

Préserver la cohérence entre les modèles UML en utilisant la logique de descriptions

Jocelyn Simmonds, Universidad de Chile

Ragnhild Van Der Straeten, Vrije Universiteit
Brussel

Viviane Jonckers, Vrije Universiteit Brussel

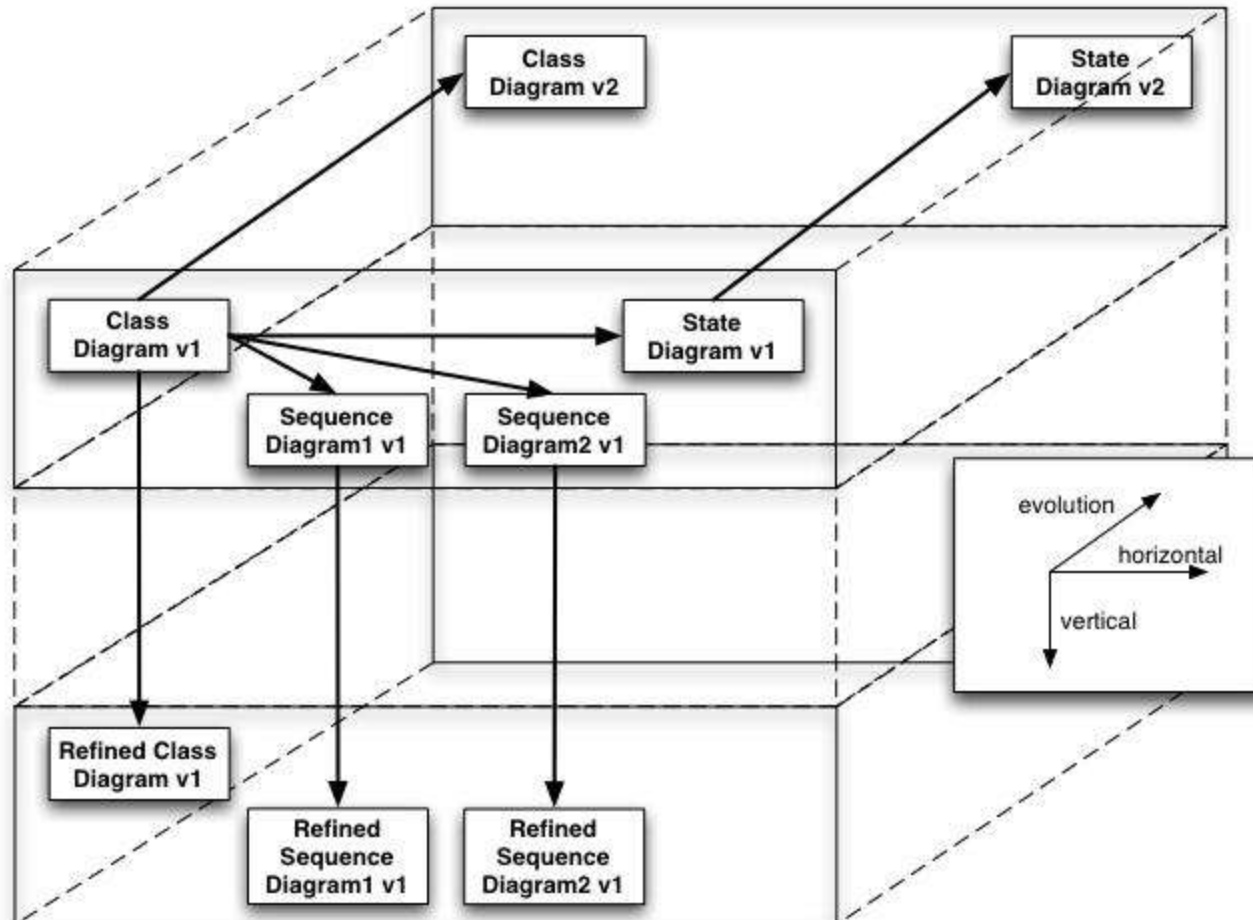
Tom Mens, Université de Mons-Hainaut



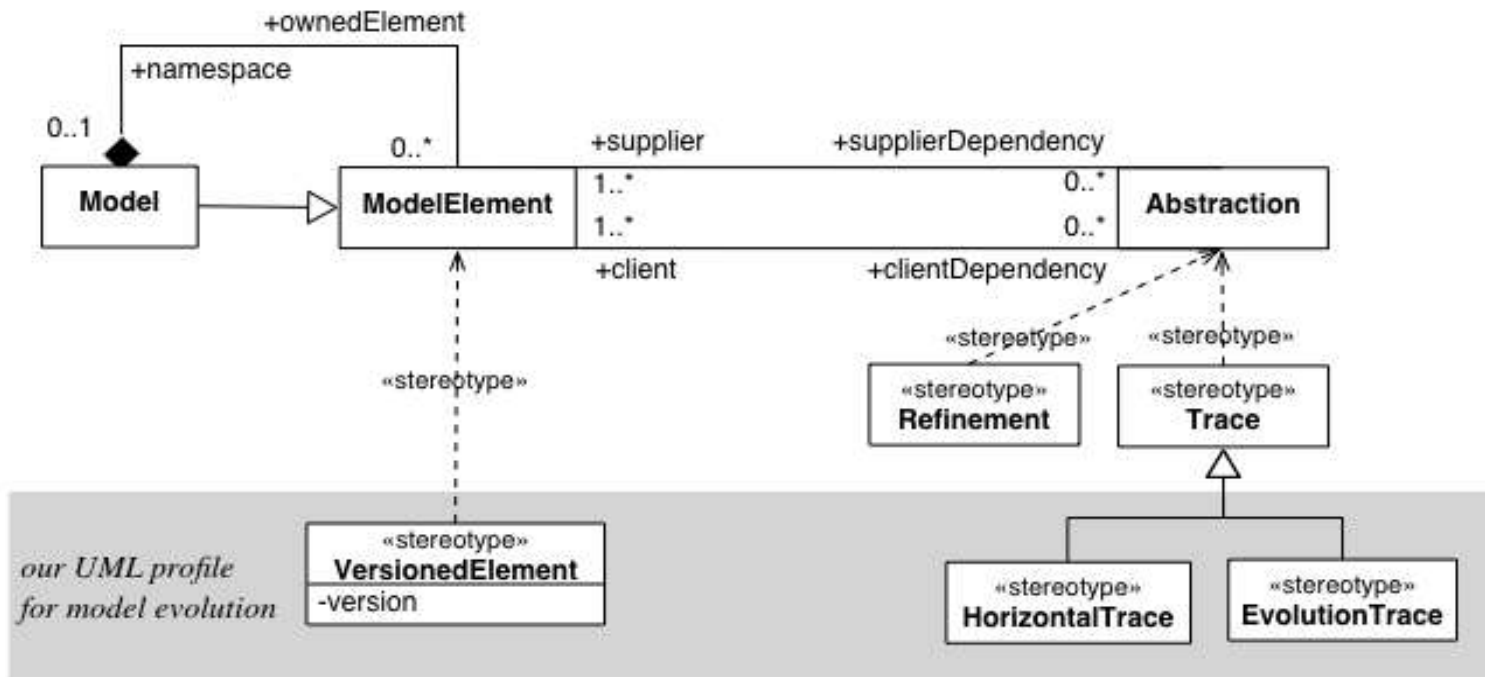
Enoncé du problème

- UML diagrams represent different views of the system design (e.g. class, sequence and state diagrams)
- Inconsistencies can be introduced due to
 - evolution
 - parallel changes made by different persons involved in the design process
- UML tools provide poor mechanisms for dealing with this problem

Cohérence des modèles



Méta modèle UML



Classification des incohérences

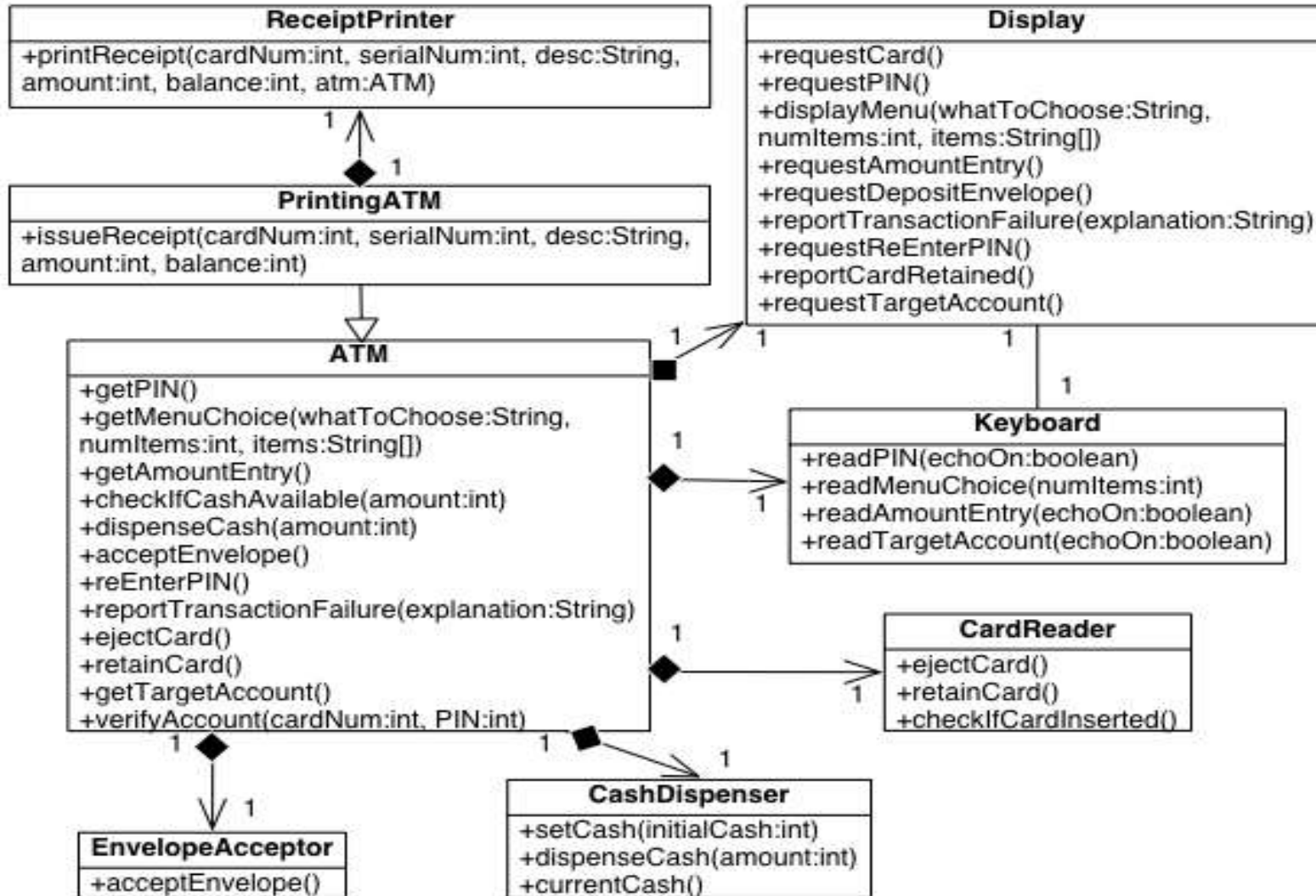
	Behavioural	Structural
Specification – specification conflicts	Invocation inconsistency Observation inconsistency	Dangling type reference Inherited association inconsistency
Specification - instance conflicts	Incompatible definition	Role specification missing Instance specification missing
Instance -instance conflicts	Invocation inconsistency Observation inconsistency	Disconnected model

**Incompatible
behaviour
inconsistency**

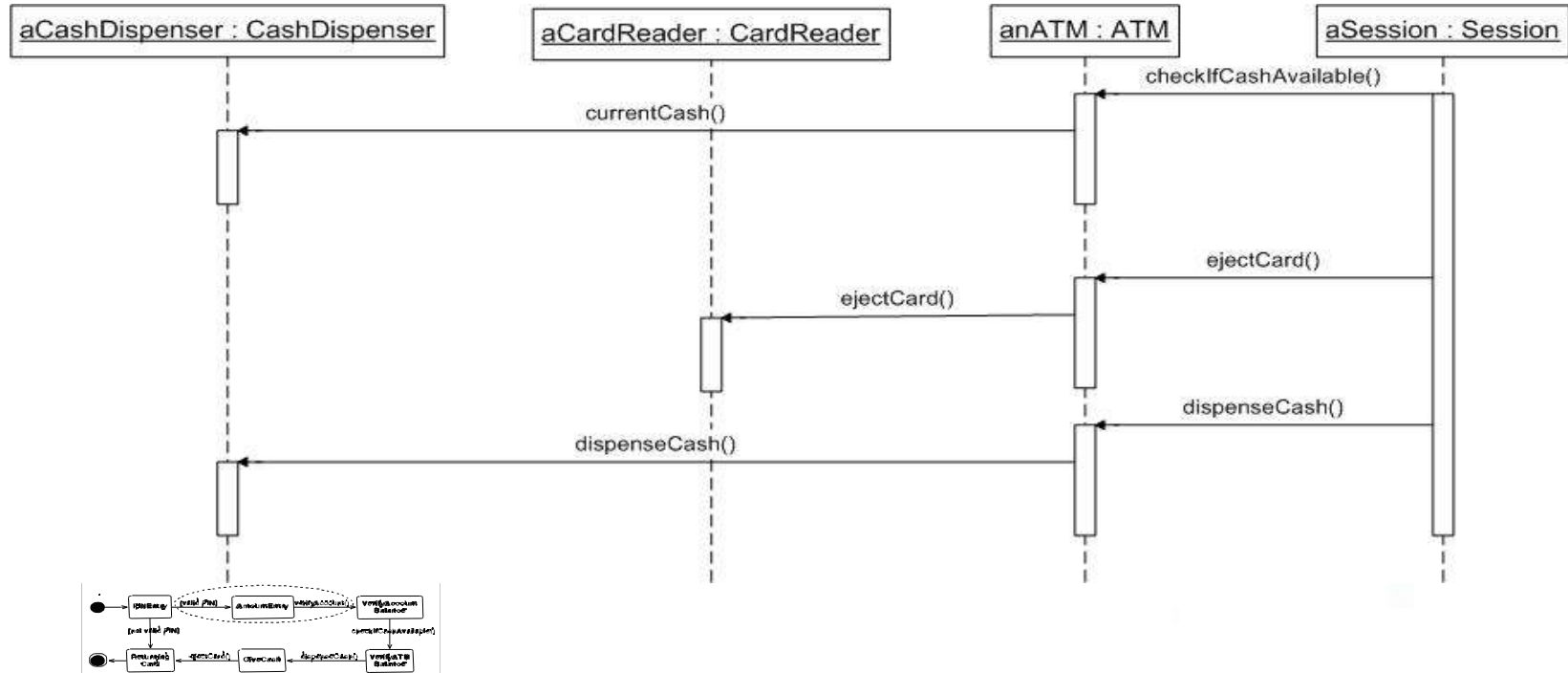
Classification des incohérences 2-dimensionnelle

- Dimension 1
 - Instance: e.g. objects unique in space and time
 - sequence diagram, state diagram
 - Specification: specification of entities at instance level
 - class diagram
- Dimension 2
 - Structural inconsistencies
 - when the structure of the system is incompatible or inconsistent
 - Behavioural inconsistencies
 - when the specification of the behaviour of the system is incompatible or inconsistent

Exemple



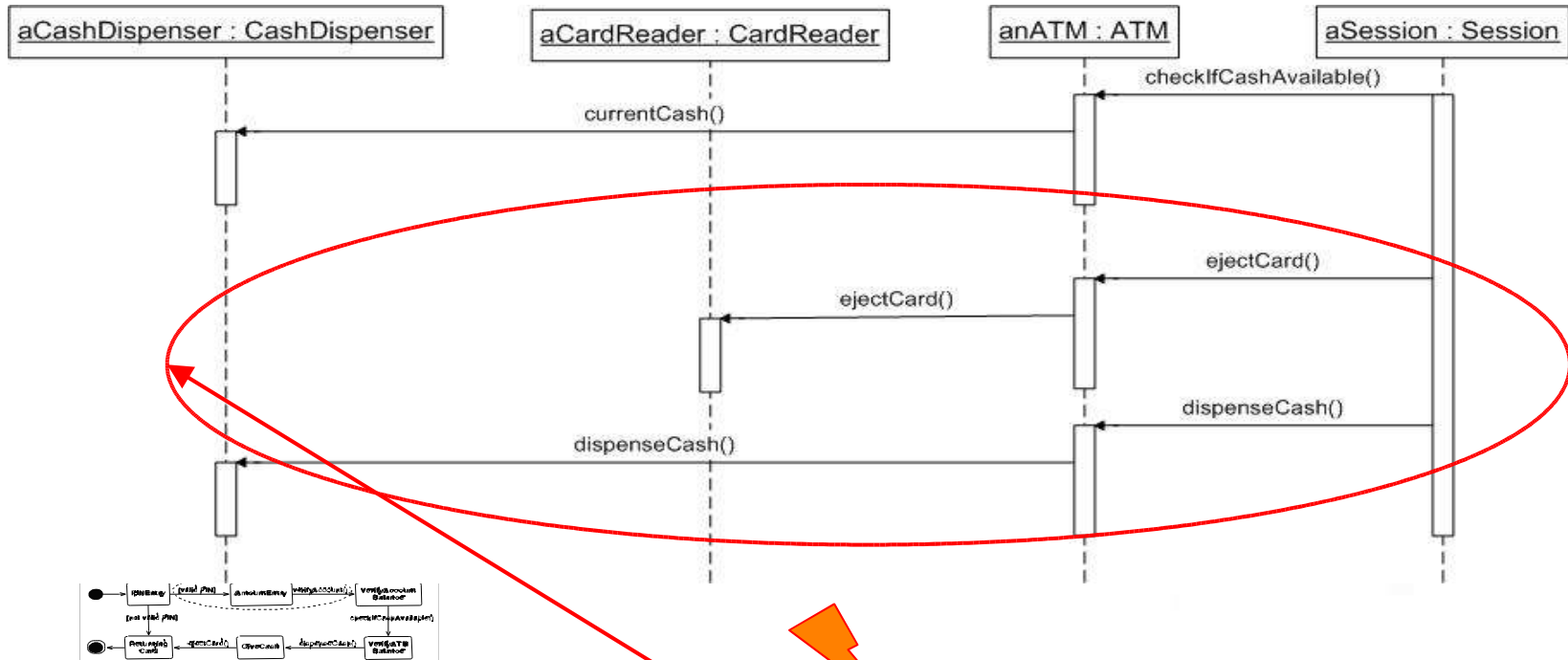
Exemple



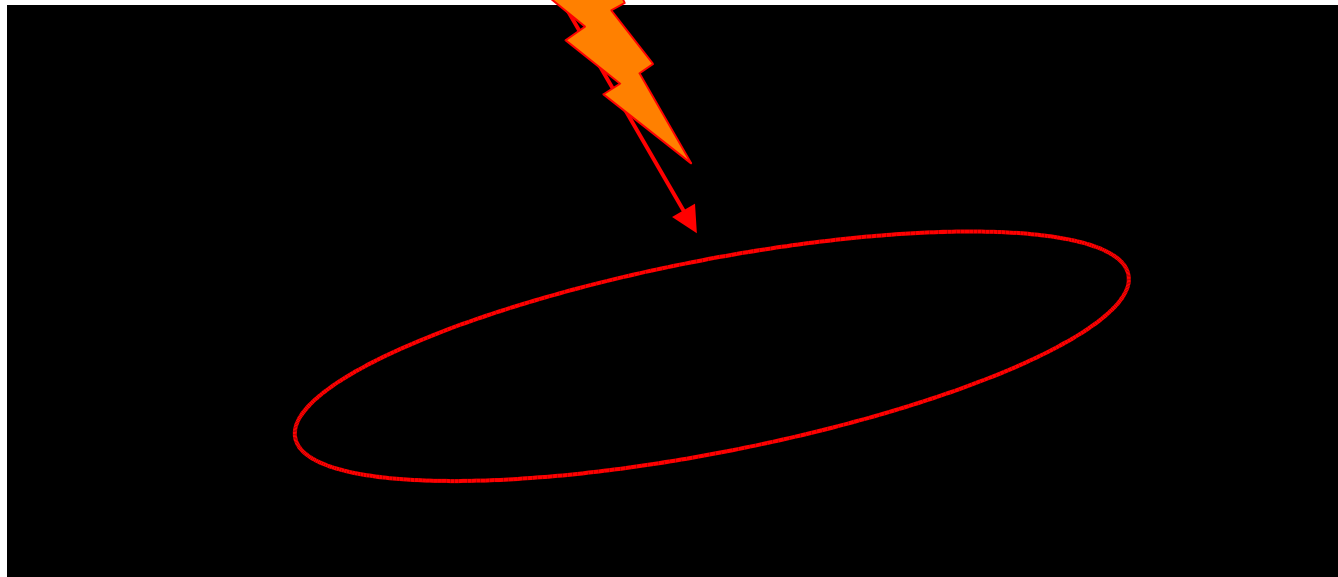
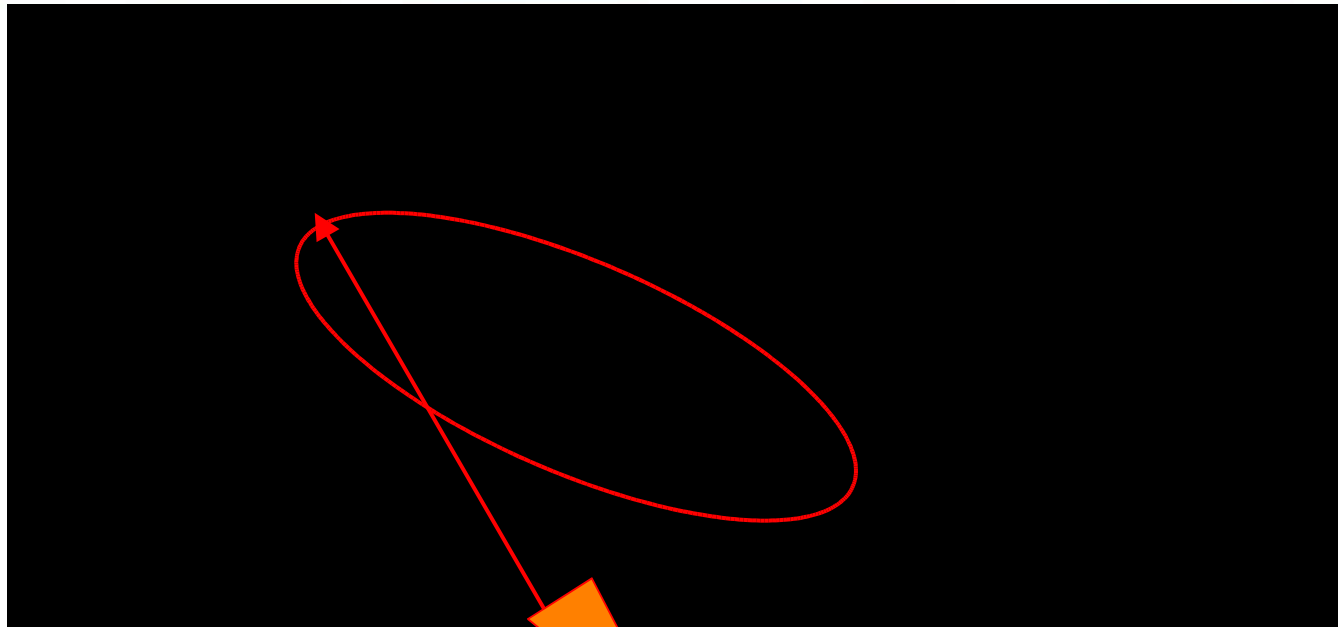
Exemple

- **Incompatible behaviour** inconsistency
 - Sequence of messages in sequence diagram is inconsistent with specification of behaviour represented by protocol state machine
 - it is not possible to find a path in the state machine that corresponds to the order established by the sequence diagram.

Exemple: incompatible behaviour



SSEL



Formalisme pour maintenir la cohérence

- a decidable formalism (to detect inconsistencies)
 - detecting inconsistencies requires answering queries in a finite time
- a generic framework for inconsistency detection/resolution
 - to facilitate adding/removing/modifying consistency rules
 - robust rules that remain valid even when the metamodel changes/evolves

Pourquoi logique de descriptions ?

- decidable two-variable fragment of first-order predicate logic
- consistency between metamodel and models is guaranteed for free
- direct correspondence of UML metamodel to DL concepts
 - classes → concepts
 - associations → roles
 - attributes → roles or concrete domain attribute (primitive types)
 - inheritance → subsumption mechanism and transitive closure
- detecting inconsistencies = answering queries

Support automatisé

- Traduction **manuelle** du méta modèle UM 

```
(LOOM:defrelation namespace :domain ModelElement  
:range Model :characteristics :single-valued)
```

```
(LOOM:defrelation ownedElement :is (:inverse  
namespace))
```

```
(LOOM:defconcept Versioned :is Stereotype (:roles  
(version)))
```

```
(LOOM:defconcept VersionedElement :is (:and  
ModelElement Versioned))
```

Support automatisé

- Traduction **automatisé** des modèles UML

The logo consists of the text 'ABox' in a blue, sans-serif font, centered within a yellow square with a thin blue border.

```
(create 'ATM class)
```

```
(tellm (:about ATM
```

```
  (name ATM)
```

```
  (Has-feature getPin)
```

```
  (Has-feature getAmountEntry)
```

```
  ...
```

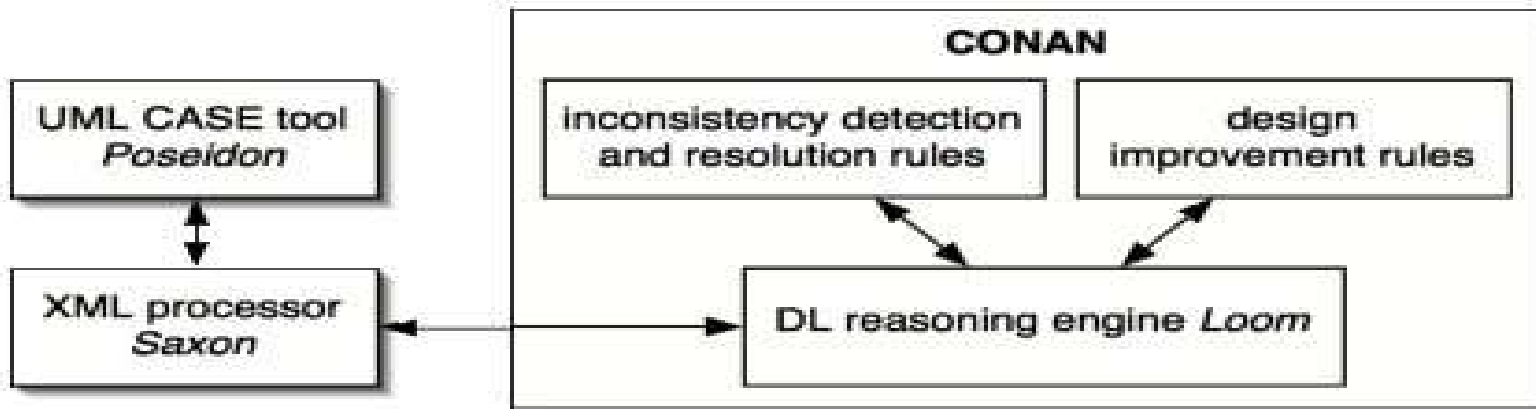
```
  (Is-parent-of PrintingATM)
```

```
  (IsAbstract false)
```

```
  (In-namespace Class-Diagram) ) )
```

Support automatisé

- UML models are
 - developed in a CASE tool (*Poseidon*)
 - exported in XMI format
 - translated into description logics format by an XSLT tool (*Saxon*)
 - imported in, and analysed with, a logic reasoning engine (*Loom*)



Expériences

Incompatible behaviour inconsistency

- Collect operations in sequence diagram

```
(retrieve (?stimulus ?callaction ?operation)
(:and
(Receiver-of ?object ?stimulus)
(Initiates ?stimulus ?callaction)
(CallAction-operation ?callaction ?
operation))))
```

Expériences

Incompatible behaviour inconsistency

```
(defun traverse-statediagram (?from-state ?from-name ?op-list)
...
;starting from a certain ?state
;retrieve the ?operation set corresponding to events on the ?
  transition
(do-retrieve (?transition ?operation ?callevnt ?state)
  (:and
    (Is-source-of ?from-state ?transition)
    (Triggered-by ?transition ?callevnt)
    (Is-occurence-of ?callevnt ?operation)
    (Is-target-of ?state ?transition) )
  (if (equalp ?operation ?current)
      (traverse-statediagram ?state (get-value ?state 'name)
        (cdr ?op- list))
      (format t "Incompatible behaviour found at state:
?from-name))))))
```

Expériences

UML (49) :

```
(traverse-sm  
  (fi VerifyAccountBalance-SM-1.0)  
  "VerifyAccountBalance"  
  (generate-received-operations (fi anATM-1.0)))
```

Current operation: |I|CHECKIFCASHAVAILABLE

Current operation: |I|DISPENSECASH

Current operation: |I|EJECTCARD

Inconsistency at state: |I|GIVECASH

NIL

Travaux futurs

- improve usability
 - integrate in UML tool
 - OCL front-end
- assess feasibility for larger examples
- other reasoning engine: Racer
- translate UML metamodel automatically
- specify model restructurings
 - detect opportunities for model refactoring
 - guarantee preservation of behaviour