

# Génération de correctifs pour les modèles partiels d'AnimUML

Mickaël Clavreul<sup>1</sup>, Frédéric Jouault<sup>1</sup>, Maxime Méré<sup>2</sup>, Matthias Brun<sup>1</sup>, Théo Le Calvar<sup>3</sup>, Matthias Pasquier<sup>4</sup>,  
and Ciprian Teodorov<sup>5</sup>

<sup>1</sup>ESEO  
Angers, France  
*first.last@eseo.fr*

<sup>2</sup>STMicroelectronics, INSA Rennes  
Le Mans, France  
*first.last@st.com*

<sup>3</sup>IMT Atlantique, LS2N,  
UMR CNRS  
6004, F-44307  
Nantes, France  
*theo.le-calvar@imt-atlantique.fr*

<sup>4</sup>ERTOSGENER  
Angers, France  
*first.last@ertosgener.com*

<sup>5</sup>Lab-STICC CNRS UMR 6285  
ENSTA Bretagne  
Brest, France  
*first.last@ensta-bretagne.fr*

Adapté de : Jouault et al. *From OCL-based model static analysis to quick fixes*. OCL Workshop, MoDELS'22.

# Sommaire

- Contexte
- Cas d'étude : AnimUML
- Approche
- Validation préliminaire
- Conclusion

# Contexte – Modélisation de systèmes

- Modélisation complexe et difficile (langages, compétences)
- Outils support
  - focus sur les concepts, la syntaxe
  - peu sur le processus de modélisation
  - l'outil devrait assister le concepteur dans la correction des erreurs de modélisation
- Besoins [1]
  - localiser
    - Rapport d'erreurs / warnings
  - comprendre
    - Exécution pas-à-pas de correctifs
  - corriger
    - Suggestions de correctifs
    - Application automatique de correctifs

[1] P. Pourali, *Tooling Advances Inspired to Address Observed Challenges of Developing UML-like Models When Using Modelling Tools*. MoDELS'18.

# Cas d'étude – AnimUML [2]

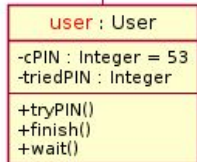
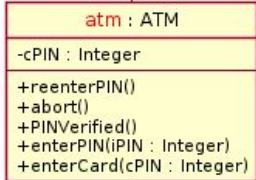
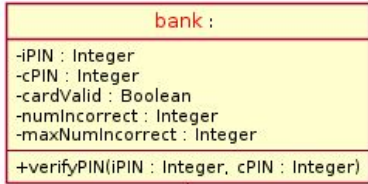
- Support de l'exécution de modèles **partiels/incomplets**, en partie **incohérents**
  - Avec outils d'analyse de leurs comportements
- **Assistance** à la mise au point de modèles complets, cohérents, corrects et conformes
  - Animation (machines à états) et traces d'exécution
  - Débogage
  - Vérification (analyse statique, *model-checking*)
- (Pédagogie) sur la méthodologie de construction de modèles
  - Petits correctifs, *feedback*

[2] Jouault et al. *Designing, animating and verifying partial UML models*. MoDELS'20.

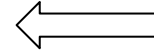
# Cas d'étude – Modèle AnimUML [2]

Instances

Classes



- [Static analysis \(auto-fix\)](#)



Or automatically

- Errors:
- Warnings:

Can be applied individually

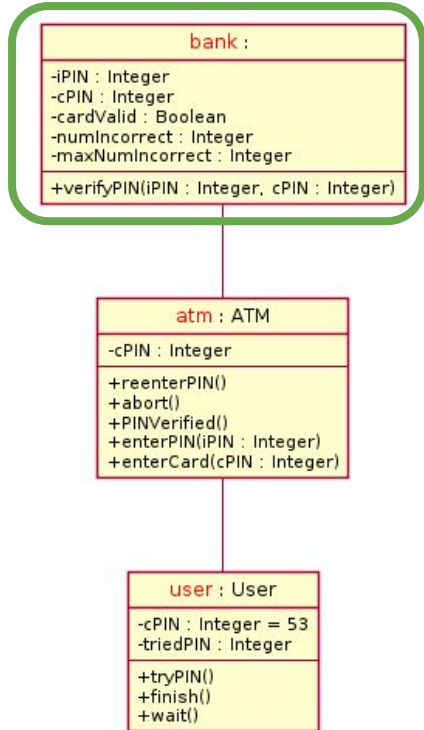
1. No class is specified for object bank.
  - [Set class to Bank.](#)
2. Class ATM of object atm not found.
  - [Create class ATM.](#)
3. Class User of object user not found.
  - [Create class User.](#)
4. No type is specified for connector connector\_bank\_atm between bank and atm.
  - [Create new association.](#)
5. No type is specified for connector connector\_user\_atm between atm and user.
  - [Create new association.](#)

[2] Jouault et al. *Designing, animating and verifying partial UML models*. MoDELS'20.

# Cas d'étude – Modèle AnimUML corrigible

Instances

Classes



- [Static analysis \(auto-fix\)](#)

- Errors:

- Warnings:

1. No class is specified for object bank.

- [Set class to Bank.](#)

2. Class ATM of object atm not found.

- [Create class ATM.](#)

3. Class User of object user not found.

- [Create class User.](#)

4. No type is specified for connector connector\_bank\_atm between bank and atm.

- [Create new association.](#)

5. No type is specified for connector connector\_user\_atm between atm and user.

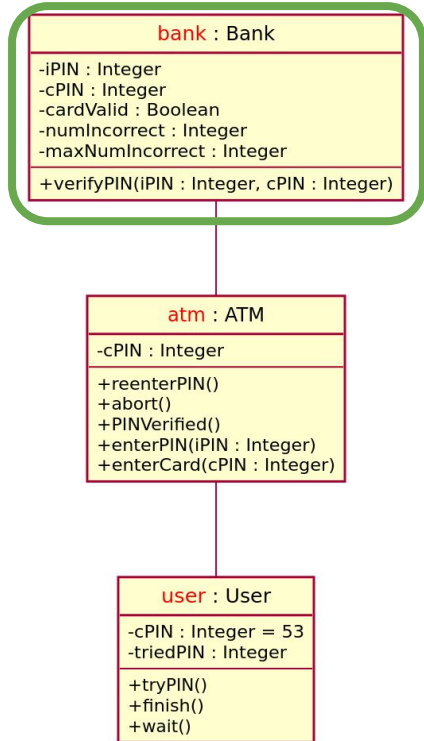
- [Create new association.](#)



# Cas d'étude – Modèle AnimUML corrigible

Instances

Classes



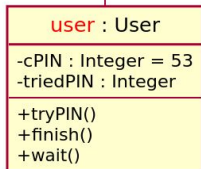
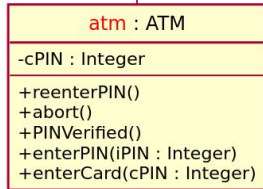
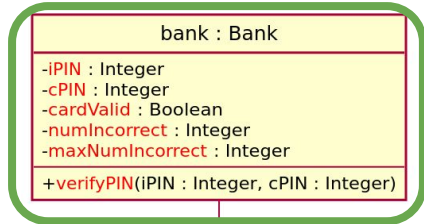
- [Static analysis \(auto-fix\)](#)
  - Errors:
  - Warninos:
    1. Class Bank of object bank not found.
      - [Create class Bank.](#)
    2. Class ATM of object atm not found.
      - [Create class ATM.](#)
    3. Class User of object user not found.
      - [Create class User.](#)
    4. No type is specified for connector connector\_bank\_atm between bank and atm.
      - [Create new association.](#)
    5. No type is specified for connector connector\_user\_atm between atm and user.
      - [Create new association.](#)



# Cas d'étude – Modèle AnimUML corrigible

Instances

Classes



- [Static analysis \(auto-fix\)](#)

- Errors:
- Warnings:
  1. Property iPIN of object bank should rather be defined on its class.
    - [Move property to class.](#)
  2. Property cPIN of object bank should rather be defined on its class.
    - [Move property to class.](#)
  3. Property cardValid of object bank should rather be defined on its class.
    - [Move property to class.](#)
  4. Property numIncorrect of object bank should rather be defined on its class.
    - [Move property to class.](#)
  5. Property maxNumIncorrect of object bank should rather be defined on its class.
    - [Move property to class.](#)
  6. Operation verifyPIN of object bank should rather be defined on its class.
    - [Move operation to class.](#)
  7. Class ATM of object atm not found.
    - [Create class ATM.](#)
  8. Class User of object user not found.
    - [Create class User.](#)
  9. No type is specified for connector connector\_bank\_atm between bank and atm.
    - [Create new association.](#)
  10. No type is specified for connector connector\_user\_atm between atm and user.
    - [Create new association.](#)



# Cas d'étude – Modèle AnimUML corrigible

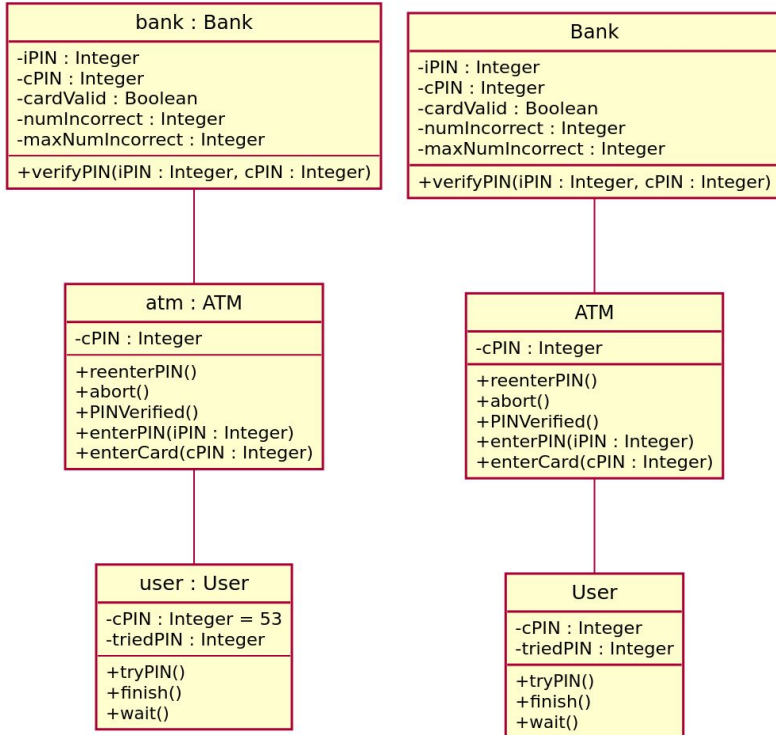
## Instances

## Classes

- [Static analysis \(auto-fix\)](#)

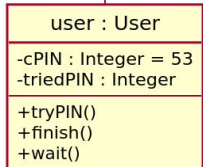
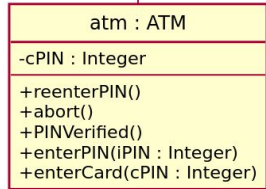
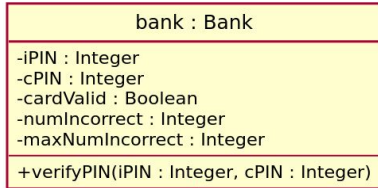


- Errors:
- Warnings:

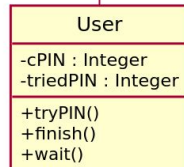
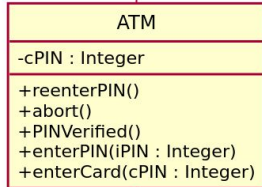
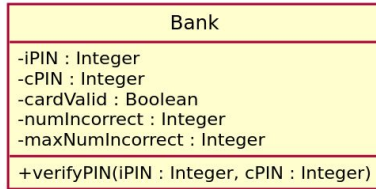


# Cas d'étude – Modèle AnimUML corrigible

## Instances



## Classes



- [Static analysis \(auto-fix\)](#)

- Errors:
- Warnings:

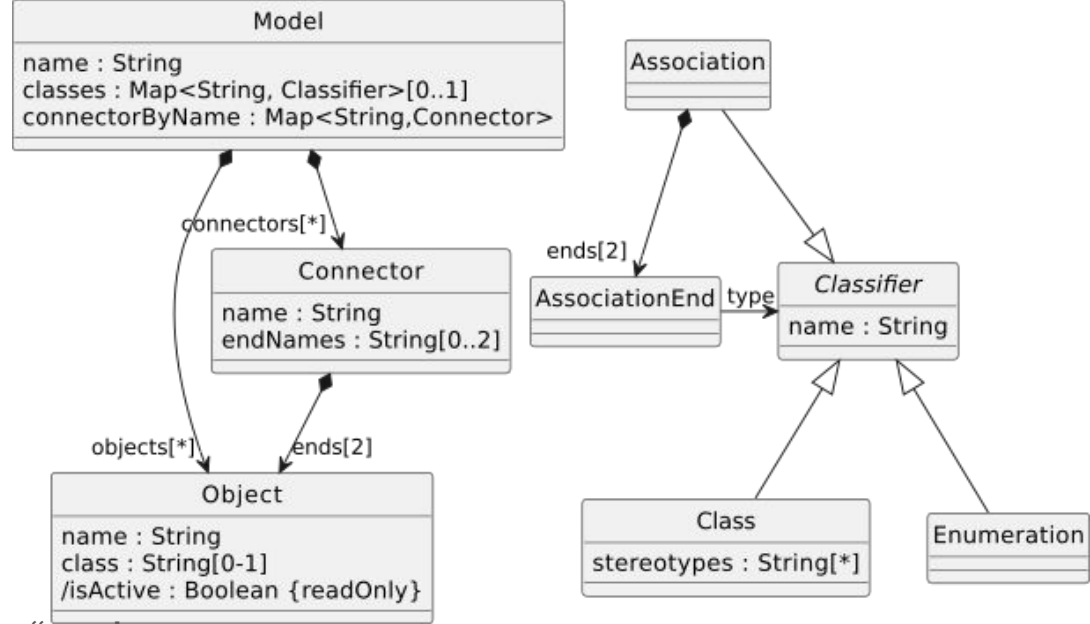
- [Auto-fix](#) applique tous les correctifs et suggestions
  - Pas-à-pas automatiquement
  - Analyse du modèle à chaque modification
- Résultat ⇒ modèle corrigé et complété, conforme

# Cas d'étude – Métamodèle AnimUML (extrait)

- Métamodèle UML partiel
  - e.g., permet objet avec nom de classe mais sans la classe correspondante

- Modèles conformes peuvent être “mal formés”

- Object::class is optional
- Model::classes may not contain all used values of Object::class
- Model::connectors between objects may not have a counterpart Association between objects classes



1. No class is specified for object bank.
  - [Set class to Bank.](#)
2. Class ATM of object atm not found.
  - [Create class ATM.](#)
3. Class User of object user not found.
  - [Create class User.](#)
4. No type is specified for connector connector\_bank\_atm between bank and atm.
  - [Create new association.](#)
5. No type is specified for connector connector\_user\_atm between atm and user.
  - [Create new association.](#)

# Cas d'étude – Analyse statique et Invariants OCL

**context** Object **inv** :

```
(not self.class.oclIsUndefined())
```

```
and not model.classes.get(self.class).oclIsUndefined()
```

1. No class is specified for object bank.
  - [Set class to Bank.](#)
2. Class ATM of object atm not found.
  - [Create class ATM.](#)
3. Class User of object user not found.
  - [Create class User.](#)

Extrait de : Jouault et al. *From OCL-based model static analysis to quick fixes*. OCL Workshop, MoDELS'22.

# Cas d'étude – Analyse statique et Invariants OCL

```
context Object inv :  
  (not self.class.oclIsUndefined())  
  and not model.classes.get(self.class).oclIsUndefined()
```

```
context Connector inv :  
  let class_end0 : Classifier = model.classes.get(self.ends->get(0).class) in  
  let class_end1 : Classifier = model.classes.get(self.ends->get(1).class) in  
  not class_end0.oclIsUndefined()  
  and not class_end1.oclIsUndefined()  
  and model.classes->select((cname,cl) | cl.oclIsTypeOf(Association)  
    and (cl.ends->get(0).type = class_end0 and cl.ends->get(1).type =  
      class_end1)) ->notEmpty()
```

1. No class is specified for object bank.
  - [Set class to Bank.](#)
2. Class ATM of object atm not found.
  - [Create class ATM.](#)
3. Class User of object user not found.
  - [Create class User.](#)
4. No type is specified for connector connector\_bank\_atm between bank and atm.
  - [Create new association.](#)
5. No type is specified for connector connector\_user\_atm between atm and user.
  - [Create new association.](#)

# Problématique

- Outillage et implémentation complexe et faillible
  - Construction de messages spécifiques requiert de décomposer les invariants
    - Complexité de la décomposition et réécriture
    - + Traçabilité des dépendances
  - Correctifs (quick-fixes) doivent être corrects
    - i.e., résoudre le problème à corriger
      - garantie difficile à prouver
      - sinon déroutant + difficulté à converger pour l'[auto-fix](#)
- Ressources limitées pour les éditeurs d'outils de modélisation
  - En particulier pour les outils académiques
- Solution ⇒ Création automatique des correctifs

# Approche

## *Génération automatique de correctifs basée sur les invariants OCL, par rétro-propagation*

- Pour les développeurs (des outils de modélisation) :
  - Écriture de règles / contraintes pour vérifier la construction des modèles
  - Ajout d'annotations
    - Sévérité (*error, warning*)
    - Messages personnalisés plus pertinents
    - Guider la création automatique des correctifs
- Génération automatique des correctifs
  - Analyse des contraintes OCL
  - Rétro-propagation
  - Expertise développeur

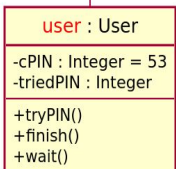
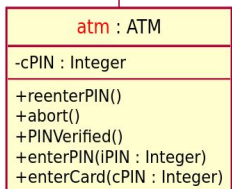
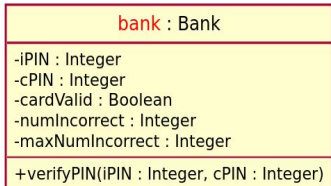
# Rétro-propagation – Exemple 1

context Object inv :

```
(not self.class.oclIsUndefined())
```

and not

```
model.classes.get(self.class).oclIsUndefined()
```



For object *bank*

<code>a = self</code>	<code>&lt;object bank&gt;</code>	
<code>b = self.class</code>	<code>OclUndefined</code>	<code>set to self.name.toUpper()</code>
<code>c = not b.ocllsUndefined()</code>	<code>false</code>	<code>true?</code>
<code>d = c and [...]</code>	<code>false</code>	<code>true?</code>



# Invariant OCL – Annotations

```
context Object inv :  
  (not self.class.oclIsUndefined()) -- @severity: warning, @value:  
self.name.firstToUpper()  
  and not model.classes.get(self.class).oclIsUndefined() -- @severity:  
warning, @value: Class.newInstance().refSetValue('name', propName)
```

# Évaluation préliminaire

- Implémentation en JavaScript
  - opérateurs OCL ↔ fonctions JS
  - Annotations
- Implémentation non-incrémentale
  - Réévaluation des règles à chaque correction
- Messages **générés** ou **fournis** par le développeur

1. No class is specified for object bank.
  - [Set class to Bank.](#)
2. Class ATM of object atm not found.
  - [Create class ATM.](#)
3. Class User of object user not found.
  - [Create class User.](#)
4. No type is specified for connector connector\_bank\_atm between bank and atm.
  - [Create new association.](#)
5. No type is specified for connector connector\_user\_atm between atm and user.
  - [Create new association.](#)

```
warning: class of object bank is undefined.
```

```
set class of object bank to: Bank
```

```
warning: Bank of classes of model is undefined.
```

```
set Bank of classes to: new class
```

```
warning: classes of model should include at least one  
element satisfying "can type connector C1 between bank  
: Bank and atm : ATM"
```

# Conclusion

- Approche pour la définition et la génération de correctifs
  - Invariants OCL + annotations
  - Évaluée avec AnimUML
- Limites
  - Quelques opérateurs OCL évalués
  - Maîtrise du processus de rétro-propagation ?
  - Correction des *quick fixes*, quelles garanties ?
- Perspectives
  - Évaluation des opérateurs OCL non abordés, intégration d'un parser OCL ?
  - Expérimentation de l'approche sur d'autres langages de modélisation, avec un public moins expert...
  - Outils de modélisation : Interacteurs / Interactions ?

# Merci de votre attention !

Questions?