Gestion de la cohérence lors de la construction de logiciels

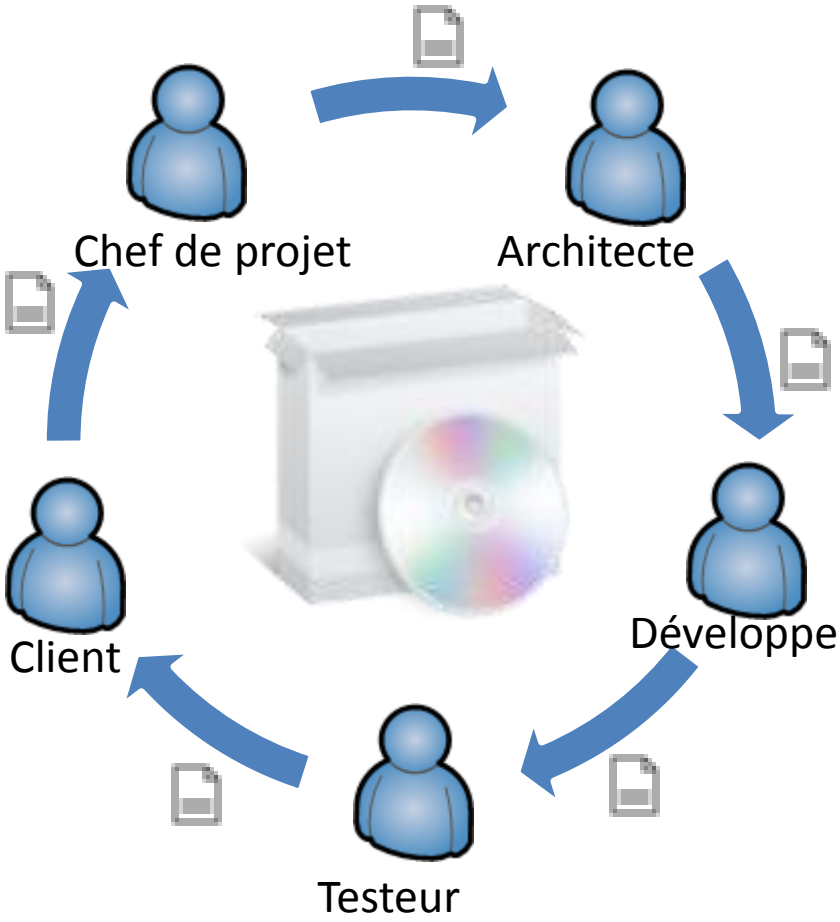
Xavier Blanc

Université Pierre et Marie Curie
Laboratoire d'Informatique de Paris 6

Agenda

- Génie logiciel et approche guidée par les modèles
- Gestion de la cohérence
  - Problématique
  - L'approche Praxis
  - Résultats obtenus
- Conclusion et perspectives

Génie Logiciel



Production de Logiciel

"Software engineering is an engineering discipline which is concerned with all aspects of software production" Sommerville

Environnement GL

"A set of management and technical tools to support software development, usually integrated in a coherent framework" [WAS90]

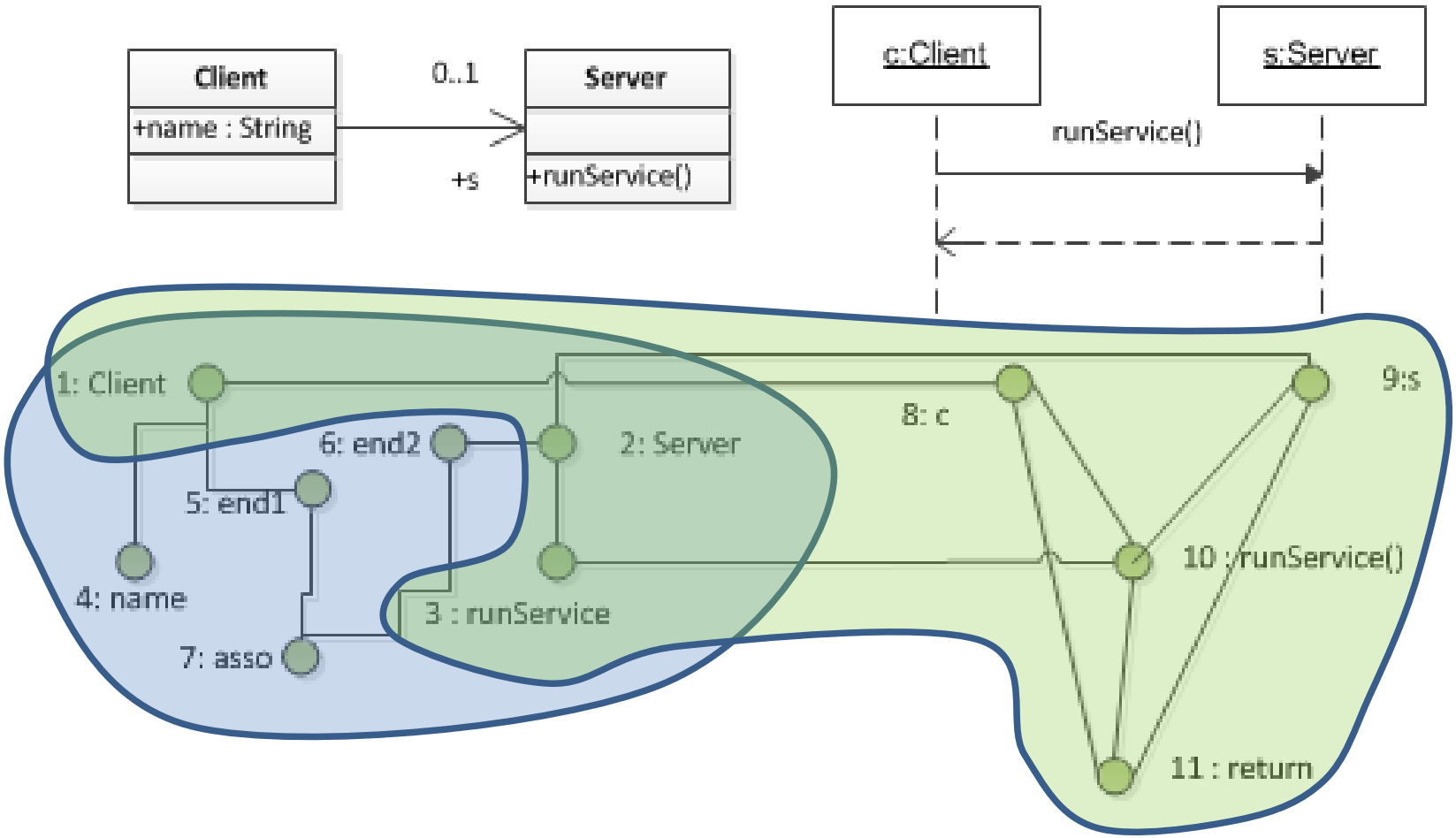
Points clés

- Intégration des données
- Intégration du contrôle
- Intégration du procédé

Approche guidée par les modèles

- Principe [OMG01][Mellor03][Selic03][Schmidt06]
  - Artefact logiciel => Modèle (Graphe typé)
  - Logiciel => décrit par 1 modèle global (N modèles)
  - Procédé => Construction du modèle global
- Impacts
  - Taille du modèle global (millions d'éléments)
  - Granularité (modèle, vue, diagramme)

UML, l'exemple classique

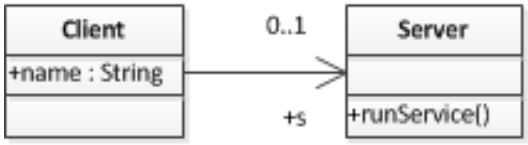


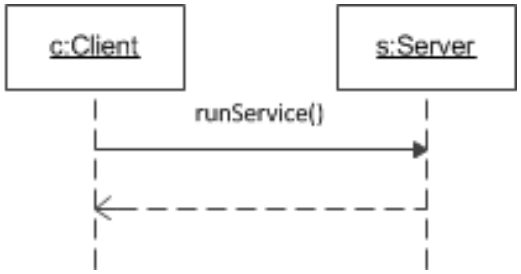
Gestion de la cohérence

- Tous les éléments du modèle global décrivent un même logiciel
- Si des éléments du modèle décrivent une même sous-partie du logiciel, ils doivent être cohérents
- Règles de cohérence
  - Syntaxique
  - Sémantique

Exemples

- Chaque message doit référencer une des opérations que possède la classe à destination du message
- Les types des lignes de vie d'une séquence sont des classes
- Pas de cycle d'héritage
- …





Vivre avec les incohérences

- Lors de l'évolution continue du modèle global, il est difficile d'assurer à tout moment la cohérence
- L'objectif devient la détection les **in**cohérences afin d'offrir un support à leur résolution
  - [Balzer91] [Finkelstein94-96] [Nentwitch03] [Elaasar04] [Mens06] [Egyed07]
- Difficultés
  - Taille des modèles
  - Fréquence des modifications

Praxis

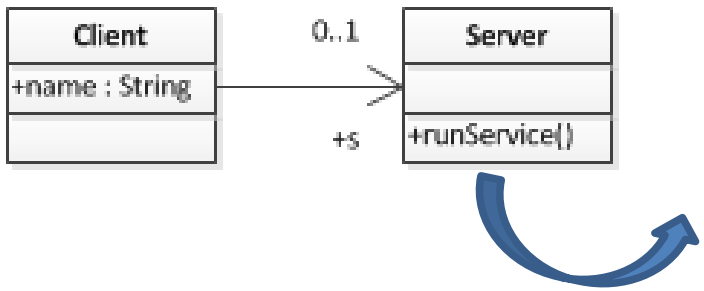
- Objectif : Détection des incohérences (ajout & suppression)
  - Modèle de grande taille
  - Avec évolution continue

- Approche
  1. Format basé sur les actions de construction
  2. Détection incrémentale

Les actions élémentaires de Praxis

6 actions

create(eltId,type)
delete(eltId)
addProperty(eltId,type,val)
remProperty(eltId,type,val)
addReference(eltId,type,val)
remReference(eltId,type,val)



1. create(1,class)
2. addProperty(1,name, 'Client')
3. create(4,attribute)
4. addProperty(4,name,'name')
5. addProperty(4,type, 'String')
6. addReference(1,attribute,4)
7. create(2,class)
8. addProperty(2,name,'Server')
9. create(3,operation)
10. addProperty(3,name,'runService')
11. addReference(2,operation,3)
12. create(7,association)
13. …

Les règles de cohérence Praxis

- Exprimées sous forme de règles logique sur la séquence de construction

Ex : Message et opération des classes:

« last » pour préciser que l'action n'a pas été annulée

```
message(M) :-
  lastCreate(M,Message),
  lastCreate(_C,Class),
  lastCreate(_O,Operation),
  lastCreate(_L,LifeLine),
  lastAddReference(M,operation,_O),
  lastAddReference(M,target,_L),
  lastAddReference(_L,type,_C),
  not (lastAddReference(C,owned,O)).
```

Détection incrémentale par règles

- Les règles expriment un filtre sur des classes d'action (classes d'équivalence)
  - Ex : Création d'un message, Référence vers une opération

$\Rightarrow$ Re-détecter la règle uniquement lorsque la modification peut être filtrée par la règle (analyse statique des règles).

Classes d'équivalence	Message Rule
CreateAssociation	Pas de re-détection
SetReferenceTarget	**Re-détection**

Détection incrémentale par fragment

- Les règles visitent certains éléments (scope de la règle).

=> Lancer la détection de la règle en pré-affectant certaines variables

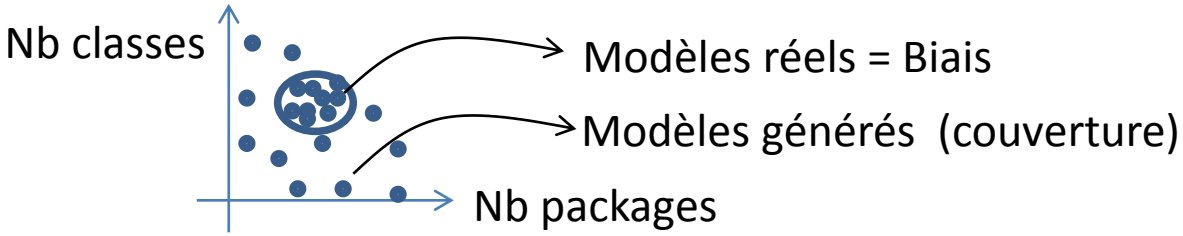
```
message(M,O,C,L) :-
  lastAddReference(M,operation,O),
  lastAddReference(M,target,L),
  lastAddReference(L,type,C),
  not (lastAddReference(C,owned,O)).
```

14. RemReference(c2,operation,o1)

```
message(M,o1,c2,L) .
```

Validation

- Expérimentation
  - Définition d'un générateur uniforme de modèles basé sur la méthode de Boltzmann [ECMFA09]



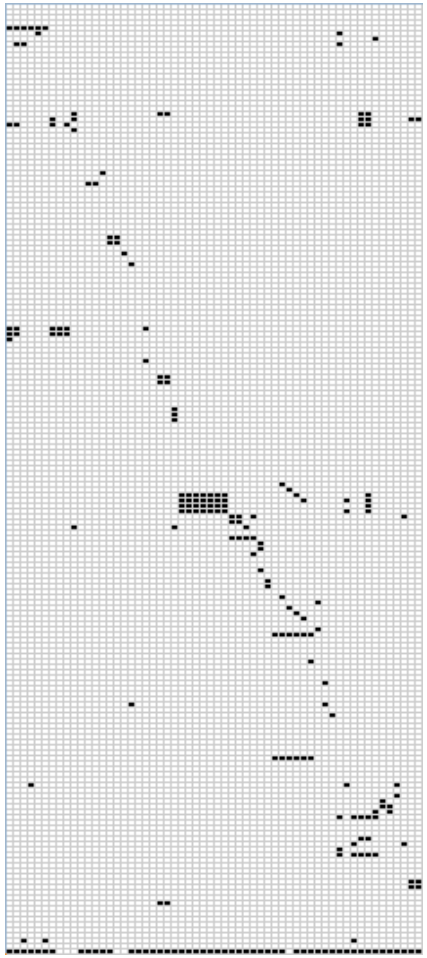
- Etude Empirique
  - Mise en œuvre de l'approche afin de mesurer quantitativement les gains qu'elle apporte
- Projets
  - IP ModelWare, IP Modelplex, ANR Movida, ANR Galaxy

Format basé sur les actions [ICSE08]

- Les évènements de l'éditeur UML sont capturés pour construire la séquence d'actions sous forme d'une base de faits Prolog
- Les règles de cohérence sont des règles Prolog
- A chaque fois que le modèle est changé, la détection est réalisée

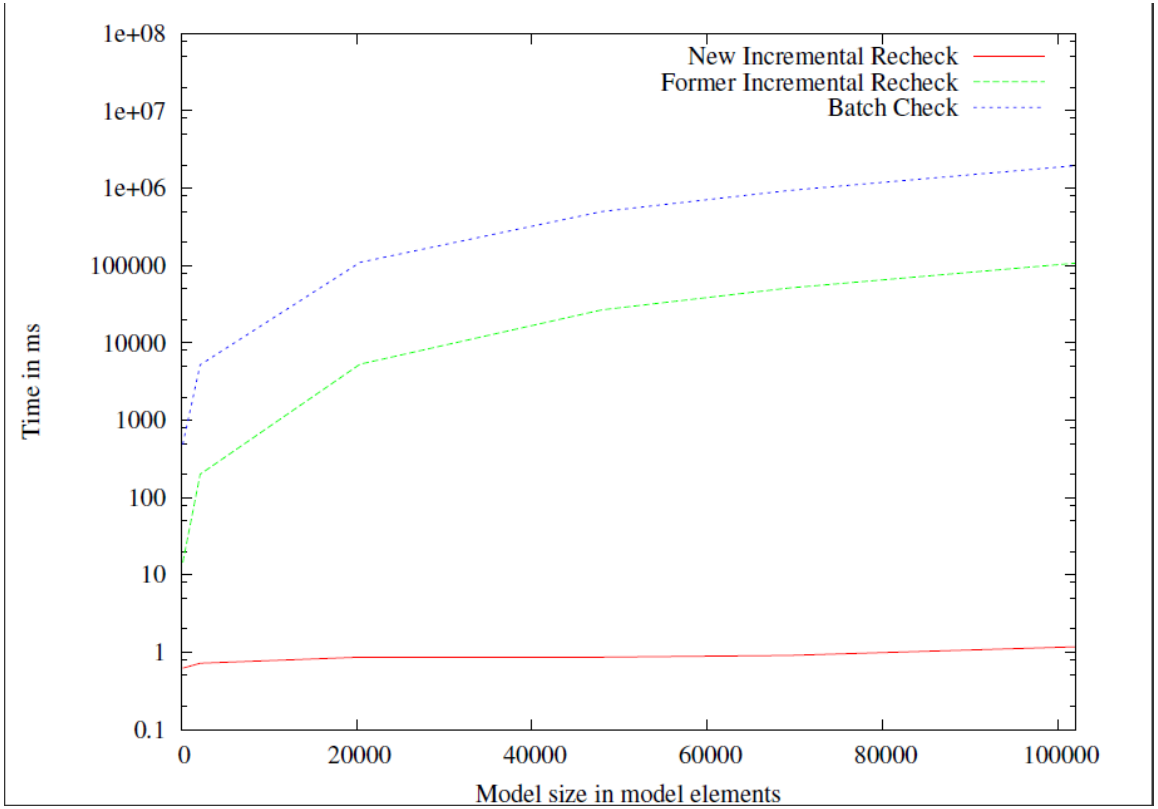
Incrément par règle [CAiSE09]

- 58 règles de cohérence
- 55 MetaClasses
- 177 classes d'équivalence
- Les règles filtrent de 1 à 11 classes
- 66.1% des classes d'équivalence sont sans effet
- 3 règles avec une complexité quadratique



Incrémentale par fragment [?10]

- Le problème n'est pas le même car on ne cherche qu'une incohérence (scope réduit)



Conclusion

- Praxis propose un langage pour la description des règles structurelles et méthodologiques.
  - Collaboration : Université de Mons, VUB, LIFL, IRISA, Thales, Airbus
- Ce langage a des propriétés intéressantes pour la détection incrémentale.
  - Validation des performances (comparable à Eyged)
- L'approche est formalisée (Alloy)
- Thèses G. De Fombelle, A. Mougenot, M. Almeida

http://meta.lip6.fr/

Perspectives

- Répartition
  - Comment répartir les modèles ?
  - Comment répartir la détection ?
  - *Approche P2P [DAIS09]*
- Résolution des cohérences
  - Comment aider le développeur à résoudre les incohérences?
  - Comment prévoir l'évolution ?
  - *Approche par planification [CAiSE10]*
- Cohérences méthodologiques
  - Comment mesurer la divergence des actions réalisées par rapport à la description du procédé ?
  - Comment mesurer l'effet de l'approche lors d'un développement ?
  - *Modélisation du procédé [TSE10]*