A propos de Rétro-ingénierie

composants logiciels, services

Pascal ANDRE

AELOS / LINA – UMR CNRS 6241
{Firstname.Lastname}@univ-nantes.fr

Exposés AeLoS

Plan de l'exposé

Plan

Introduction

Contexte : Vérification, documentation et alignement

- Vérification avec des modèles à composants (et services) : Kmelia...
- Documentation de code
- Alignement de processus métiers avec des architectures

Objectifs : Abstraction, restructuration, vérification

- extraire des architectures logicielles
- extraire des modèles pour vérifier des propriétés structurelles ou dynamiques
- restructurer les applications, faire migrer

Autres motivations : analyse, qualité et métriques

- extraire des modèles, des patterns
- mesurer des critères qualité
- comparer des modèles

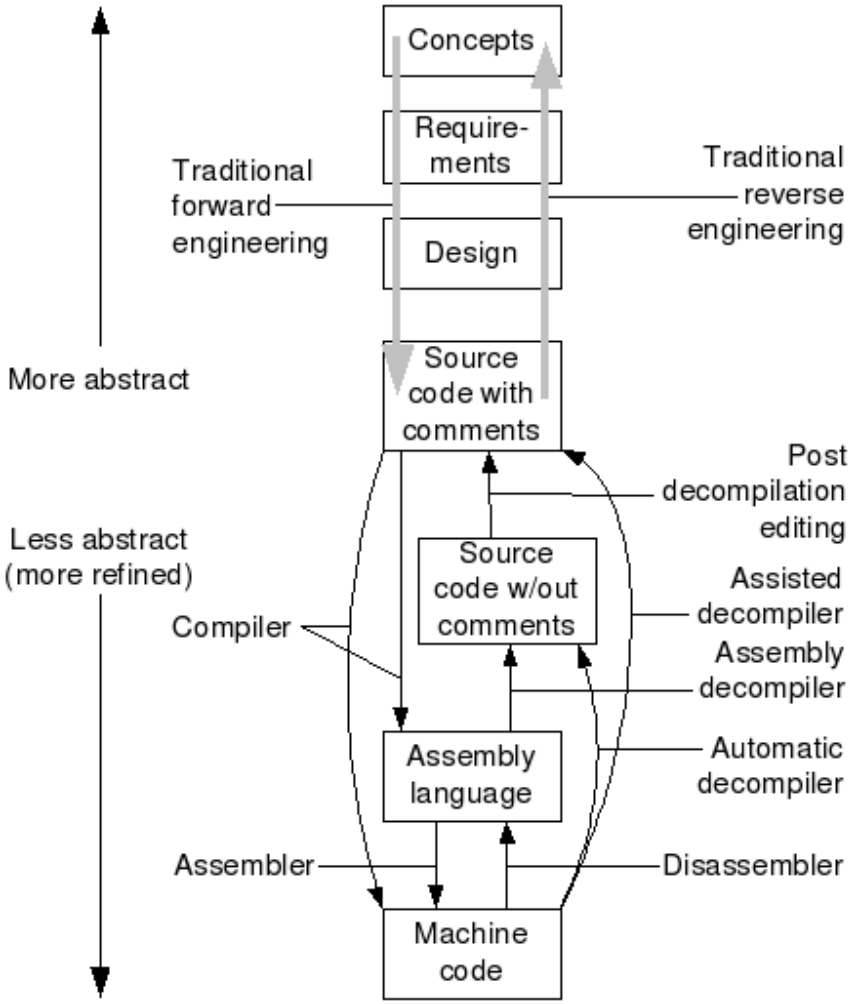
Rétro-ingénierie

Généralités

- Problématique vaste
  - décompilation / extraction / slicing / abstraction
  - analyse de programmes / vérification / mesure
  - documentation / restructuration / optimisation

WCRE, CSMR, ...

Décompilation/Rétro-ingénierie



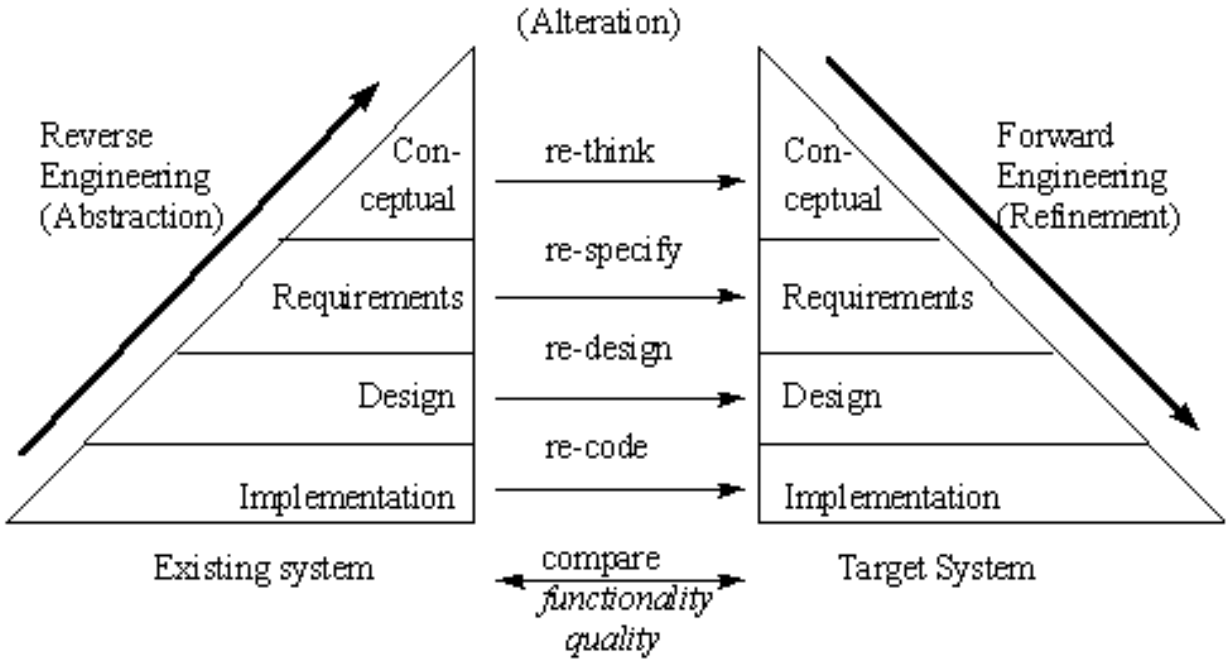
http://www.program-transformation.org/Transform/DecompilationAndReverseEngineering

Introduction

Généralités

- Problématique
  - décompilation / extraction / slicing / abstraction
  - analyse de programmes / vérification / mesure
  - documentation / restructuration / optimisation
- Rétro-ingénierie / rétro-conception
  - ? quelle base de départ (source, doc, modèles, tests...)
  - ? méthode d'ingénierie identifiable
  - ? un ou plusieurs raffinements (séquentiel, parallèle)
  - ? mélange d'aspects +/- orthogonaux
  - ? métamodèles

Ingénierie/Rétro-ingénierie



http://www.program-transformation.org/Transform/DecompilationAndReverseEngineering

Liaison

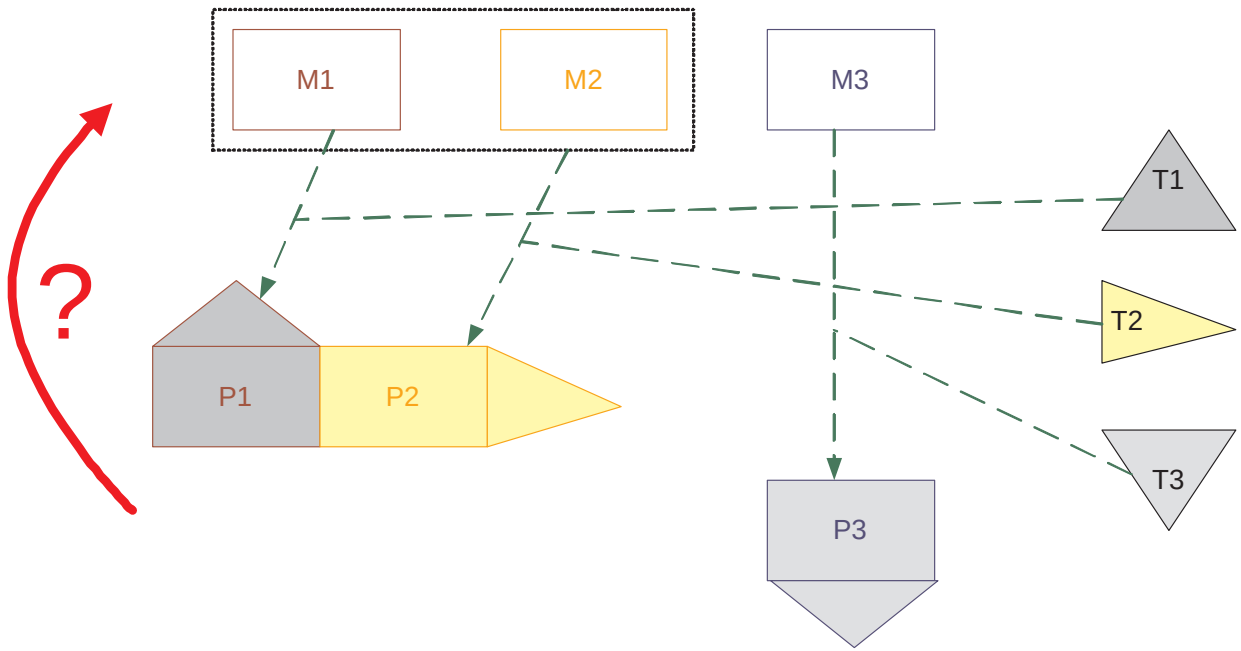
- Unidirectionnel / bidirectionnel
- Avec/sans information, traçabilité
- Roundtrip et co-évolution

Introduction

Généralités

- Problématique
  - décompilation / extraction / slicing / abstraction
  - analyse de programmes / vérification / mesure
  - documentation / restructuration / optimisation
- Rétro-ingénierie / rétro-conception
  - ? quelle base de départ (source, doc, modèles, tests...)
  - ? méthode d'ingénierie identifiable
  - ? un ou plusieurs raffinements (séquentiel, parallèle)
  - ? mélange d'aspects +/- orthogonaux
  - ? métamodèles
- Hétérogénéité / Interopérabilité
  - ? approche : procédural, objet, service...
  - ? langages et concepts : C, C++, Java, Smalltalk...
  - ? mise en œuvre : ad-hoc, guides, patterns, transformation...
  - ? frameworks : persistance, distribution, communication...

Hétérogénéité/Interopérabilité



`http://www.program-transformation.org/Transform/DecompilationAndReverseEngineering` $\Longrightarrow$ ingénierie (cf exposé `gencode.pdf`)

Introduction

Généralités

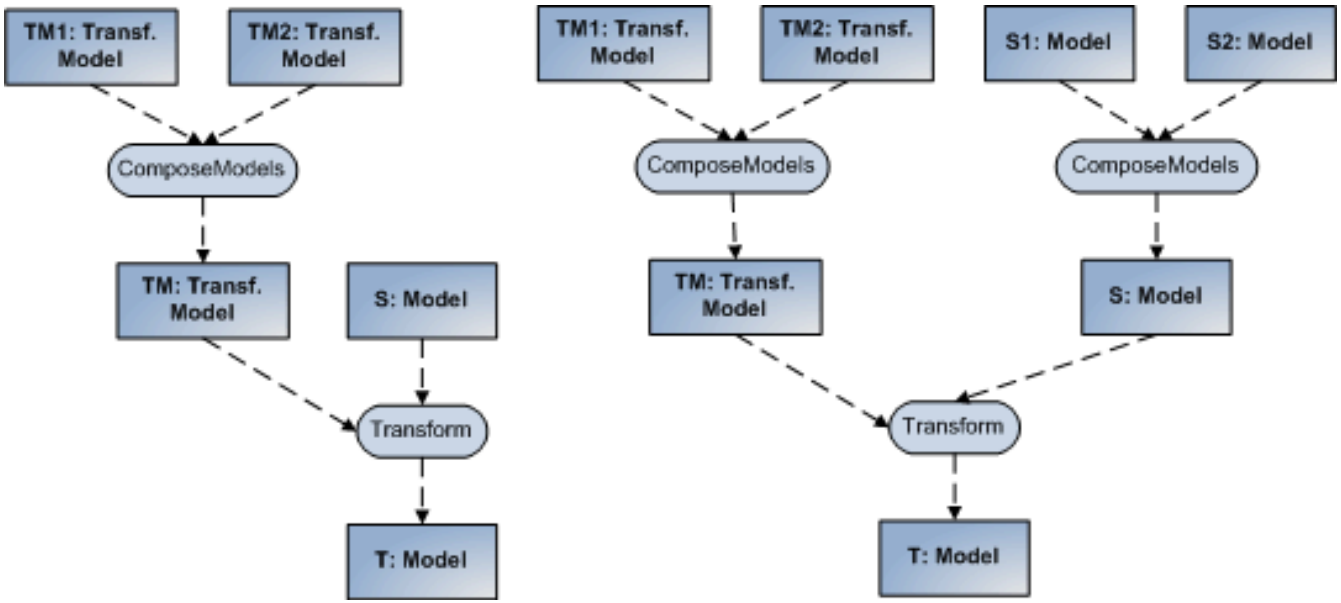
- Problématique
- Rétro-ingénierie / rétro-conception
- Hétérogénéité

Thèmes connexes

- IDM
- Traçabilité
- Evolution

IDM

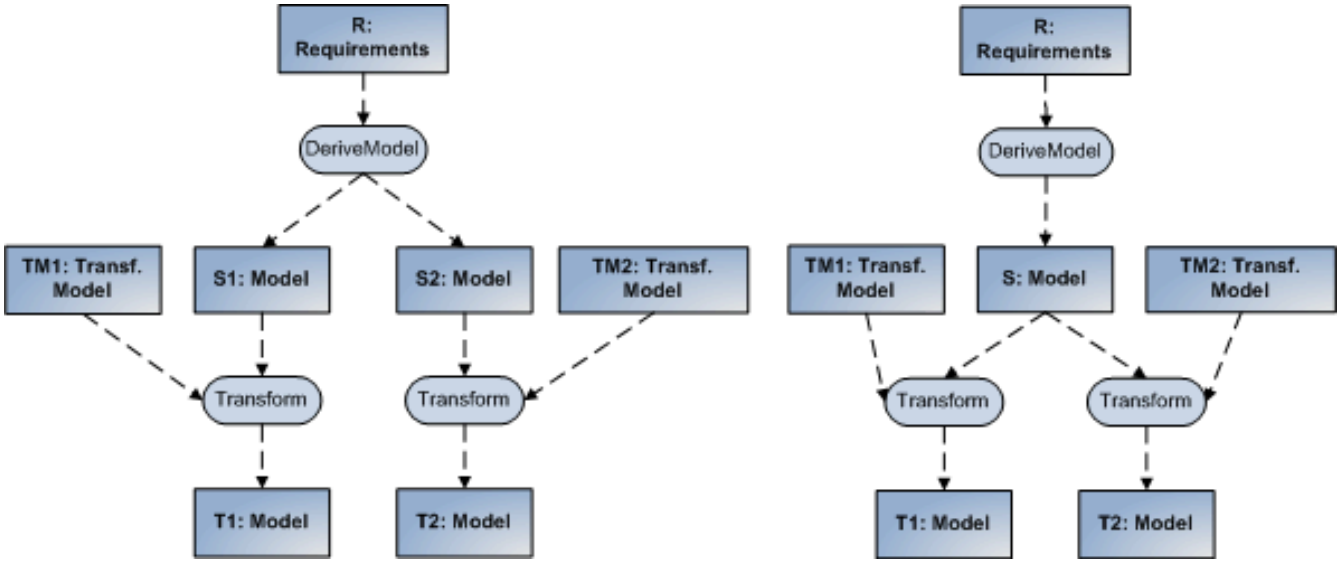
Fusion



http://www.theenterprisearchitect.eu/archive/2008/03/14/model_driven_engineering

IDM

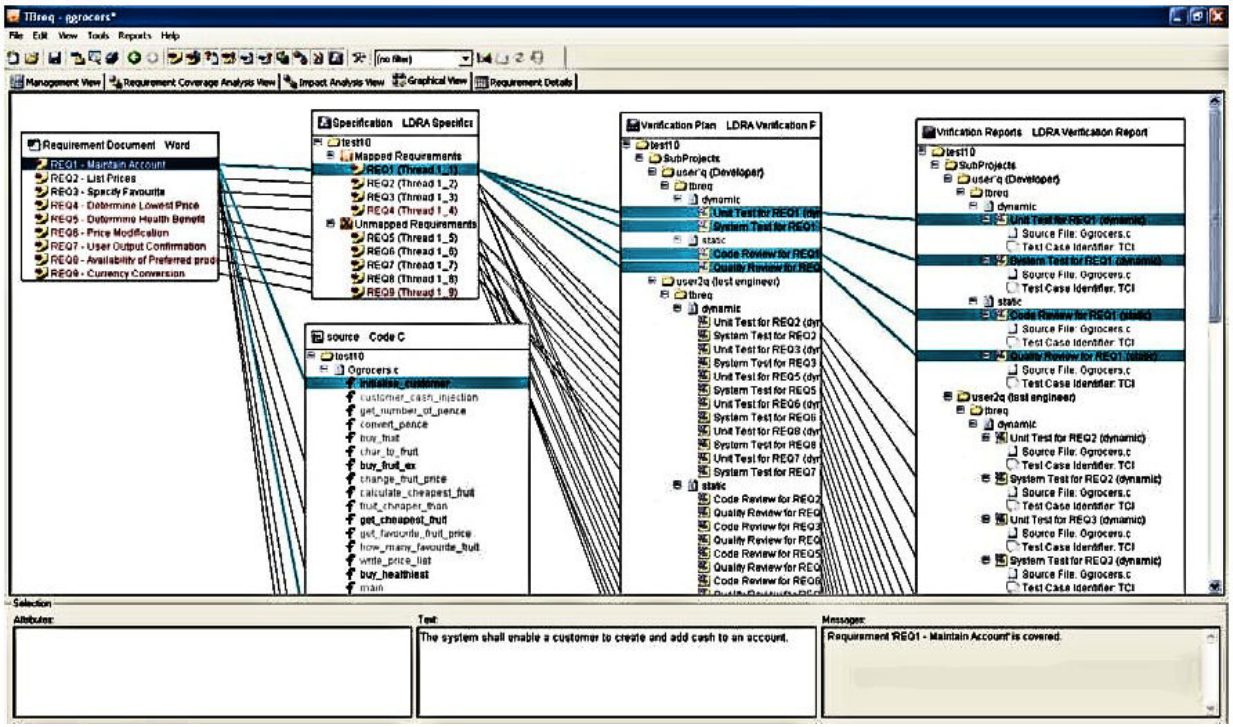
Décomposition



http://www.theenterprisearchitect.eu/archive/2008/03/14/model_driven_engineering

Rétro-ingénierie et métamodélisation [Favre and Musset, 2006]

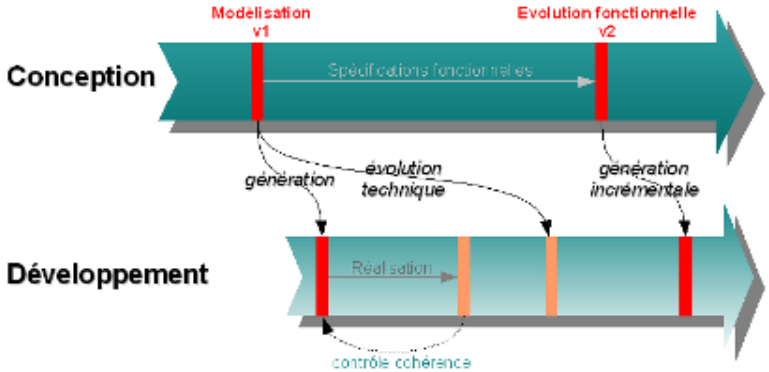
Traçabilité



LDRA TBreq montre ici la traçabilité entre les besoins, la conception le code source et la vérification finale

`http://www.techonlineindia.com/article/11-09-14/cwe_what_developers_of_connected_embedded_systems_need_to_know.aspx`

Evolution



CreateAccountAction.java | Weblog.xmi

```
public ActionForward create(ActionMapping mapping, ActionForm form, HttpS
    log.debug("Starting");
    String returnCode = PAGE SELF;
The generated code shouldn't be modified, move the text between the user tagseateAccountForm) form;

    String myString = new String();

    //Start of user code method create

    System.out.println(myString + createAccountForm);
    // TODO Write here the action code for create

    //End of user code
    return mapping.findForward(returnCode);
}
```

Erreurs | Javadoc | Déclaration | Console | Journal des erreurs

1 erreur, 30 avertissement(s), 0 info(s) (Le filtre a détecté 31 sur 32 éléments)

Description	Ressource	Dans le dossier
The generated code shouldn't be modified, move the text between the user tags	CreateAcc...	WebLog/JavaSource/fr/

Obeo Traceability `http://www.obeo.fr/pages/obeo-traceability/`

Introduction

Généralités

- Problématique
- Rétro-ingénierie / rétro-conception
- Hétérogénéité

Thèmes connexes

- IDM
- Traçabilité
- Evolution

Problème complexe

$\Longrightarrow$ ciblage

Introduction

Généralités

- Problématique
- Rétro-ingénierie / rétro-conception
- Hétérogénéité

Thèmes connexes

- IDM
- Traçabilité
- Evolution

Problème complexe

$\Longrightarrow$ ciblage

Introduction / focus

Trois situations

Documentation

Java $\longrightarrow$ UML
DEA Sami Dakka 2004

Vérification

Java $\longrightarrow$ CBSE
projet Econet 2007-2008

Urbanisation, Alignement

code $\longrightarrow$ BPM
Thèse Jonathan Pépin 2013

Discussion sans prétention, échange

Introduction / focus

Trois situations

Documentation

Java $\longrightarrow$ UML
DEA Sami Dakka 2004

Vérification

Java $\longrightarrow$ CBSE
projet Econet 2007-2008

Urbanisation, Alignement

code $\longrightarrow$ BPM
Thèse Jonathan Pépin 2013

Discussion sans prétention, échange

Introduction / focus

Trois situations

Documentation

Java $\longrightarrow$ UML
DEA Sami Dakka 2004

Vérification

Java $\longrightarrow$ CBSE
projet Econet 2007-2008

Urbanisation, Alignement

code $\longrightarrow$ BPM
Thèse Jonathan Pépin 2013

Discussion sans prétention, échange

Introduction / focus

Trois situations

Documentation

Java $\longrightarrow$ UML
DEA Sami Dakka 2004

Vérification

Java $\longrightarrow$ CBSE
projet Econet 2007-2008

Urbanisation, Alignement

code $\longrightarrow$ BPM
Thèse Jonathan Pépin 2013

Discussion sans prétention, échange

Plan

Documentation

Java ⟶ UML

Rétro-ingénierie vers UML

- Domaine exploré : ingénierie, rétro-ingénierie [Kollmann et al., 2002]
- Modèle relationnel ⟶ DC UML (Hibernate, ...)
  la proximité de modèles facilite les transformations
- Java ⟶ DC UML
  par défaut on documente la structure du code problème de la représentation visuelle
- Java ⟶ DS UML (message sequence charts)
  extraction difficile dans l'absolu problème de la représentation visuelle
- Java ⟶ DET UML (statecharts)
  extraction depuis MSC [Maier and Zündorf, 2003], ...,
  travaux sur le model-checking [Parizek et al., 2006]

De nombreux outils Rational, Fujaba, Obeo Designer, Modelio...
`http://en.wikipedia.org/wiki/List_of_UML_tools`

Expérimentation

Fujaba

Comparaison d'outils

Fujaba, ArgoUML, Rational, ...

Rétro-ingénierie vers UML DC

- séparation types primitifs / classes
- séparation classes utilitaires / classes applicatives
- patterns
  - associations (agrégation, composition ?)
  - interfaces et héritage multiple
  - classe-associations

Expérimentations

peu convaincantes ... Voir aussi BlueJ, StarUML, BoUML, ...

Expérimentation

Fujaba

Comparaison d'outils

Fujaba, ArgoUML, Rational, ...

Rétro-ingénierie vers UML DC

- séparation types primitifs / classes
- séparation classes utilitaires / classes applicatives
- patterns
  - associations (agrégation, composition ?)
  - interfaces et héritage multiple
  - classe-associations

Expérimentations

peu convaincantes ... Voir aussi BlueJ, StarUML, BoUML, ...

Expérimentation

Fujaba

Comparaison d'outils

Fujaba, ArgoUML, Rational, ...

Rétro-ingénierie vers UML DC

- séparation types primitifs / classes
- séparation classes utilitaires / classes applicatives
- patterns
  - associations (agrégation, composition ?)
  - interfaces et héritage multiple
  - classe-associations

Expérimentations

peu convaincantes ... Voir aussi BlueJ, StarUML, BoUML, ...

Expérimentation

Fujaba

Comparaison d'outils

Fujaba, ArgoUML, Rational, ...

Rétro-ingénierie vers UML DC

- séparation types primitifs / classes
- séparation classes utilitaires / classes applicatives
- patterns
  - associations (agrégation, composition ?)
  - interfaces et héritage multiple
  - classe-associations

Expérimentations

peu convaincantes ... Voir aussi BlueJ, StarUML, BoUML, ...

Rétro-conception Java $\longrightarrow$ UML

Le problème reste ouvert

- général / spécifique
- structure / comportement
- faible abstraction

Ingénierie/Frameworks/Patterns/Information disponible

- Plain Java / Enriched Java
  - infrastructures : EJB, CORBA, SOAP...
  - Java support : RMI, JMS
  - Java basic : Class, threads
- Abstraction
  - Composition
  - Généralisation
  - Masquage
- Annotation
  - traçabilité
  - round-trip
  - évolution

Rétro-conception Java $\longrightarrow$ UML

Le problème reste ouvert

- général / spécifique
- structure / comportement
- faible abstraction

Ingénierie/Frameworks/Patterns/Information disponible

- Plain Java / Enriched Java
  - infrastructures : EJB, CORBA, SOAP...
  - Java support : RMI, JMS
  - Java basic : Class, threads
- Abstraction
  - Composition
  - Généralisation
  - Masquage
- Annotation
  - traçabilité
  - round-trip
  - évolution

Bilan

Idées

Etude à relancer...

Comparaison d'outils récents

AGL MDA ...

Evolution des techniques et outils

MDA et profils UML

Proposition étagée par les profils

UML Java — JMS, EJB, SOA
UML Tech
UML Appli
UML Business
...

Bilan

Idées

Etude à relancer...

Comparaison d'outils récents

AGL MDA ...

Evolution des techniques et outils

MDA et profils UML

Proposition étagée par les profils

UML Java

UML Tech

UML Appli

UML Business

...

JMS, EJB, SOA

Bilan

Idées

Etude à relancer...

Comparaison d'outils récents

AGL MDA ...

Evolution des techniques et outils

MDA et profils UML

Proposition étagée par les profils

UML Java — JMS, EJB, SOA
UML Tech
UML Appli
UML Business
...

Bilan

Idées

Etude à relancer...

Comparaison d'outils récents

AGL MDA ...

Evolution des techniques et outils

MDA et profils UML

Proposition étagée par les profils

UML Java JMS, EJB, SOA
UML Tech
UML Appli
UML Business
...

Bilan

Idées

Etude à relancer...

Comparaison d'outils récents

AGL MDA ...

Evolution des techniques et outils

MDA et profils UML

Proposition étagée par les profils

UML Java — JMS, EJB, SOA
UML Tech
UML Appli
UML Business
...

Plan

Vérification de modèles à composants

A partir du code

Vérifier des programmes — Java

- trop complexe
- trop de préoccupations croisées
- trop "spécifique"

Vérifier des modèles — composants et services

- \+ outillage
- \- pas universel
- \- raffinement

Lien modèles programmes — métamodèles et transformations

- rétro-ingénierie (abstraction)
- alignement des deux "modèles", annotation de Java (JML)
- raffinement, génération (preuve)

Vérification de modèles à composants

A partir du code

Vérifier des programmes — Java

- trop complexe
- trop de préoccupations croisées
- trop "spécifique"

Vérifier des modèles — composants et services

+ outillage
- pas universel
- raffinement

Lien modèles programmes — métamodèles et transformations

- rétro-ingénierie (abstraction)
- alignement des deux "modèles", annotation de Java (JML)
- raffinement, génération (preuve)

Vérification de modèles à composants

A partir du code

Vérifier des programmes — Java

- trop complexe
- trop de préoccupations croisées
- trop "spécifique"

Vérifier des modèles — composants et services

+ outillage
- pas universel
- raffinement

Lien modèles programmes — métamodèles et transformations

- rétro-ingénierie (abstraction)
- alignement des deux "modèles", annotation de Java (JML)
- raffinement, génération (preuve)

Concepts à l'exécution

Paradigmes à l'exécution (traçabilité) :

- Données & actions
  - types de base, fonctions
  - types utilisateur, fonctions et méthodes
  - expressions de base
  - communications et appels de service
- composants et services
  - composant = structure + process (thread, runnable)
  - service offert = LTS + process
  - service requis = passerelle
- assemblage
  - canaux & synchronisation
  - mappings de données et messages
  - encapsulation
- Composites
  - Promotion
  - Héritage

Vérification

Econet

Séparer structure et comportement

- concepts différentes
- techniques différentes
- évolution différente

Contexte

- plain java
- plusieurs cibles (SOFA [Bures et al., 2006], Kmelia [André et al., 2010], KADL [Pavel et al., 2005], FRACTAL [Bruneton et al., 2006])
  - IDM

Architecture

- CoCoME
- spécification par modèles UML[Rausch et al., 2008] (composants)
- une implantation Java type
- implantation d'autres modèles (Sofa, rCOS, Fractal...)

Vérification

Econet

Séparer structure et comportement

- concepts différentes
- techniques différentes
- évolution différente

Contexte

- plain java
- plusieurs cibles (SOFA [Bures et al., 2006], Kmelia [André et al., 2010], KADL [Pavel et al., 2005], FRACTAL [Bruneton et al., 2006])
  - IDM

Architecture

- CoCoME
- spécification par modèles UML[Rausch et al., 2008] (composants)
- une implantation Java type
- implantation d'autres modèles (Sofa, rCOS, Fractal...)

Vérification

Econet

Séparer structure et comportement

- concepts différentes
- techniques différentes
- évolution différente

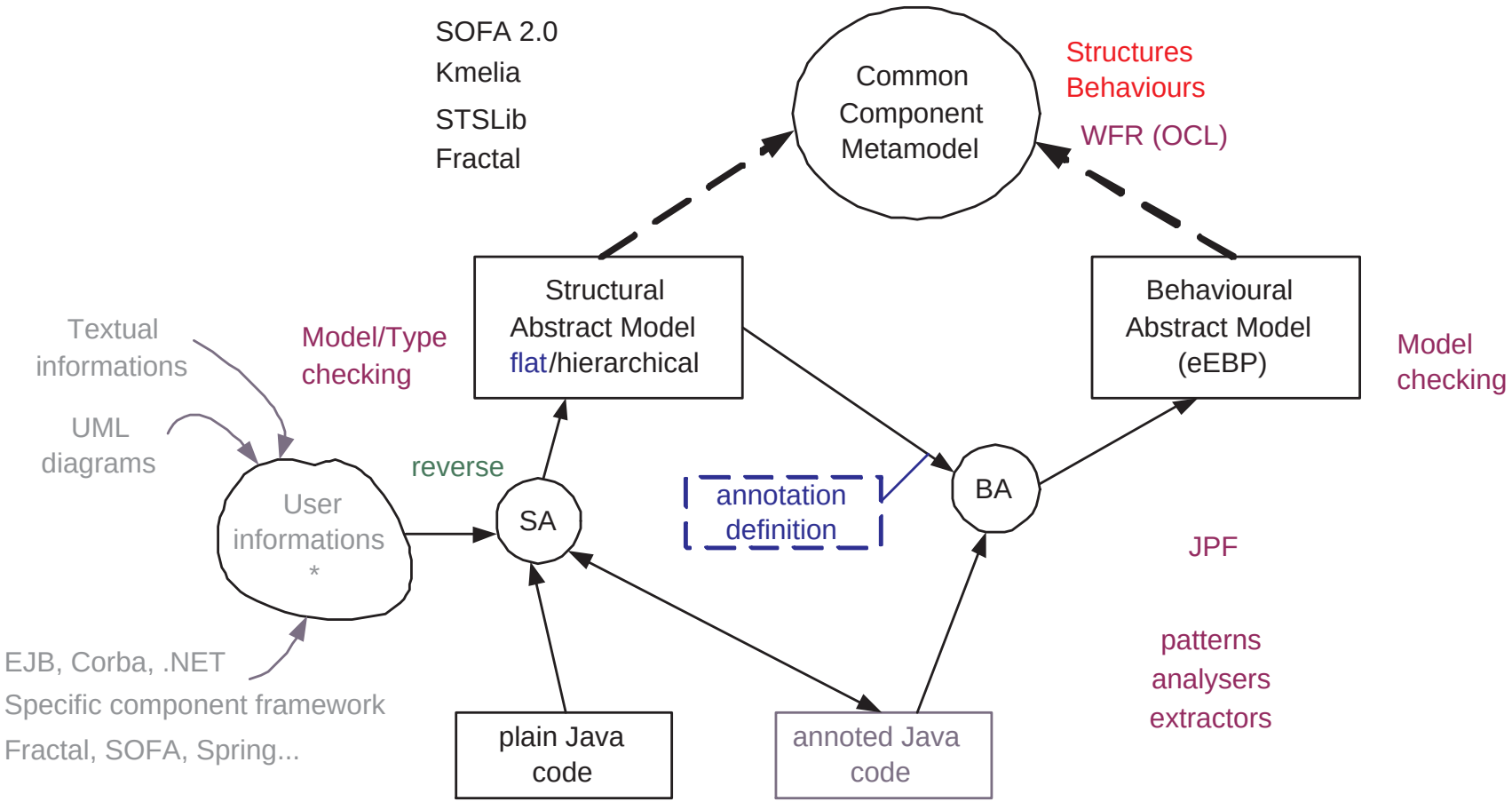
Contexte

- plain java
- plusieurs cibles (SOFA [Bures et al., 2006], Kmelia [André et al., 2010], KADL [Pavel et al., 2005], FRACTAL [Bruneton et al., 2006])
  - IDM

Architecture

- CoCoME
- spécification par modèles UML[Rausch et al., 2008] (composants)
- une implantation Java type
- implantation d'autres modèles (Sofa, rCOS, Fractal...)

Architecture de la proposition



www.lina.sciences.univ-nantes.fr/aelos/projects/econet/

Structure

OO $\longrightarrow$ CBSE

Plusieurs approches pour identifier une structure à composants

- Alignement de deux modèles (architecture / implantation)
  - analyse des noms et des structures, y compris de paquetages
  - identificationde patterns composants (interface/classes abstraites/classes concrètes)
  - modèles complémentaires (fractal) ou raffinement

  besoin d'implantation systématique (pas d'exception ou de "raccourcis")
- Extraction d'architecture à composants
  - A partir d'une rétro-ingénierie de classes UML
    pas d'intérêt majeur
  - Par clusterisation à partir de métriques (qualité, cohésion, couplage)
    pas simple de fixer les seuils et de détecter les composites, dépend souvent de la structure et non des communications
  - Par identification (système à base de règles)
    solution choisie ici

Autres approches : component recovery (code C ou cobol...)
Facteurs influents : style architectural, framework, pattern

Extraction de composants

Java $\longrightarrow$ CBSE

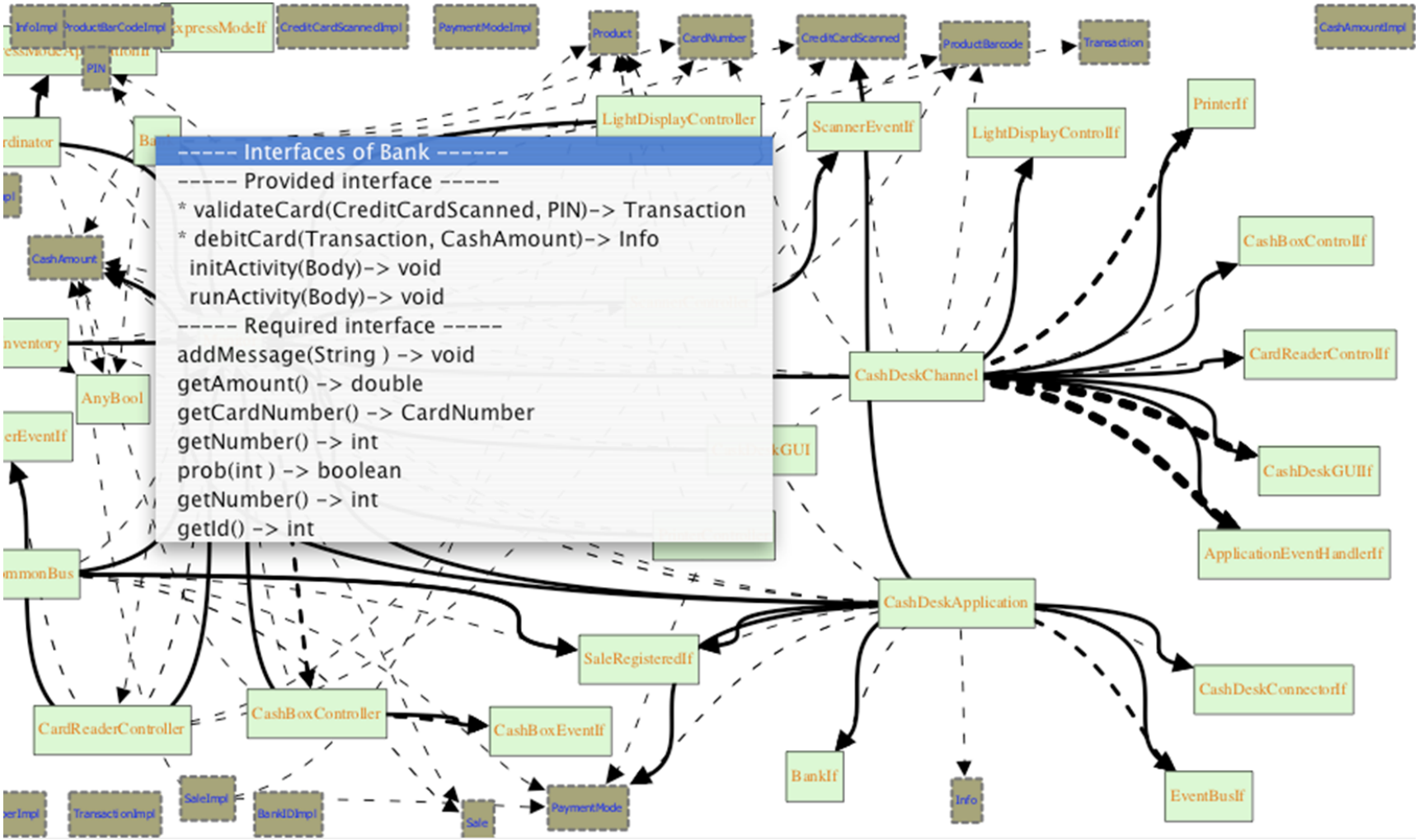
- Séparer composants et type de données
- Pas d'hypothèses sur la structure du code source (mais une programmation "composant" donne de meilleurs résultats)
- inférence des communications (canaux)
- les composants sont des classes
- définition de règles d'extraction (*find the components*) plus empirique que statistique
- heuristiques : propriétés architecturales (de bonne "componentisation")
  - contexte du projet
  - intégrité de communication (préserve encapsulation, canaux)
  - analyse des limites (communications "directes")

L'outil proposé (basé sur Eclipse JDT) extrait :

- composants et types de données
- services offerts et requis
- composites
- canaux de communication

Modèle influent : ArchJava

Architecture



www.lina.sciences.univ-nantes.fr/aelos/projects/econet/ [Anquetil et al., 2009]

Comportement

Java $\longrightarrow$ CBSE

Rétro-ingénierie vers BP ou LTS

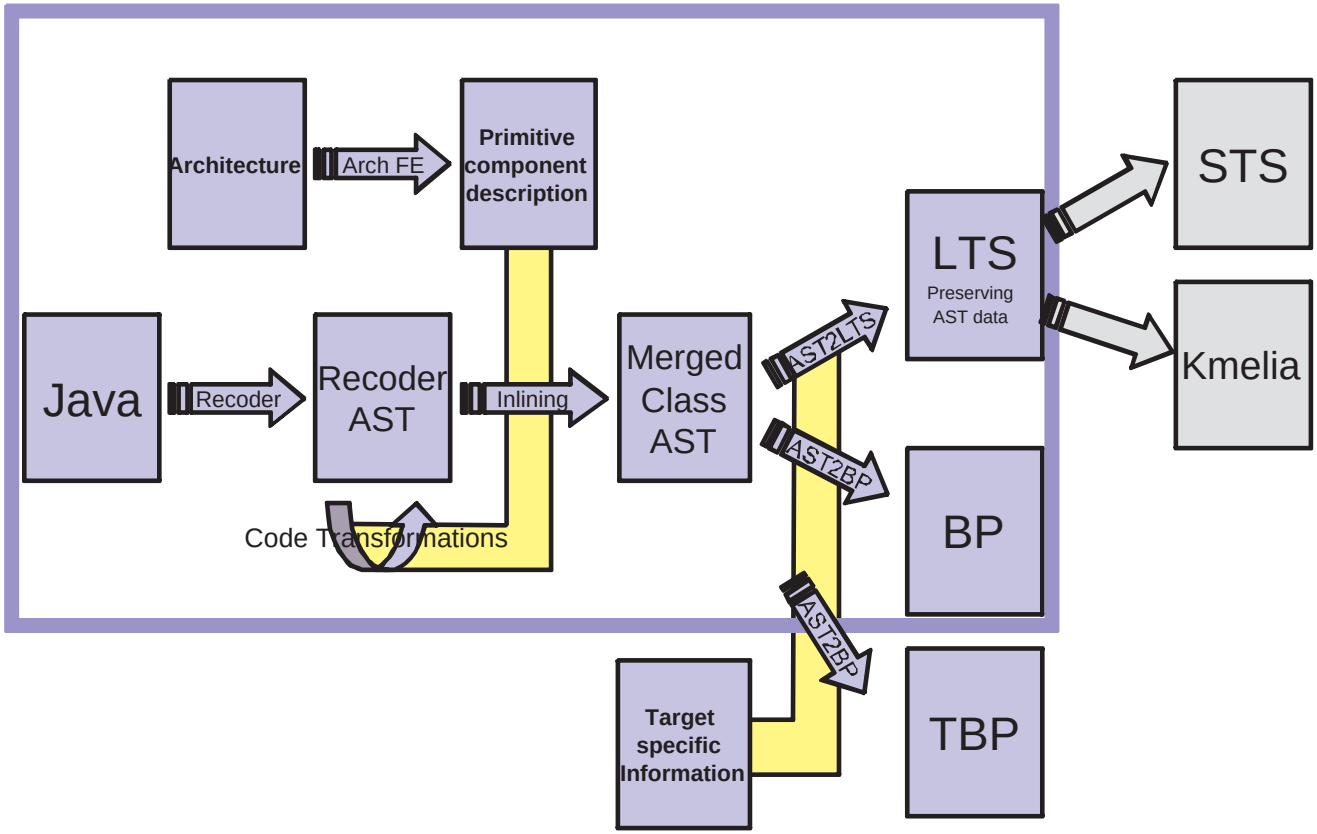
- Code source Java
- Annotations "composants" du code, y compris des points d'entrée
- réaction aux envois de messages (transformations AST)
- détection de threads autonomes
- conversion (si nécessaire) des BP en LTS

Technique complémentaire : *slicing* selon la structure composants pour éliminer des parties inutiles dans les protocoles

Détails dans [Poch and Plasil, 2009]

Autres approches : Bandera Program and interface slicing for reverse engineering [Beck and Eichmann, 1993]

Architecture de l'extraction de comportements



http://d3s.mff.cuni.cz/~poch/jabstractor.html [Poch and Plasil, 2009]

Métamodèle

Expérimentations

Métamodèle à composants

- Common Meta Model Definition
- API subset
- OCL WFR
- extensions Java

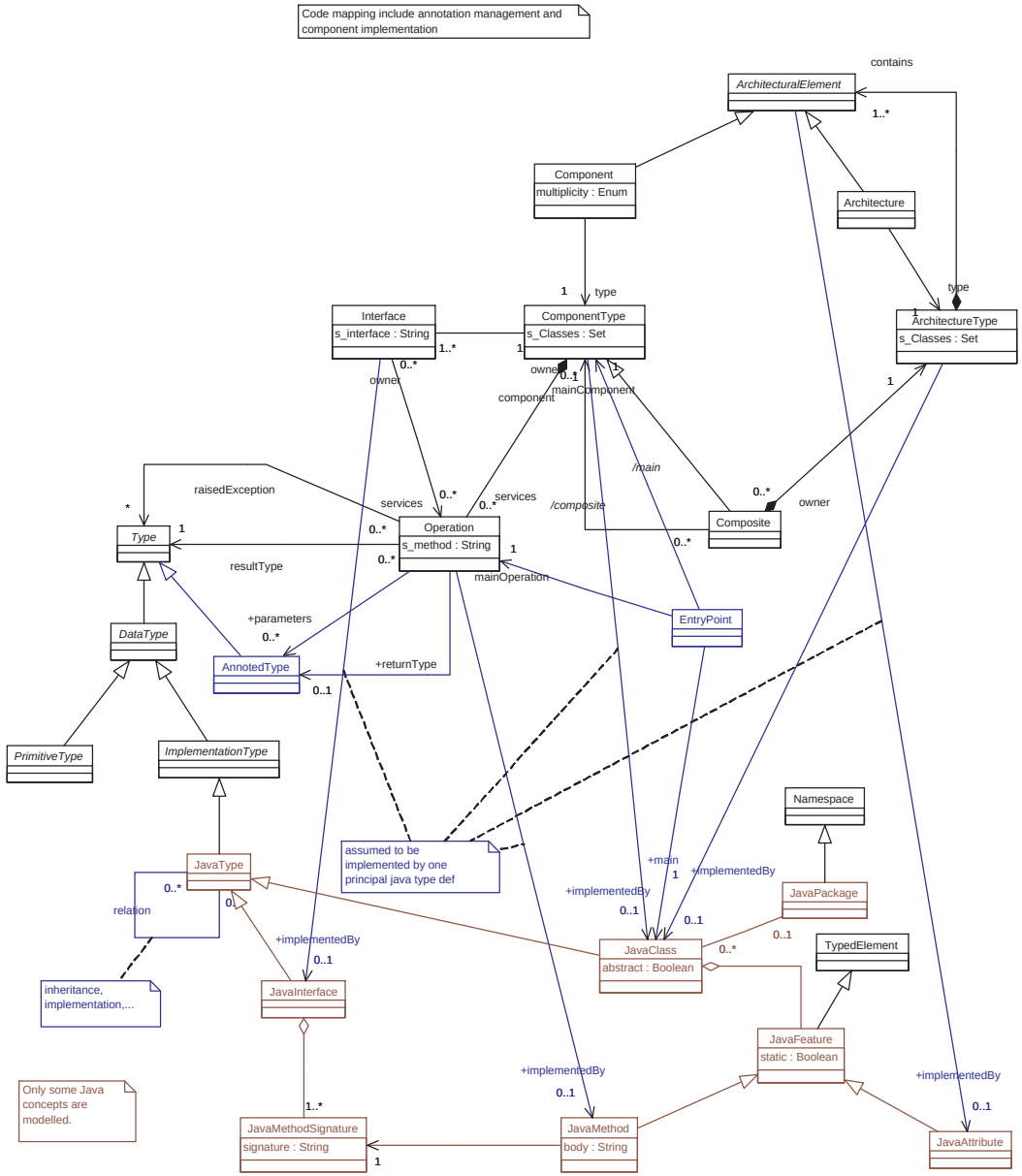
Implantations

- OCLE pour la vérification de contraites `http://lci.cs.ubbcluj.ro/ocle/`
- EMF pour la génération de code
- oAW pour avoir un référentiel flexible des templates de génération de code

- Vérification cohérence des métamodèles
- Génération de code (via EMF)
- Test sur une partie du CoCoME

Plus sur OCLE [Chiorean et al., 2008, Chiorean et al., 2011]

Architecture



Bilan

Idées

- Support métamodèles
- Stratégie à base de règles paramétrique
- Complémenter par les approches à métriques
- Besoin de critères de conformité

Etude à poursuivre...

Comparaison d'outils récents

Composants UML ?

Evolution des techniques et outils de génération de code

MDA (séparation applicatif/technique) ? profils UML ?

Proposition étagée par les profils

UML Java	JMS, EJB, SOA
UML Composant	
UML Applicatif	
UML Métier	

Bilan

Idées

- Support métamodèles
- Stratégie à base de règles paramétrique
- Complémenter par les approches à métriques
- Besoin de critères de conformité

Etude à poursuivre...

Comparaison d'outils récents

Composants UML ?

Evolution des techniques et outils de génération de code

MDA (séparation applicatif/technique) ? profils UML ?

Proposition étagée par les profils

UML Java	JMS, EJB, SOA
UML Composant	
UML Applicatif	
UML Métier	

Bilan

Idées

- Support métamodèles
- Stratégie à base de règles paramétrique
- Complémenter par les approches à métriques
- Besoin de critères de conformité

Etude à poursuivre...

Comparaison d'outils récents

Composants UML ?

Evolution des techniques et outils de génération de code

MDA (séparation applicatif/technique) ? profils UML ?

Proposition étagée par les profils

UML Java — JMS, EJB, SOA
UML Composant
UML Applicatif
UML Métier

Bilan

Idées

- Support métamodèles
- Stratégie à base de règles paramétrique
- Complémenter par les approches à métriques
- Besoin de critères de conformité

Etude à poursuivre...

Comparaison d'outils récents

Composants UML ?

Evolution des techniques et outils de génération de code

MDA (séparation applicatif/technique) ? profils UML ?

Proposition étagée par les profils

UML Java	JMS, EJB, SOA
UML Composant	
UML Applicatif	
UML Métier	

Plan

Alignement métier - applications

Urbanisation

Co-évolution

- processus métier
- systèmes supports (applications)

"l'alignement du Systèmes d'Information (SI) avec la stratégie représente une des principales préoccupations des Directions des SI depuis une dizaine d'années" [Thevenet, 2009]

Aligner des modèles

composants et services

+ mapping, traçabilité, évolution
- processus métier
- composants logiciels (code)

Connecter au bon niveau

rétro-ingénierie (abstraction)

- déploiement, binaire
- source
- architectures (différents nivaux)
- métiers

Alignement métier - applications

Urbanisation

Co-évolution

- processus métier
- systèmes supports (applications)

"l'alignement du Systèmes d'Information (SI) avec la stratégie représente une des principales préoccupations des Directions des SI depuis une dizaine d'années" [Thevenet, 2009]

Aligner des modèles

composants et services

+ mapping, traçabilité, évolution
- processus métier
- composants logiciels (code)

Connecter au bon niveau

rétro-ingénierie (abstraction)

- déploiement, binaire
- source
- architectures (différents nivaux)
- métiers

Alignement métier - applications

Urbanisation

Co-évolution

- processus métier
- systèmes supports (applications)

"l'alignement du Systèmes d'Information (SI) avec la stratégie représente une des principales préoccupations des Directions des SI depuis une dizaine d'années" [Thevenet, 2009]

Aligner des modèles

composants et services

+ mapping, traçabilité, évolution
- processus métier
- composants logiciels (code)

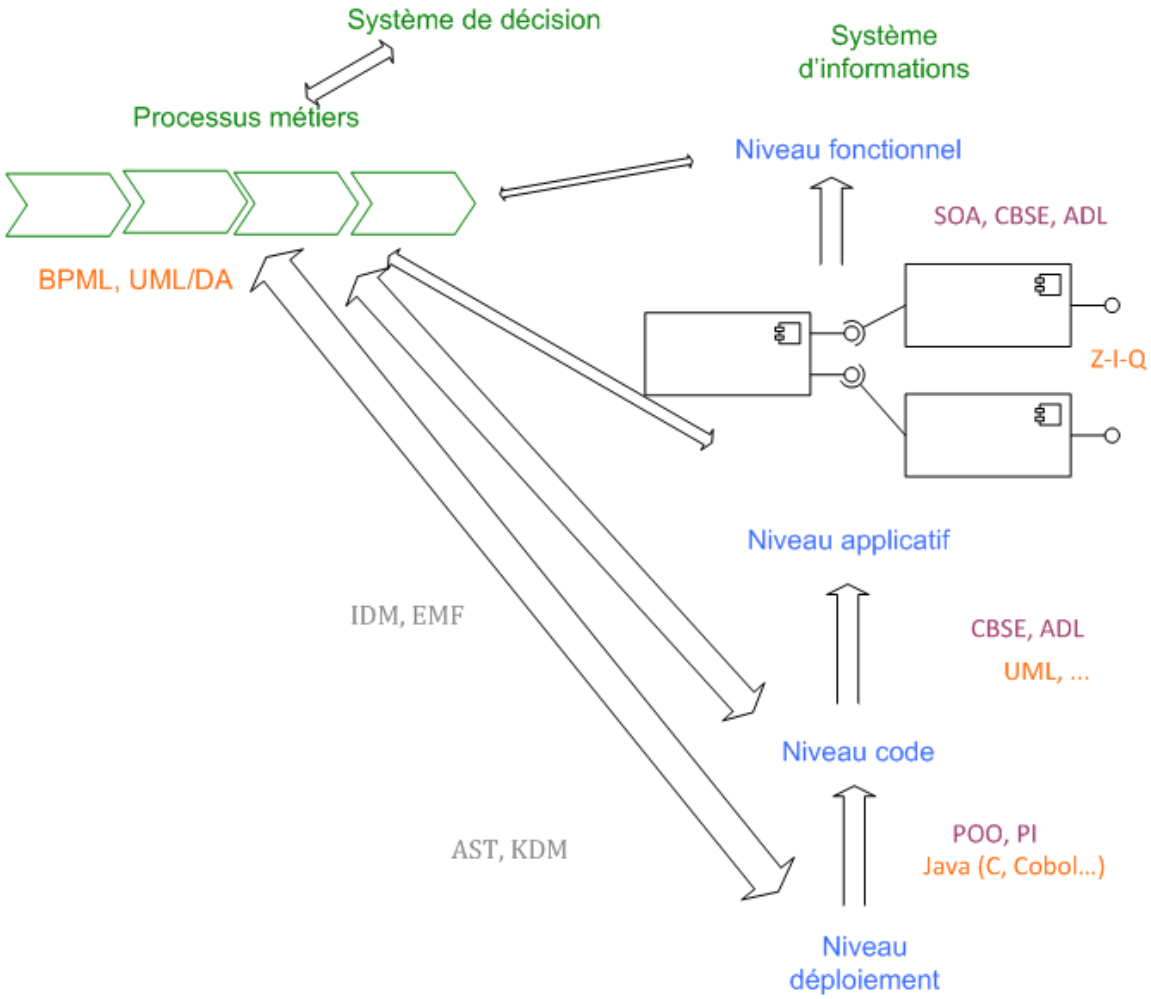
Connecter au bon niveau

rétro-ingénierie (abstraction)

- déploiement, binaire
- source
- architectures (différents nivaux)
- métiers

Alignement métier - applications

Niveaux d'alignement



source : JP

Alignement métier - applications

Approche

Cartographie à différents niveaux d'architecture et de modèles

- architecture fonctionnelle (SOA...)
- urbanisation (zone, îlots, quartiers)
- architectures composants, services
- packages, modules, sources
- déploiement, frameworks technique, nœuds physiques

Lien modèles programmes

- la structure prime (au départ)
- mapping, annotations (support alignement, traçabilité)
- séparation technique / applicatif
- procéder par "petites" abstractions
- environnement KDM, EMF

métamodèles et transformations

ça débute...

Alignement métier - applications

Approche

Cartographie à différents niveaux d'architecture et de modèles

- architecture fonctionnelle (SOA...)
- urbanisation (zone, îlots, quartiers)
- architectures composants, services
- packages, modules, sources
- déploiement, frameworks technique, nœuds physiques

Lien modèles programmes

métamodèles et transformations

- la structure prime (au départ)
- mapping, annotations (support alignement, traçabilité)
- séparation technique / applicatif
- procéder par "petites" abstractions
- environnement KDM, EMF

ça débute...

Alignement métier - applications

Approche

Cartographie à différents niveaux d'architecture et de modèles

- architecture fonctionnelle (SOA...)
- urbanisation (zone, îlots, quartiers)
- architectures composants, services
- packages, modules, sources
- déploiement, frameworks technique, nœuds physiques

Lien modèles programmes

métamodèles et transformations

- la structure prime (au départ)
- mapping, annotations (support alignement, traçabilité)
- séparation technique / applicatif
- procéder par "petites" abstractions
- environnement KDM, EMF

ça débute...

Plan

Conclusion

Résumé

Evolutions technologiques $\Longrightarrow$ rétro-ingénierie nécessaire

- domaine ancien
- techniques à usage large
- nombreuses facettes $\Longrightarrow$ techniques différentes (complémentaires ?)
- préoccupations transverses : évolution, traçabilité, qualité
- approche modèle est un plus

Trois applications

- documentation
- vérification de programmes
- alignement

Conclusion

Résumé

Evolutions technologiques $\Longrightarrow$ rétro-ingénierie nécessaire

- domaine ancien
- techniques à usage large
- nombreuses facettes $\Longrightarrow$ techniques différentes (complémentaires ?)
- préoccupations transverses : évolution, traçabilité, qualité
- approche modèle est un plus

Trois applications

- documentation
- vérification de programmes
- alignement

Conclusion

Retours

Quelques enseignements

- Patrimoine important $\Longrightarrow$ rétro-ingénierie indispensable
- Pas de solution miracle
- La proximité des concepts paie ex : BDR-UML
- Approche pragmatique
  - procéder par petit pas => nombreuses strates idem en ingénierie
  - séparer les aspects (technique/applicatif, structure/comportement)
  - rétro-conception si conception systématique - standard - génératif sinon analyse, évaluation
  - migration par modèle et non par code
  - aide souvent nécessaire (utilisateurs, référentiels...)
- Prendre en compte l'évolution

Références I


André, P., Ardourel, G., Attiogbé, C., and Lanoix, A. (2010).
Using assertions to enhance the correctness of kmelia components and their assemblies.
ENTCS, 263 :5 – 30.
Proceedings of FACS 2009.


Anquetil, N., Royer, J.-C., Andre, P., Ardourel, G., Hnetynka, P., Poch, T., Petrascu, D., and Petrascu, V. (2009).
Javacompext : Extracting architectural elements from java source code.
In *Reverse Engineering, 2009. WCRE '09. 16th Working Conference on*, pages 317–318.


Beck, J. and Eichmann, D. (1993).
Program and interface slicing for reverse engineering.
In *ICSE '93 : Proceedings of the 15th international conference on Software Engineering*, pages 509–518, Los Alamitos, CA, USA. IEEE CS Press.


Bruneton, E., Coupaye, T., Leclercq, M., Quéma, V., and Stefani, J.-B. (2006).
The Fractal Component Model and Its Support in Java.
Software Practice and Experience, 36(11-12).


Bures, T., Hnetynka, P., and Plasil, F. (2006).
Sofa 2.0 : Balancing advanced features in a hierarchical component model.
In *SERA '06 : Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications*, pages 40–48. IEEE CS.

Références II


Chiorean, D. I., Ober, I., and Petrascu, V. (2011).
Avoiding ocl specification pitfalls.
ECEASST, 52.


Chiorean, D. I., Petrascu, V., and Petrascu, D. (2008).
How my favorite tool supporting ocl must look like.
ECEASST, 15.


Favre, J.-M. and Musset, J. (2006).
Rétro-ingénierie dirigée par les métamodèles : Concepts, Outils, Méthodes.
In *2èmes Journées sur l'Ingénierie Dirigée par les Modèles*.


Kollmann, R., Selonen, P., and Stroulia, E. (2002).
A study on the current state of the art in toolsupported uml-based static reverse engineering.
In *Proceedings of the Ninth Working Conference on Reverse Engineering*, pages 22–32.


Maier, T. and Zündorf, A. (2003).
The fujaba statechart synthesis approach.
In *in : 2nd International Workshop on Scenarios and State Machines : Models, Algorithms, and Tools, ICSE 2003*. Wiley-VCH.


Parizek, P., Plasil, F., and Kofron, J. (2006).
Model checking of software components : Combining java pathfinder and behavior protocol model checker.
Software Engineering Workshop, 0 :133–141.

Références III


Pavel, S., Noyé, J., Poizat, P., and Royer, J.-C. (2005).
A java implementation of a component model with explicit symbolic protocols.
In *Proceedings of the 4th International Workshop on Software Composition (SC'05)*, volume 3628 of *LNCS*, pages 115–125. Springer-Verlag.


Poch, T. and Plasil, F. (2009).
Extracting behavior specification of components in legacy applications.
In Lewis, G. A., Poernomo, I., and Hofmeister, C., editors, *CBSE*, volume 5582 of *Lecture Notes in Computer Science*, pages 87–103. Springer.


Rausch, A., Reussner, R., Mirandola, R., and Plasil, F., editors (2008).
The Common Component Modeling Example : Comparing Software Component Models, volume 5153 of *LNCS*.
Springer, Heidelberg.


Thevenet, L.-H. (2009).
Proposition d'une modélisation conceptuelle d'alignement stratégique : La méthode INSTAL.
These, Université Panthéon-Sorbonne - Paris I.