30/10/2012

# Gouvernance et étude de l'impact du changement des processus métiers sur les architectures orientées services

Soutenance de thèse

**Karim DAHMAN – François CHAROY – Claude GODART**

UNIVERSITÉ DE LORRAINE

Loria

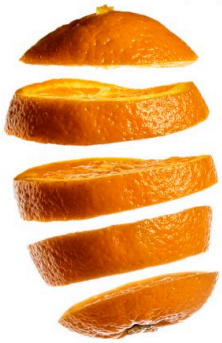
SCORE

# Evolutions des processus métiers

- **Processus métiers**
  - Réaliser un produit ou un service selon une logique métier
  - Automatiser (informatiser) les opérations de réalisation
- **Evolutions des processus métiers**
  - Faire évoluer → changer la logique métier
  - Changer → adapter le Système Informatique (**SI**)
  - Adapter → modifier la logique qui « automatise » le métier

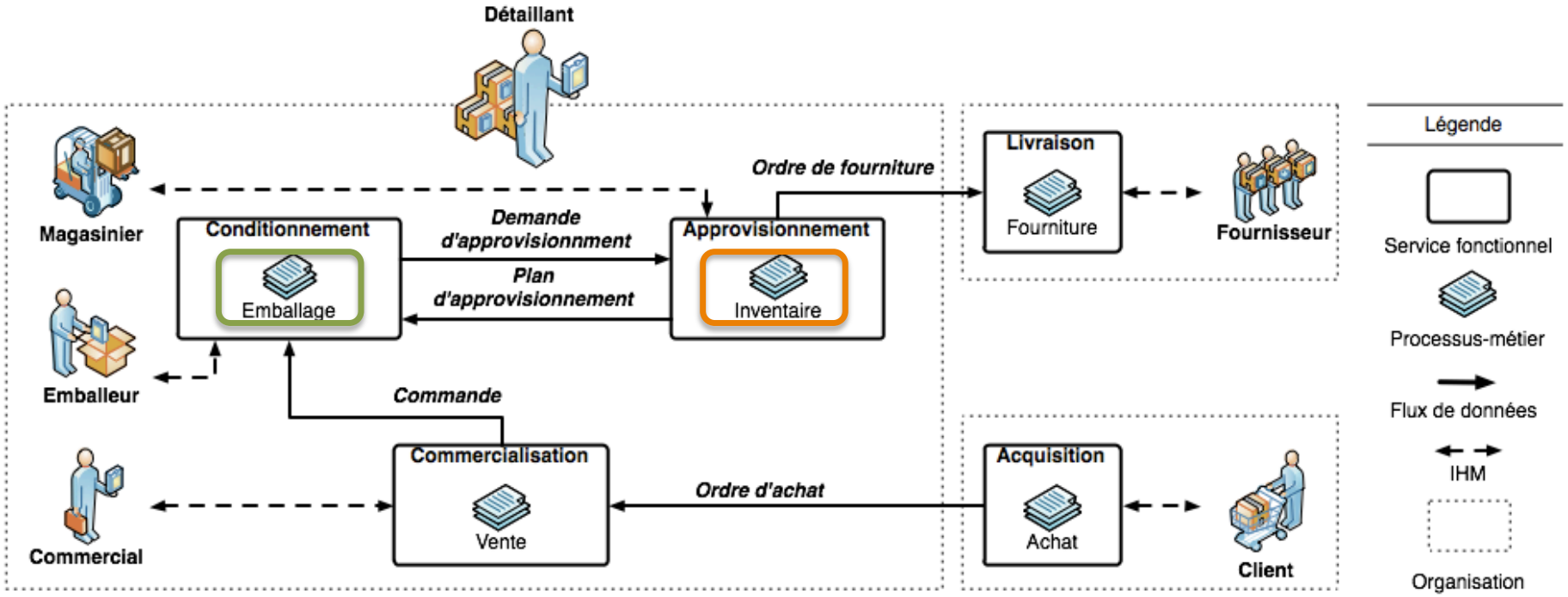
# Impact du changement sur le SI

- SI : une **solution logicielle** selon le **style SOA**
  - SOA : *Service Oriented Architecture*
    - Le « service » est le paradigme structurant de « l'automatisation »
  - Séparation des logiques : **métier**, **fonctionnelle**, **applicative**
  - Traçabilité des compositions (processus, service, composant)
- Notre intérêt pour l'étude de **l'impact du changement**
  - **Concevoir** une solution **selon les principes du style SOA**
  - **Modifier** la solution **selon les évolutions métiers**



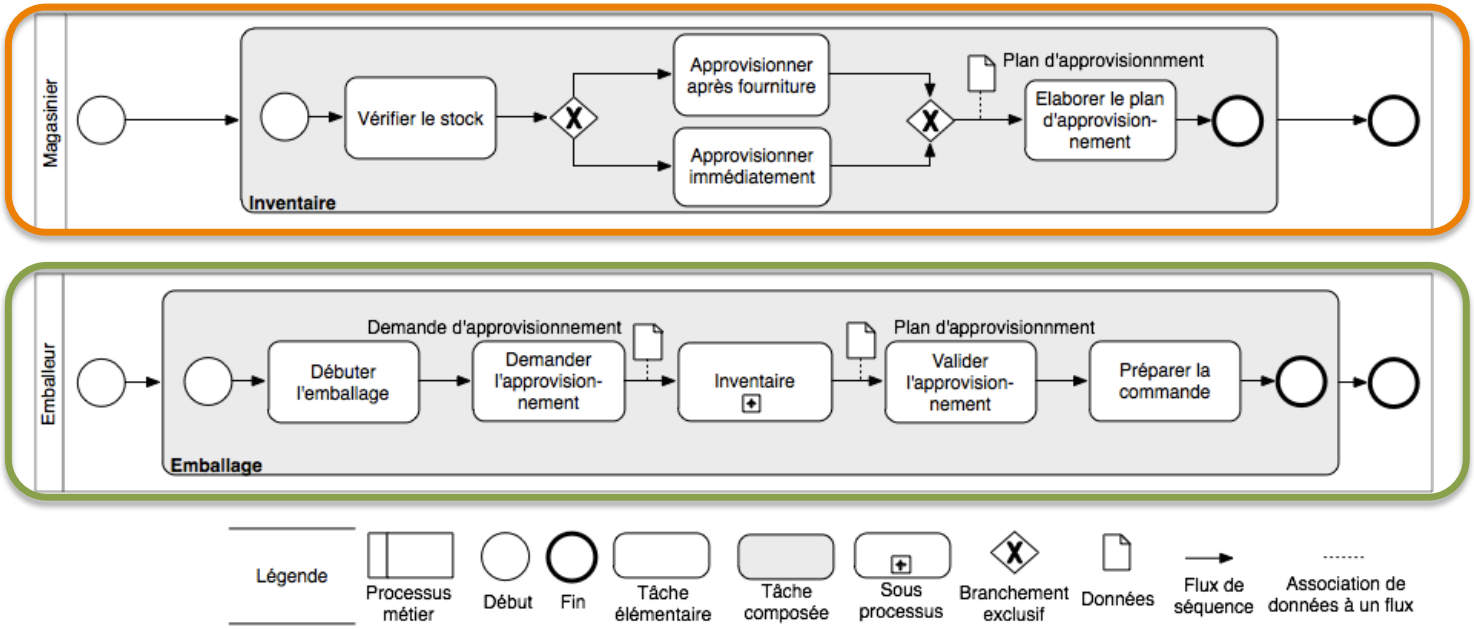
# Ingénierie logicielle du SI

- **Développer** une solution axée sur le métier
  - Problème : **garantir l'alignement entre les concepts**
    - Ex. : automatiser une procédure de commercialisation au détail



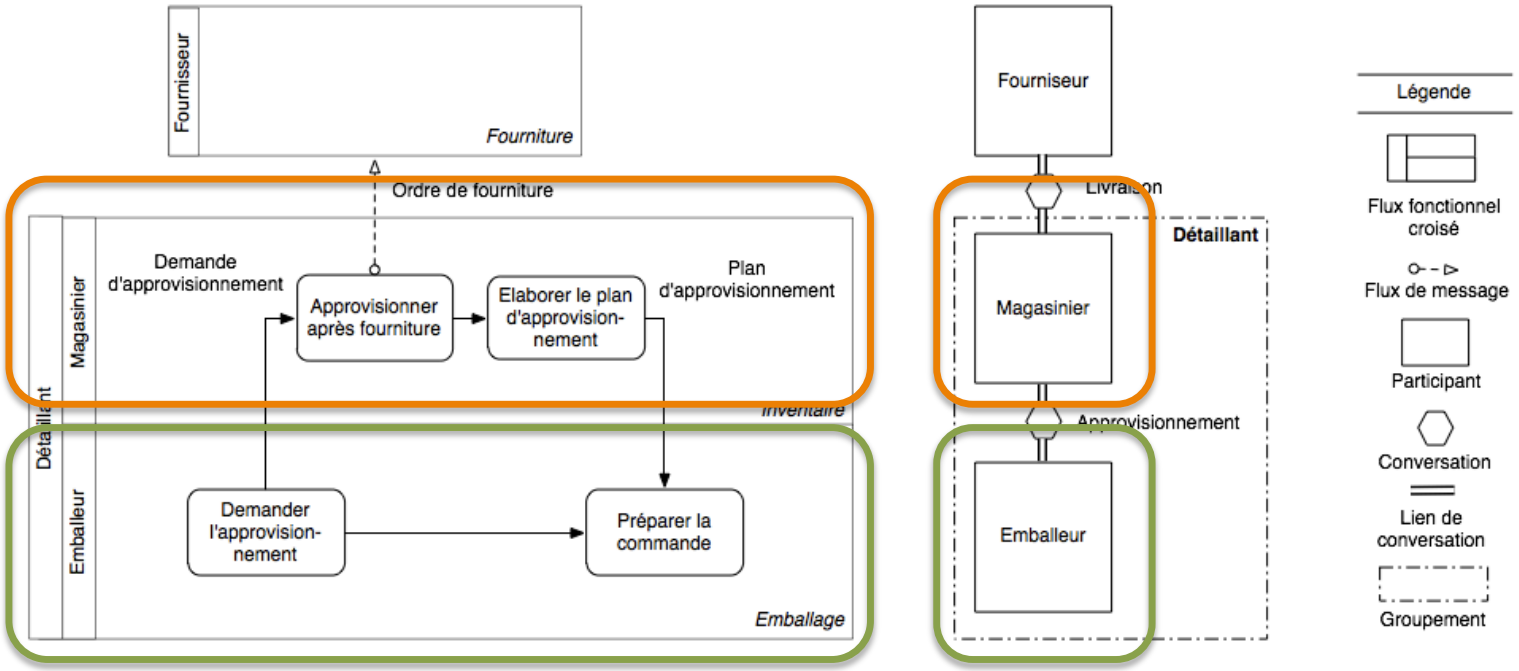
# Architecture métier : BPMN

- **Modéliser** les processus métiers
  - BPMN : *Business Process Modeling Notation*
  - Décrire la logique métier des **processus privés**
  - Définir les **responsabilités** entre les unités organisationnelles



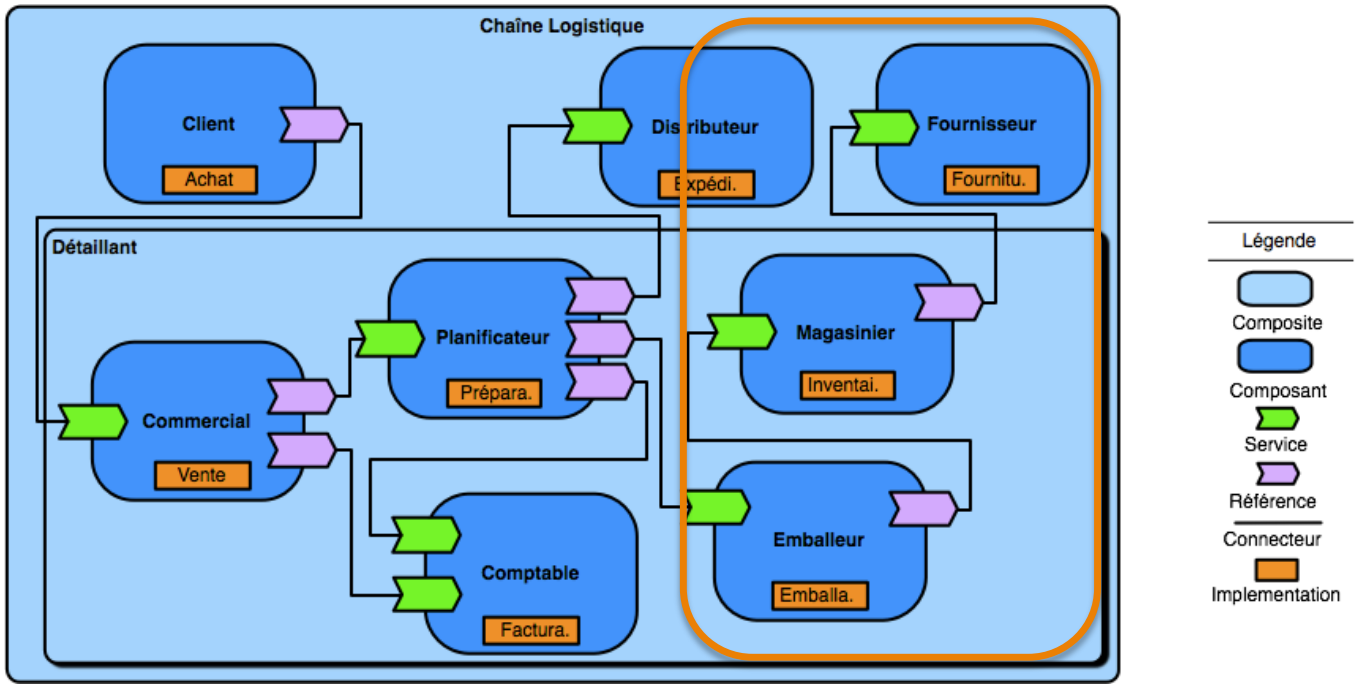
# Architecture fonctionnelle : BPMN

- **Modéliser** les services fonctionnels
  - BPMN : *Business Process « collaboration » Modeling Notation*
  - Spécifier les **collaborations** (processus publics) selon le métier
  - Structurer les **conversations** selon la logique fonctionnelle



# Architecture applicative : SCA

- **Modéliser** les composants applicatifs
  - SCA : *Service Component Architecture*
  - Construire les **configurations** selon le style **SOA**
  - Assembler les **domaines** selon la logique fonctionnelle



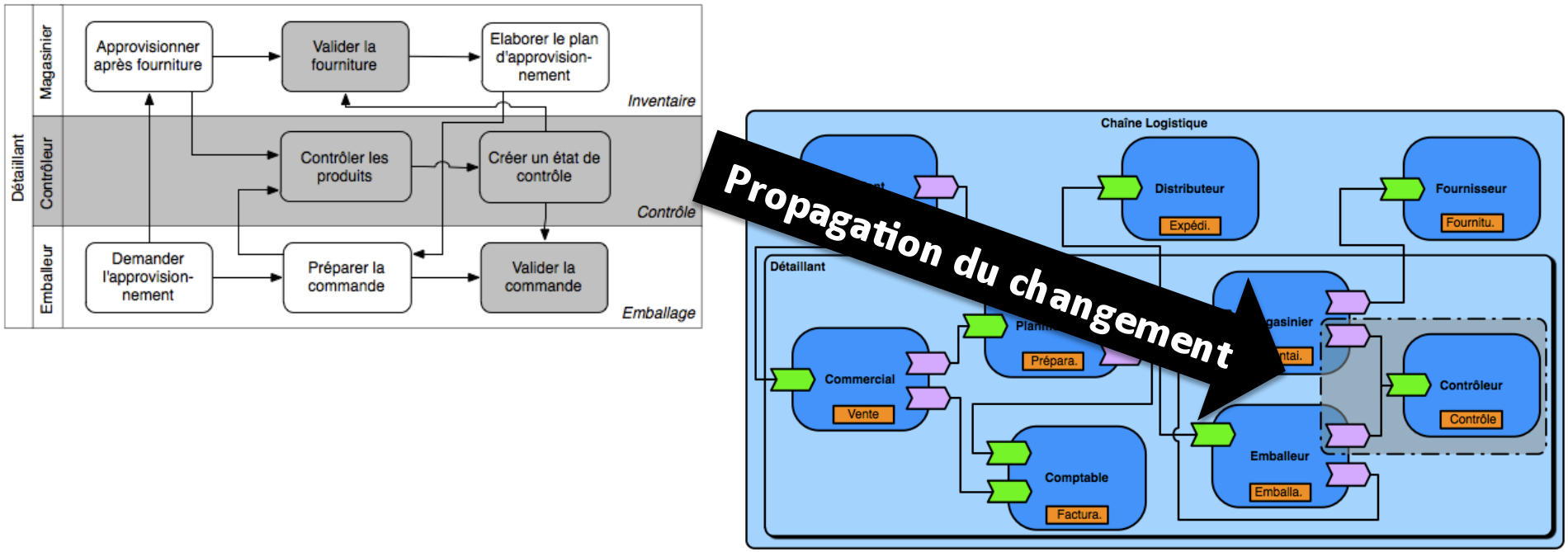
# Changement des processus


Si le modèle BPMN **change …**

Alors … le modèle SCA
doit être « **adaptée** » **!**

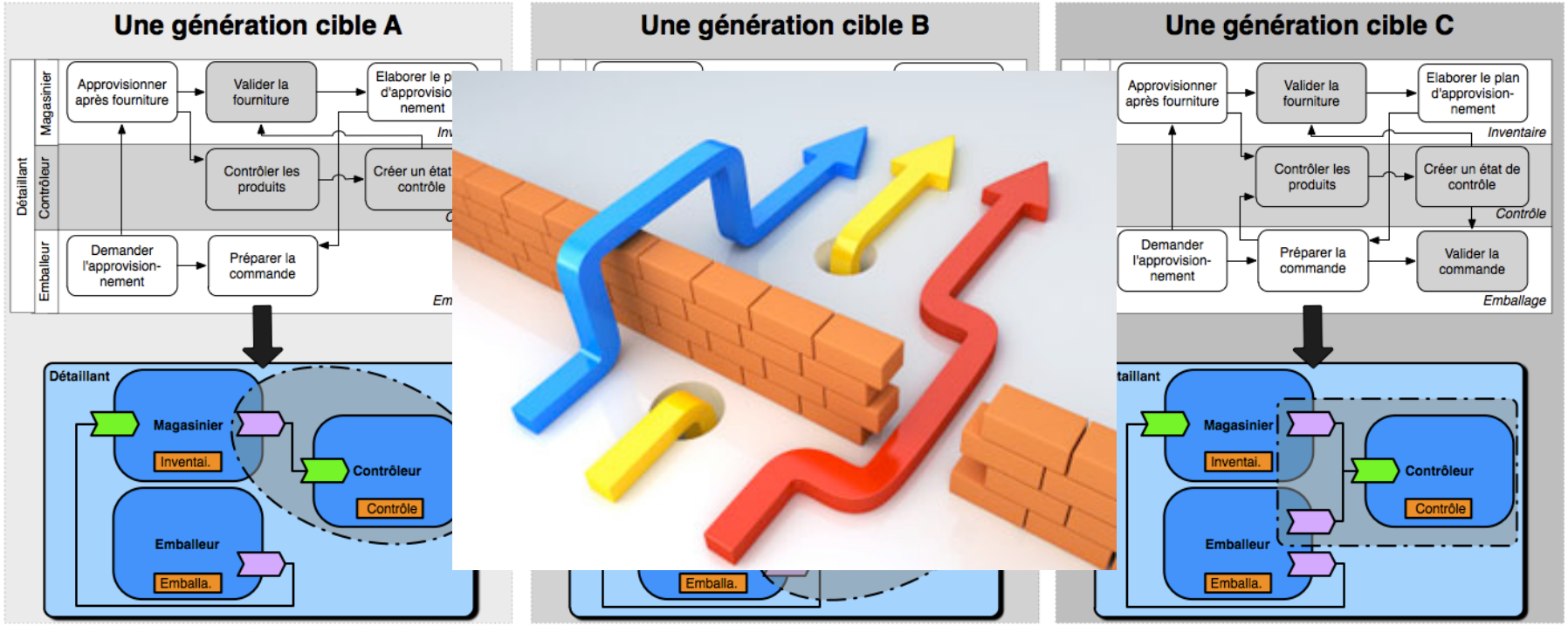
# Réingénierie logicielle du SI

- Ajout d'un service de contrôle qualité
  - Problème : **assurer l'alignement des architectures**
  - **Faire évoluer** le métier → **changer** les processus
  - Changer → **adapter** les services et les composants



# Alternatives de changement

- Plusieurs alternatives de conception



# Problème(s) du changement

- Questions **méthodologiques**
  - Comment **préserver** l'alignement des architectures ?
  - Que faut-il **changer**, ajouter ou supprimer ?
  - Comment **propager** le changement ?
  - Comment **évaluer** l'impact ?

# Automatisation du tissage

- Nos objectifs
  - **Automatiser la production** de(s) modèle(s) d'architecture(s)
  - **Assister les actions :** préserver, changer, propager, évaluer

- Notre proposition
  - Utiliser une **« approche dirigée par les modèles »**
  - Avec des notations « conceptuellement » découplées

# Cadre de l'étude du changement

- Démarche d'analyse
  - **Générative** : transformation automatisée
  - Outils pour tisser (construire) une solution


- Démarche de conception
  - **Itérative** : synchronisation assistée
  - Outils pour changer (reconstruire) la solution

# Etat de l'art

- Outil de développement

  **Eclipse STP** : *SOA Tools Platform*

  (+) Transformation automatisée BPMN-SCA

  (−) Tissage non viable, restrictif, pas de propagation des changements

- Langages de transformation

  **ATL** : *ATLAS transformation Language*

  (+) Implémentation du standard QVT

  (−) Pas de synchronisation, complexité des règles imbriquées

  **TGG** : *Triple Graph Grammars*

  (+) Abstraction : modèle - graphe, changement - état, propager - réévaluer

  (−) Inadaptée pour un *mapping* impérative et multivalué (réévaluations)

# Nos travaux : trois contributions

- Moyens pour **construire** et **changer** une solution


1. **Transformation** automatisée BPMN-SCA

   Tissage de modèles et alignement architectural

2. **Synchronisation** assistée BPMN-SCA

   Propagation et alignement fonctionnel

3. **Outillage** pour la simulation de l'impact

   Evaluation du changement des modèles

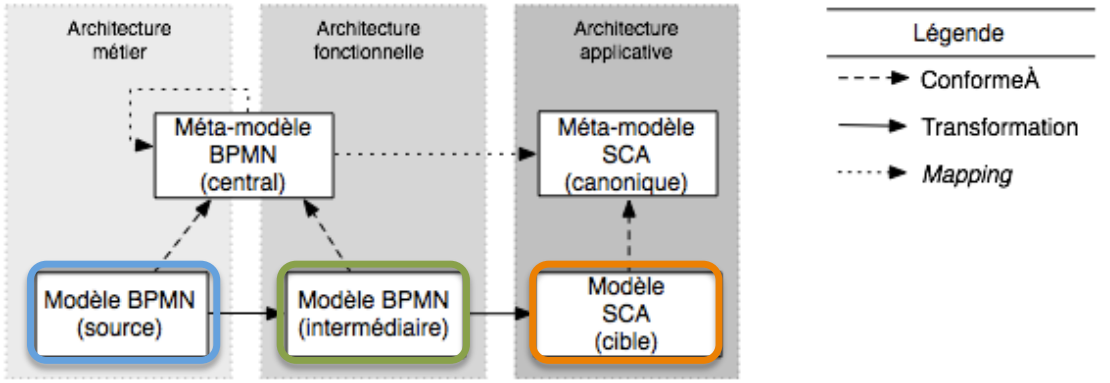
# Première contribution

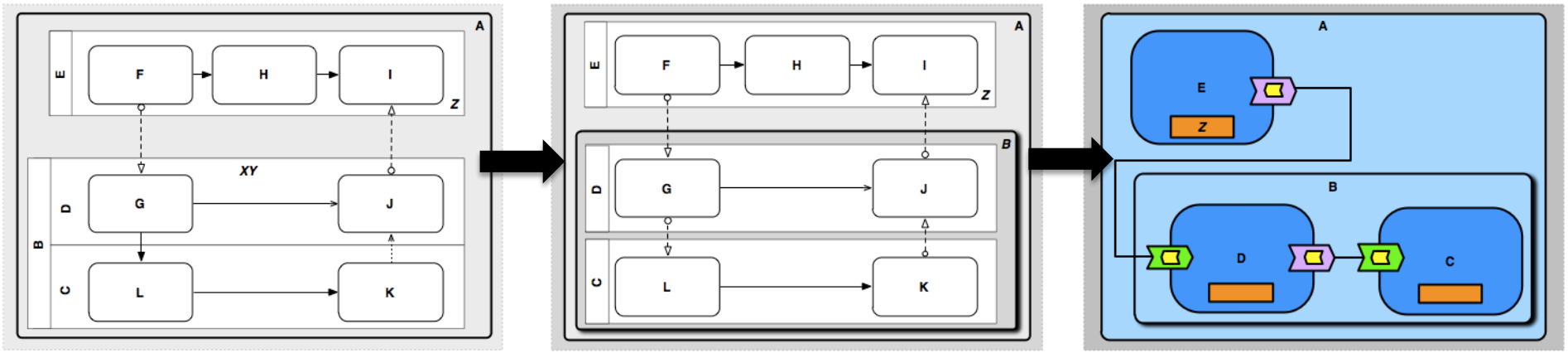
1. **Transformation** automatisée BPMN-SCA

   Tissage de modèles et alignement architectural

# Transformation de modèles

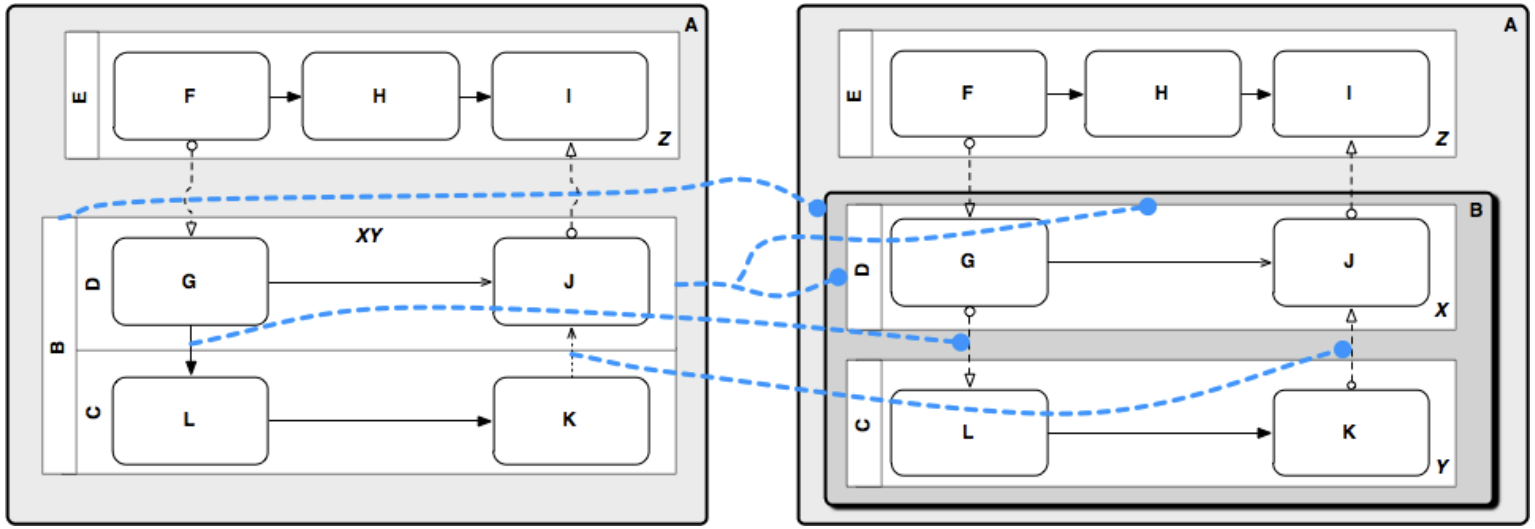
- ***Mapping*** : $BPMN_s$ - $BPMN_i$ – $SCA_c$
  - Règles de correspondance



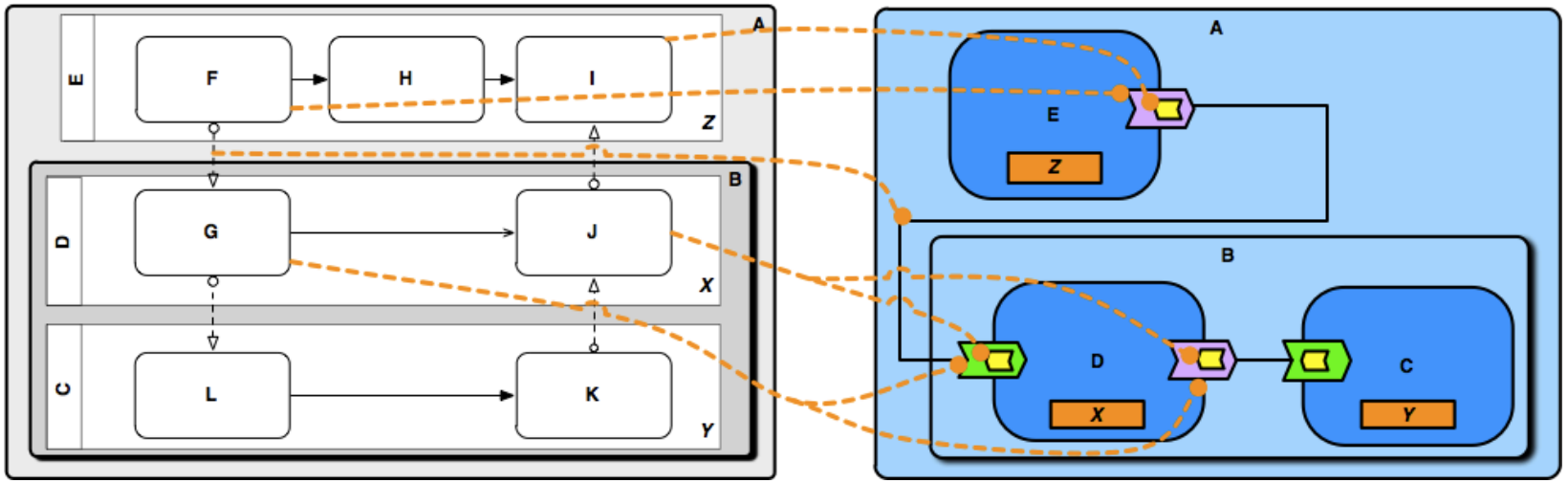
# Transformation $BPMN_s$ - $BPMN_i$

- Production d'un modèle $BPMN_i$ **intermédiaire**
  - Partitionner les flux fonctionnels croisés du modèle source
  - Règles de correspondances imbriquées et ordonnées



# Transformation $BPMN_i$ - $SCA_c$

- Production d'un modèle $SCA_c$ **cible**
  - Transformer les interactions de service en connecteurs
  - *Mapping* : impératif et multivalué



# Formalisation (transformation)

- Un outil réalise une multifonction composée $\boldsymbol{trans_s}$
  - Exécution du *mapping* ($BPMN_s$ - $BPMN_i$ – $SCA_c$)

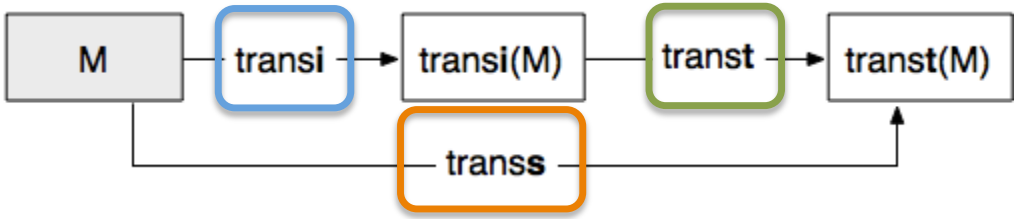
Le *mapping* est **impératif** et **multivalué** !

Comment **vérifier** qu'un outil éxecute une transformation « **correcte** » ?

# Transformation correcte

- ***trans***$_s$: transformation unidirectionnelle **non-bijective**
  - Propriétés de correction de l'exécution d'une transformation
    - **Uniforme, déterministe, valide, stable, autonome, idempotente**

$$trans_s(M) = trans_t \circ trans_i(M)$$



# Deuxième contribution

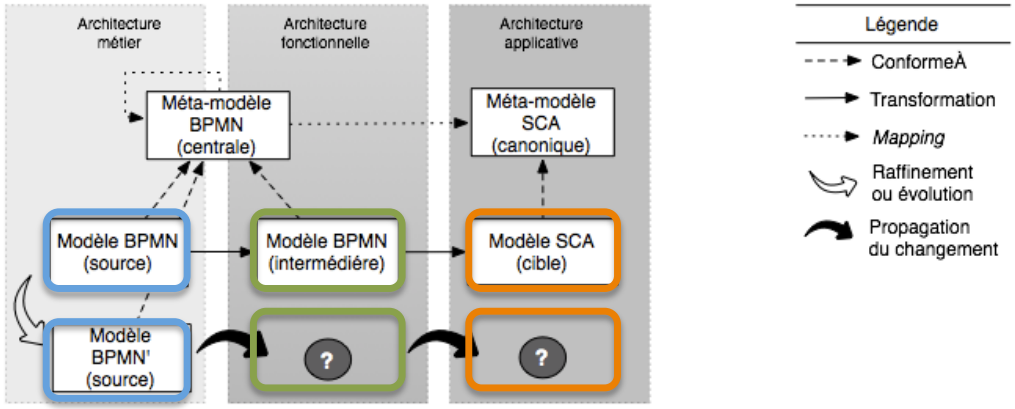
1. **Transformation** automatisée BPMN-SCA

   Tissage des modèles et alignement architectural

Si le modèle BPMN **source change,**
alors le modèle SCA **cible**
doit être « **synchronisé** » **!**

# Conception itérative

- **Générations** de modèles (actuels, cible, feuille de route)

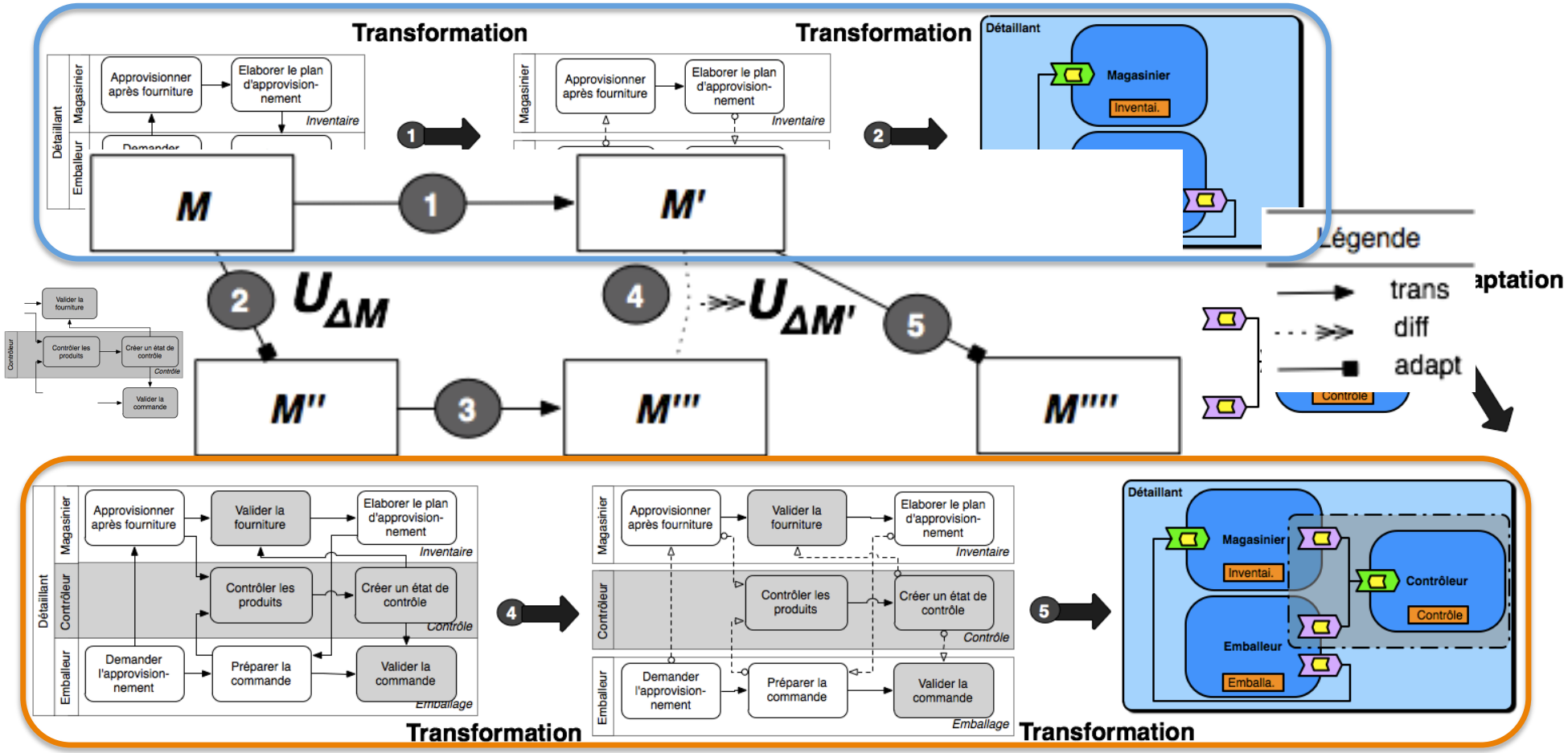



Pour des changements minimes …

Comment **synchroniser** les modèles d'une manière « **incrémentale** » ?

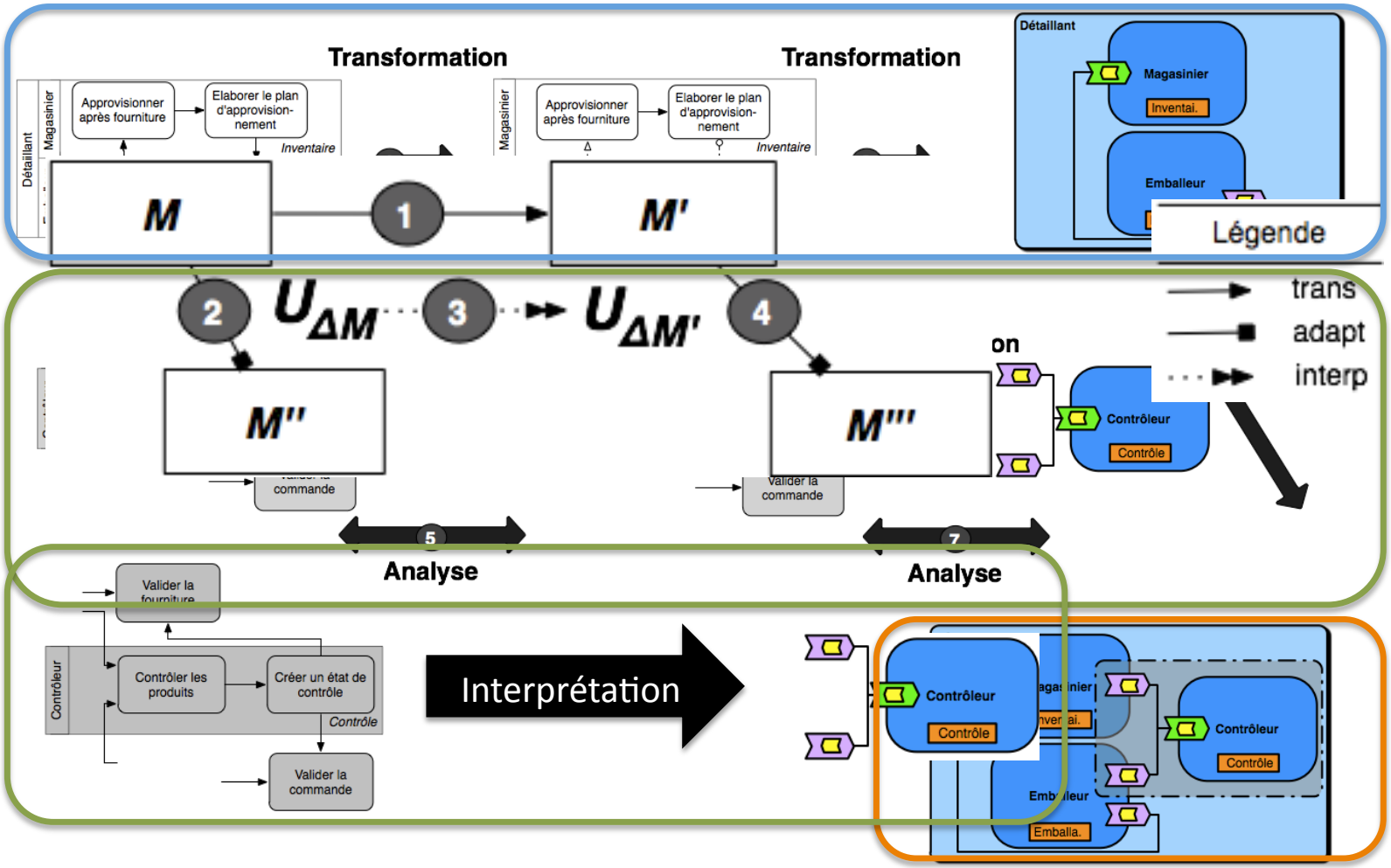
# Synchronisation différentielle

- **Transformer** et **différencier** les modèles



# Synchronisation incrémentale

- **Identifier** et **interpréter** les changements



# Synchronisation incrémentale

- Etapes de notre synchronisation

a. **Détection** des changements du modèle source
   **Identifier** les règles qui doivent être réévaluées

b. **Analyse de l'impact** sur le modèle cible
   **Interpréter** les changements et vérifier la cohérence

c. **Propagation du changement** au modèle cible
   **Appliquer** les changements en adaptant le modèle

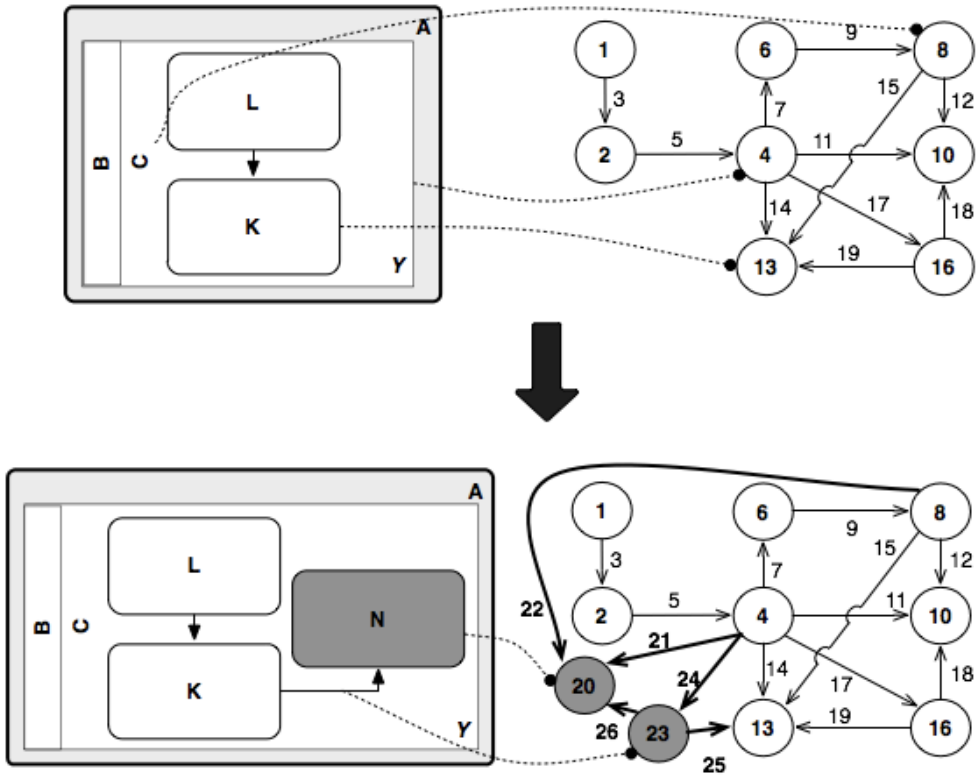
# Synchronisation incrémentale

- Etapes de notre synchronisation

a. **Détection** des changements du modèle source
   **Identifier** les règles qui doivent être réévaluées

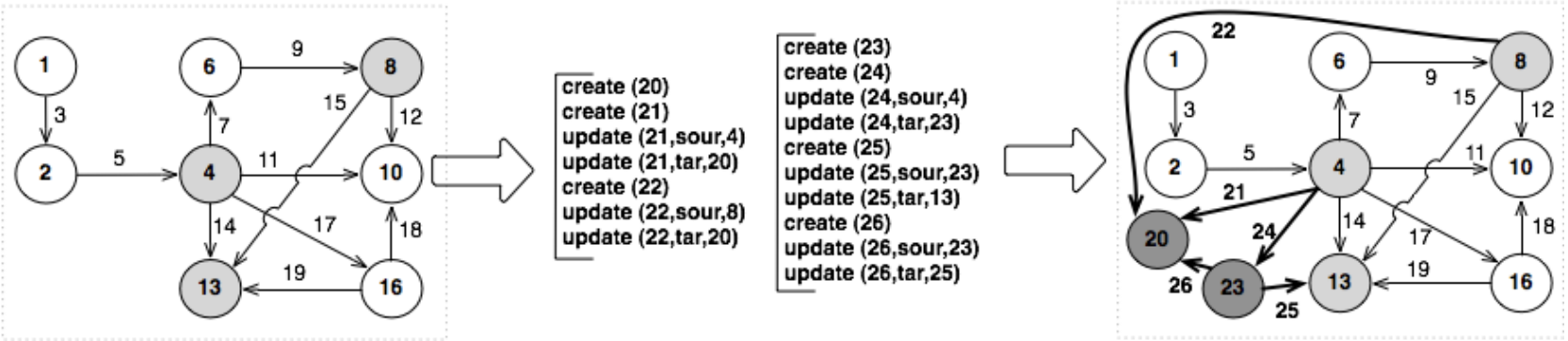
# Détection du changement (1/2)

- **Détecter** les changements structurels
  - Abstraction : modèle → **graphe** « étiqueté typé orienté »



# Détection du changement (2/2)

- **Grammaire** de réécriture des graphes
  - Réécriture → **productions conditionnelles**



# Sémantique opérationnelle

- Ensemble minimal et complet d'opérations élémentaires
  - Conditions d'application avec des **assertions**
  - Changement → **séquence** d'opérations
    - Bien formée (sans contradiction), valide (conformité)
    - Normalisée (sans redondance) : simplification

| Opération | Pré-condition | Invariant | Post-condition |
|---|---|---|---|
| $create(\phi)$ | $-\phi$ | $-(\phi, *, *)$ | $+\phi$ |
| $destroy(\phi)$ | $+\phi$ | $-(\phi, *, *)$ | $-\phi$ |
| $update(\phi, \mu, \nu)$ | | | |
| $undo(\phi, \mu, \nu)$ | | | |

| Assertion | Contredit | Description |
|---|---|---|
| $+\phi$ | $-\phi$ | Contradiction pour l'existence de l'élément |
| $+(\phi, \mu, \nu)$ | $-(\phi, \mu, \nu)$ | Contradiction pour une propriété de l'élément |
| $+(\phi, \mu, \nu)$ | $-(\phi, *, *)$ | |

| $\delta_l \circ \delta_k(\vec{r})$ | *create* | *destroy* | *update* | *undo* |
|---|---|---|---|---|
| *create* | . | **simplifier** | . | . |
| *destroy* | *create* | . | . | . |
| *update* | . | . | . | **simplifier** |
| *undo* | . | *destroy* | *update* | . |

# Synchronisation incrémentale

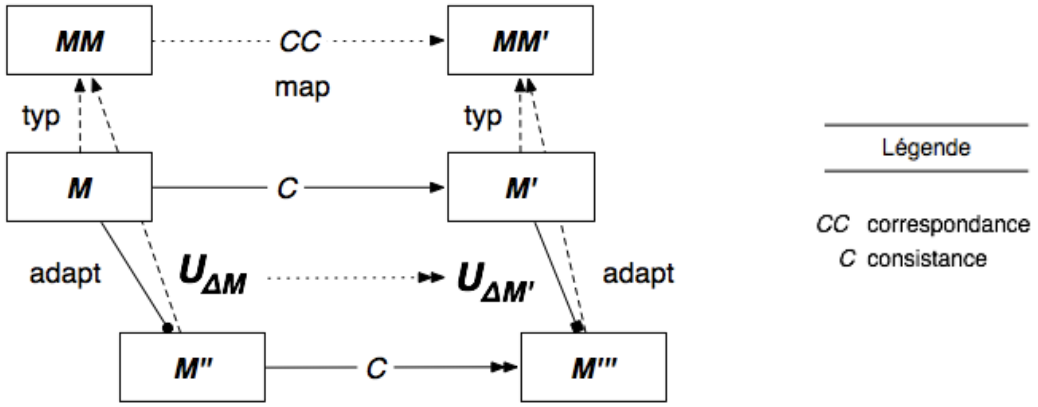
- Etapes de notre synchronisation

a. **Détection** des changements du modèle source
   **Identifier** les règles qui doivent être réévaluées

Comment « **interpréter** » les opérations du modèle source au modèle cible ?

# Analyse de l'impact (2/2)

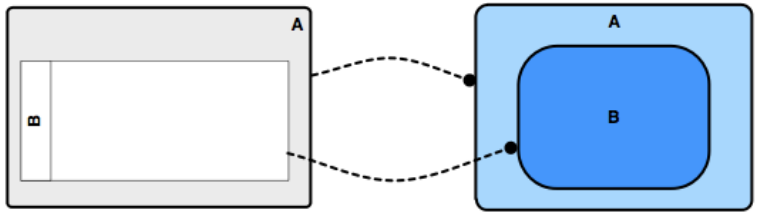
- **Vérifier** la cohérence des paires de modèles
  - Changements synchronisés → modèles cohérents



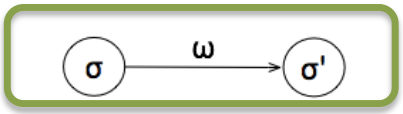

**Définition 16 (Changements synchronisés)** *Deux changements* $U_{\Delta_{\mathcal{M}}} \in \Delta_{\mathcal{M}}$ *et* $U_{\Delta_{\mathcal{M}'}} \in \Delta_{\mathcal{M}'}$ *modifiant respectivement deux modèles cohérents* $M \in \mathcal{M}$ *et* $M' \in \mathcal{M}'$ *tel que* $(M, M') \in C$ *sont dits synchronisés ssi ils produisent des modèles cohèrents :* $C(adapt(M, U_{\Delta_{\mathcal{M}}}), adapt(M', U_{\Delta_{\mathcal{M}'}}))$.

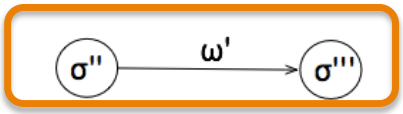
# Analyse de l'impact (1/2)

- **Interpréter** les séquences d'opérations
  - Correspondance : $(Collaboration, Composite), (Participant, Component), (Contain, Contain) \in \mathcal{CC}.$




  - Cohérence :




$$update(\sigma, typ, Collaboration), update(\sigma', typ, Participant), update(\omega, typ, Contain),$$
$$update(\omega, sour, \sigma), update(\omega, tar, \sigma'),$$

  - Consistance : $(\sigma, \sigma''), (\sigma', \sigma'''), (\omega, \omega') \in \mathcal{C}.$

Règle d'interprétation

$$update(\sigma'', typ, Composite), update(\sigma''', typ, Component), update(\omega', typ, Contain),$$
$$update(\omega', sour, \sigma''), update(\omega', tar, \sigma''').$$

# Synchronisation incrémentale

- Etapes de notre synchronisation

a. **Détection** des changements du modèle source
   **Identifier** les règles qui doivent être réévaluées

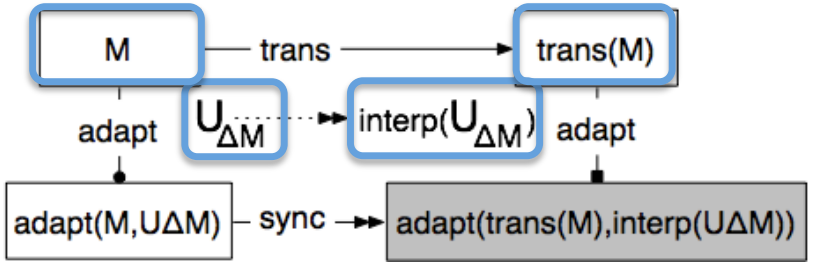
b. **Analyse de l'impact** sur le modèle cible
   **Interpréter** les changements et vérifier la cohérence

Comment « **adapter** » le modèle cible
avec les **opérations interprétées** ?

# Adaptation du modèle cible

- Synchronisation incrémentale
  - **Equivalente** à une transformation
- Mais la **complexité**
  - Proportionnelle **aux changements** du modèle source
  - Non au modèle source modifié

$$sync(M, trans(M), U_{\Delta_M}) = trans \circ adapt(M, U_{\Delta_M}) = adapt(trans(M), interp(U_{\Delta_M}))$$



# Formalisation (synchronisation)

- Un outil réalise une multifonction composée ***sync***$_s$
  - Réévaluation du *mapping* ($BPMN_s$ - $BPMN_i$ – $SCA_c$)

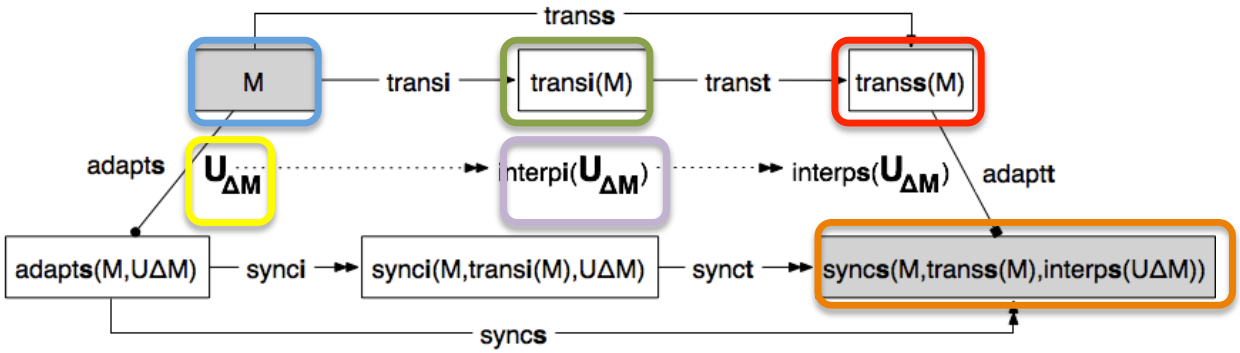
Le *mapping* est **impératif** et **multivalué** !

Comment **vérifier** qu'un outil éxecute une synchronisation « **correcte** » ?

# Synchronisation correcte

- ***sync*$_s$** : synchronisation unidirectionnelle **non-bijective**
  - Propriétés de correction de l'exécution d'une synchronisation
    - **Uniforme**, **déterministe**, **valide**, **stable**, **autonome**, **idempotente**

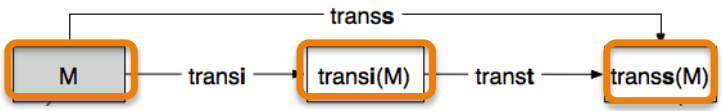
$$sync_s(M, trans_s(M), U_{\Delta_M})) = sync_t(trans_i(M), trans_s(M), interp_i(U_{\Delta_M}))$$



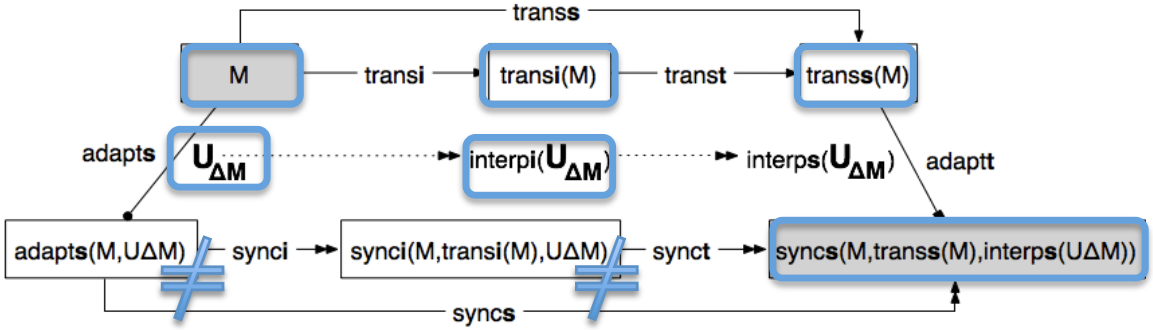
# Démonstrations théoriques

- Définitions, théorèmes, démonstrations

  - **Transformation** $BPMN_s$-$BPMN_i$-$SCA_c$ : $trans_s = trans_t \circ trans_i$




  - **Synchronisation** $BPMN_s$-$BPMN_i$-$SCA_c$ : $sync_s \neq sync_t \circ sync_i$



# Troisième contribution

1. **Transformation** automatisée BPMN-SCA

   Tissage de modèle et alignement architectural

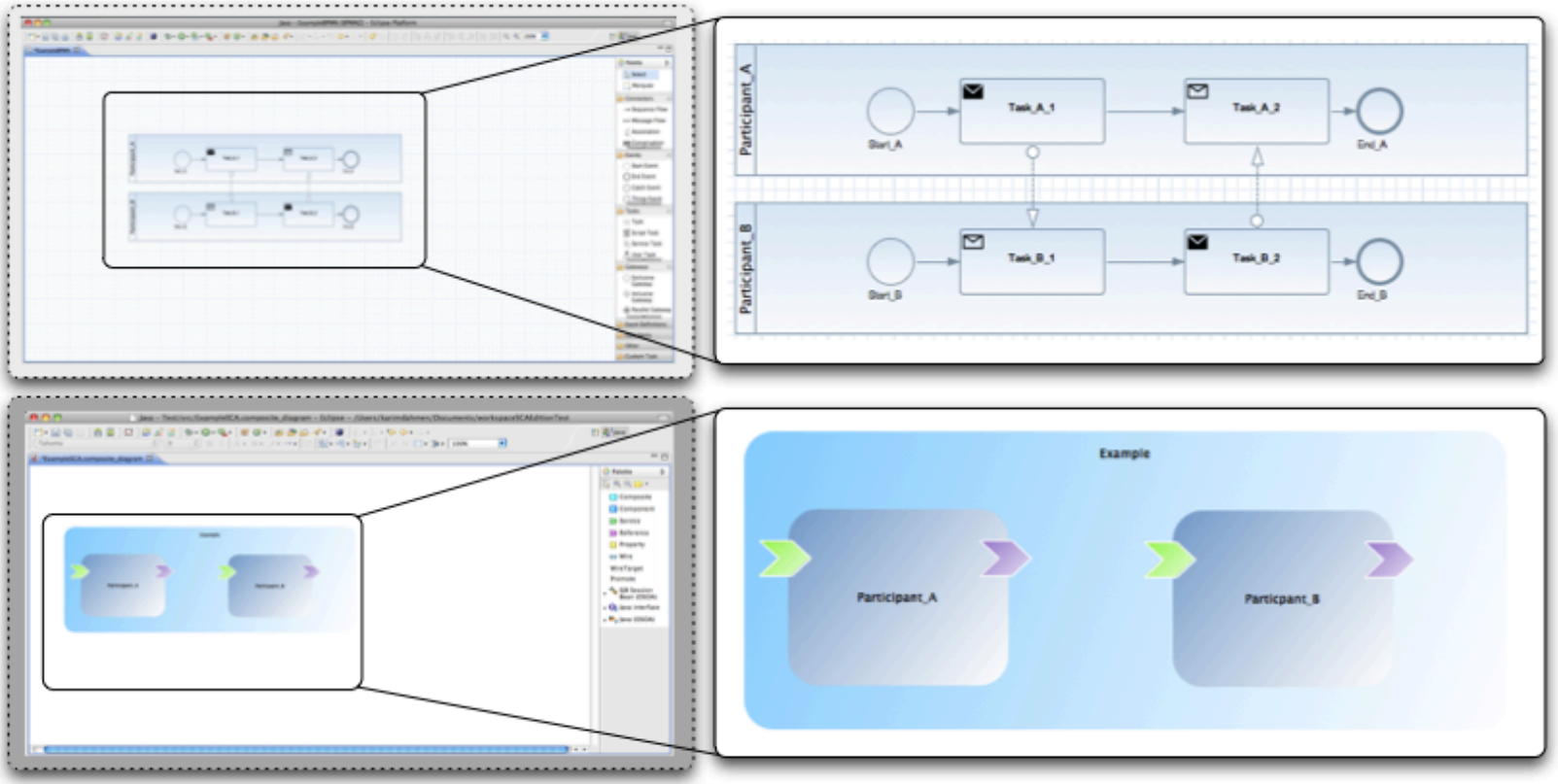
2. **Synchronisation** incrémentale BPMN-SCA

   Propagation et alignement fonctionnel

Comment **valider** les résultats théoriques
et « **mesurer** » le **changement** des modèles ?

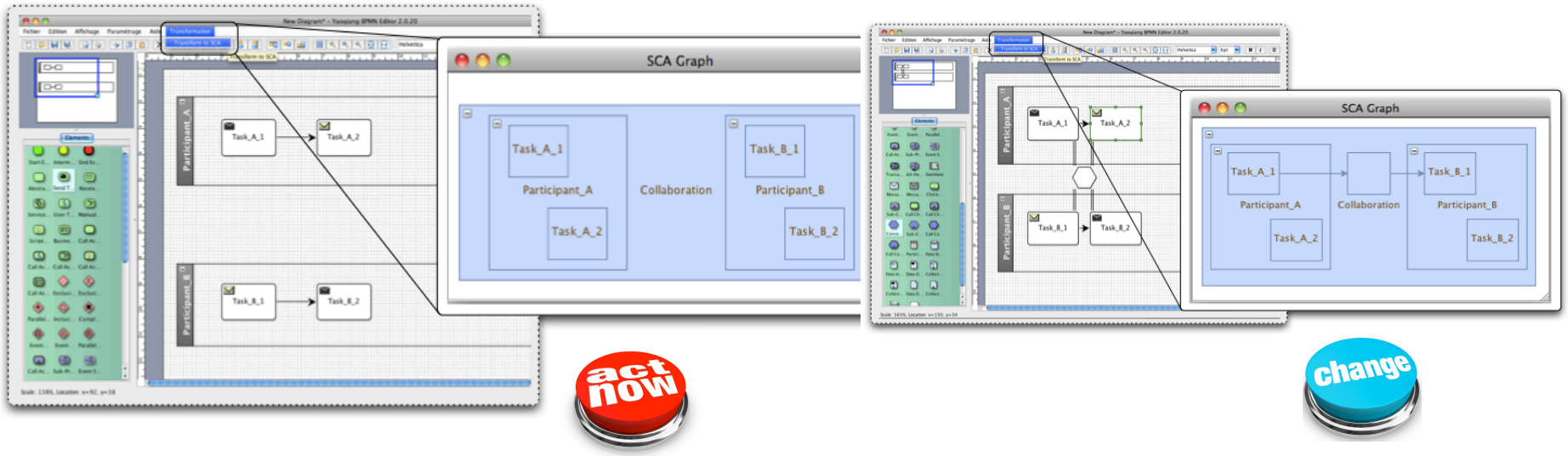
# Mise en œuvre (transformation)

- **Environnement** pour valider la transformation
  - Intégré à *Eclipse Modeling Framework* (ATL)



# Mise en œuvre (synchronisation)

- **Synchroniseur** pour valider la synchronisation
  - Utilisant *BPMN 2.0 Editor* et *Drools*



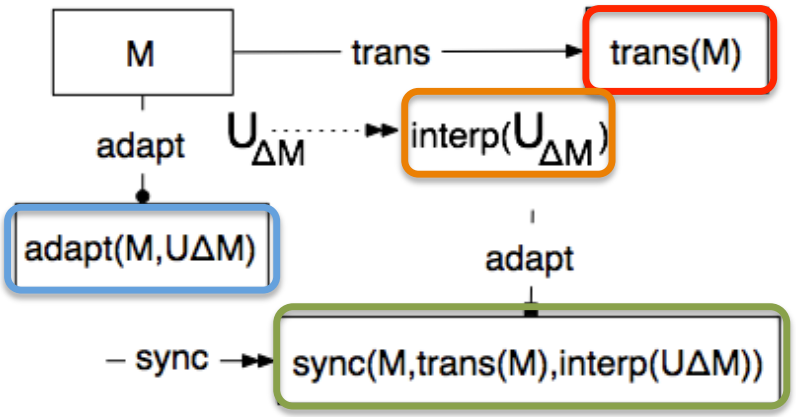

Comment « **mesurer** »
le changement des modèles ?

# Métriques du changement

- Quantifier le changement des modèles
  - **Métriques** : **taille** (normalisée), sémantique et complexité
  - **Unité** : opération élémentaire dans une séquence

$size(U_{\Delta_M})$ : nombre de *create* et *destroy* (taille du changement)

$\overline{size}(M) = size(M)$ : nombre d'opérations après normalisation (cardinal)




Quelle est « **l'efficacité** »
de l'**interprétation des changements**
par rapport à
la **transformation de modèles** ?

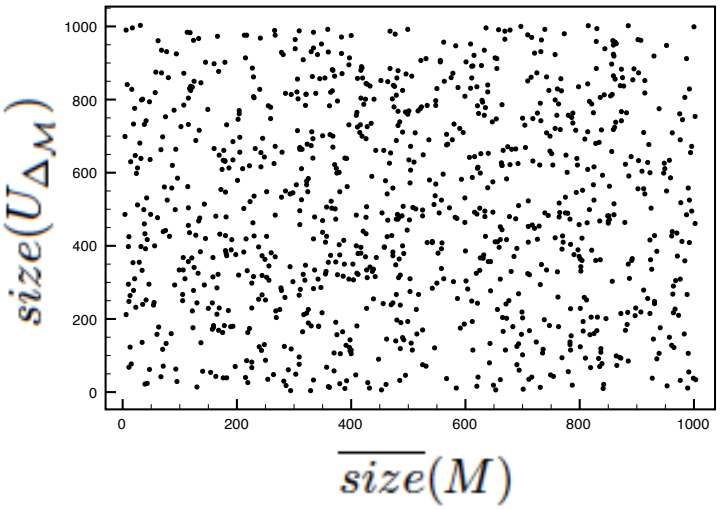
# Protocole de l'expérimentation

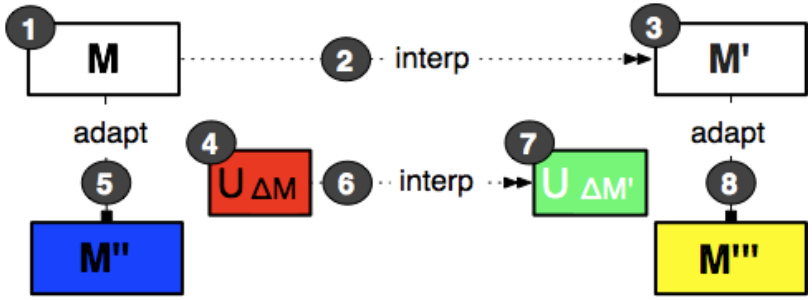
- Objectif
  - Observer le **comportement** de la fonction d'interprétation
- Hypothèses
  - Modification sur un modèle → séquences d'opérations
  - Modèle vide → séquence d'opération vide
  - Modèle modifié → séquence d'opérations « dénombrables »

# Jeu de tests

- Séquences aléatoires
  - 1000 **modèles** avec (0 < $\overline{size}(M)$ ≤ 1000)
  - 1000 **changements** avec (0 < $size(U_{\Delta_M})$ ≤ 1000)

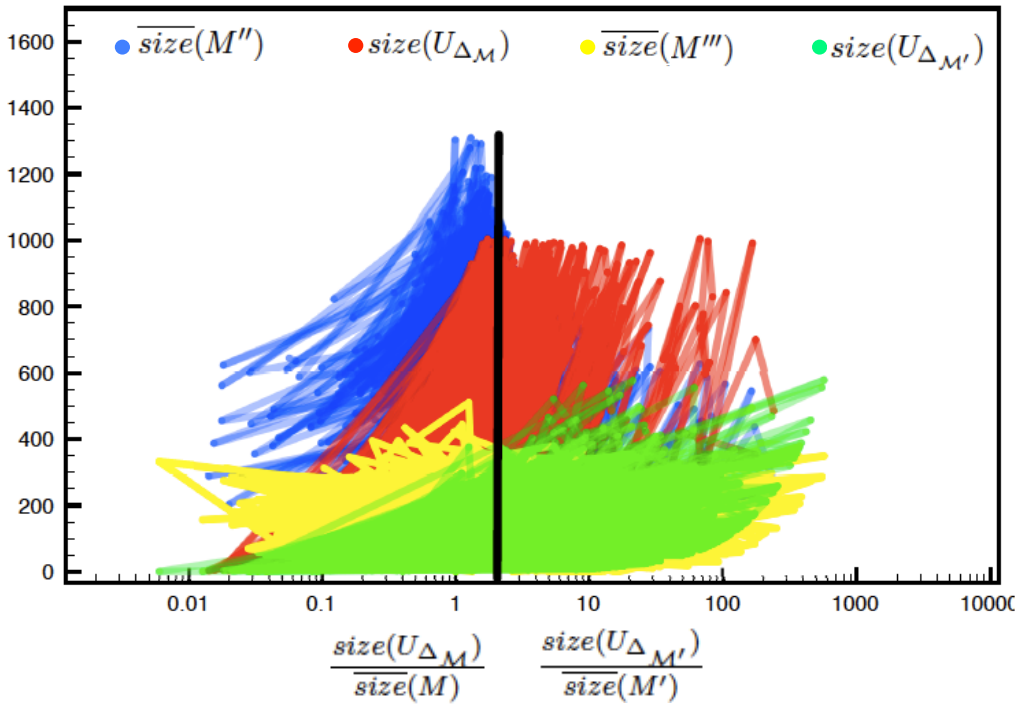
[Aléatoires : opérations (*create, destroy*) qui activent ou non le *mapping*]

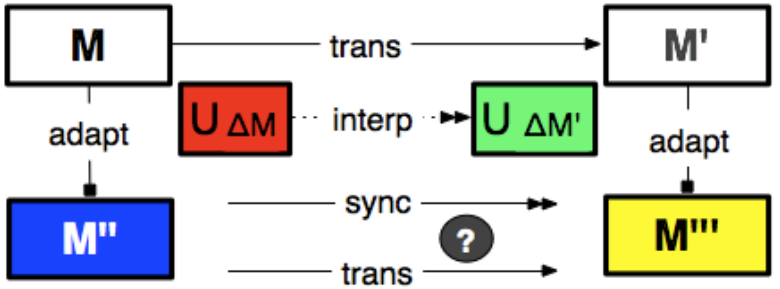


# Observations (interprétation)

- Efficace tant que : $size(U_{\Delta_{\mathcal{M}}}) \leq \overline{size}(M)$ → $size(U_{\Delta_{\mathcal{M}'}}) \leq \overline{size}(M')$



# Questions de départ

- **Méthodologie** → démarche(s) + outil(s)
  - Pour **préserver** l'alignement → cohérence
  - Pour **changer** les modèles → itérative
  - Pour **propager** le changement → propagation
  - Pour **évaluer** l'impact → évaluation

# Bilan des contributions

- **Démarche d'analyse** (transformation BPMN - SCA)
  - Cohérence des architectures, traçabilité et lisibilité du SI
- **Démarche de conception** itérative (synchronisation)
  - Propagation incrémentale : détection, analyse, adaptation
- **Outillage** pour l'évaluation de l'impact du changement
  - Simulation des alternatives de conception

# Maîtrise de l'impact du delta

- **Procédures** pour **concevoir** et **modifier** une solution
  - Cadre de **gouvernance des changements** des processus métiers sur les architectures orientées services (SOA)
  - Approche **dirigée** par les modèles

# Limites

- Imputables **aux langages** BPMN et SCA
  - L'expressivité du modèle SCA dépend du modèle BPMN
- Imputables **à notre approche** incrémentale
  - Le maintien d'un contexte d'exécution de la transformation

# Extensions envisageables

- Etendre l'**étude** de l'impact du changement
  - A d'autres notations et des changements de complexité réelle
- Enrichir le ***mapping*** BPMN - SCA
  - Processus « exécutables » en configurations « déployables »
    - Ex. : donnés en SDO, conversationnel en policyset (WS-Policy)
- Définir une synchronisation **bidirectionnelle**

# Merci de votre attention

**L'efficacité** de la conception
provient d'une **méthode agile** pour **étendre ou adapter**
des solutions **durables**
en utilisant des **services flexibles**
et en favorisant la **réutilisation** !

Mosaïque de verre d'après "*Jeune fille devant un miroir*" de PICASSO