Consistency Maintenance

Ragnhild Van Der Straeten
SSEL, Vrije Universiteit Brussel
Brussel
Kinds of Consistencies
### Classification of model inconsistencies

<table>
<thead>
<tr>
<th>Specification – specification conflicts</th>
<th>Behavioural</th>
<th>Structural</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Invocable inconsistency</td>
<td>• Dangling type reference</td>
</tr>
<tr>
<td></td>
<td>• Observation inconsistency</td>
<td>• Inherited association inconsistency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Role specification missing</td>
</tr>
<tr>
<td>Specification - instance conflicts</td>
<td>• Incompatible definition</td>
<td>• Instance specification missing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instance - instance conflicts</td>
<td>• Invocation inconsistency</td>
<td>• Disconnected model</td>
</tr>
<tr>
<td></td>
<td>• Observation inconsistency</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Incompatible behaviour</td>
<td></td>
</tr>
</tbody>
</table>
Example

```
aCashDispenser : CashDispenser
aCardReader : CardReader
anATM : ATM
aSession : Session
```

- PINEntry
  - [valid PIN] AmountEntry
  - [not valid PIN] Returning Card

- verifyAccount()
- checkIfCashAvailable()

- ejectCard()
- dispenseCash()
Formalism for consistency management

• consistency management requires:
  – a decidable formalism (to detect inconsistencies)
    • detection of inconsistencies requires answering queries over sets of individuals
  – a generic framework for consistency detection/resolution
    • to facilitate adding/removing/modifying consistency rules
Why description logics?

- decidable two-variable fragment of first-order predicate logic
- consistency between metamodel and models is guaranteed for free
- straightforward mapping of UML metamodel
  - classes → concepts
  - associations → roles
  - attributes → roles or concrete domain attribute
  - inheritance → subsumption mechanism and transitive closure
- detecting inconsistencies = answering queries
But, there is more...

- Wide range of software artefacts:
  - requirements, architectures, design models, source code, tests
- Evolution
  - e.g. refactoring
- Refinement

??Other inconsistencies??
Other approaches...

- Graph transformation schemes (e.g. by Bottoni et al.)
- Model for change propagation based on graph rewriting (e.g. by Rajlich)
- Prolog, SOUL (e.g. by R. Wuyts and K. Mens)

??Requirements which must be met by an approach??

??Does THE approach exist??

??Tool support??