

**Un langage de description d'architectures
multi-niveaux
pour l'évolution directe et la rétro-évolution
de logiciels à base de composants**

Huaxi (Yulin) ZHANG

LGI2P / Ecole des Mines d'Alès - Nîmes – France

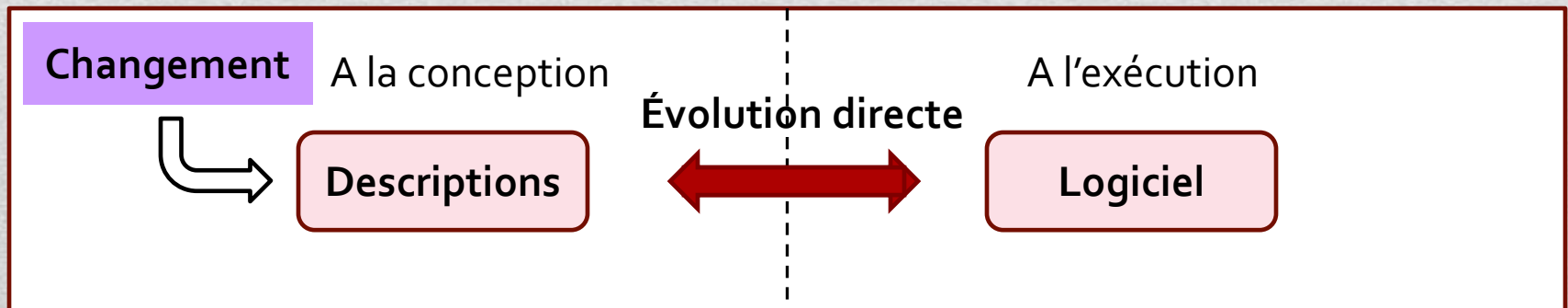
Directeur de thèse: Marianne Huchard

Encadrants: Christelle Urtado, Sylvain Vauttier



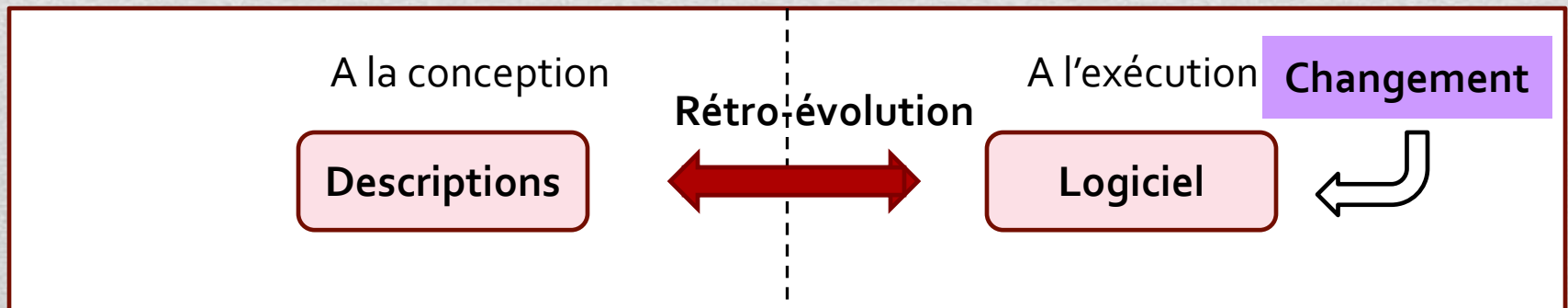
Contexte

- ❖ Génie logiciel à base de composants
 - Réutilisation de composants « sur étagère »
- ❖ Architecture logicielle
 - Langage de description d'architecture
- ❖ Évolution dynamique
 - Cohérence entre les descriptions architecturales et le logiciel qui s'exécute
 - Pour éviter la dérive et l'érosion d'architecture



Contexte

- ❖ Génie logiciel à base de composants
 - Réutilisation de composants « sur étagère »
- ❖ Architecture logicielle
 - Langage de description d'architecture
- ❖ Évolution dynamique
 - Cohérence entre les descriptions architecturales et le logiciel qui s'exécute
 - Pour éviter la dérive et l'érosion d'architecture



Problématique

- ❖ Développement à base de composants
 - Processus centré architecture basé sur la réutilisation
- ❖ Évolution des architectures
 - Processus d'évolution contrôlée, centré architecture
 - Évolution directe et rétro évolution
- ❖ Un ADL
 - Support du processus de développement à base de composants
 - Intégrant l'expression du changement
 - Support du processus d'évolution centré architecture

Plan

- ❖ Partie I: Développement à base de composants centré architecture
 - Processus de développement centré architecture
 - Adéquation des ADL existants au développement centré architecture
 - Dedal: un langage de description d'architectures multi-niveaux

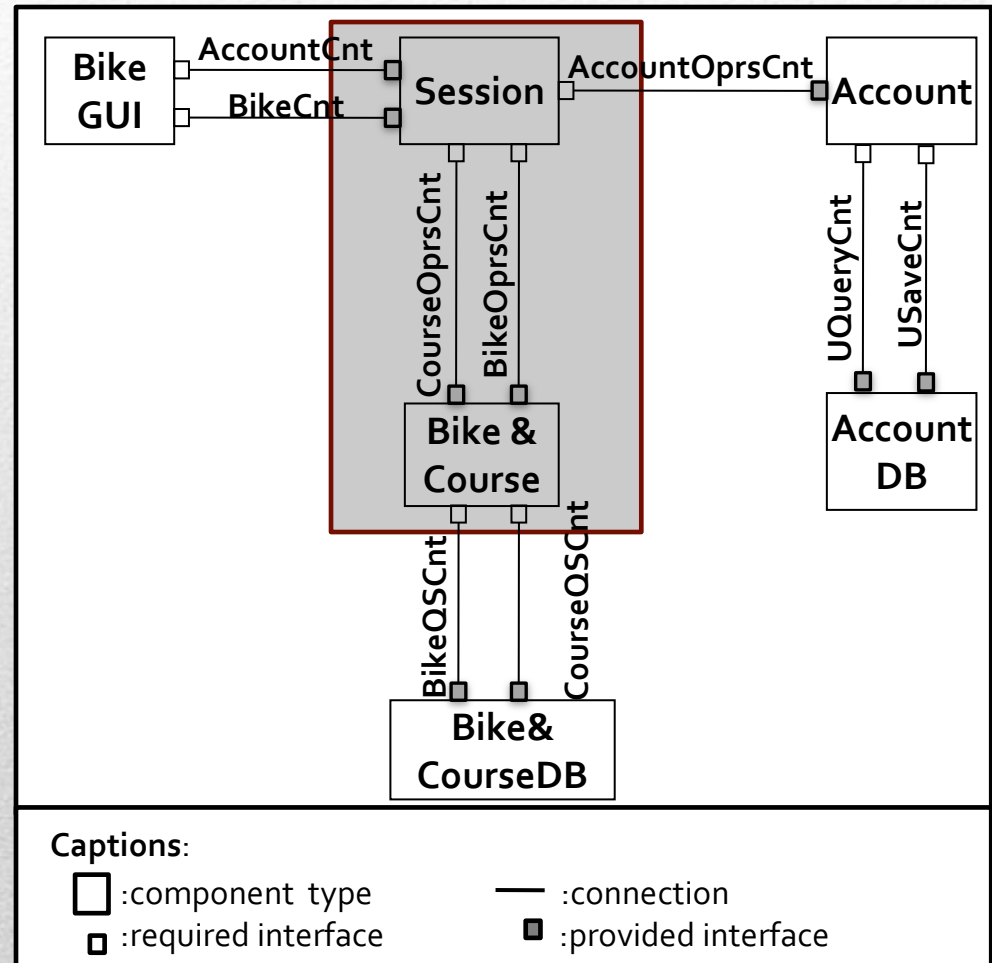
- ❖ Partie II: Evolution centrée architecture
 - Problématique de l'évolution des architectures
 - Support de l'évolution dans les ADL existants
 - Gestion du changement
 - Processus d'évolution contrôlée, centré architecture

- ❖ Prototype

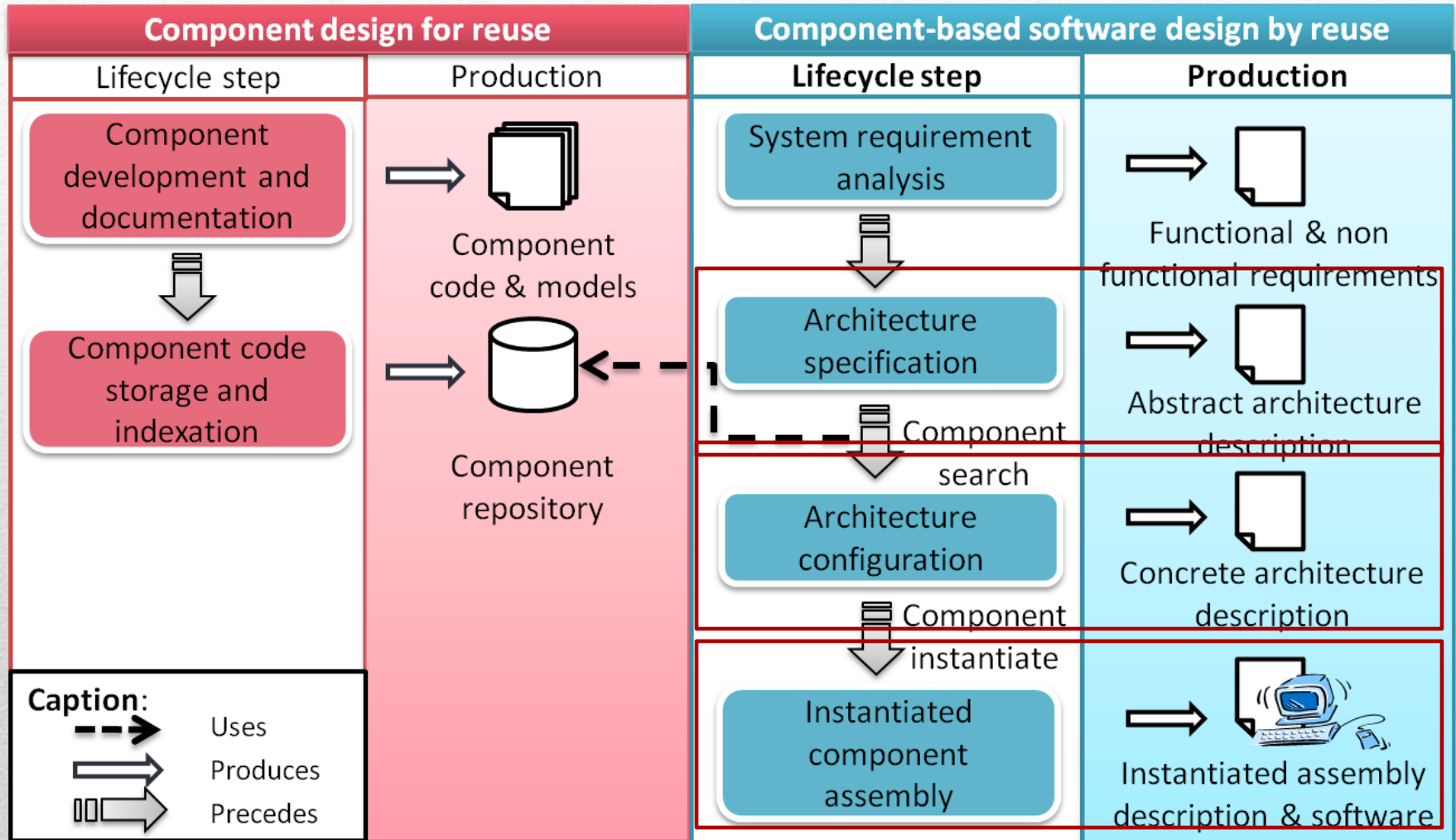
- ❖ Conclusion

Processus de développement par réutilisation de composants

- ❖ Centré sur la définition d'une architecture
- ❖ Utilisation d'un Langage de Description d'Architecture (ADL)
 - Identification des composants à réutiliser
 - Définition des connexions entre composants



Développement à base de composants centré architecture



Raffinement des composants

Spécification

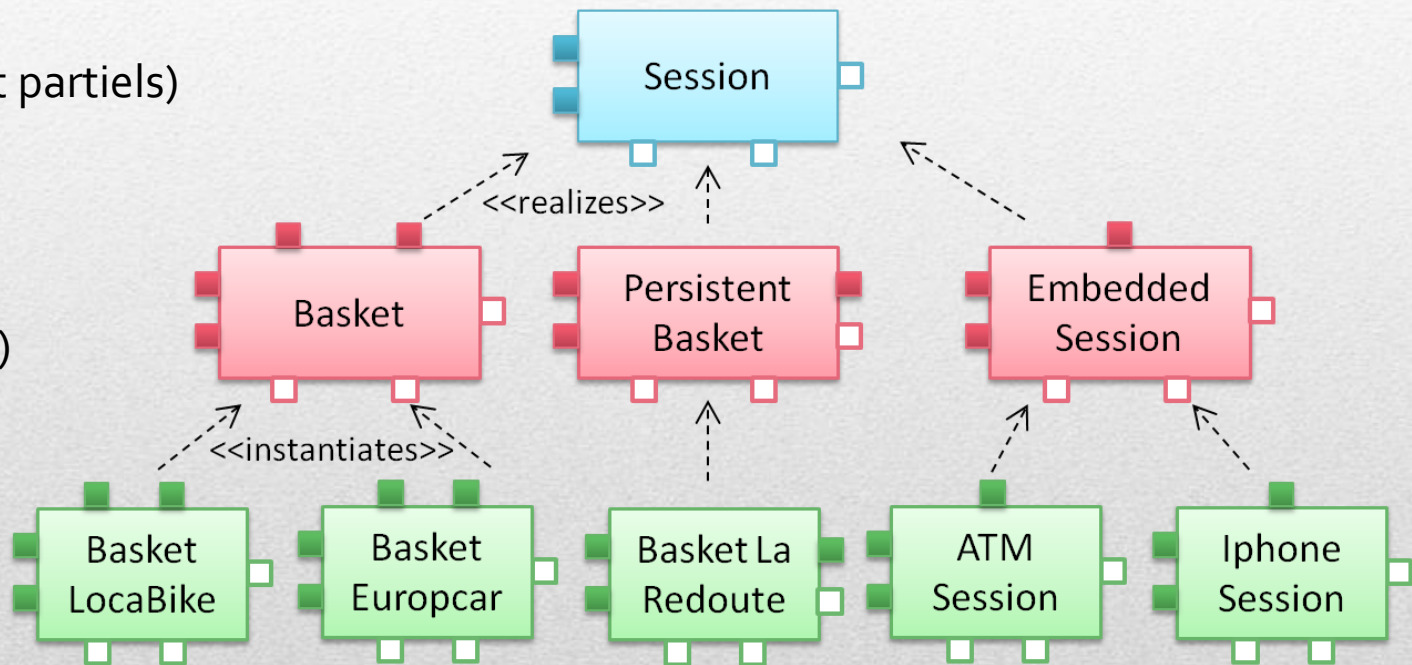
(types abstraits et partiels)

Configuration

(classes concrètes)

Assemblage

(instances spécifiques)



Trois niveaux d'architecture dans les ADL existants

❖ Spécification

- Types de composants abstraits, partiels
- Connexions
- Comportement d'architecture

❖ Assemblage

- Instances de composants
- Instances de connecteurs
- Contraintes d'assemblage

❖ Configuration

- Classes de composants
- Classes de connecteurs



SOFA 2.0

Wright

Fractal ADL

Darwin

xADL 2.0

Unicon

Trois niveaux d'architecture dans les ADL existants

❖ Spécification

- Types de composants abstraits, partiels
- Connexions
- Comportement d'architecture

❖ Assemblage

- Instances de composants
- Instances de connecteurs
- Contraintes d'assemblage

❖ Configuration

- Classes de composants
- Classes de connecteurs

C2SADEL

SOFA 2.0

Wright

Fractal ADL

Darwin

xADL 2.0

Unicon

Trois niveaux d'architecture dans les ADL existants

❖ Spécification

- Types de composants abstraits, partiels
- Connexions
- Comportement d'architecture

❖ Configuration

- Classes de composants
- Classes de connecteurs

❖ Assemblage

- Instances de composants
- Instances de connecteurs
- Contraintes d'assemblage

C2SADEL

SOFA 2.0

Wright

Fractal ADL






















Darwin

xADL 2.0

Unicon

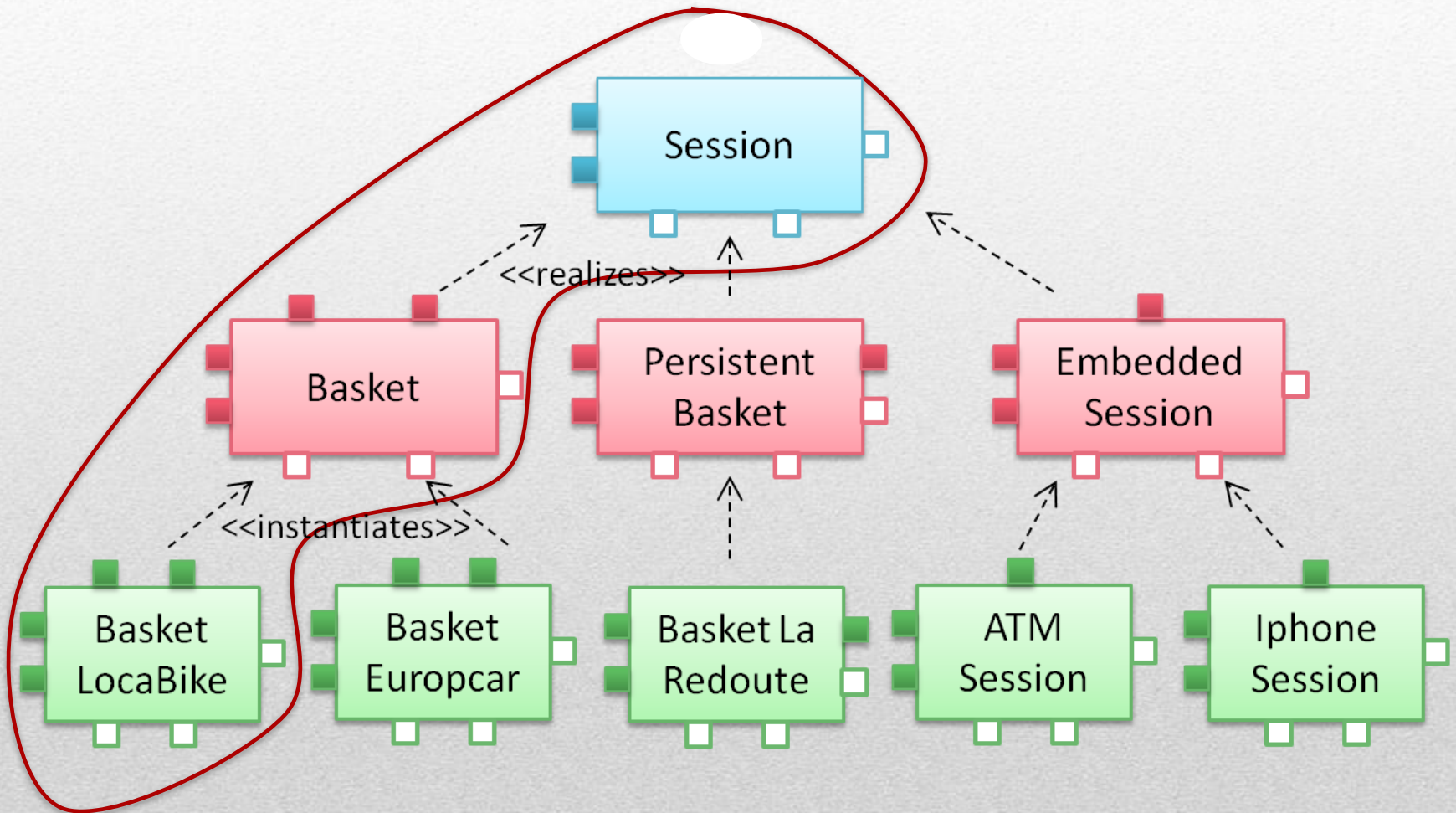
Synthèse

- ❖ Pragmatiques : centrés sur la définition de la configuration des architectures (implémentation)
- ❖ Pas de support au processus de développement par réutilisation

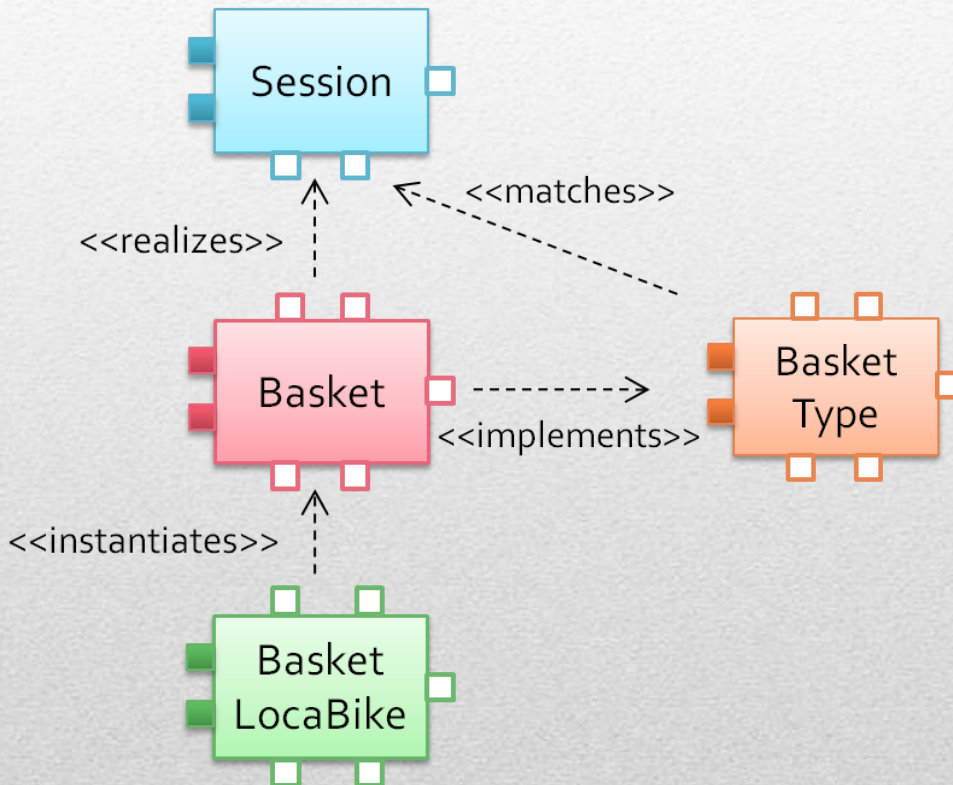
ADL	Spécification	Configuration	Assemblage
C2SADEL			
Wright			
Darwin			
Unicon			
SOFA 2.0			
Fractal ADL			
xADL 2.0			

Dedal: un langage de description d'architecture

- ❖ Multi-niveaux: Trois niveaux d'abstraction explicites
 - Spécification abstraite d'une architecture
 - Configuration concrète d'une architecture
 - Assemblage d'instances de composants
- ❖ Support des trois étapes du processus de développement
 - Pour la représentation séparée des décisions de conception
 - Pour la réutilisation des modèles abstraits pour définir des architectures concrètes



Description de composants en Dedal



```

component_role Session
required_interfaces BikeOprs; CourseOprs;
                        AccountOprs
provided_interfaces Account; Bike
component_behavior . . .
  
```

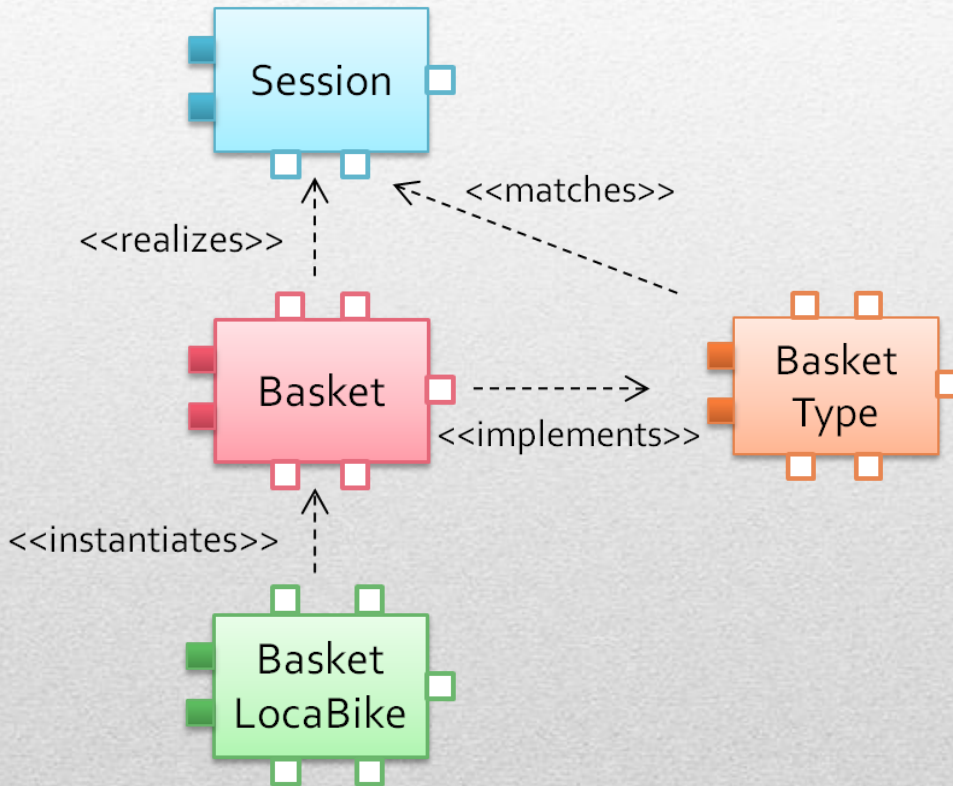
```

component_type BasketType
required_interfaces BikeOprs ; CourseOprs;
                        AccountOprs; CampusOprs; AccessoryOprs
provided_interfaces Account; Bike
component_behavior . . .
  
```

```

component_class Basket
implements BasketType
content fr.ema.locabike.Basket
attributes string company; string currency
versionID 1.0
  
```

Description de composants en Dedal



```

component_role Session
required_interfaces BikeOprs; CourseOprs;
                        AccountOprs
provided_interfaces Account; Bike
component_behavior . . .
  
```

```

component_class Basket
implements BasketType
content fr.ema.locaBike.Basket
attributes string company; string currency
versionID 1.0
  
```

```

component_instance BasketLocaBike
instance_of Basket (1.0)
initiation_state company="LocaBike";
                    concurrency="euro"
  
```

Spécification abstraite d'une architecture en Dedal

- ❖ Expression des décisions architecturales
 - Liste des rôles de composants
 - Liste des connexions (topologie)
 - Comportement d'architecture

```
specification BRSSpec
component_roles Session; BikeCourse
connections
  connection CourseOprsCnt
    client Session.CourseOprs
    server BikeCourse.CourseOprs
  connection BikeOprsCnt
    client Session.BikeOprs
    server BikeCourse.BikeOprs
architecture_behavior
(!Session.Bike.selectBike;
 ?Session.BikeOprs.selectBike;
 ! BikeCourse.BikeOprs.selectBike;
 ? BikeCourse.BikeQS.selectBike;
)
+
...
versionID 1.0
```

Extrait de la spécification

Configuration concrète d'une architecture en Dedal

- ❖ Raffinement d'une spécification
- ❖ Expression des choix d'implémentation
 - Liste des classes de composants
 - Sélectionnées
 - Réutilisées dans les différents rôles
 - Liste des classes de connecteurs

```
configuration BRSSConfig  
implements BRSSpec (1.0)  
component_classes  
  Basket (1.0) as Session;  
  BikeTrip (1.0) as BikeCourse;  
versionID 1.0
```

Extrait de la configuration

Assemblage d'instances de composants en Dedal

- ❖ Raffinement d'une configuration
- ❖ Expression des décisions d'instanciation / déploiement
 - Liste des instances de composants
 - Spécifiques
 - Utilisées dans certain rôles
 - Liste des contraintes d'assemblage

```
assembly BRSAAss
instance_of BRSConfig (1.0)
component_instances
  BikeTripC1 as BikeCourse ;
  BikeTripDBC1 as BikeCourseDB ;
assembly_constraints
  BikeCourse.currency="Euro" ;
  BikeCourseDB.company=
      BikeCourse.company
versionID 1.0
```

Extrait de l'assemblage

Contributions – Partie I

- ❖ Proposition d'un processus de développement à base de composants centré architecture
- ❖ Analyse du support d'un processus de développement basé sur la réutilisation de composants par les ADL existants
- ❖ Proposition d'un ADL support au processus de développement d'architectures par réutilisation de composants
 - Définition d'architectures concrètes par raffinements successifs
 - Représentation séparée des décisions de conception
 - Intégration des mécanismes de recherche et de sélection des composants à réutiliser

Plan

- ❖ Partie I: Développement à base de composants centré architecture
 - Processus de développement centré architecture
 - Adaptation des ADL existants au développement centré architecture
 - Dedal: un langage de description d'architectures multi-niveaux

- ❖ Partie II: Evolution centrée architecture
 - Problématique de l'évolution des architectures
 - Support de l'évolution dans les ADL existants
 - Gestion du changement
 - Processus d'évolution centré architecture

- ❖ Prototype

- ❖ Conclusion

Problématique de l'évolution d'architecture

- ❖ Gestion du changement d'une architecture
 - Opérations de changement (ajout, retrait, modification)
 - Caractérisation, traçabilité des changements

- ❖ Maintien de la cohérence des différentes descriptions d'une architecture
 - Cohérence interne à une description
 - En vérifiant la syntaxe et la sémantique
 - Cohérence entre les descriptions
 - Pour éviter la dérive et l'érosion des descriptions
 - En vérifiant la sémantique des relations (« realizes », « instantiates »)

Support du changement

❖ Moment du changement

- Statique
- Dynamique

❖ Anticipation

- Anticipée
- Non-anticipée

❖ Type de changement

- Sémantique
- Structure

❖ But de changement

- Correction
- Perfectionnement
- Adaptation

❖ Niveau du changement

- Spécification
- Configuration
- Assemblage

Support du changement dans les ADL existants

ADL	Time	Anticipation	Change type	Change purpose	Level of change
C2SADEL	runtime	unanticipated	structure	⊗	configuration
Darwin	runtime	anticipated, unanticipated	structure	⊗	configuration
Dynamic Wright	runtime	anticipated	structure	⊗	configuration
SOFA 2.0	runtime	anticipated	structure	⊗	configuration
xADL 2.0	runtime	unanticipated	structure	⊗	configuration
MAE	runtime	unanticipated	semantic	perfective	configuration

Support de l'évolution dans les ADL existants

- ❖ Vérification de cohérence
- ❖ Test de l'évolution
- ❖ Propagation des changements
- ❖ Versionnement

ADL	Consistency checking	Evolution test	Change propagation	Versionning
C2SADEL	Refinement	⊗	⊗	⊗
Darwin	State	⊗	⊗	⊗
Dynamic Wright	Name, Interaction, Deadlock	⊗	⊗	⊗
SOFA 2.0	Behavior	⊗	⊗	Architecture (state-based)
xADL 2.0	Refinement	⊗	Top-down	⊗
MAE	Sub-typing	Substitution	⊗	Component (change-based)

Synthèse des Travaux existants

- ❖ Peu de support à l'expression du changement
- ❖ Centrés sur la vérification de la cohérence locale
- ❖ Peu de mécanismes de propagation des changements
 - Changements réalisés sur les configurations
 - Changements propagés aux architectures en cours d'exécution (évolution directe)

Gestion du changement en Dedal

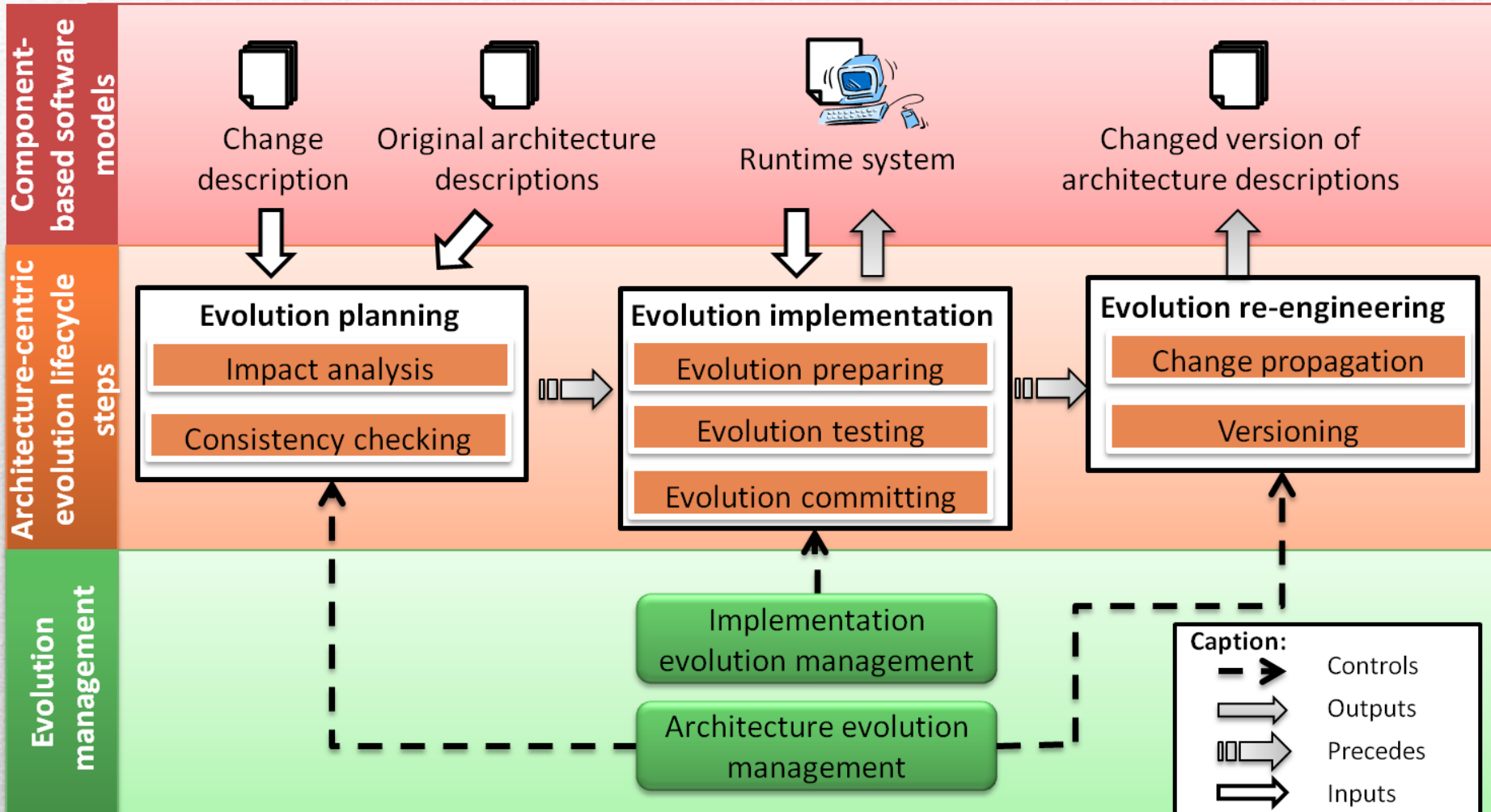
- ❖ Représentation explicite des changements :
 - Éléments de modèle de première classe
 - Décrits par un ensemble étendu de propriétés

- ❖ Versionnement :
 - Des descriptions d'architectures
 - Spécifications
 - Configurations
 - Assemblages
 - Des éléments d'architecture
 - Classes de composants

```
change additionStationData
time dynamic
level configuration
operation addition
artifact component_class
           is StationData
purpose perfective
origin configuration
```

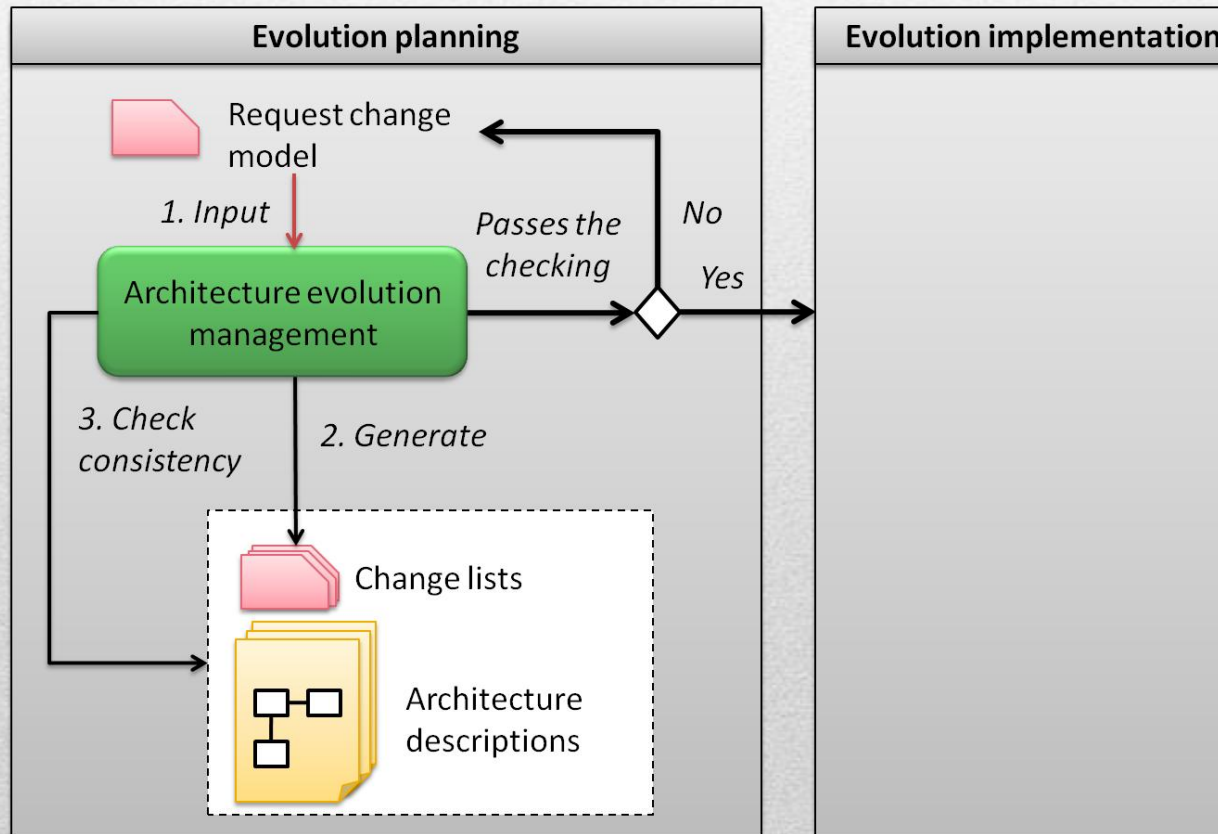
```
configuration BRSConfig
...
versionID 2.0
pre_version 1.0
change additionStationData
```

Processus d'évolution contrôlée, centré architecture



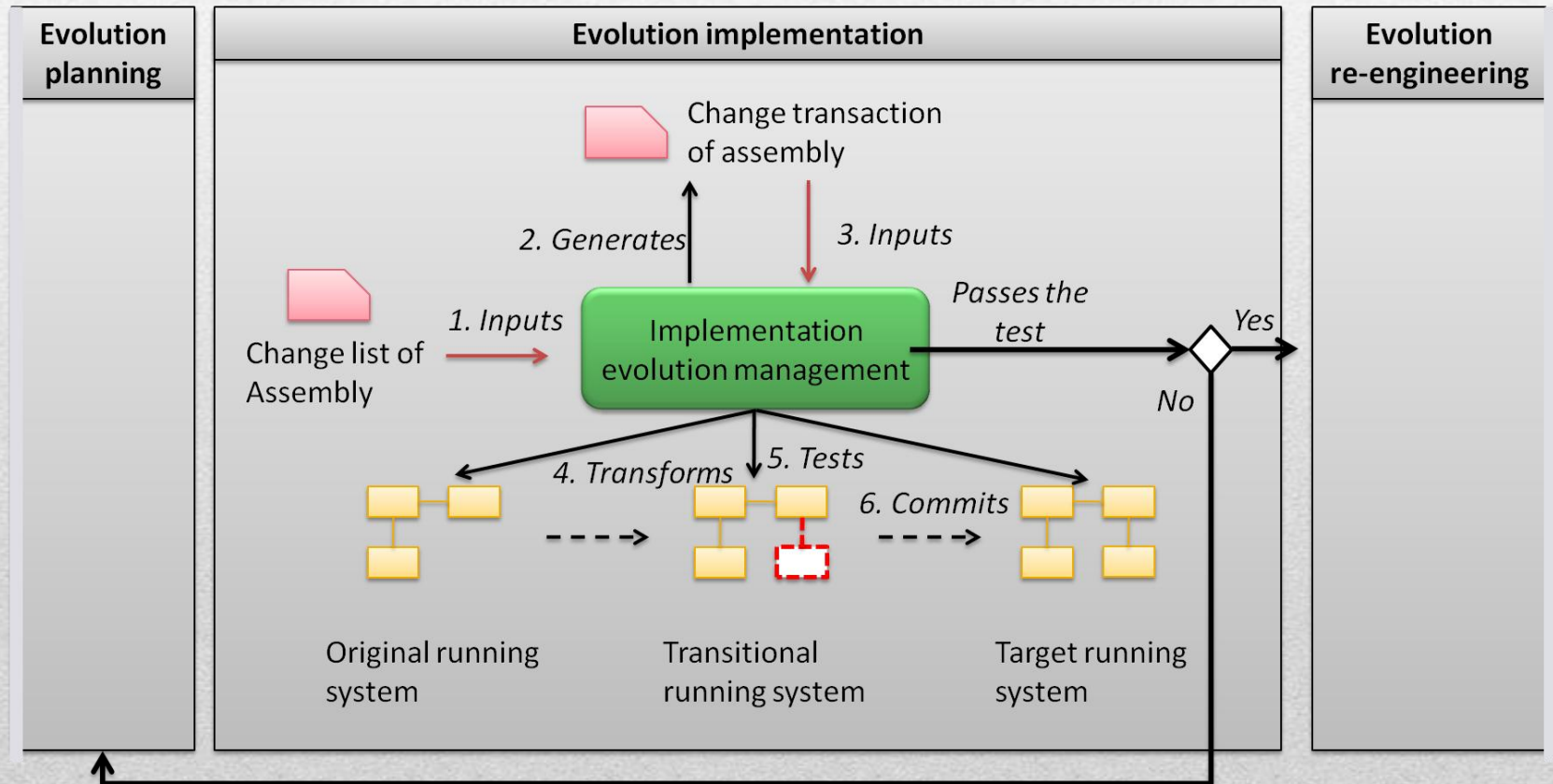
Planification de l'évolution

- ❖ Analyse de l'impact
- ❖ Vérification de cohérence



Implémentation de l'évolution

- ❖ Changement transactionnel
- ❖ Processus graduel d'évolution



Environnement de gestion de l'évolution dynamique

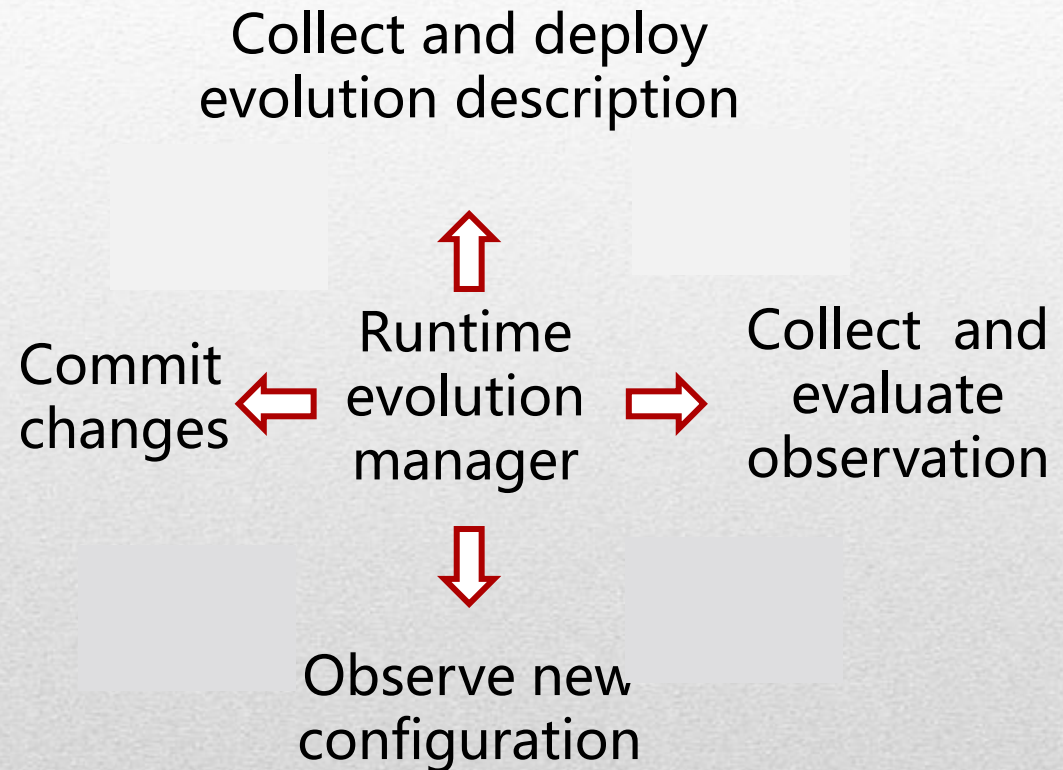
- ❖ Conteneur de composants
 - Gestion de l'évolution
- ❖ Composant
 - Observation des changements d'états (historiques)
- ❖ Connecteur
 - Contrôles des connexions
 - Contrôles des flux de données
 - Contrôles temporels (horodatage)

Mécanisme d'évolution graduelle

❖ Architecture transitoire

❖ Quatre étapes

- Étape de préparation
- Étape d'évaluation (test)
- Étape de surveillance
- Étape de confirmation



Exemple

❖ Préparation

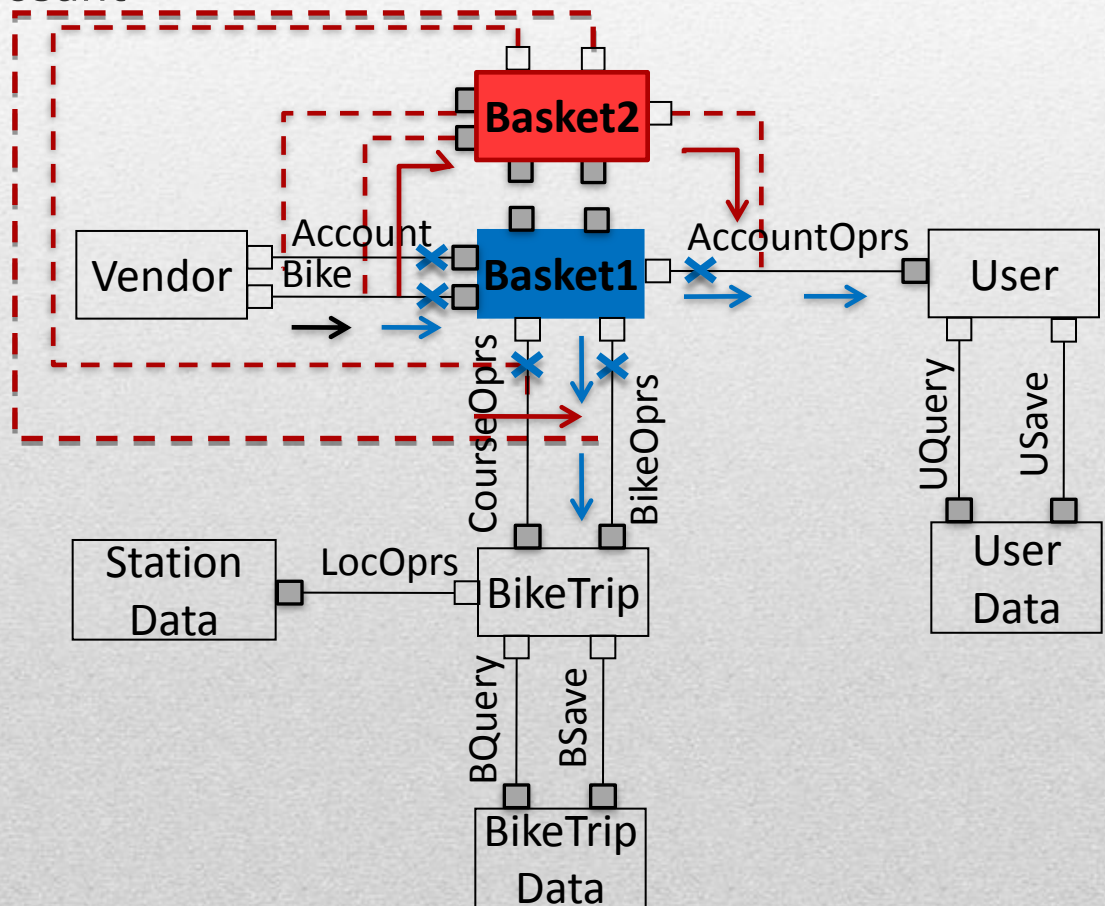
- Ajout d'un nouveau composant
- Connexion avec le nouveau composant
- Migration état

❖ Evaluation

- Passive
(Basket1 dominant)
- Active
(Basket2 dominant)

❖ Surveillance

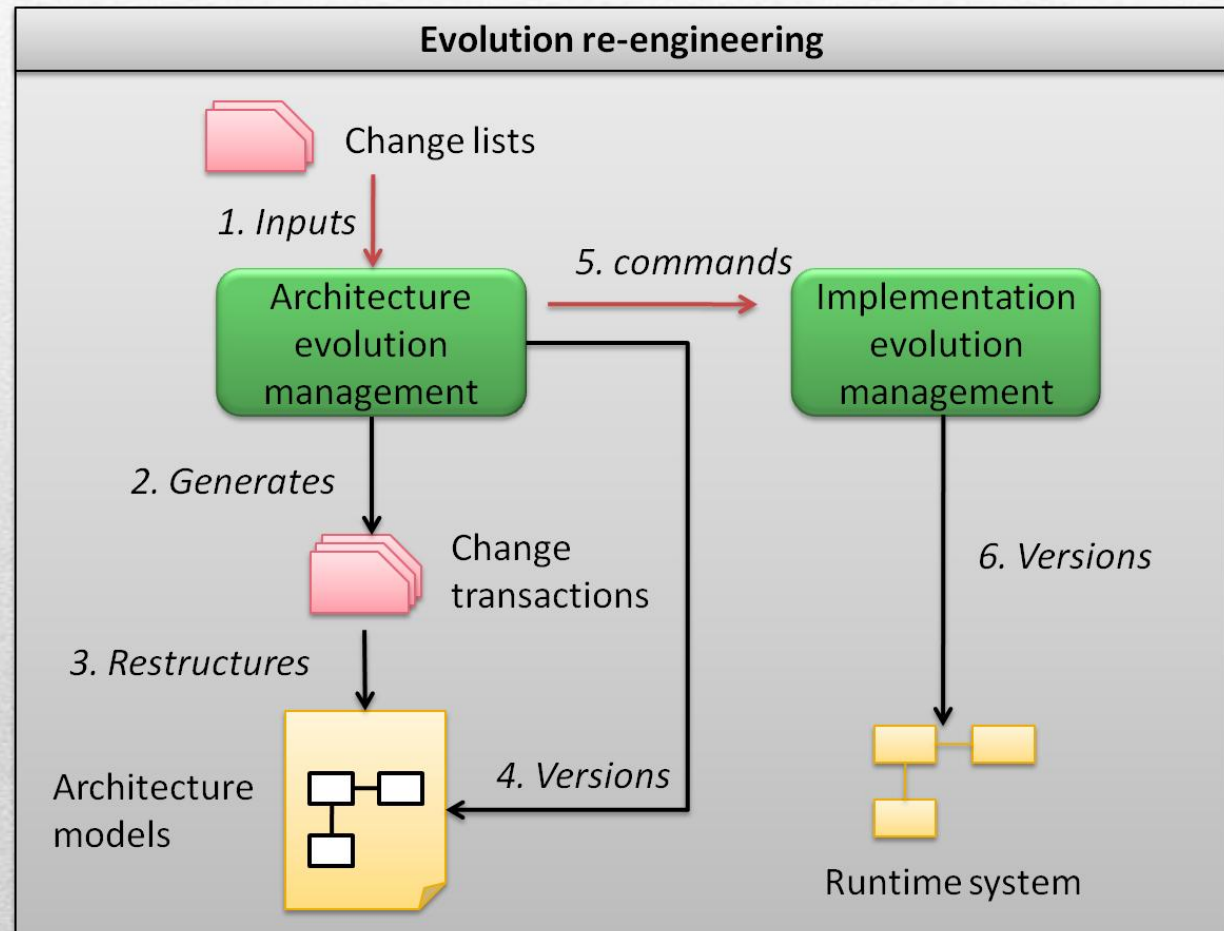
❖ Confirmation



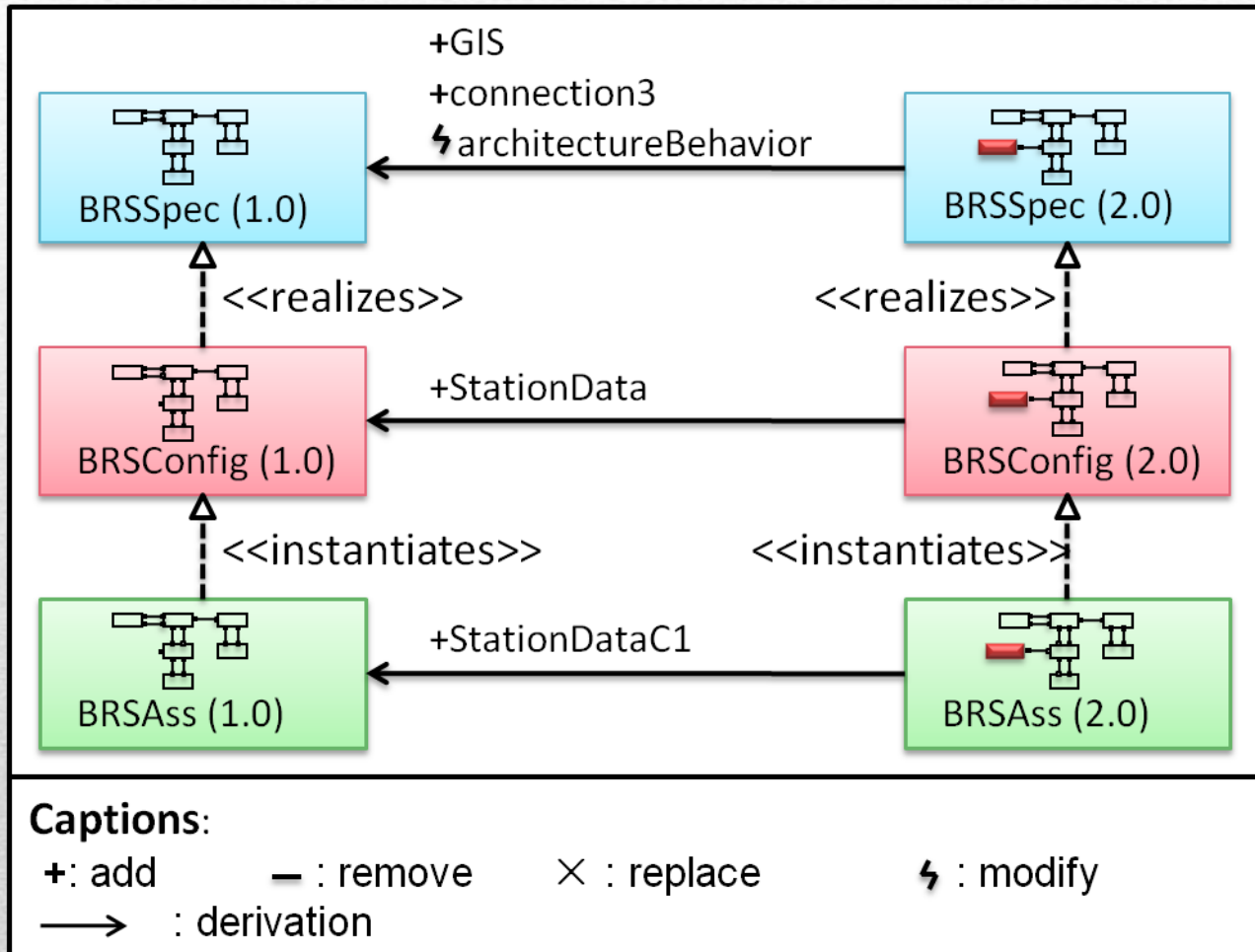
Ré-ingénierie par l'évolution

❖ Propagation des changements

❖ Versionnement



Arbres de versions



Contributions – Partie II

- ❖ Analyse de l'expression du changement et de la gestion de l'évolution par les ADL existants
- ❖ Proposition d'un ADL intégrant l'expression du changement et la gestion du versionnement
 - Expression d'un changement
 - Versionnement des architectures basé sur les changements
- ❖ Proposition d'un processus d'évolution contrôlée, centré architecture
 - Vérification de la cohérence
 - Propagation du changement
 - Évolution graduelle (multi-versions, test, architecture transitoire)

Plan

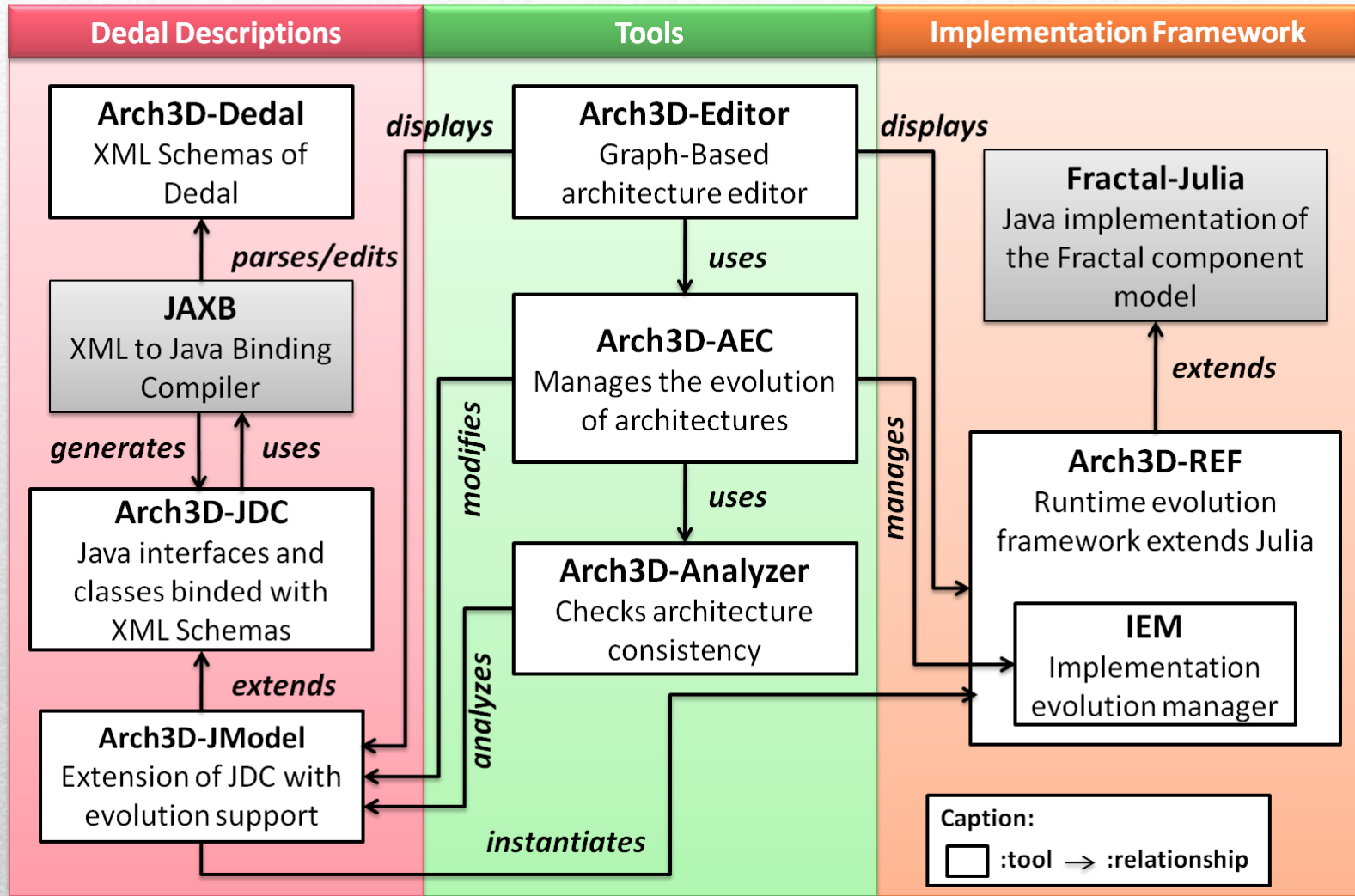
- ❖ Partie I: Développement à base de composant centré architecture
 - Processus de développement centré architecture
 - Adaptation des ADL existants au développement centré architecture
 - Dedal: un langage de description d'architectures multi-niveaux

- ❖ Partie II: Evolution centrée architecture
 - Problématique de l'évolution des architectures
 - Support de l'évolution dans ADL existants
 - Gestion du changement
 - Processus d'évolution contrôlée, centré architecture

- ❖ **Prototype**

- ❖ **Conclusion**

Arch3D



Vue d'Arch3D - Editor

The screenshot displays the Arch3D Editor interface, which is divided into several panes. On the left is the **Architecture Explorer** showing a tree view of the project structure. The main area is split into a **Graph Editor** and an **XML Editor**. The Graph Editor shows three levels of abstraction: 1) **Specification**: A simple connection between `clientRole` and `server1Role` via `connection1`. 2) **Configuration**: A connector `connector1` between `clientClass:clientRole` and `serverClass:server1Role`. 3) **Assembly**: A connector instance `connectorIns` between `clientIns:clientClass` and `serverIns:serverClass`. The XML Editor on the right shows the corresponding code for each level, including BNF and Table expressions for specification, configuration, and assembly.

```
specification clientServerSpec
component_roles
  clientRole;
  server1Role;
connections
  connection connection1
  client clientRole.r1
  server server1Role.p
architecture_behavior
  clientRole.r1.findName?;serverRole.s1.findName
versionID 1.0

configuration clientServerConfig
implements clientServerSpec
component_classes
  clientClass as clientRole;
  serverClass as server1Role;
connector_classes
  connector1 as connection1;
versionID 1.0

assembly clientServerAss
instance_of clientServerConfig
component_instances
  clientIns as clientRole;
  serverIns as server1Role;
connector_instances
  connectorIns;
assembly_rules
  MaxInstanceNbr(ClientRole)=8 && MinInstanceNbr(ClientRole)=5
versionID 1.0
```

Plan

- ❖ Partie I: Développement à base de composant centré architecture
 - Processus de développement centré architecture
 - Adaptation des ADL existants au développement centré architecture
 - Dedal: un langage de description d'architectures multi-niveaux

- ❖ Partie II: Evolution centrée architecture
 - Problématique de l'évolution des architectures
 - Support de l'évolution dans les ADL existants
 - Gestion du changement
 - Processus d'évolution contrôlée, centré architecture

- ❖ Prototype

- ❖ Conclusion

Contributions

❖ Développement à base de composants

- Proposition d'un processus de développement à base de composants centré architecture
- Analyse du support d'un processus de développement basé sur la réutilisation de composants par les ADL existants
- Proposition d'un ADL support au processus de développement d'architectures par réutilisation de composants

❖ Évolution des architectures

- Analyse de l'expression du changement et de la gestion de l'évolution par les ADL existants
- Proposition d'un ADL intégrant l'expression du changement et la gestion du versionnement
- Proposition d'un processus d'évolution contrôlée, centré architecture

❖ Implémentation d'un prototype de suite logicielle

Perspectives

❖ Théoriques

- Formalisation des relations entre les niveaux
 - Aspects statique et dynamique
- Conception d'une bibliothèque de composants et d'architectures
- Gestion autonome de l'évolution
- Formalisation des processus de développement et d'évolution (SPEM)
- Intégration d'un ADL multi-niveaux au métamodèle d'UML 2.0

❖ Pratiques

- Intégration à un IDE standard (Eclipse)
- Expérimentations à partir de données issues de vrais projets

Publications

- ❖ **Dedal: un ADL à trois dimensions pour gérer l'évolution des architectures à base de composants.**
H. Y. ZHANG, C. URTADO and S. VAUTTIER.
In Proceedings of Conférence Francophone sur les Architectures Logicielles (CAL2010).
- ❖ **Connector-driven process for the gradual evolution of component-based software.**
H. Y. ZHANG, C. URTADO and S. VAUTTIER.
In Proceedings of the 20th Australian Software Engineering Conference (ASWEC2009). IEEE. Gold Coast, Australia, April 2009.
- ❖ **Connector-driven gradual and dynamic software assembly evolution.**
H. Y. ZHANG, C. URTADO and S. VAUTTIER.
In Proceedings of the International Conference on Innovation in Software Engineering (ISEo8), M. Mohammadian editor, pages 345-350, IEEE. Vienna, Austria, December 2008.