# ISWIM For Testing – A Model Driven Approach

Tony Clark

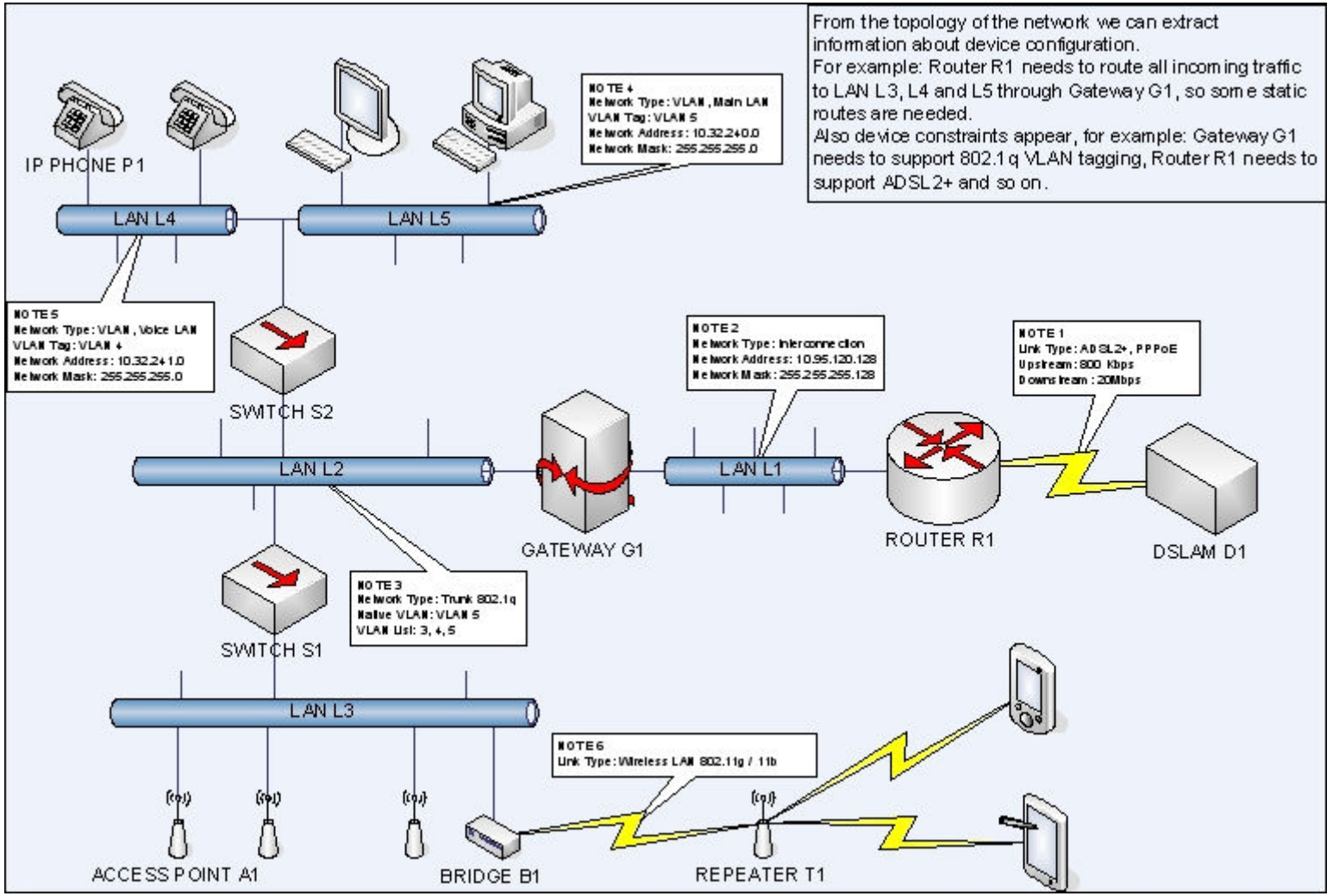
tony.clark@tvu.ac.uk
http://itcentre.tvu.ac.uk/~clark

# Overview

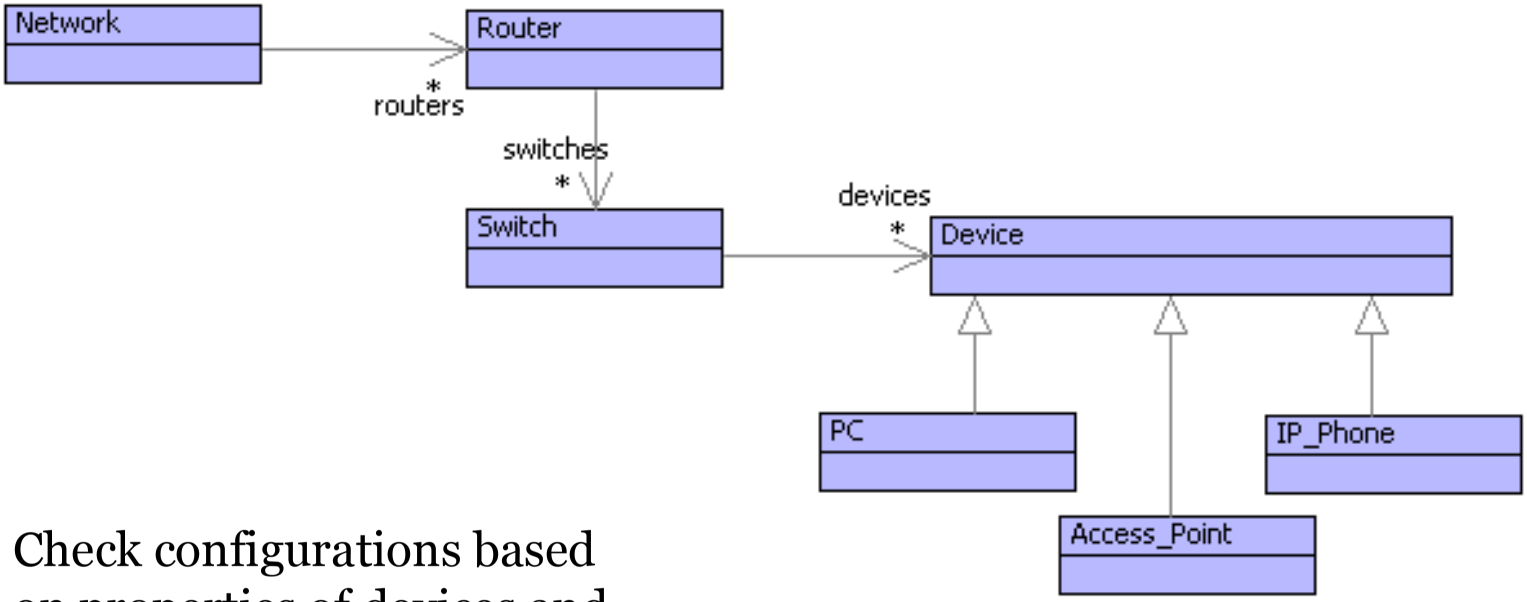
- A Requirement for Constraints:
  - Telecomms
  - Aerospace
  - Business Migration
  - Information Systems, SOA
- ISWIM Constraint Representation
- ISWIM for testing
- Conclusion

# Modelling with Constraints

# Telecomms

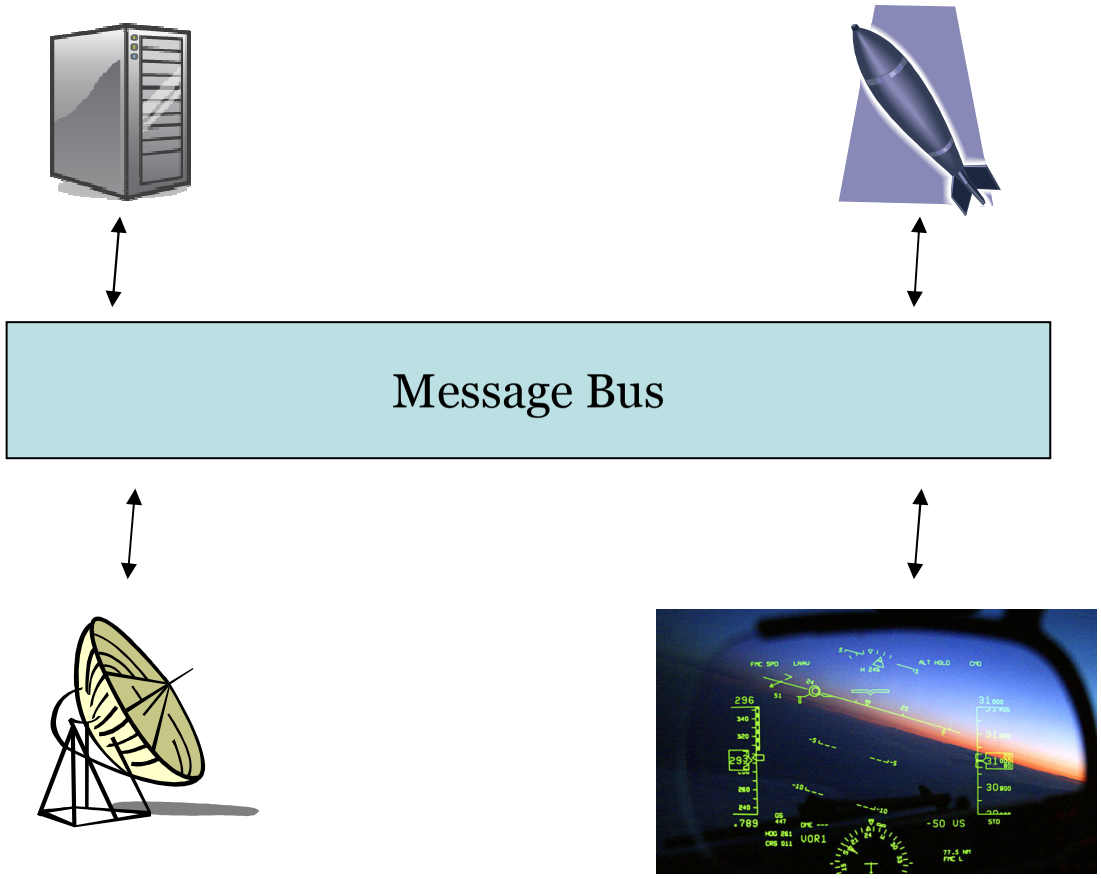

From the topology of the network we can extract information about device configuration.
For example: Router R1 needs to route all incoming traffic to LAN L3, L4 and L5 through Gateway G1, so some static routes are needed.
Also device constraints appear, for example: Gateway G1 needs to support 802.1q VLAN tagging, Router R1 needs to support ADSL2+ and so on.
IP PHONE P1
NOTE 4
Network Type: VLAN, Main LAN
VLAN Tag: VLAN 5
Network Address: 10.32.240.0
Network Mask: 255.255.255.0
LAN L4
LAN L5
NOTE 5
Network Type: VLAN, Voice LAN
VLAN Tag: VLAN 4
Network Address: 10.32.241.0
Network Mask: 255.255.255.0
SWITCH S2
NOTE 2
Network Type: Interconnection
Network Address: 10.95.120.128
Network Mask: 255.255.255.128
NOTE 1
Link Type: ADSL2+, PPPoE
Upstream: 800 Kbps
Downstream: 20Mbps
LAN L2
LAN L1
GATEWAY G1
ROUTER R1
DSLAM D1
NOTE 3
Network Type: Trunk 802.1q
Native VLAN: VLAN 5
VLAN List: 3, 4, 5
SWITCH S1
LAN L3
NOTE 6
Link Type: Wireless LAN 802.11g / 11b
ACCESS POINT A1
BRIDGE B1
REPEATER T1

# Network Models



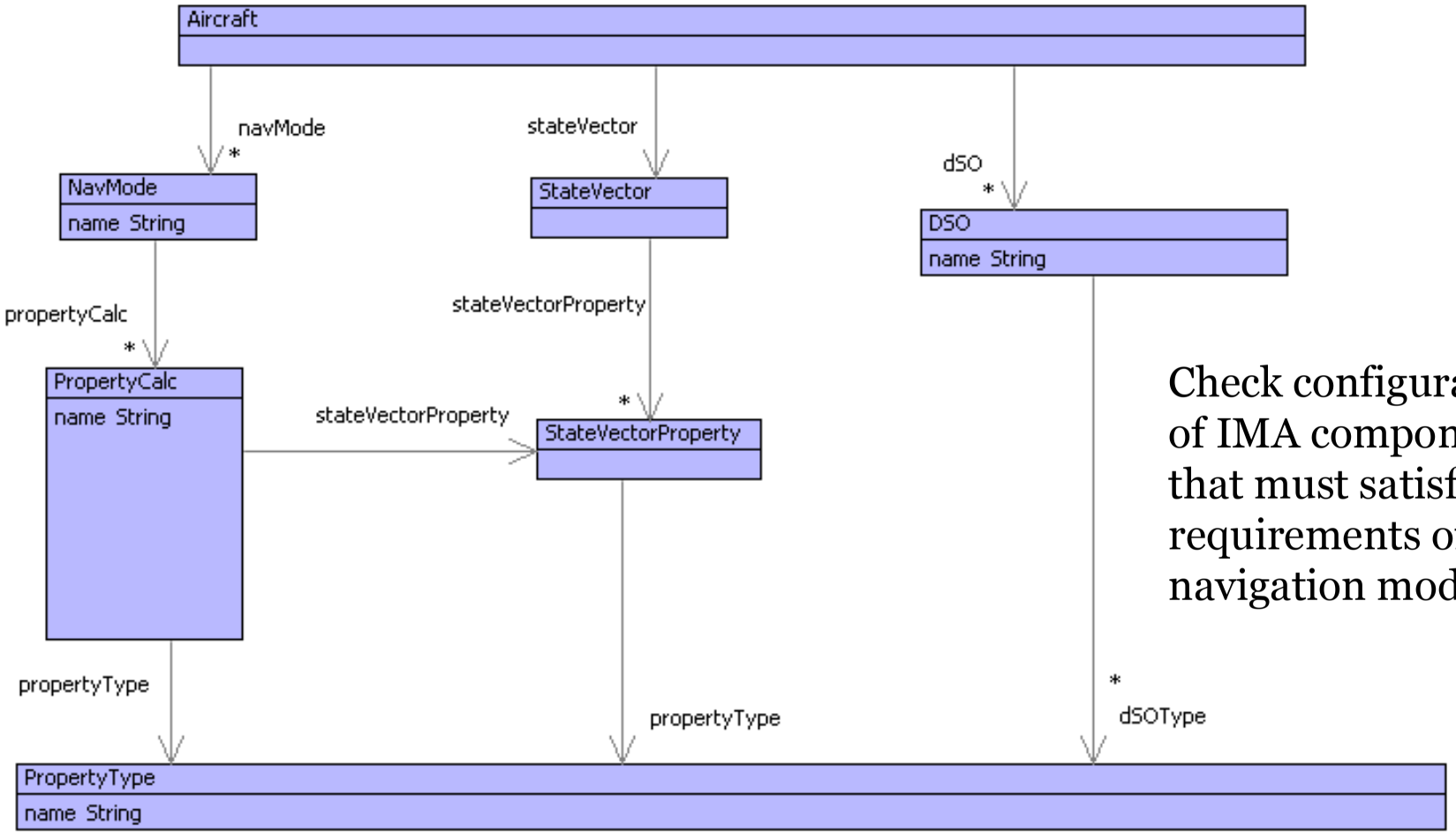

Check configurations based on properties of devices and network components.

# Aerospace



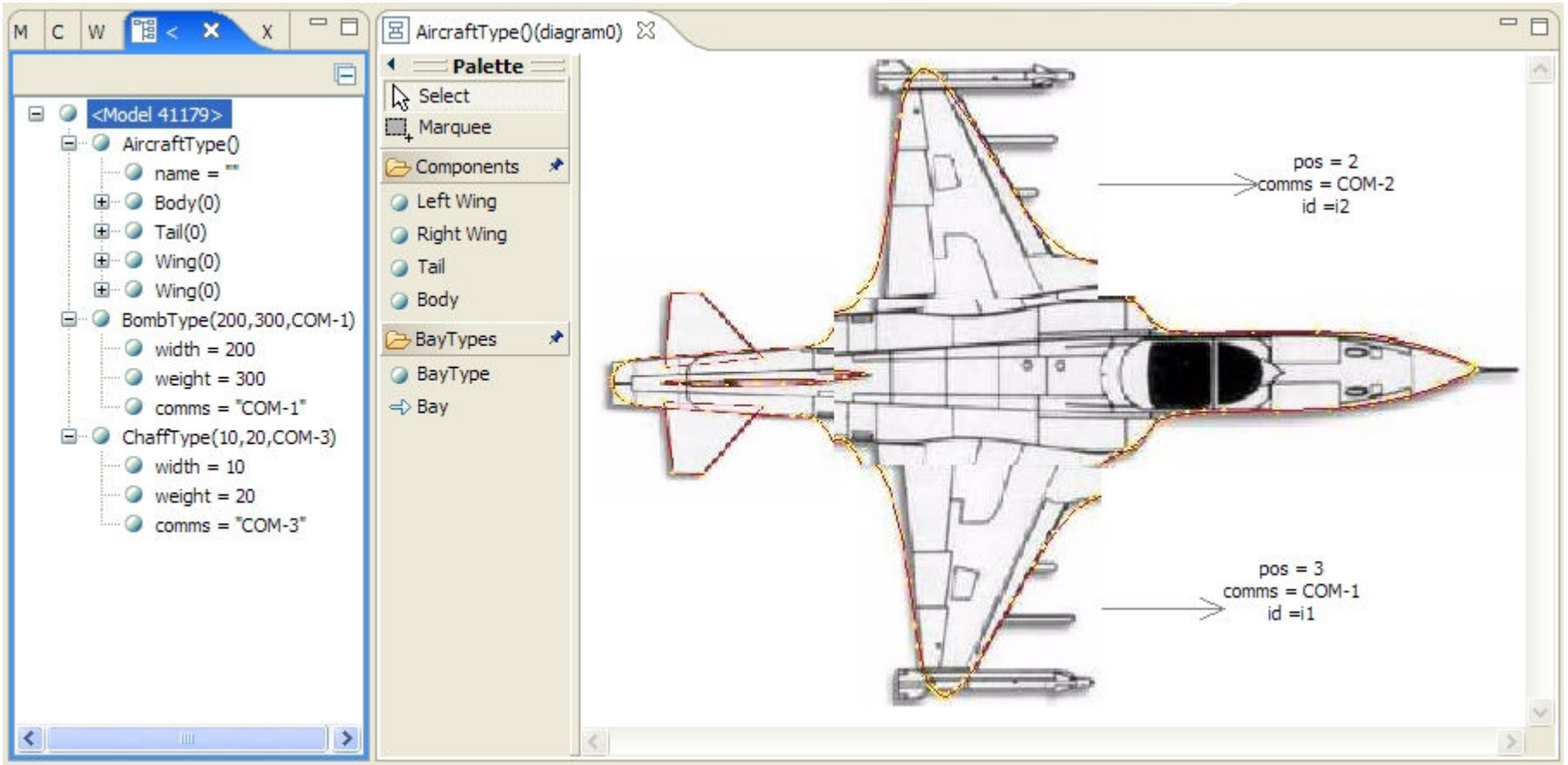

Integrated Modular Avionics (IMA)

# IMA Models

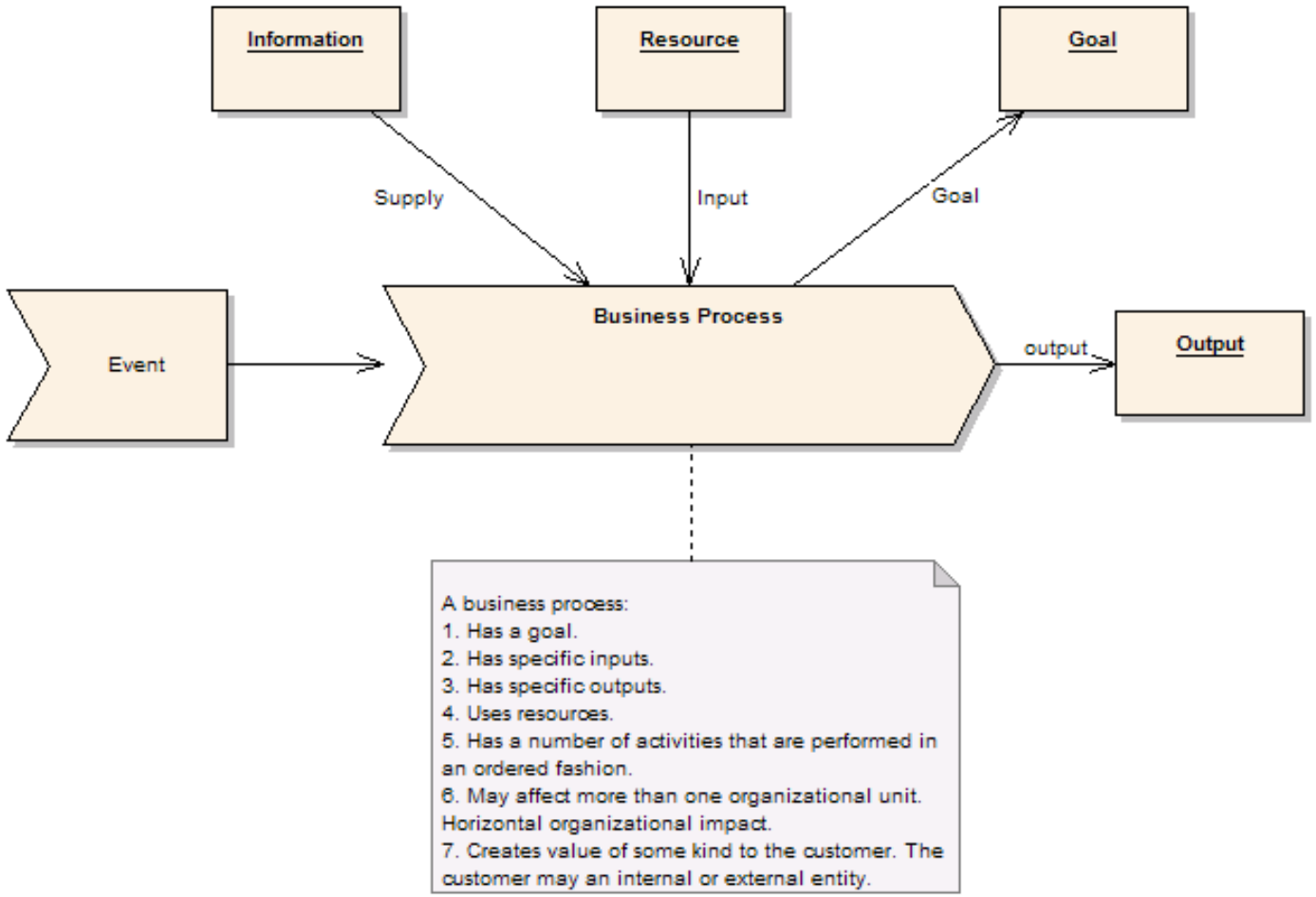



Check configurations of IMA components that must satisfy requirements of navigation modes.

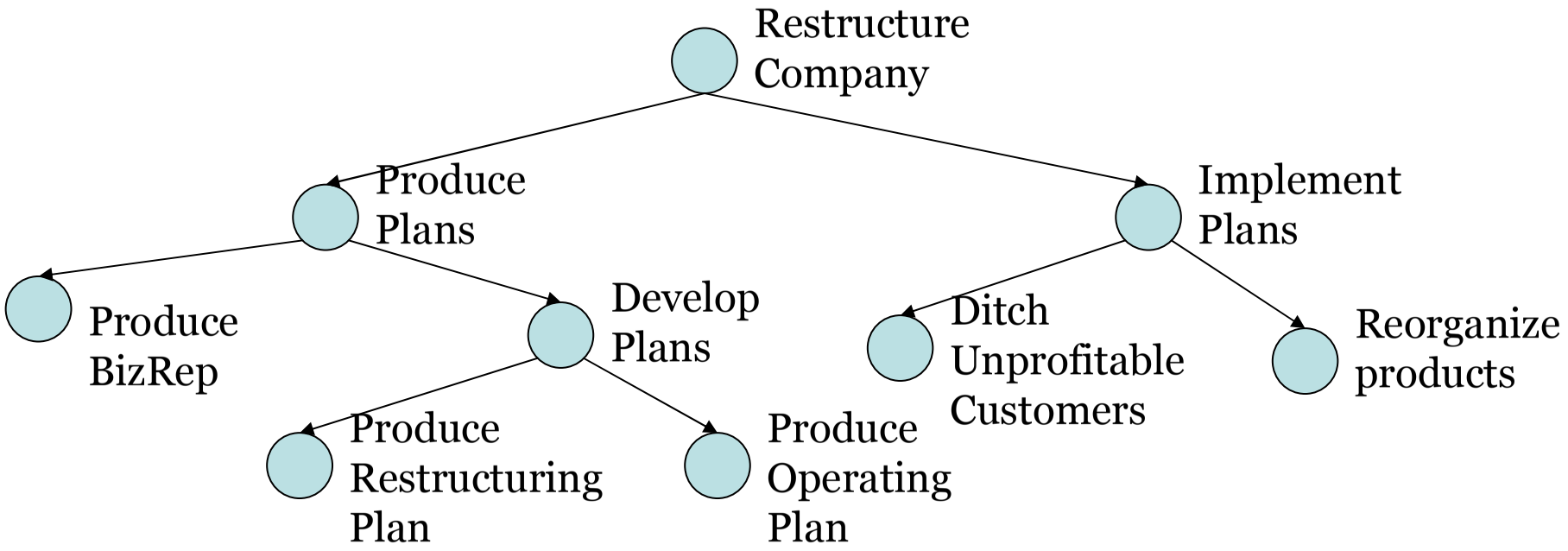
# Payload Configurations

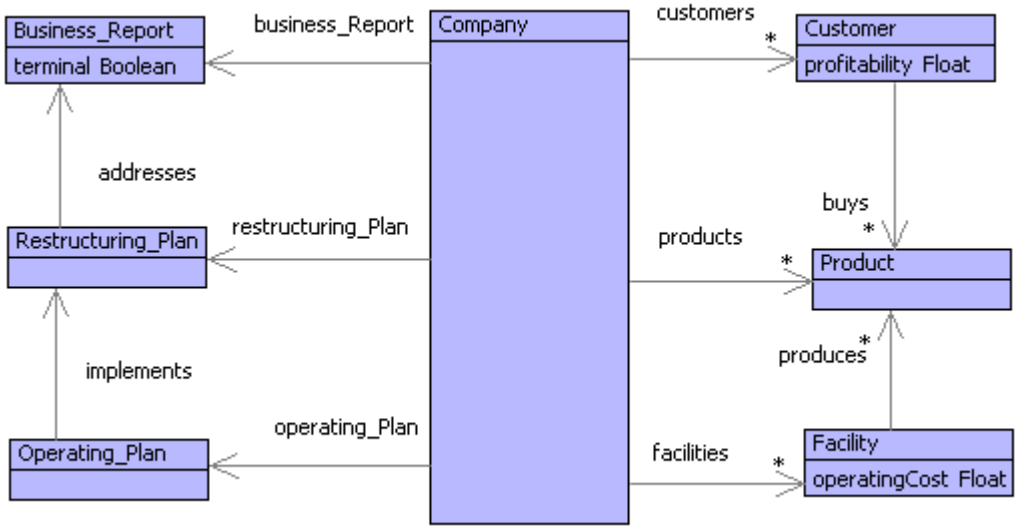


# Business Processes



# Restructuring A Company




Check sequences of company snapshots – do they satisfy the business goals?

# Web Services and SOA

FRANTZEN, TRETMANS, DE VRIES

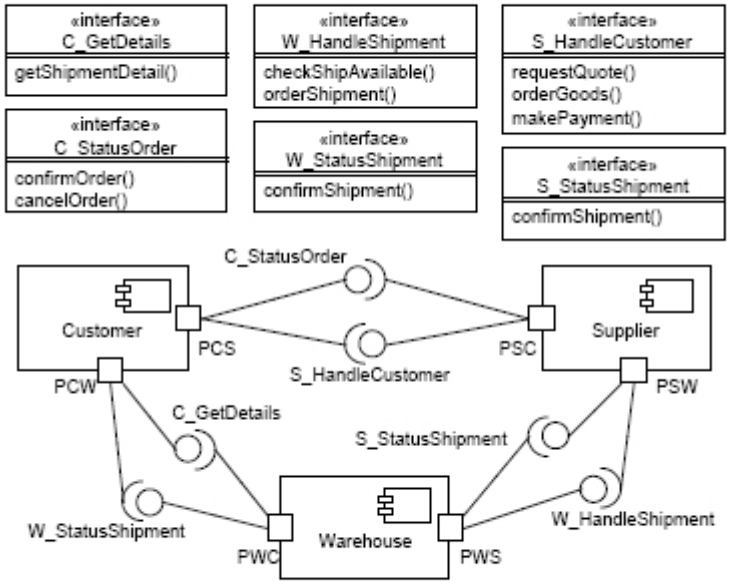



Fig. 1. The procurement protocol setup

FRANTZEN, TRETMANS, DE VRIES

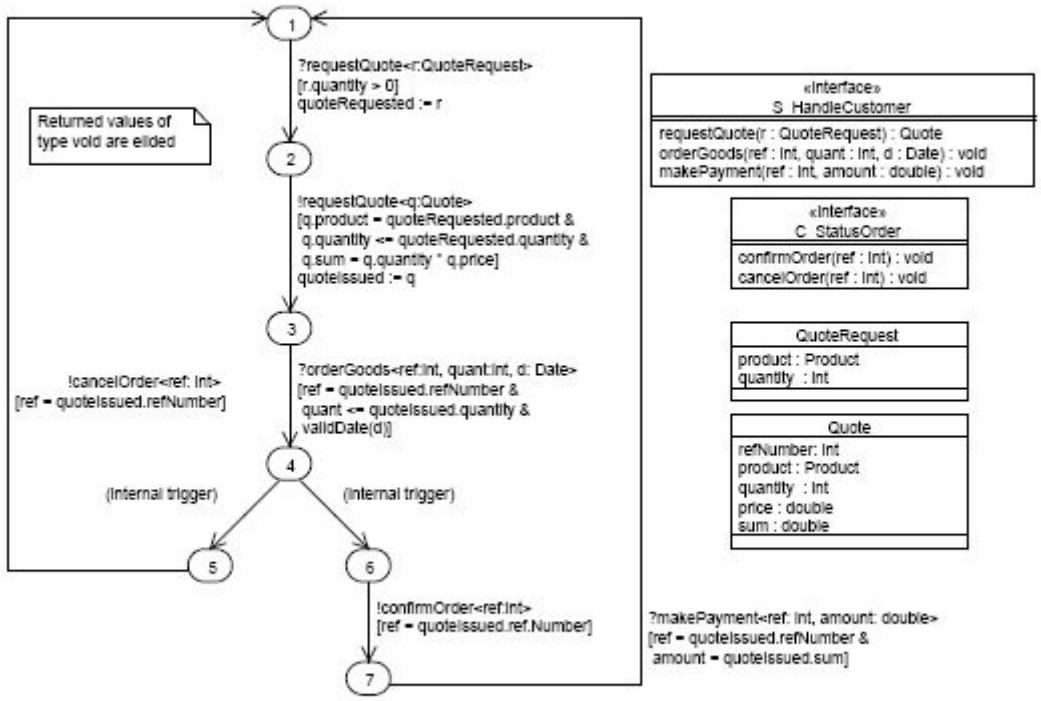
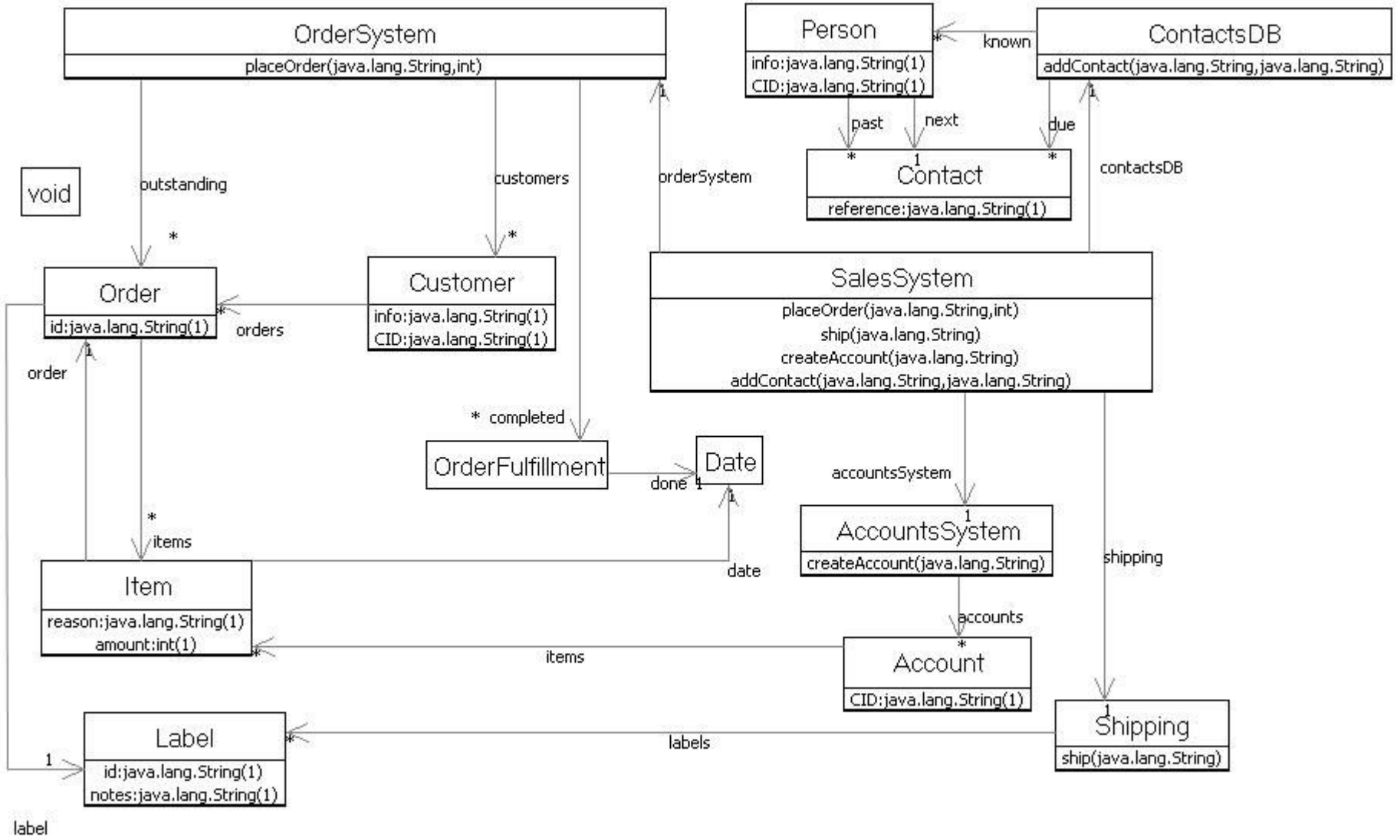



Fig. 4. A detailed STS specification for the PSC port

# Information Systems



# ISWIM Constraint Representation

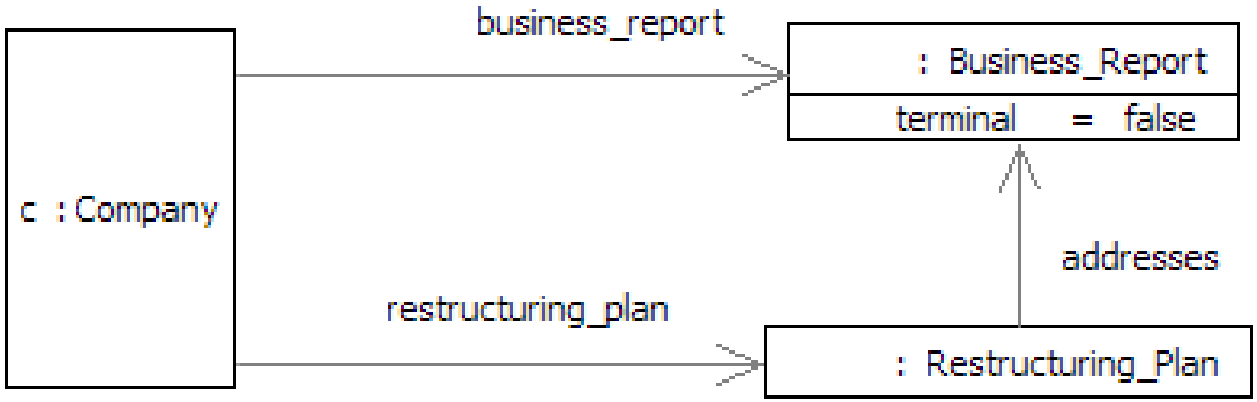
# Constraints

- What not to be: OCL, Java.
- Visual language (why?), with examples:
  - Simple state.
  - Variables.
  - And/Or/Not
  - Quantification
  - Predicates.

# Snapshots

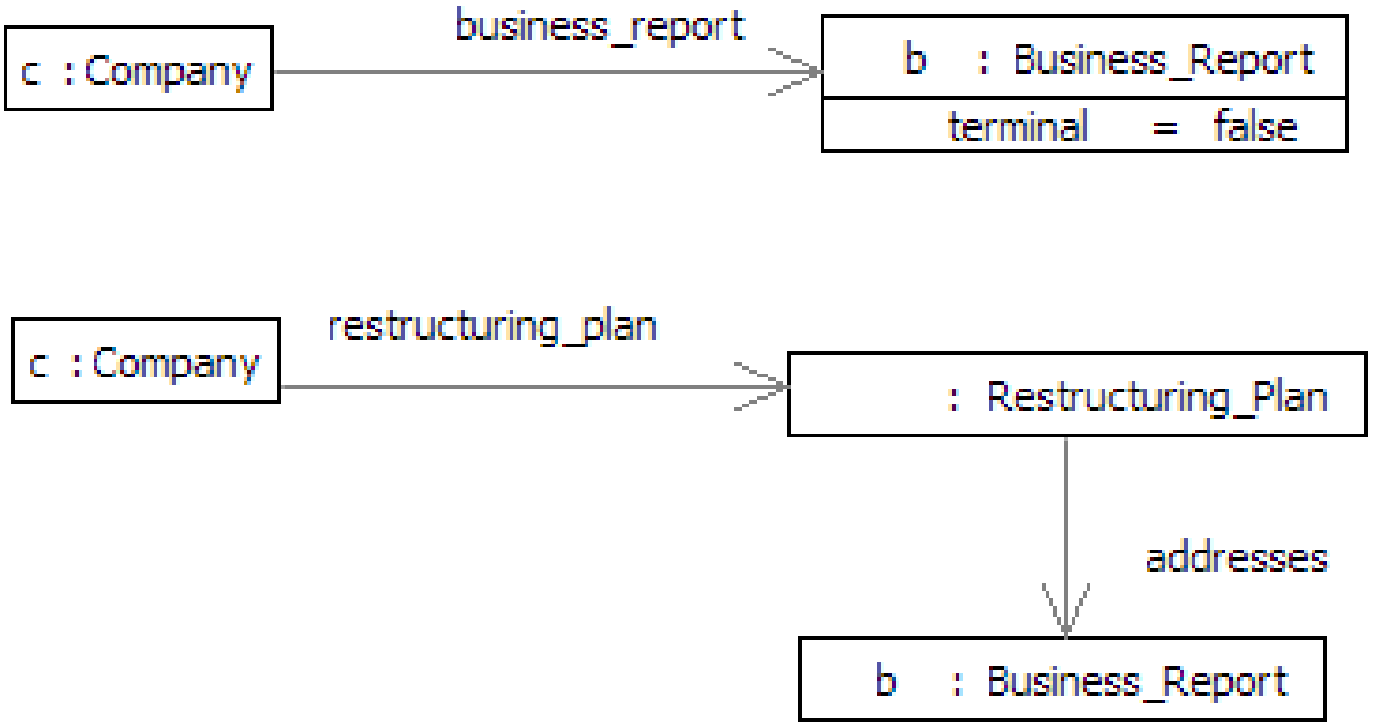
- Objects, slots and links.
- Context must be given:
  - Parameters
  - Root object
  - Self in state machine.
  - Observations in goals.
- Conjunction of contents.
- Negation.

# Snapshots as Constraints

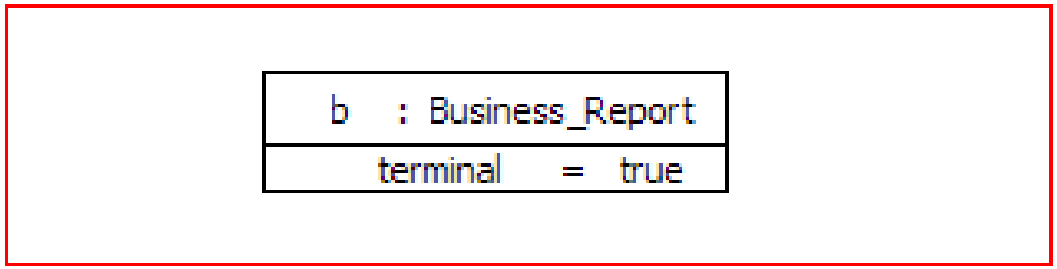



```
not c.business_report.terminal and
c.restructuring_plan.addresses = c.business_report
```

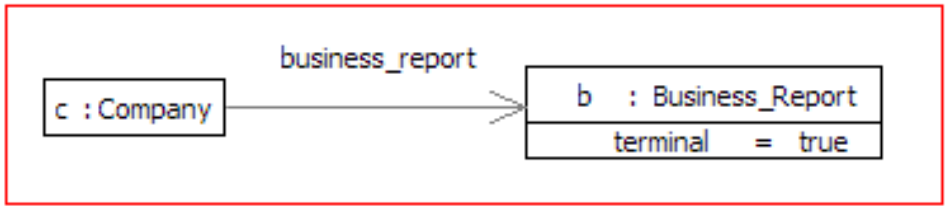
# Sharing via Identifiers
## (separation of concerns)



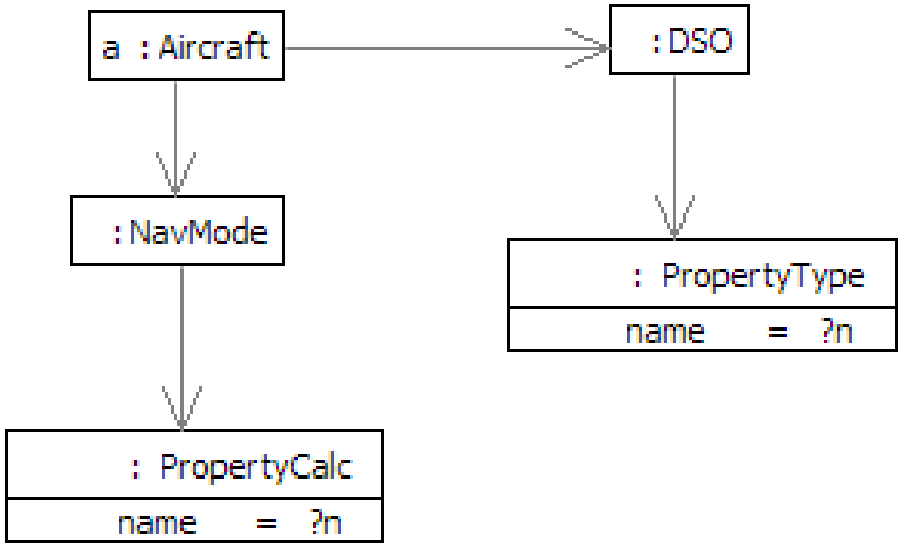
# Negation

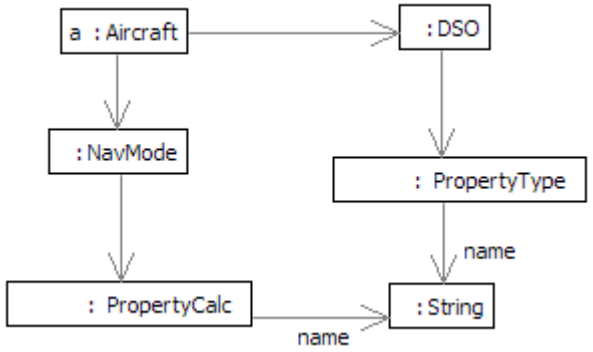

... but not ...



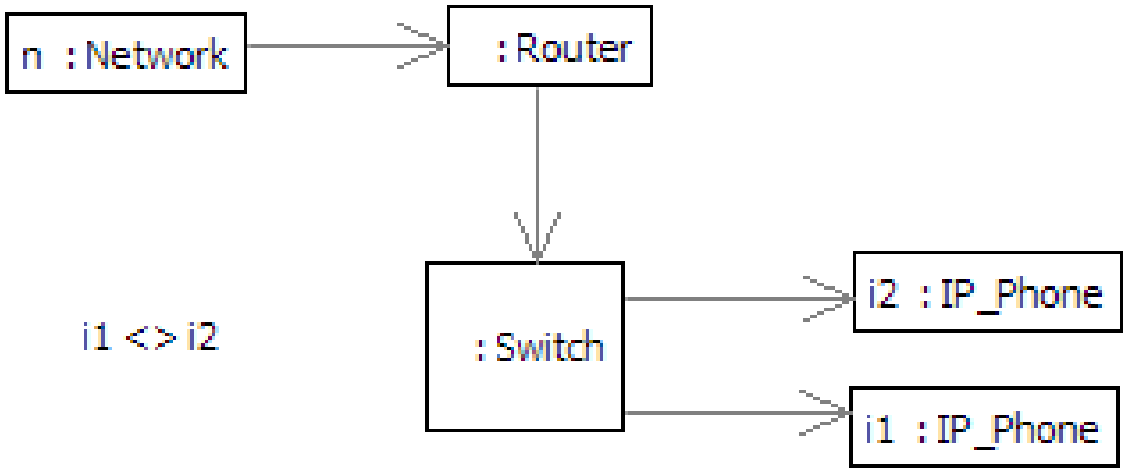
# Unknowns Tied Together




... same as ...

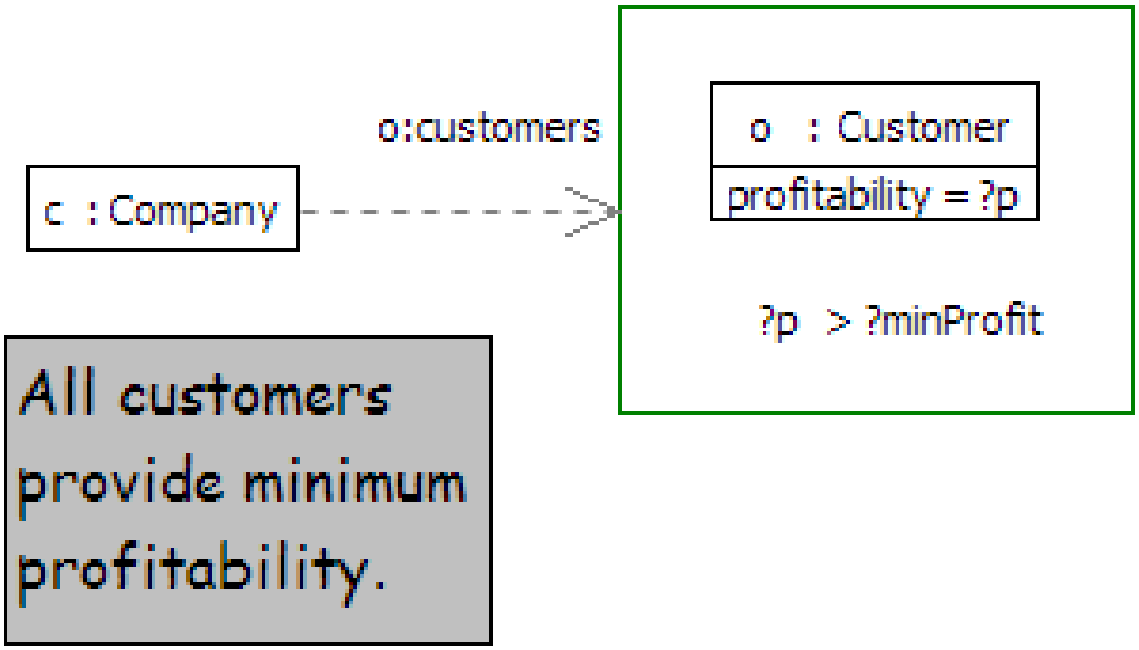


# Links Over Collections

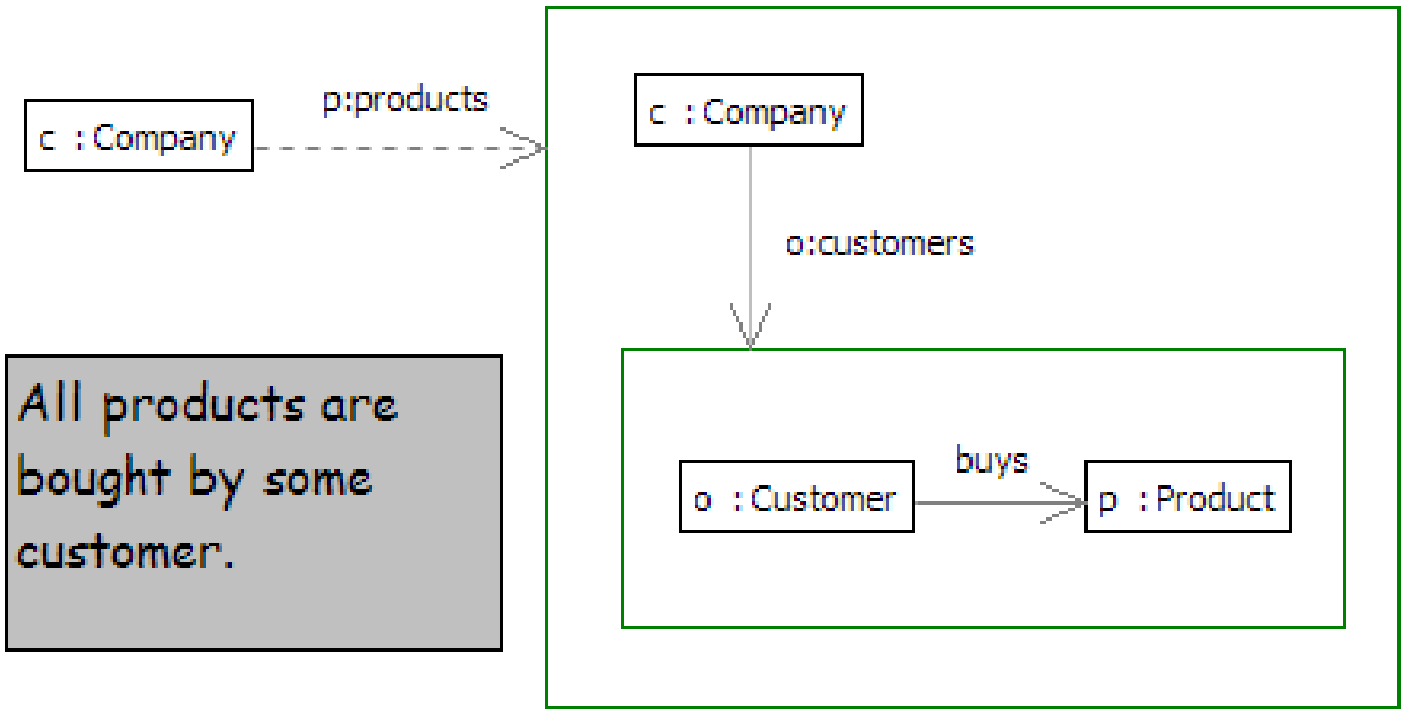



The network has at least two IP phones.

# Universal Quantification

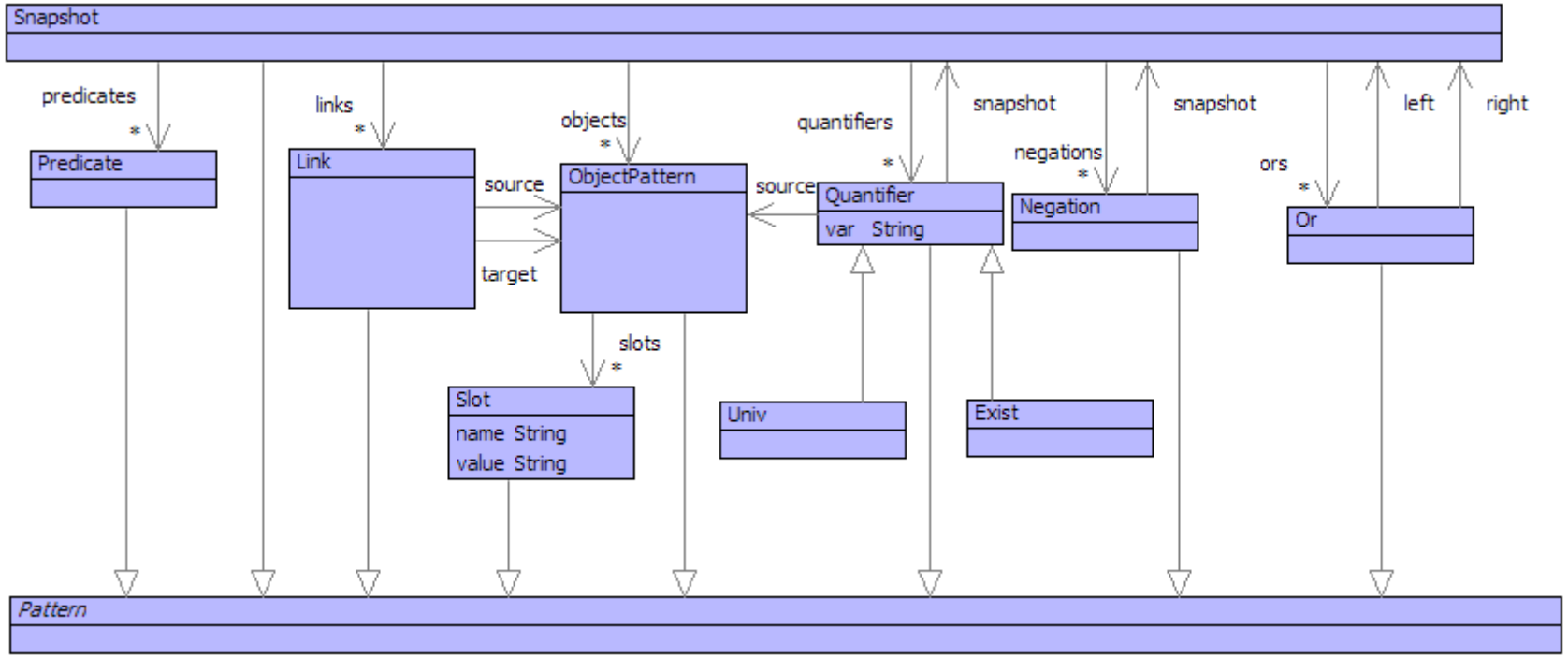


# Existential Quantification

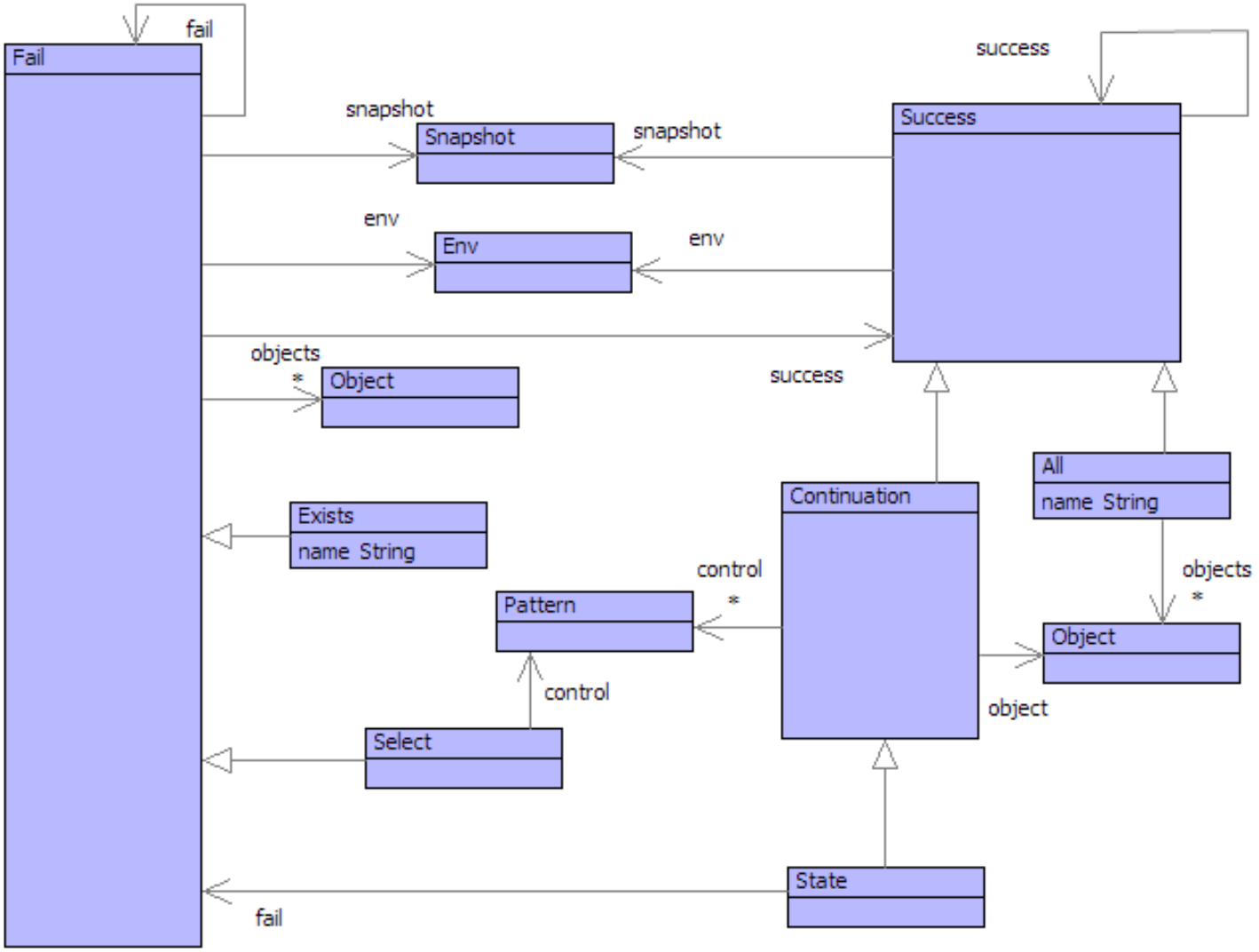


# Snapshot Checking Machine

# Syntax Model



# An Execution Machine



# Machine Transitions

| | | | |
|---|---|---|---|
| $(Obj(i,S):c,e,s,k,f)$ | $\rightarrow$ | $(S+Q+L+c,o,e,s,k,f)$ | when $i \in dom(e)$ where $Q = quant(s)$ $L = links(o,s)$ $o = e(i)$ |
| | $\rightarrow$ | $(c,e,s,k,f)$ | otherwise |
| $(n =?v:c,o[n=w],e,s,k,f)$ | $\rightarrow$ | $(c,o[n=w],e[v=w],s,k,f)$ | when $e(v) = w$ \| $v \notin dom(e)$ |
| | $\rightarrow$ | $fail(f)$ | otherwise |
| $(n=v:c,o[n=w],e,s,k,f)$ | $\rightarrow$ | $(c,o[n=w],e,s,k,f)$ | when $v = w$ |
| | $\rightarrow$ | $fail(f)$ | otherwise |
| $(\forall(v:n,s'):c,o[n=O],e,s,k,f)$ | $\rightarrow$ | $all(O,v,e,s',State(c,o[n=O],e,s,k),f)$ | |
| $(\exists(v:n,s'):c,e,o[n=O],s,k,f)$ | $\rightarrow$ | $exists(O,v,e,s',State(c,o[n=O],e,s,k),f)$ | |
| $(\rightarrow^n Obj(i,S):c,o[n=o'],e,s,k,f)$ | $\rightarrow$ | $(c,o[n=o'],s,f)$ | when $e(i) = o'$ |
| | $\rightarrow$ | $(Obj(i,S):c,o[n=o'],e[i=o'],s,k,f)$ | when $i \notin dom(e)$ |
| | $\rightarrow$ | $fail(f)$ | otherwise |
| $(\rightarrow^n Obj(i,S):c,o[n=O],e,s,k,f)$ | $\rightarrow$ | $select(Obj(i,S),O,c,e,s,k,f)$ | |

# Auxiliary Machine Defs

$$all(\{\}, v, e, s, k, f) = restart(k, f)$$
$$all(O + o, v, e, s, k, f) = (roots(s, e), e[v = o], s, All(O, v, e, s, k), f)$$

$$exists(\{\}, v, e, s, k, f) = fail(f)$$
$$exists(O + o, v, e, s, k, f) = (roots(s, e), e[v = o], s, k, Exists(O, v, e, s, k, f))$$

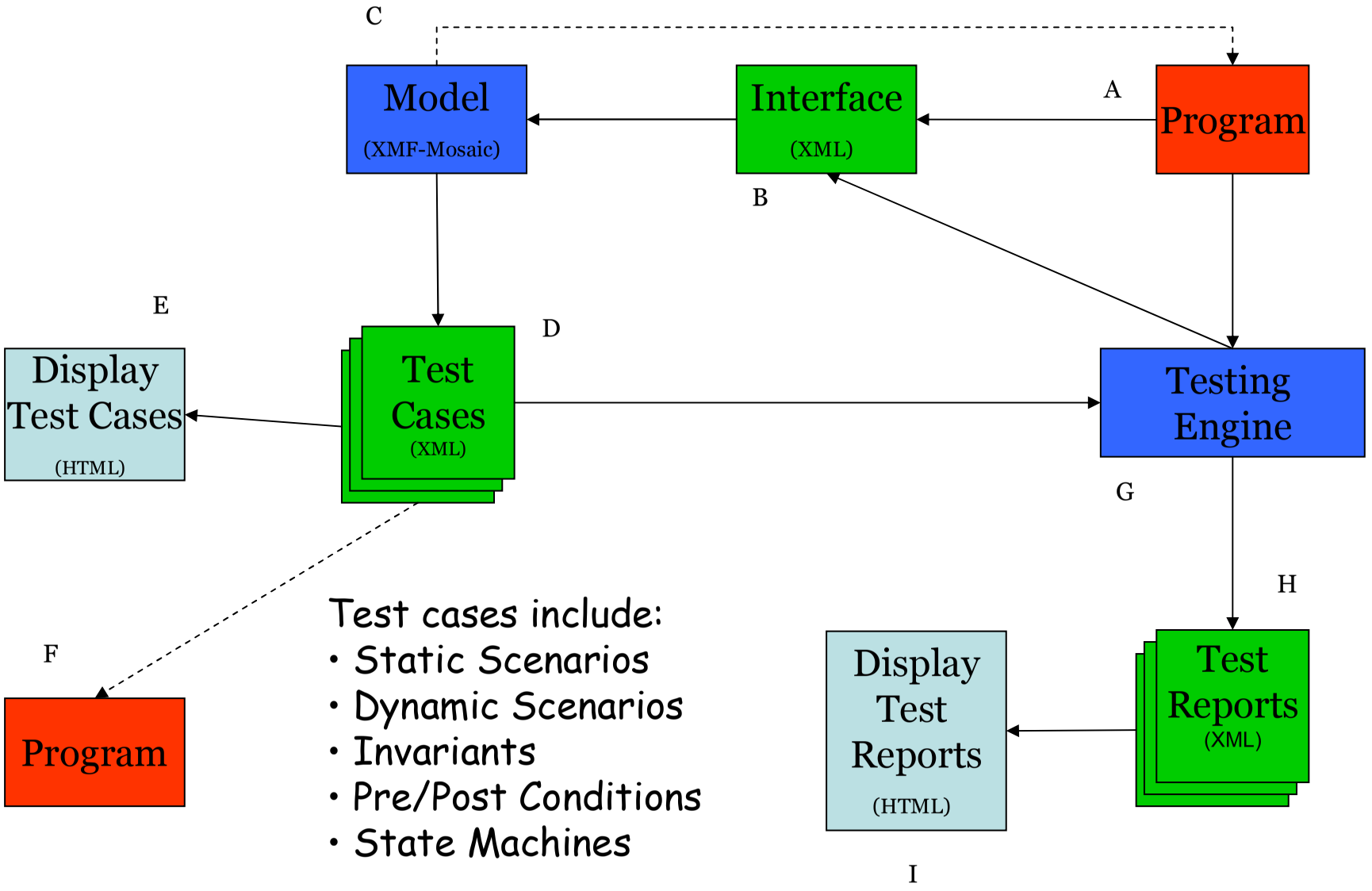
$$restart(State(c, o, e, s, k), f) = (c, o, e, s, k, f)$$
$$restart(All(O, v, e, s, k), f) = all(O, v, e, s, k, f)$$

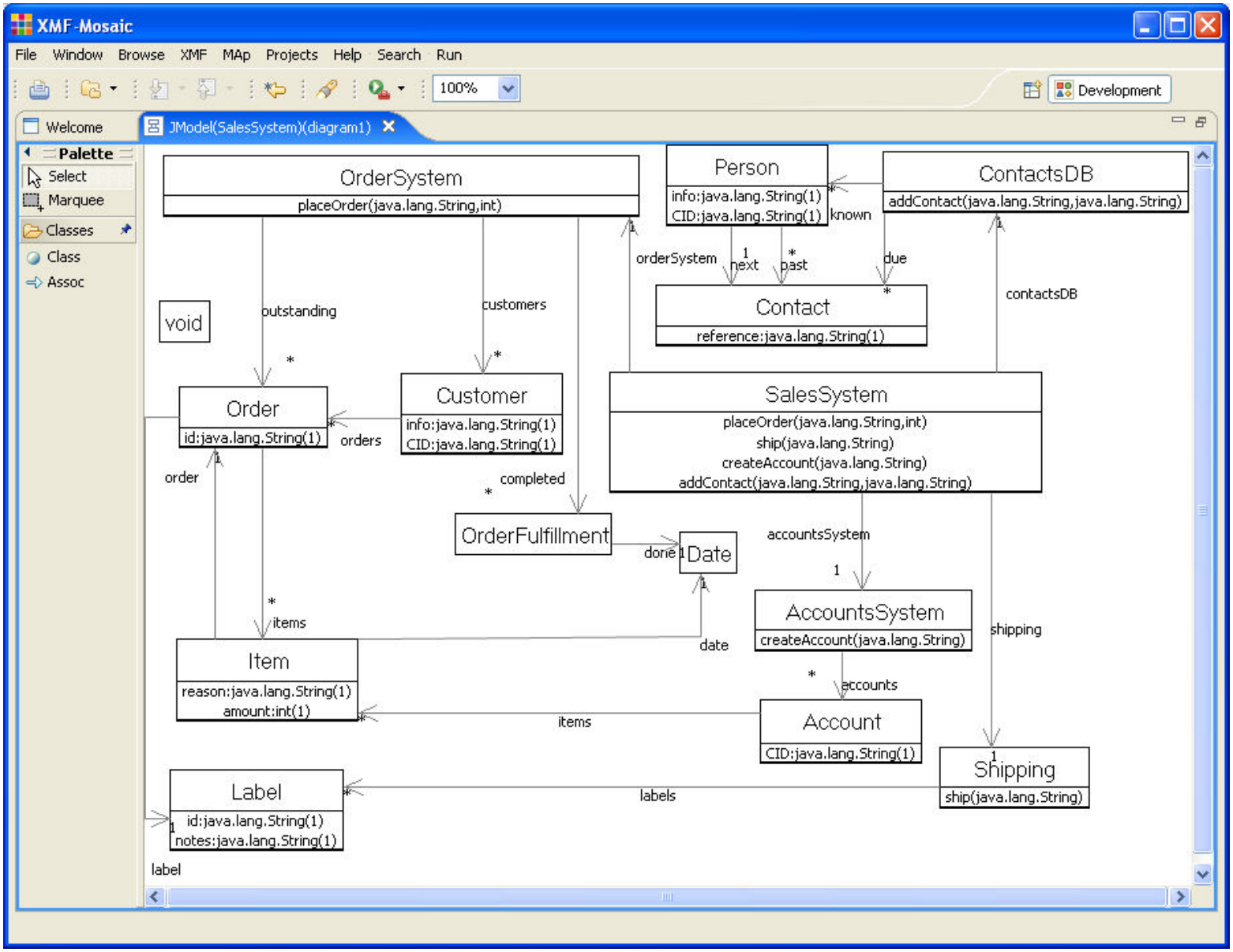
$$select(p, \{\}, c, e, s, k, f) = fail(f)$$
$$select(Obj(i, S), O + o, c, e, s, k, f) = (Obj(i, S) : c, e[i = o], o, s, k, Select(p, O, c, e, s, k, f))$$

$$fail(Exists(O, v, e, s, k, f)) = exists(O, v, e, s, k, f)$$
$$fail(Select(p, O, c, e, s, k, f)) = select(p, O, c, e, s, k, f)$$

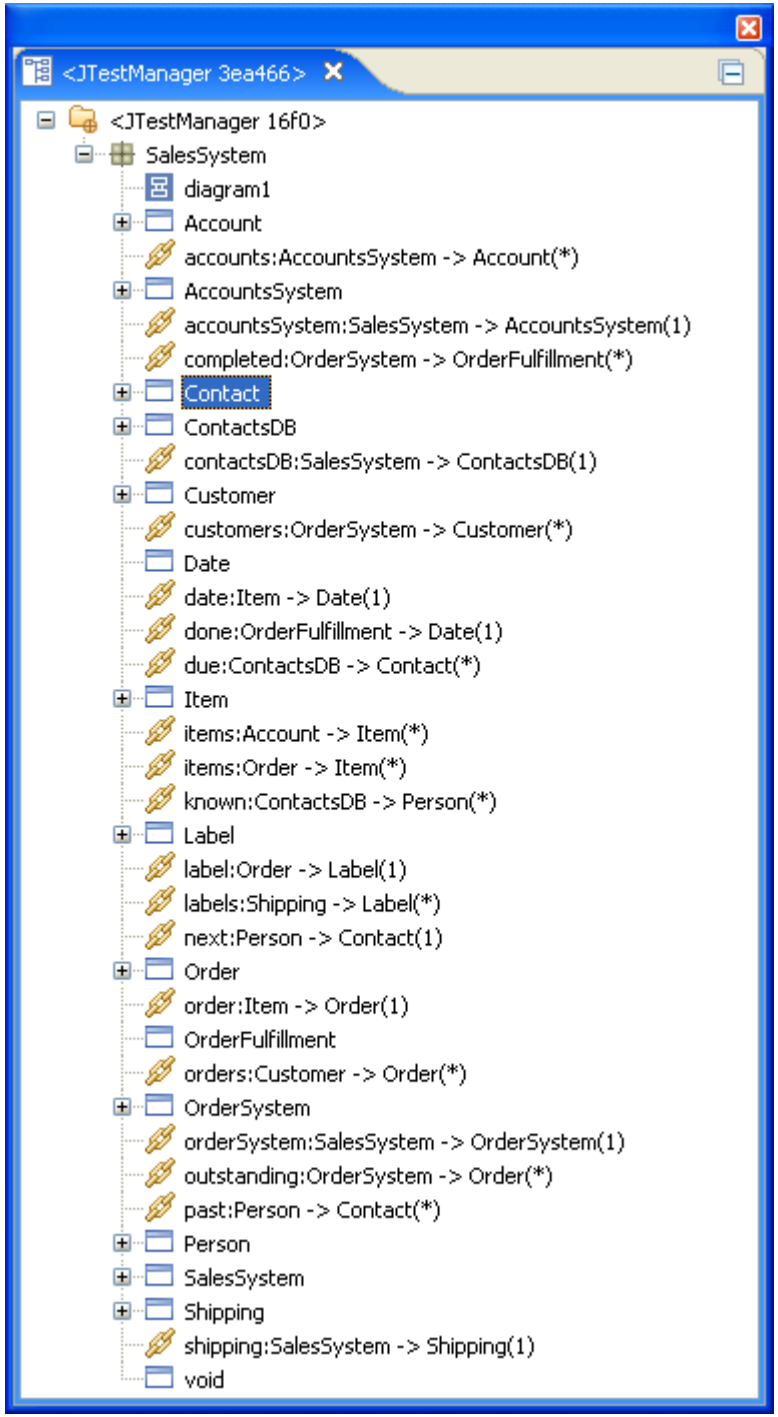
# An Implementation –
# ISWIM for Model Driven Testing

# Testing Architecture






A sales system keeps info on customers, accounts and their orders.
Customer ids (CID) are unique, but custotheir mer information may be distributed.



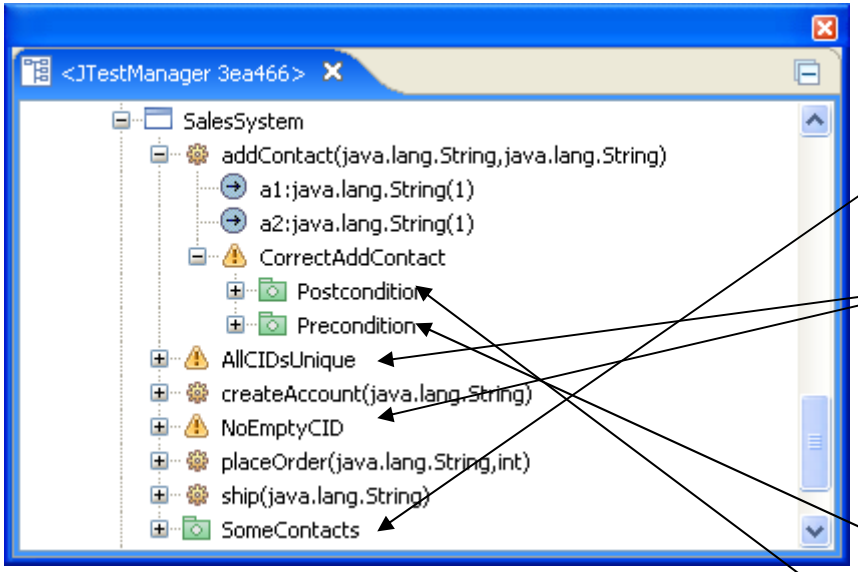

# Model Browser

A model is constructed that shows the structure and behaviour of the application.

The model could be extracted from an implementation or could be constructed by hand or could be constructed from another modelling tool.

The example shows the classes extracted from a Java application.

# Test Specification



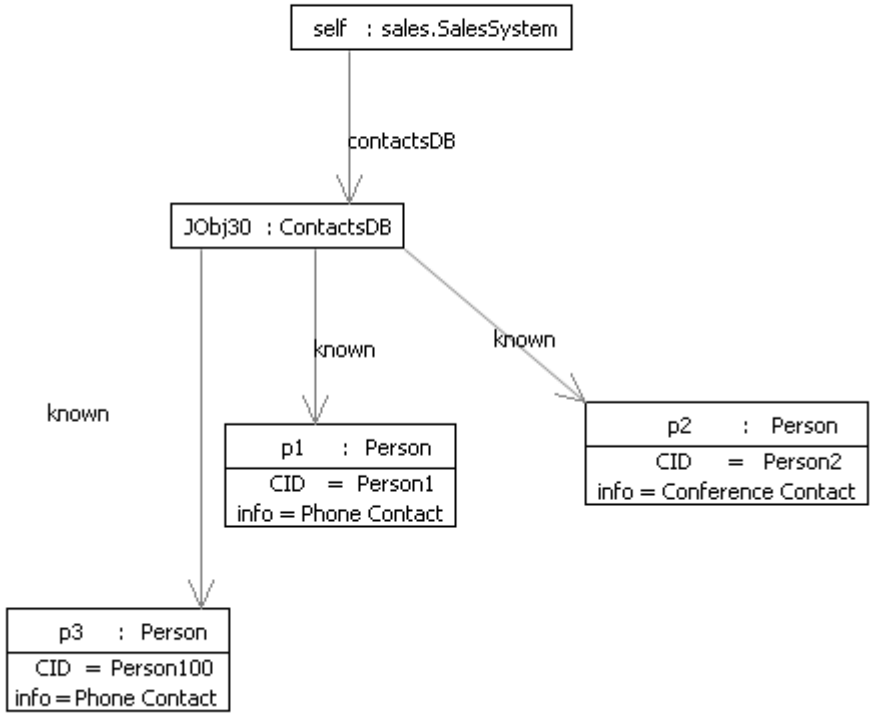

Each class may have:

- Static scenarios defining typical configurations of instances. These may be positive and negative examples.
- Invariants defining conditions that must hold true at all times.

Each operation may have specifications defining:

- Precondition defining an expectation of the method.
- Postcondition defining a corresponding guarantee if the precondition is satisfied.

# Static Scenarios




SomeContacts

A static scenario defines a collection of objects that conforms to the model.

The tool will support the construction of the scenarios by completing slots and links where possible.

Types of values can be checked by the tool.

The scenario represents an implementation independent definition of application data.

It might represent a correct, or an illegal, configuration.

The scenario can be exported in XML and recreated as implementation data.

# Example Invariant

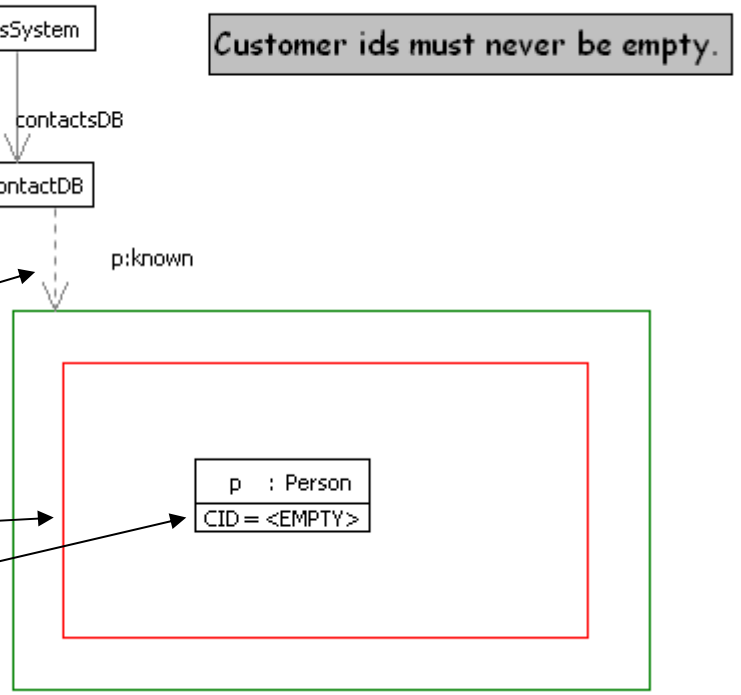
For any sales system,

With a contacts database,

For each known person p,

It is not the case,

That the customer id value is EMPTY.




NoEmptyCID

# Another Invariant

For any sales system,

With a contacts database,

For each known person p1,

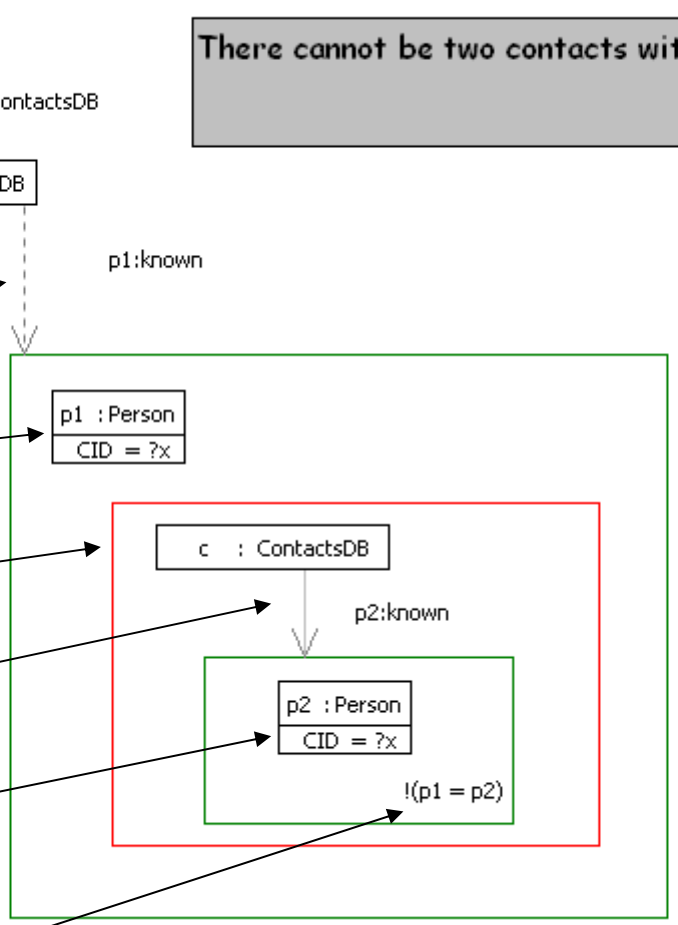
With an id x,

It is not the case that,

There is a known person p2,

With the id x,

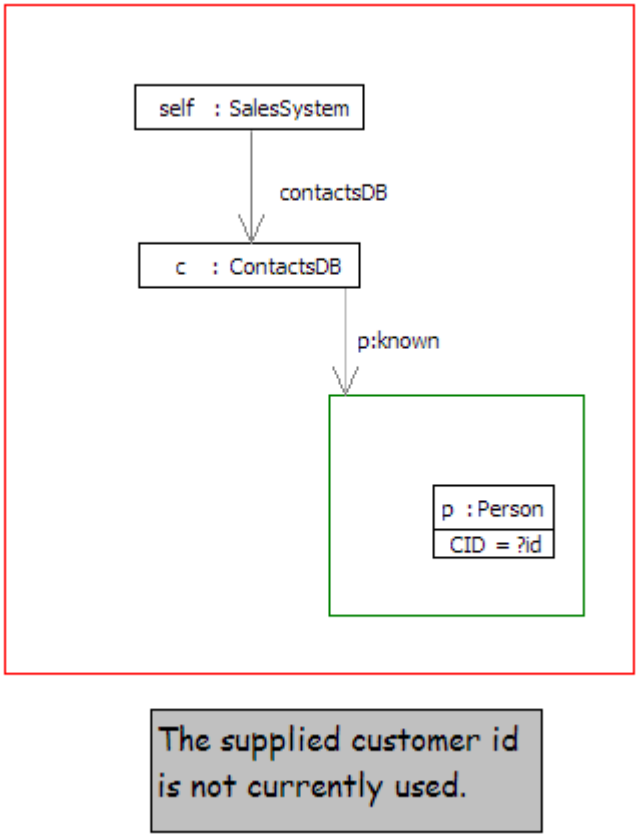
Where p1 and p2 are different people.



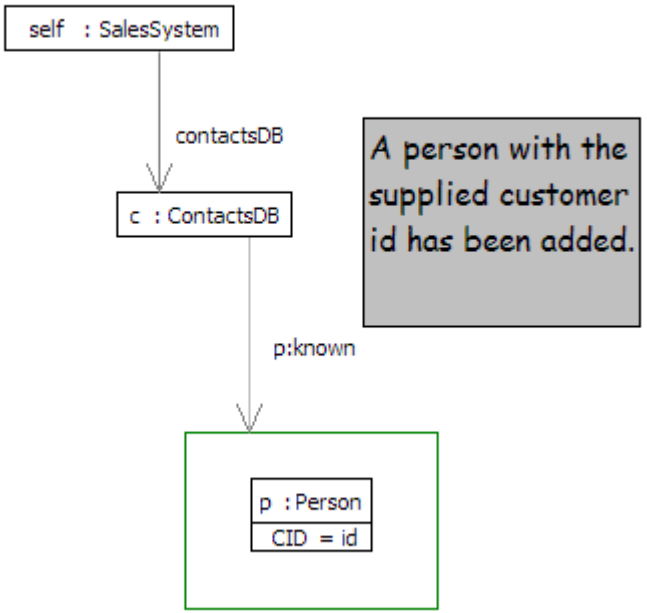

There cannot be two contacts with the same customer id.

AllCIDsUnique

# addContact(String id,String info)

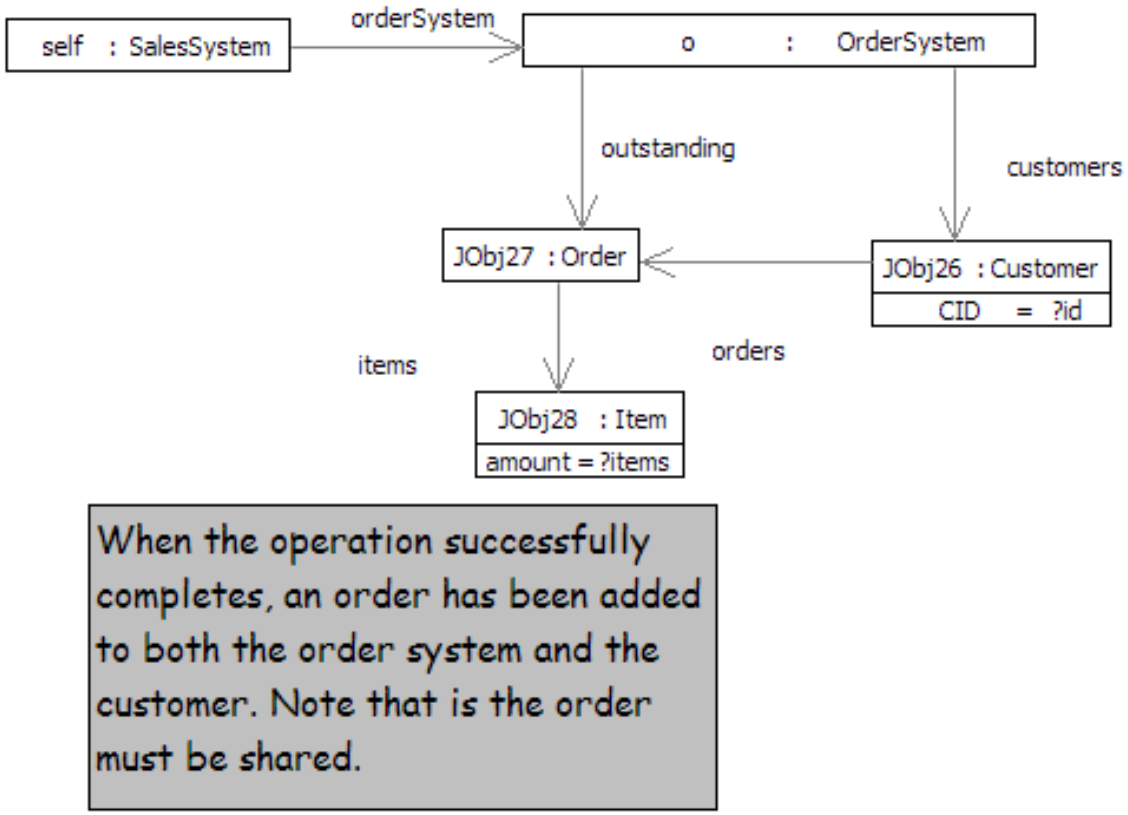



Pre-condition



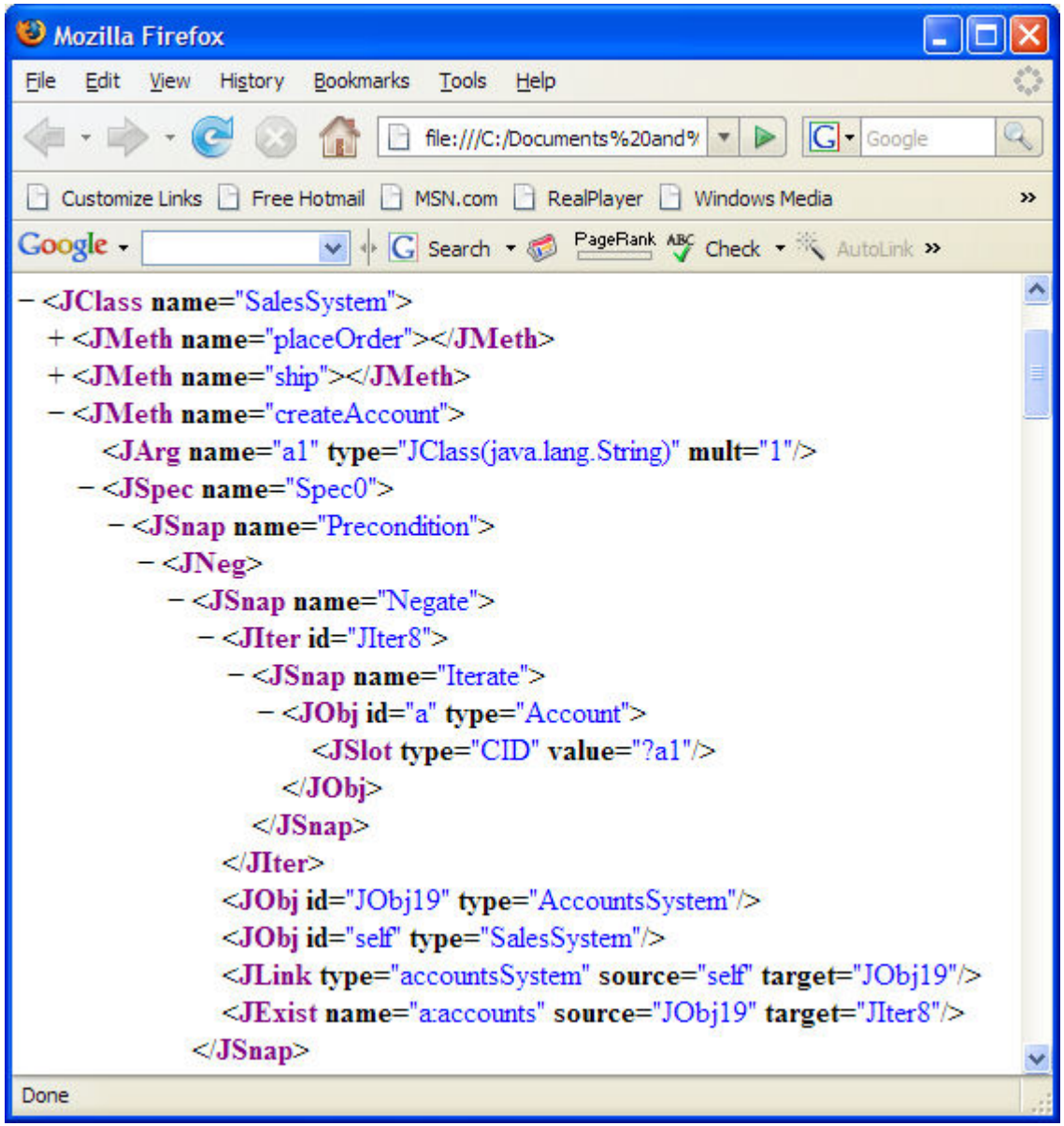

Post-condition

# placeOrder(String id,int items)




post-condition

# Export as XML



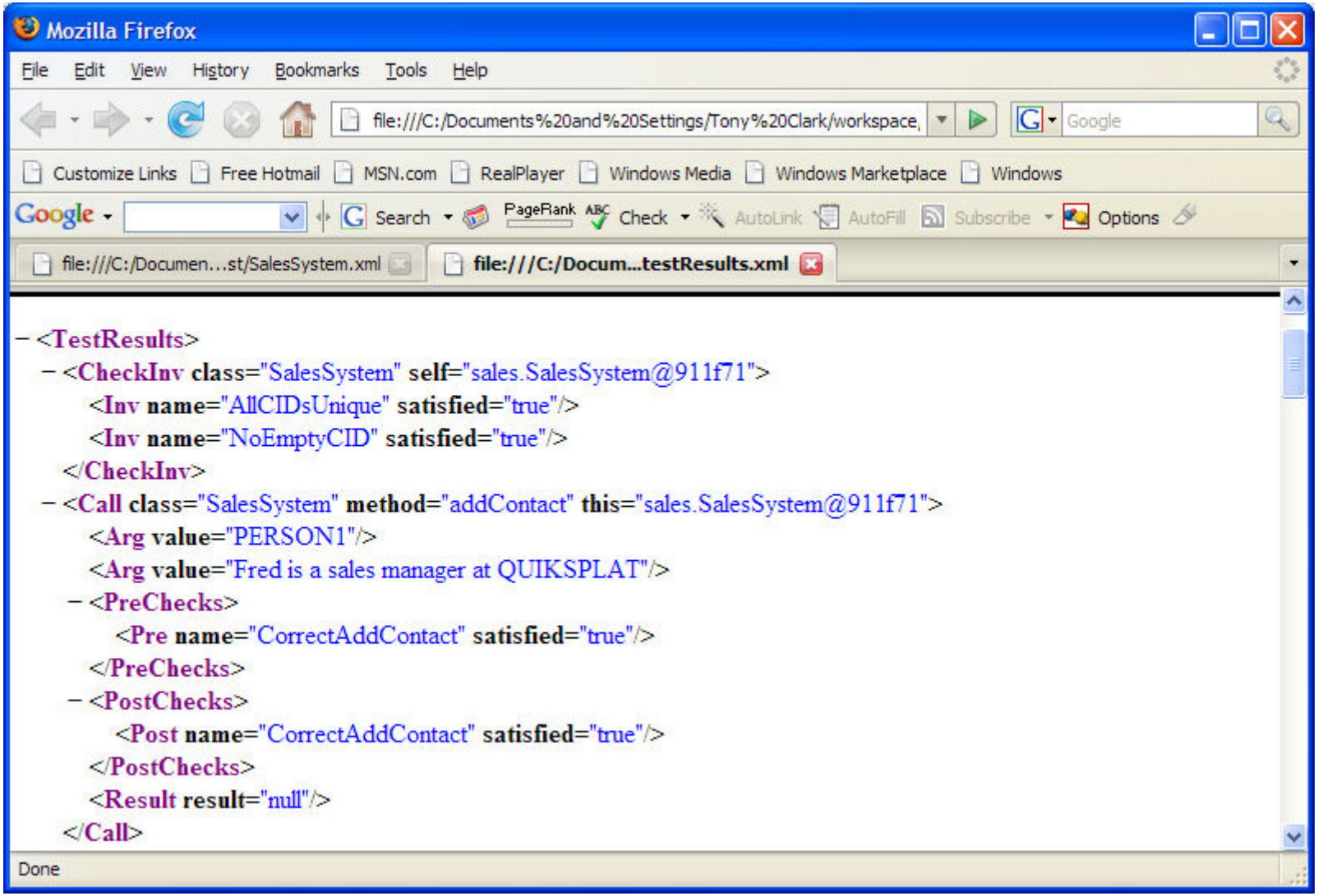
# Modify Source (or use class loader)

```
public class SalesSystem {

  ContactsDB contactsDB = new ContactsDB();
  OrderSystem orderSystem = new OrderSystem();

  public void addContact(String CID,String info) {

    JTest test = JTest.startCall("test/SalesSystem","addContact",this,CID,info);

    if(legal(CID)) contactsDB.addContact(CID,info);

    test.endCall(null);

  }

  public void placeOrder(String CID,int amount) {

    JTest test = JTest.startCall("test/SalesSystem","placeOrder",this,CID,amount);

    orderSystem.placeOrder(CID,amount);

    test.endCall(null);
  }
}
```

# A Test Harness

```
public static void main(String[] args) {

  SalesSystem sales =
   (SalesSystem)JTest.build("test/SalesSystem", "SomeContacts");

  JTest.startRecording("C:/testResults.xml");

  JTest.checkInv("test/SalesSystem", sales);
  sales.addContact("PERSON1","Fred is a sales manager.");
  System.out.println();


  sales.placeOrder("PERSON1",10);

  JTest.stopRecording();
}
```

# Test Results



# Conclusion

- Constraints occur everywhere in modelling
- Constraints can be expressed as snapshots (ISWIM).
- Constraints can be executed in various ways to suit the solution.
- ISWIM-constraints can be implemented as a tool for model driven testing.