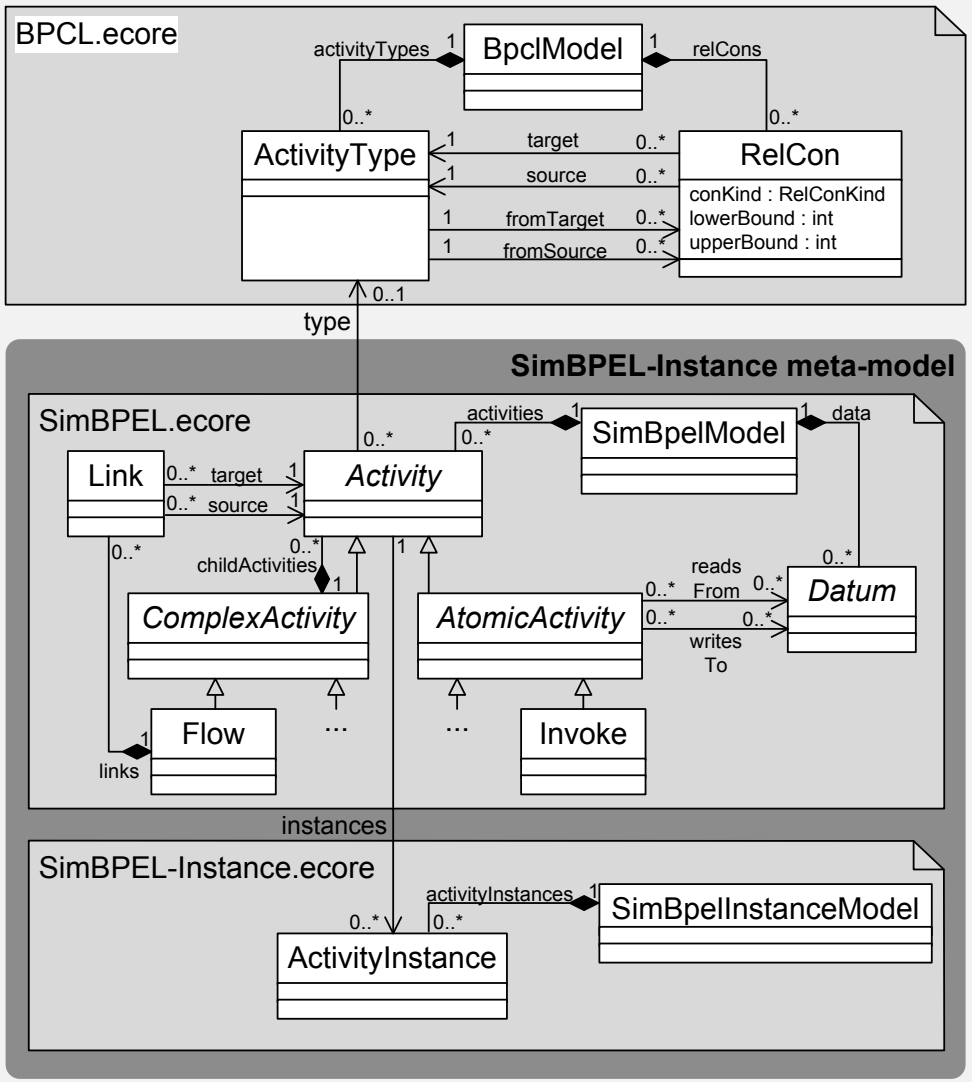
- Adherence to **explicitly modeled constraints**
- Requires **additional models** for professional constraints
- Violation may lead to **professional problems**
- **Example**: amounts are checked twice but coverage is not checked

Coarse Architecture

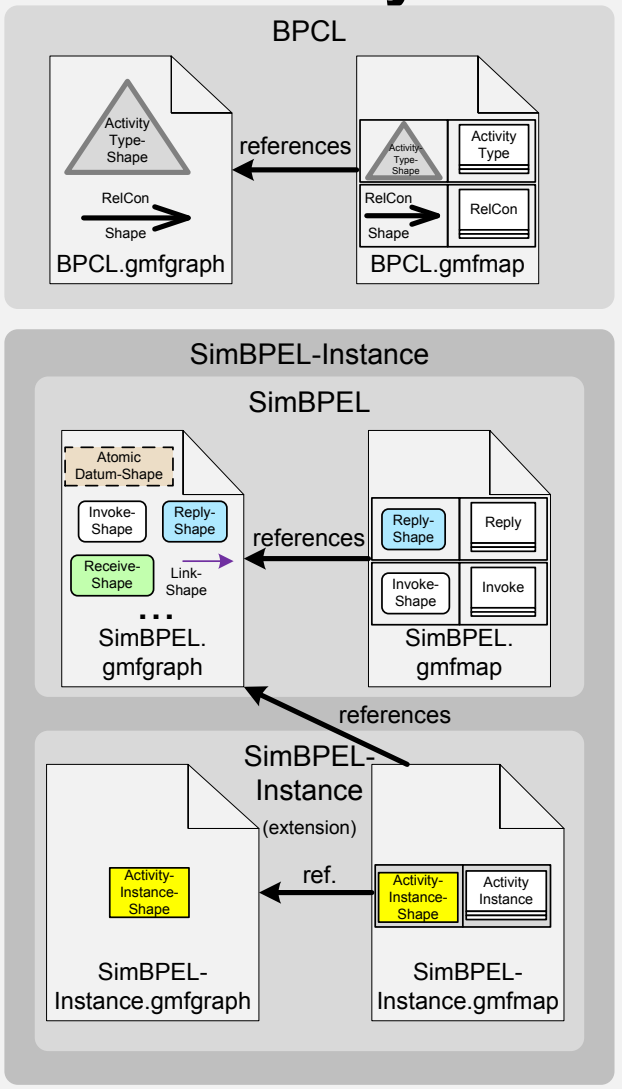


Integrated Meta-Model

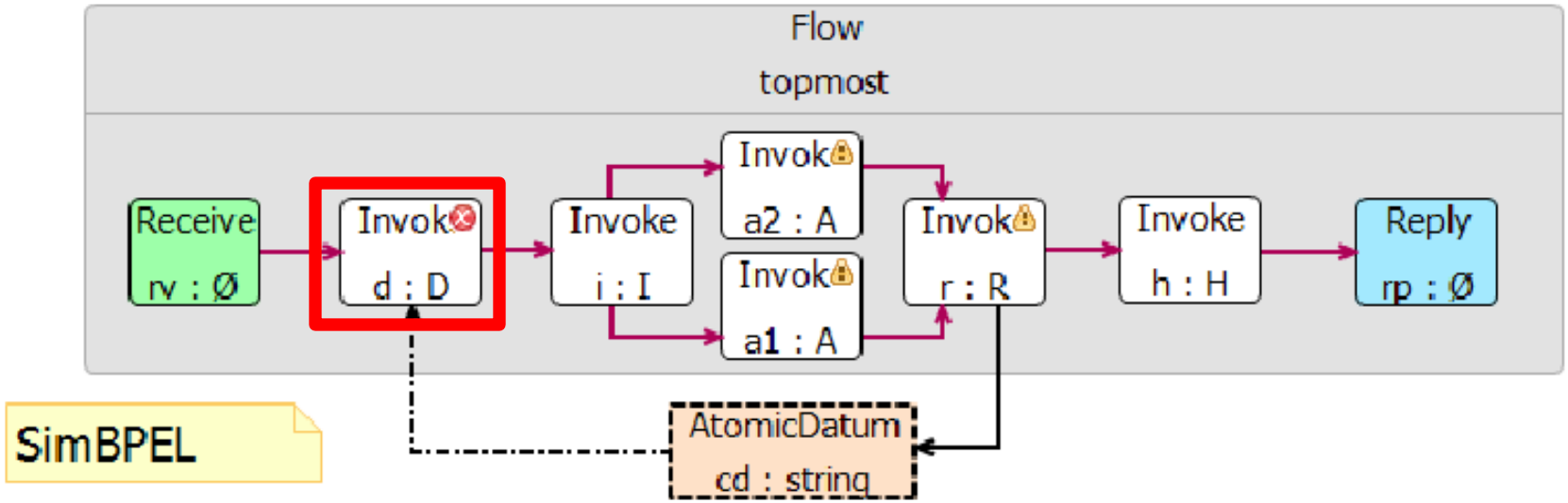
abstract syntax



concrete syntax

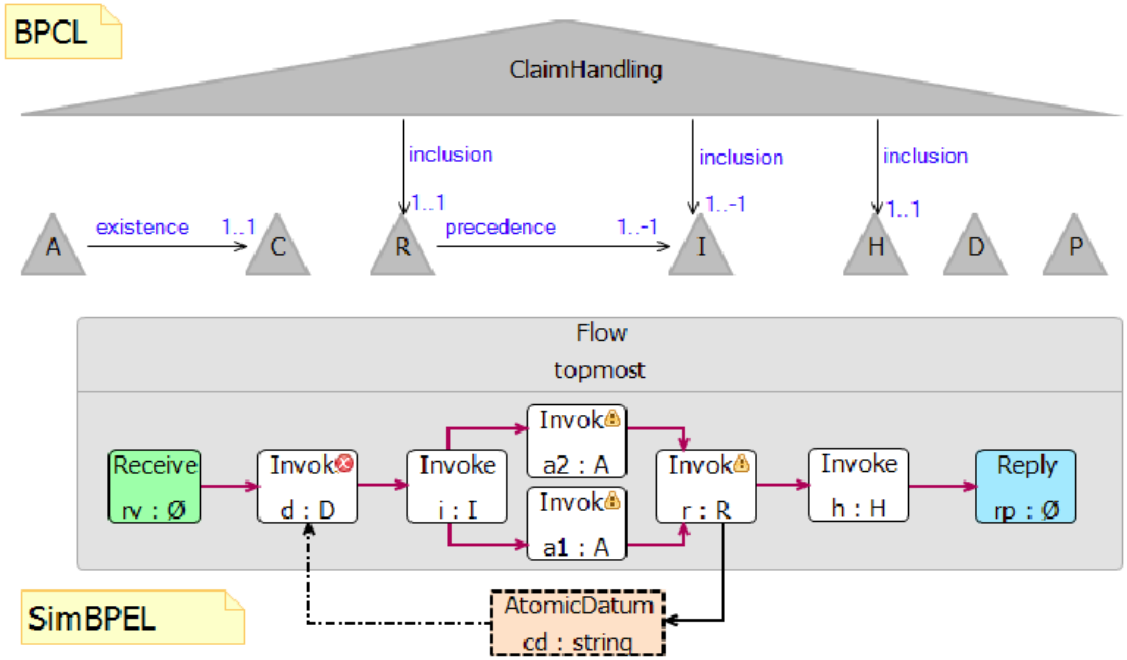


Using OCL for Correctness Checks




```
context AtomicActivity
inv variablesInitialized:
   Datum.allInstances()->forAll(d|self.readsFrom->includes(d)
   implies AtomicActivity.allInstances()->exists(a|
   a.writesTo->includes(d) and a.allSuccs->includes(self) ))
```

Using OCL for Compliance Checks




```
context ActivityType
def: allAdjPrec: Set(RelCon) =
    self.fromSource->select(rcl|rcl.conKind = RelConKind::PRECEDENCE)->asSet()

context RelCon
def: matchBounds(c: Integer): Boolean =
    self.lowerBound <= c and self.upperBound >= 0 implies self.upperBound >= c

context Activity
inv invCompliesPrecedence:
   self.typedAs.allAdjPrec->forAll(rcl|rcl.matchBounds(
   self.allSuccs->select(act|act.typedAs = rcl.target)->size() ))
```

Conclusion

- Different process model kinds on **different abstraction layers**
- Syntax of models **experimental** → frequent changes
- Models are **related** to each other
  - Using the very **same mechanism** for correctness and compliance checks
    - **integrated** meta-model
    - **textual OCL-constraints**
  - **Reuse** of concrete syntax definition (SimBPEL vs. SimBPEL-Instance)
    - reduce effort
    - consistent visualization
- **GMF + OCL** match these requirements

Thank You!

Questions?