Introduction à l'approche ADM

Modernisation du patrimoine logiciel par les modèles

GDR-GPL # Action IDM
Rennes, 2012

Olivier Le Goaer
olivier.legoaer@univ-pau.fr

Agenda

- Introduction et problématique
- Modernisation par les modèles
- ADM : La boîte à outils de l'OMG
- L'exemple de BluAge Reverse
- Conclusions

Introduction et problématique

Vers la modernisation du SI

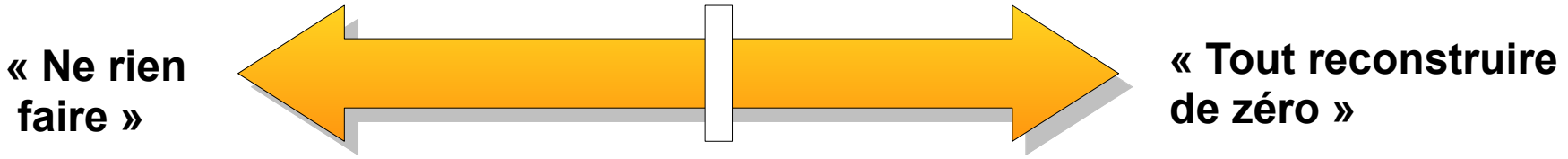
- Une prépondérance des SI
  - Grands comptes (banques, assurance, mutuelle, …)
  - SI devenus stratégiques, voir critiques pour le métier
- Des situations d'urgence
  1. Obsolescence technologique rapide, effets de mode
  2. Évolutions rapides des entreprises (structure/réglementation)
  3. Départ des « sachant » des entreprises
- Un défi industriel et scientifique majeur
  - Coûts importants, ROI difficile à évaluer
  - Verrous technologiques

Logiciels patrimoniaux (*legacy*)

- Patrimoine logiciel recouvre 2 notions :
  - Valeur : le SI est un **actif** de l'entreprise
  - Temps : le SI hérité est devenu **vieux**
- Modernisation du patrimoine logicielle
  - Conserver la valeur du SI tout en le mettant au goût du jour
- Projets de modernisation du SI
  - Migration du SI : le projet vise à re-localiser le SI sur une plateforme technologique récente. Typiquement iso-fonctionnelle.
  - Refonte du SI : le projet vise à rebâtir le SI pour repartir sur de meilleures bases. Les fonctionnalités peuvent être repensées.
  - ...

Se préparer à évoluer

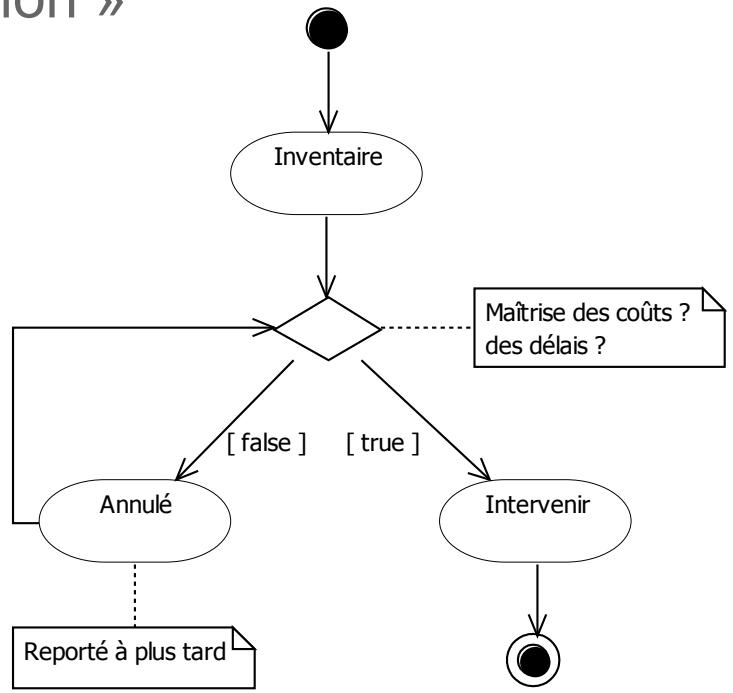
- Une option raisonnable



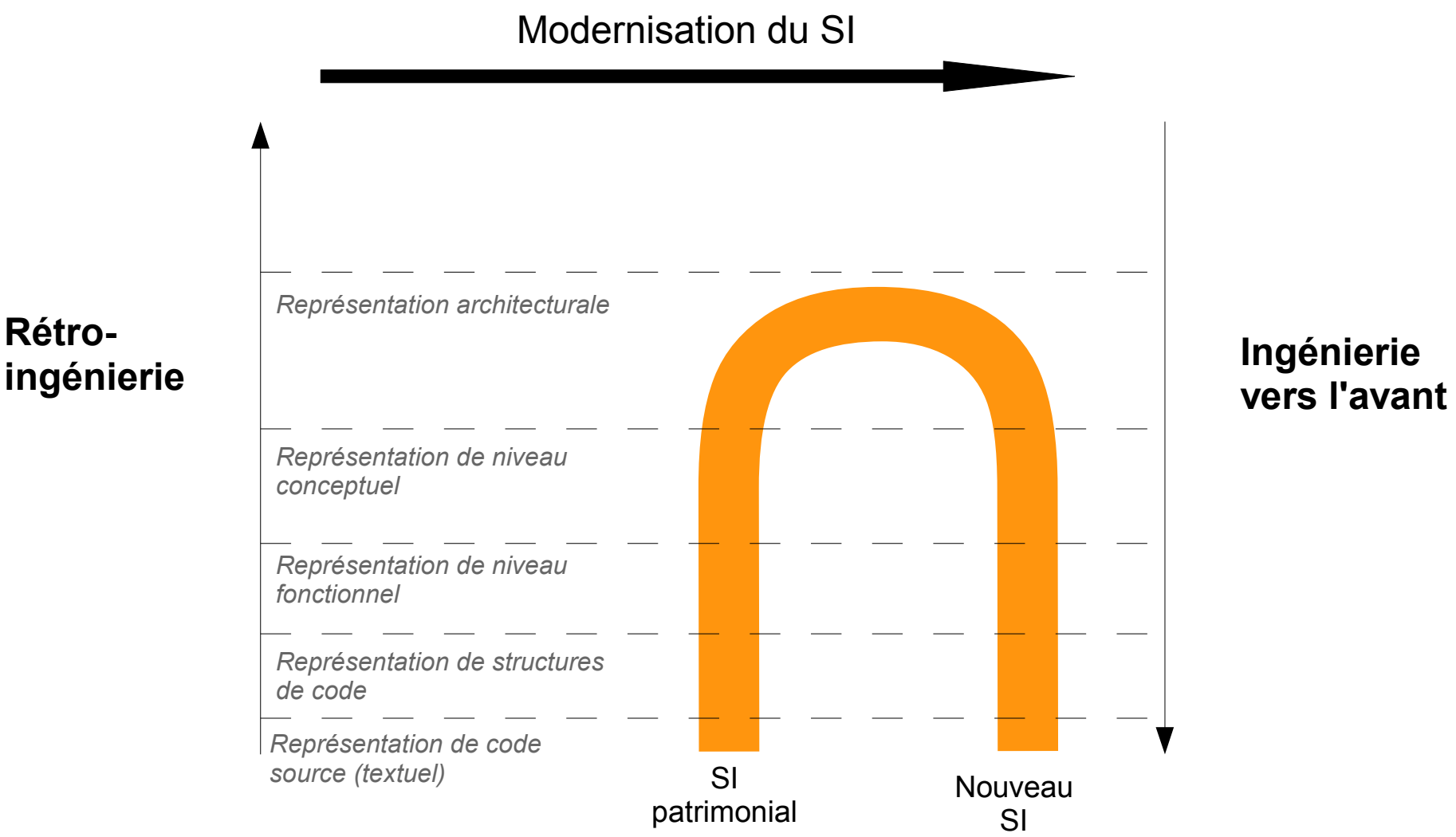
- Faire le point sur la situation
  - Inventorier le portefeuille des logiciels de chaque entreprise
  - L'occasion de (re)découvrir ce qui se cache sous l'empilement de couches logicielles (*« Software Archeology »*)
- Prendre une décision
  - Comprendre en vue d'estimer la complexité, les coûts
  - Avis d'experts, d'analystes

AGL pour la modernisation

- Atelier pour assister les projets de modernisation logicielle
  - « Computer-aided software modernization »
  - Haut-degré d'automatisation
- A chacune des 3 étapes clés
  - Inventaire
  - Décision
  - Intervention
- Bénéfices attendus
  - Fiable, prévisible et rentable



Modèle du « fer à cheval »

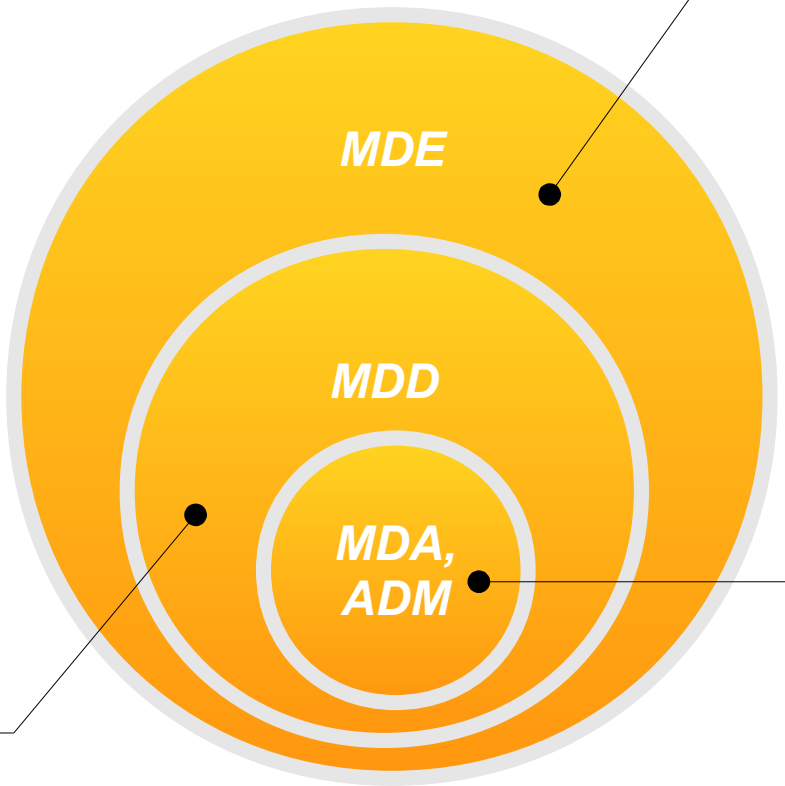


Point de vue ingénierie

1. Retro-ingénierie (*reverse-engineering*)
   - Objectifs : inventorier les artefacts du patrimoine logiciel
   - Tâches : auditer, fouiller, cartographier, ...
2. Ré-ingenierie (*re-engineering*)
   - Objectifs : revitaliser ces artefacts
   - Tâches : améliorer, nettoyer, refactoriser, remplacer, ...
3. Ingénierie avant (*forward-engineering*)
   - Objectifs : cibler de nouvelles plateformes technologiques, intégrer des notions nouvelles
   - Tâches : générer du code, documenter, tester, ...

Modernisation par les modèles

Le MD*

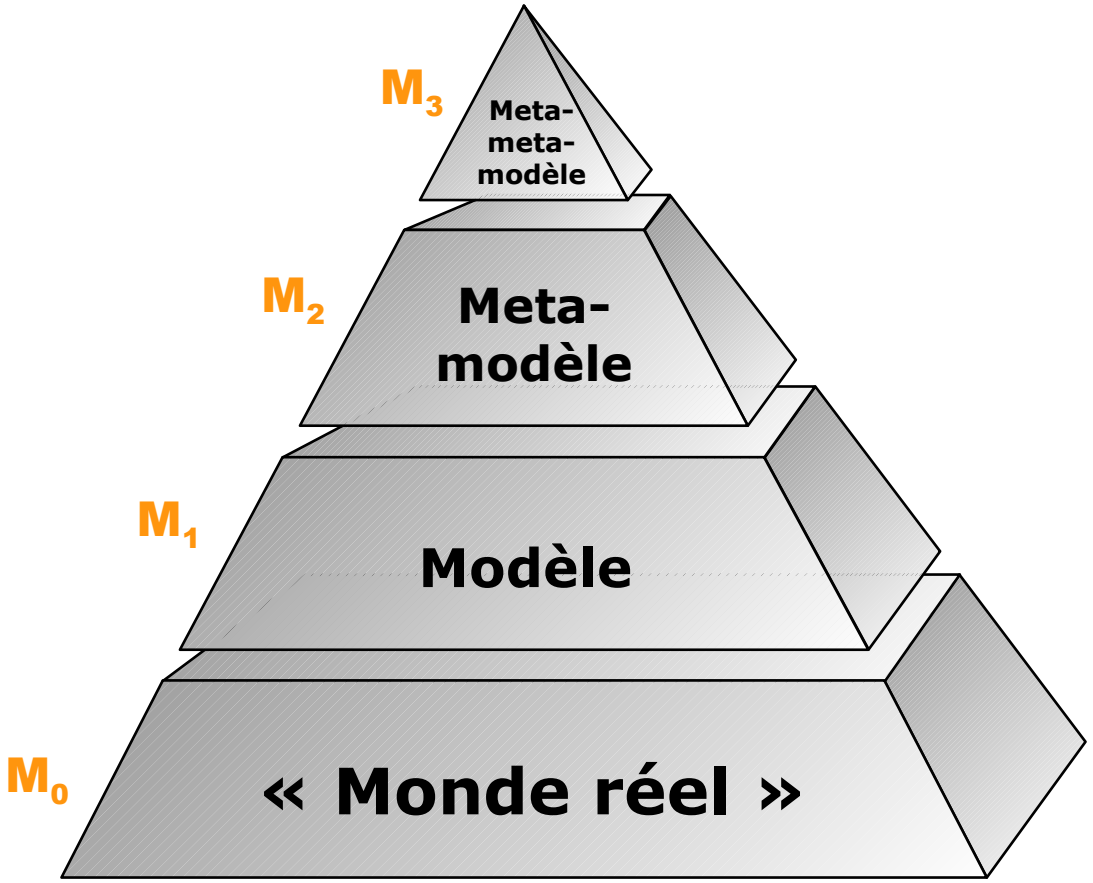


Toute activité qui considère les modèles comme citoyens de première classe (à traiter en priorité)

Vision de l'*OMG* : raffinement des modèles (PIM/PSM)

Les modèles couvrent le cycle de développement et de maintenance des systèmes logiciels

La pile de métamodélisation



Le MOF

Le métamodèle UML et d'autres métamodèles

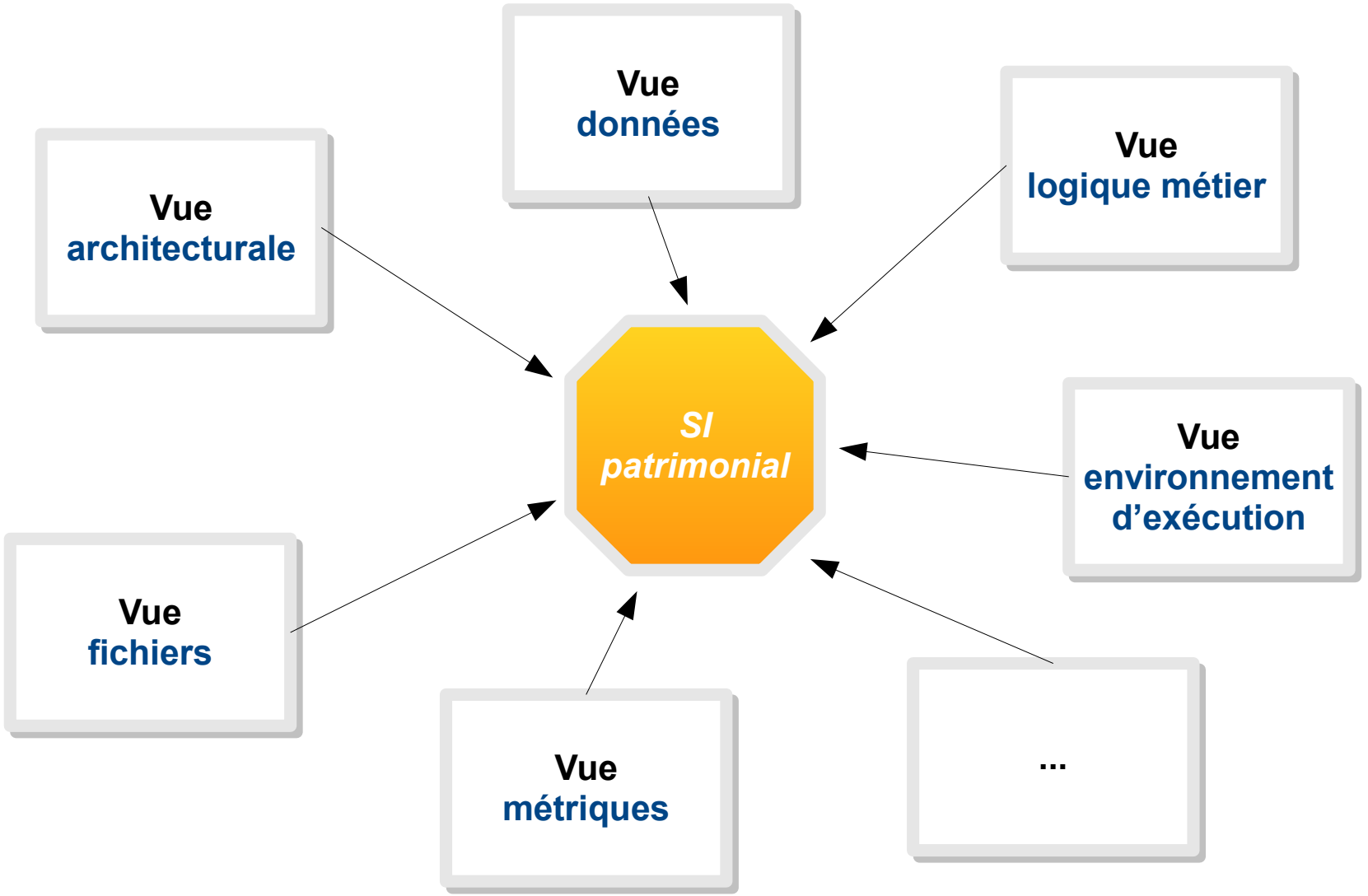
Des modèles UML et d'autres modèles

Usages variés de ces modèles

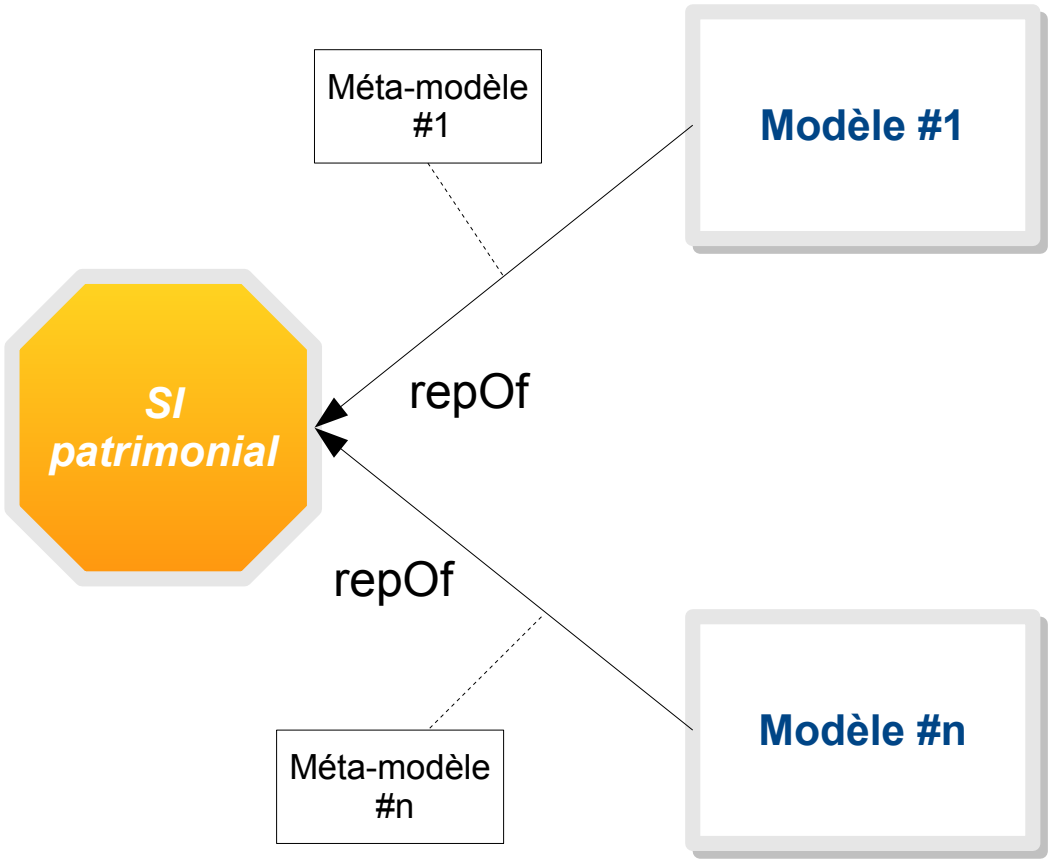
Model-driven Modernization

- La modernisation est un processus complexe
  - Nécessite une forte montée en abstraction !
- L'IDM comme technologie idéale
  - Capturer un point de vue sur le SI sous la forme d'un **modèle** (conforme à un **méta-modèle**)
  - Le traiter sous la forme d'une **transformation**
- Un projet de modernisation fera intervenir de nombreux de modèles qu'il faudra gérer
  - Méga-modélisation
  - Bus à modèles
  - ...

Approche multi-vues



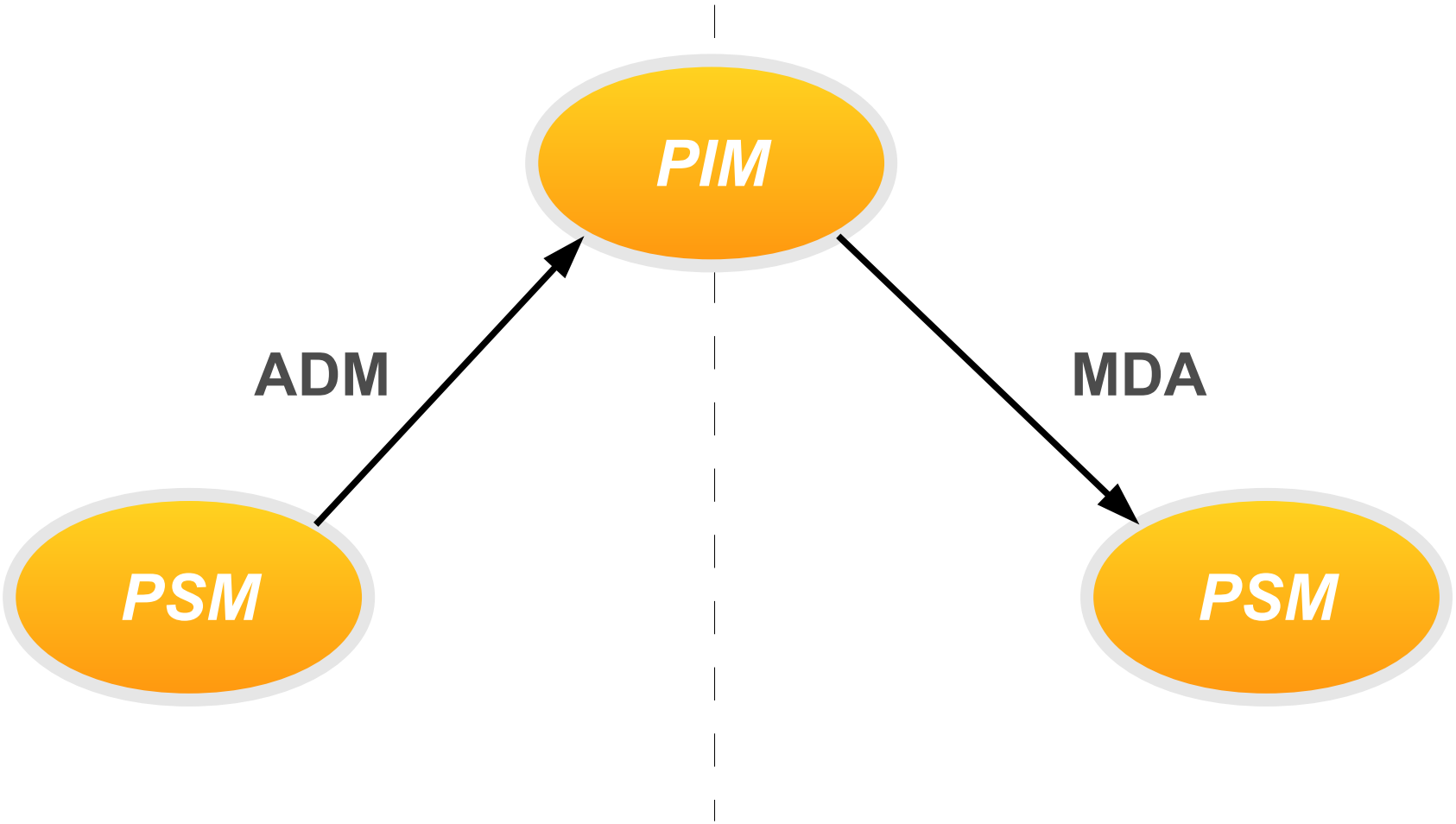
Multi-vues par les modèles



Approches ADM et MDA de l'OMG

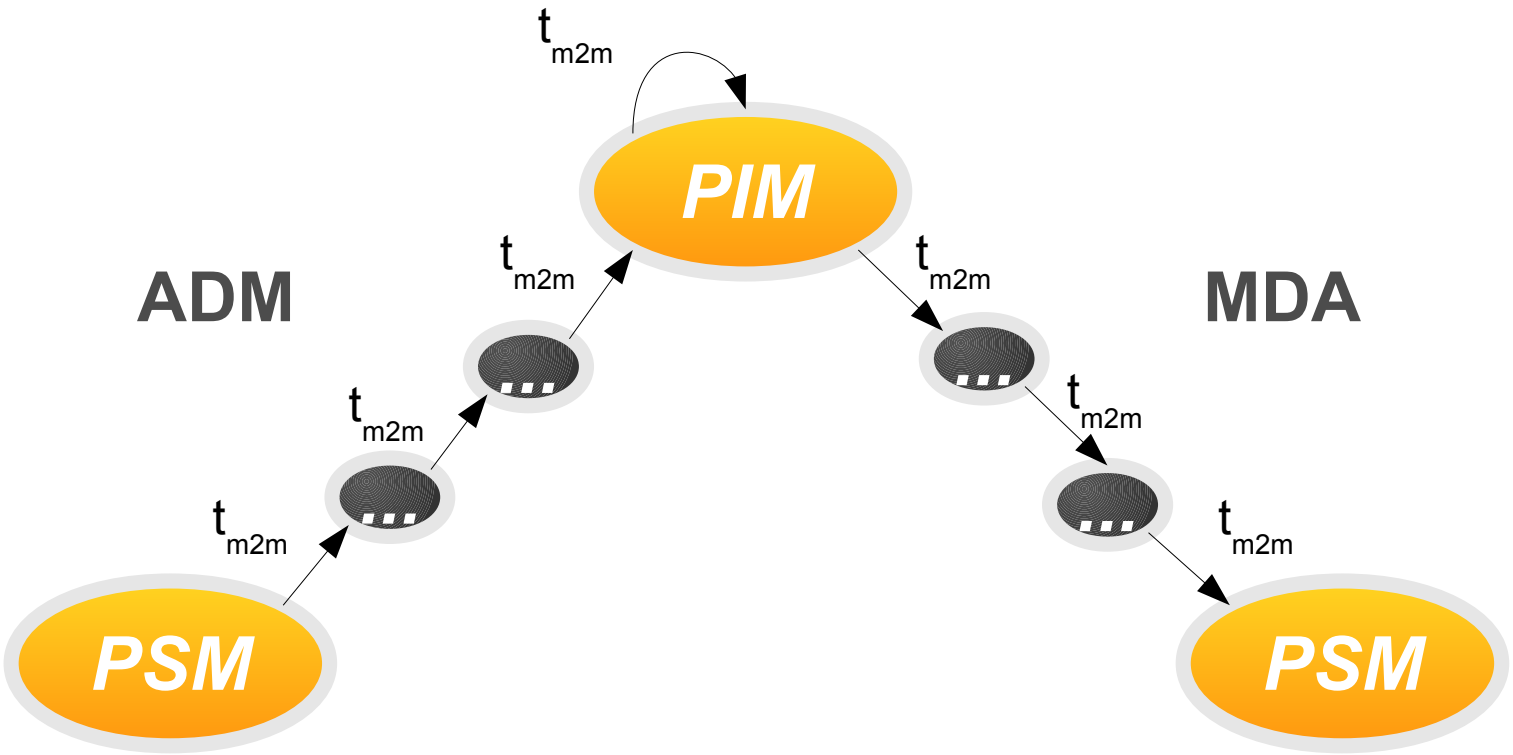
- Le patrimoine a été l'un des gros obstacles à la percée du MDA
  - Tout ré-écrire « from scratch » n'est pas viable.
  - Que fais-t-on de l'existant (legacy) ? Il faut le réutiliser...
- Effet miroir
  - ADM est le processus bottom-up (rétro-ingénierie)
  - MDA est le processus top-down (ingénierie vers l'avant)
- Forces
  - Utilisation de standards
  - Complétude (processus end-to-end)
  - Interopérabilité entre les outils

La modernisation selon l'OMG



Discrétisation du processus

- Chaîne de transformations
- Modèles (et méta-modèles) intermédiaires



ADM : La boîte à outils de l'OMG

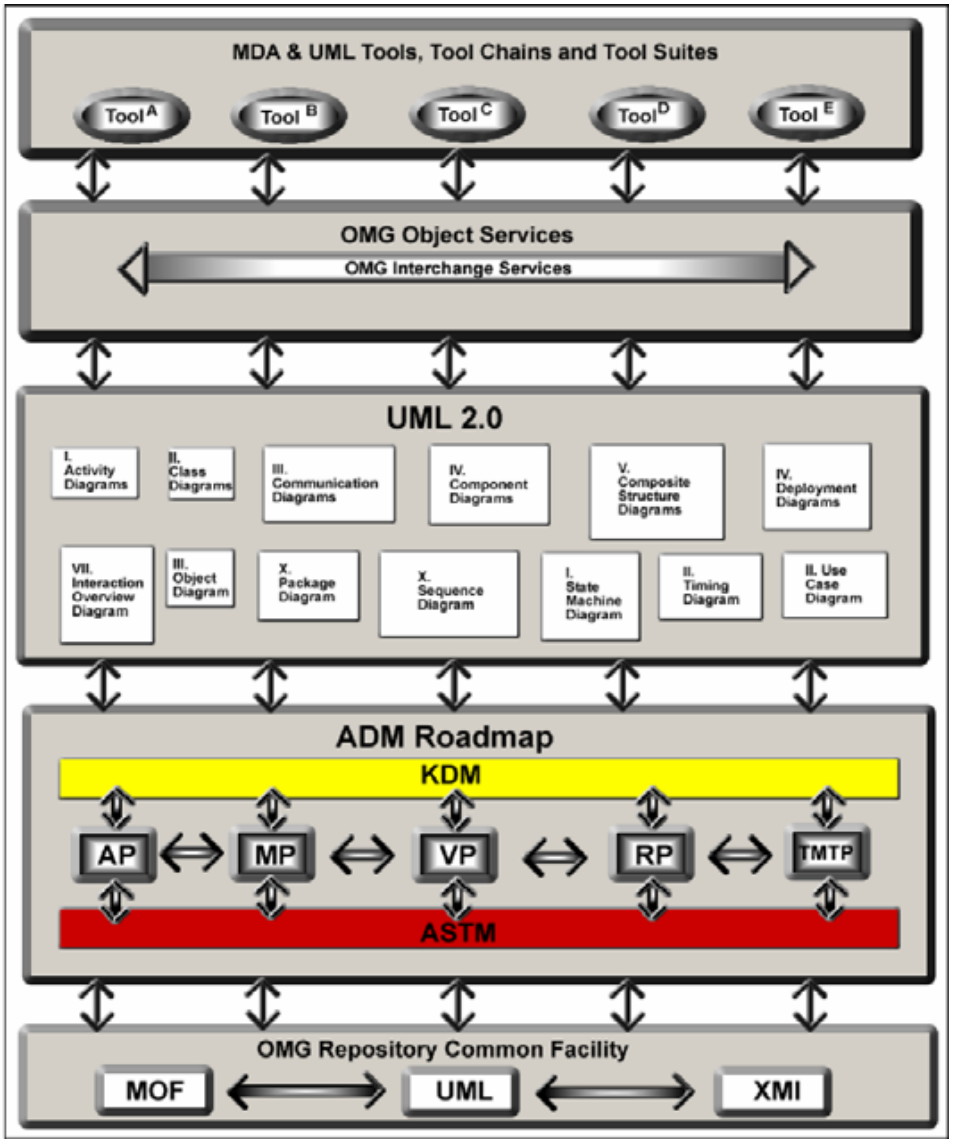
L' « ADM Task Force » de l'OMG

- Groupe créé en 2003
- Mission
  - « Create specifications and promote industry consensus on modernization of existing applications »
- Feuille de route
  - 7 appels à propositions (*Request For Proposal)*
  - 12 scénarios de modernisation
- Pilotage
  - Djenana Campara et William Ulrich
- Processus de standardisation non finalisé...

Les 7 RFPs de la feuille de route

1. Knowledge Discovery Meta-Model (KDM) Package
2. Abstract Syntax Tree Meta-Model (ASTM) Package
3. Analysis Package (AP)
4. Metrics Package (MP)
5. Visualization Package (VP)
6. Refactoring Package (RP)
7. Target Mapping & Transformation Package (TMTP)

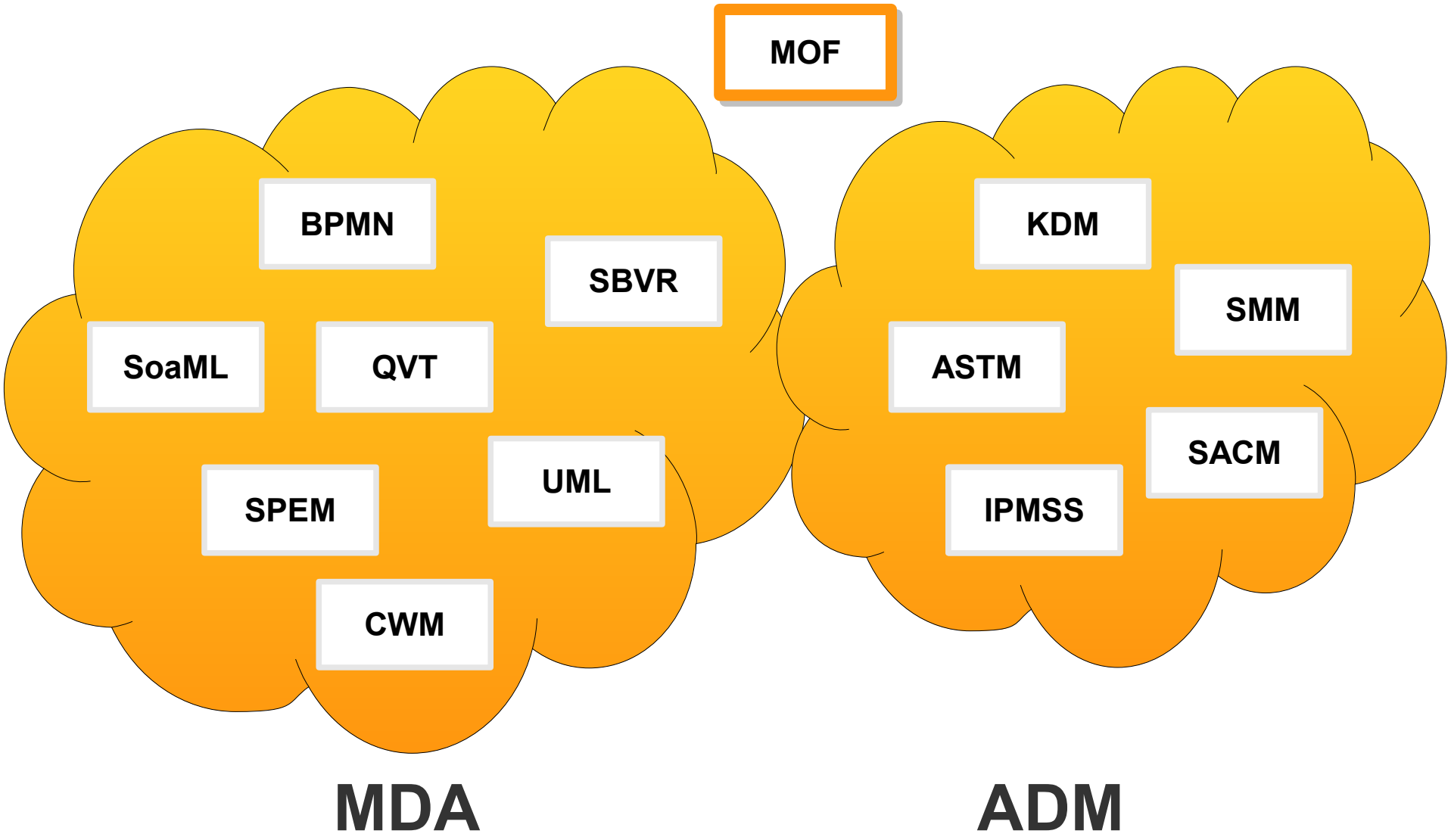
Vision d'ensemble de l'OMG



ADM, juste une boite à outils ?

- L'OMG offre uniquement des spécifications
  - Leurs implémentations ne sont pas fournies
  - Spécification largement informelles (i.e. langage naturel)
- L'OMG ne définie aucune méthodologie
  - Pas de «Unified Process» pour la modernisation logicielle
  - Les 12 scénarios se font attendre (devaient servir de guides !)
- Il manque le mode d'emploi...
  - Phase de découverte et instanciation des modèles ?
  - Transformations/tâches nécessaires ?
  - Articulation des différents standards entre eux ?

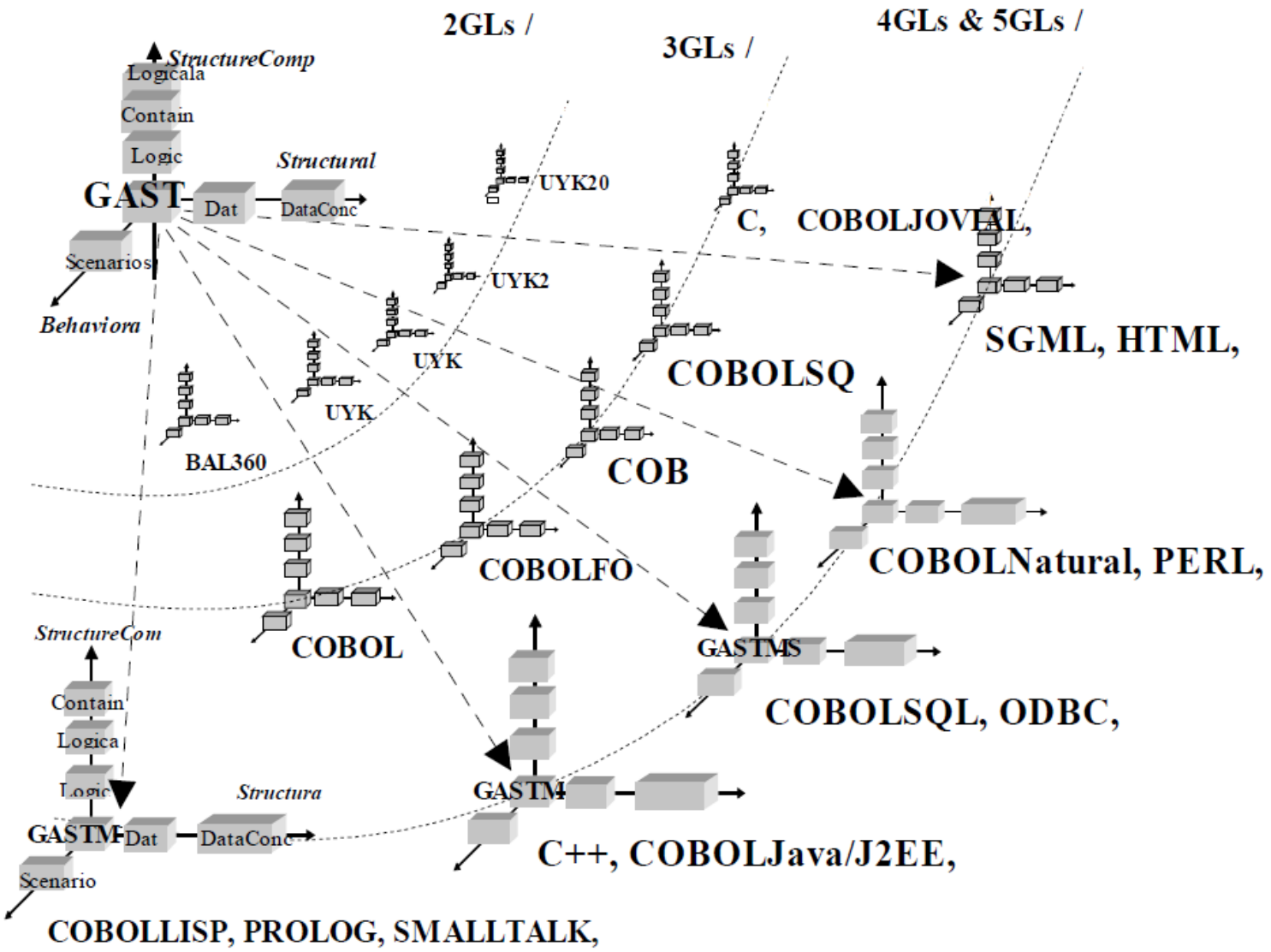
Écosystème OMG (metamodèles)



Focus sur ASTM

- Abstract Syntax Tree Metamodel (ASTM)
  - Prévu pour une modélisation bas-niveau, fidèle au code source
  - Supporte différentes familles de langages : programmation essentiellement, d'interrogation, de transformation, …
- ASTM = GASTM + SASTMs
  - GASTM (Generic ASTM) : metamodèle commun pour représenter un code source. Vise à unifier les langues syntaxiques (2 à 5 GL)
    - *Méta-types : DeclarationOrDefinition, Expression, Literal*, **etc.**
  - SASTM (Specific ASTM) : metamodèle dédié à chaque langage
    - *Méta-types : TernaryOperator (Java), MoveStatement (COBOL),* **etc.**

ASTM et langages

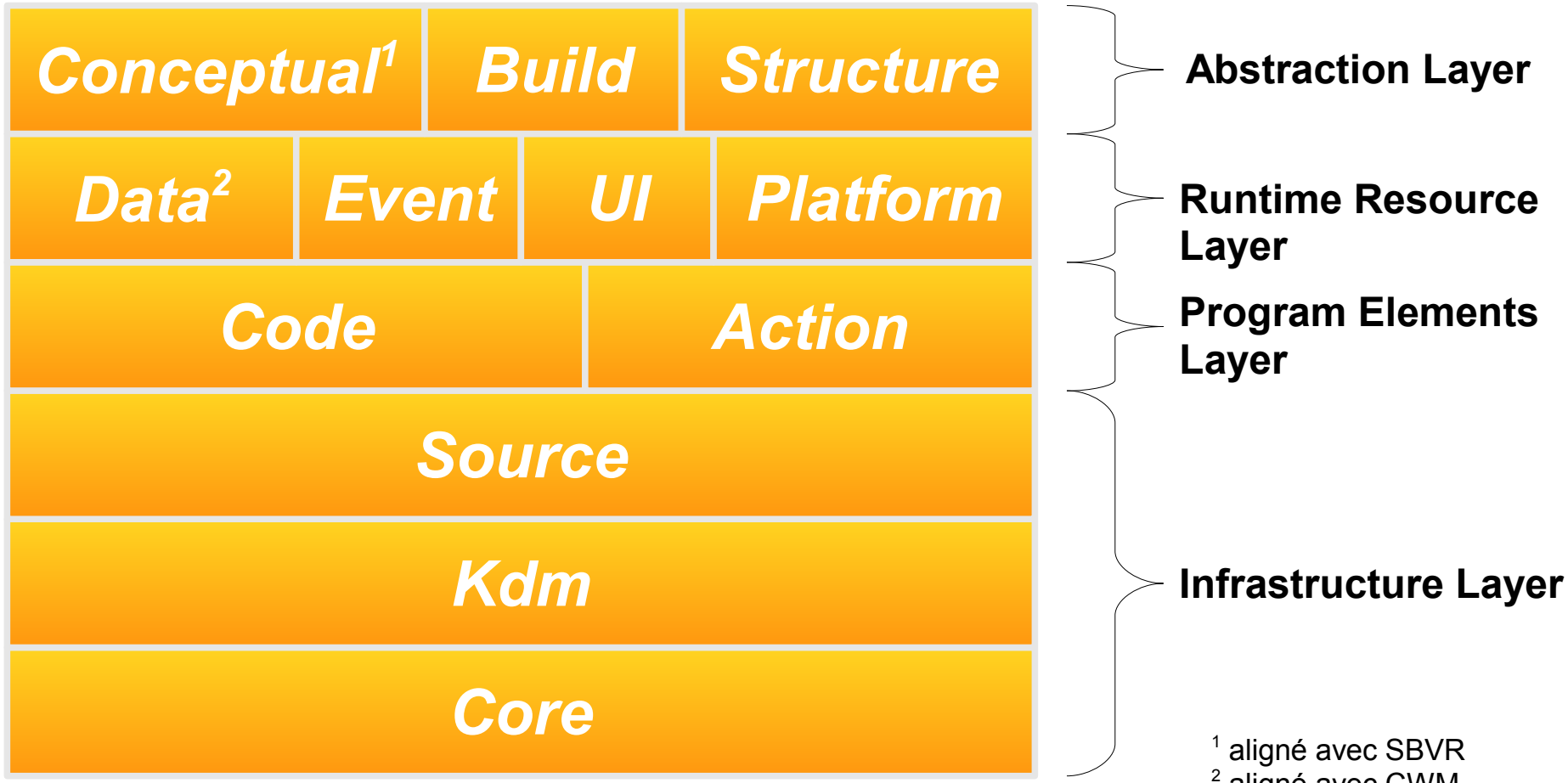


Focus sur KDM

- Knowledge Discovery Metamodel (KDM)
  - Prévu pour une modélisation haut-niveau, une « carte » du SI patrimonial selon plusieurs points de vues (cf. diapo 17)
  - Fournit différents « micro-langages » couvrant les aspects comportement, structure, et données du SI
  - Sert de modèle pivot à partir duquel on va dériver des modèles interopérables (i.e. des PIMs)
    - *UML2, SOAML, BPMN, SBVR, …*
- Implémentation disponible
  - KDM Analytics (http://www.kdmanalytics.com/)
    - *Eclipse EMF plugin*
    - *Les modèles sont créés manuellement :-/*

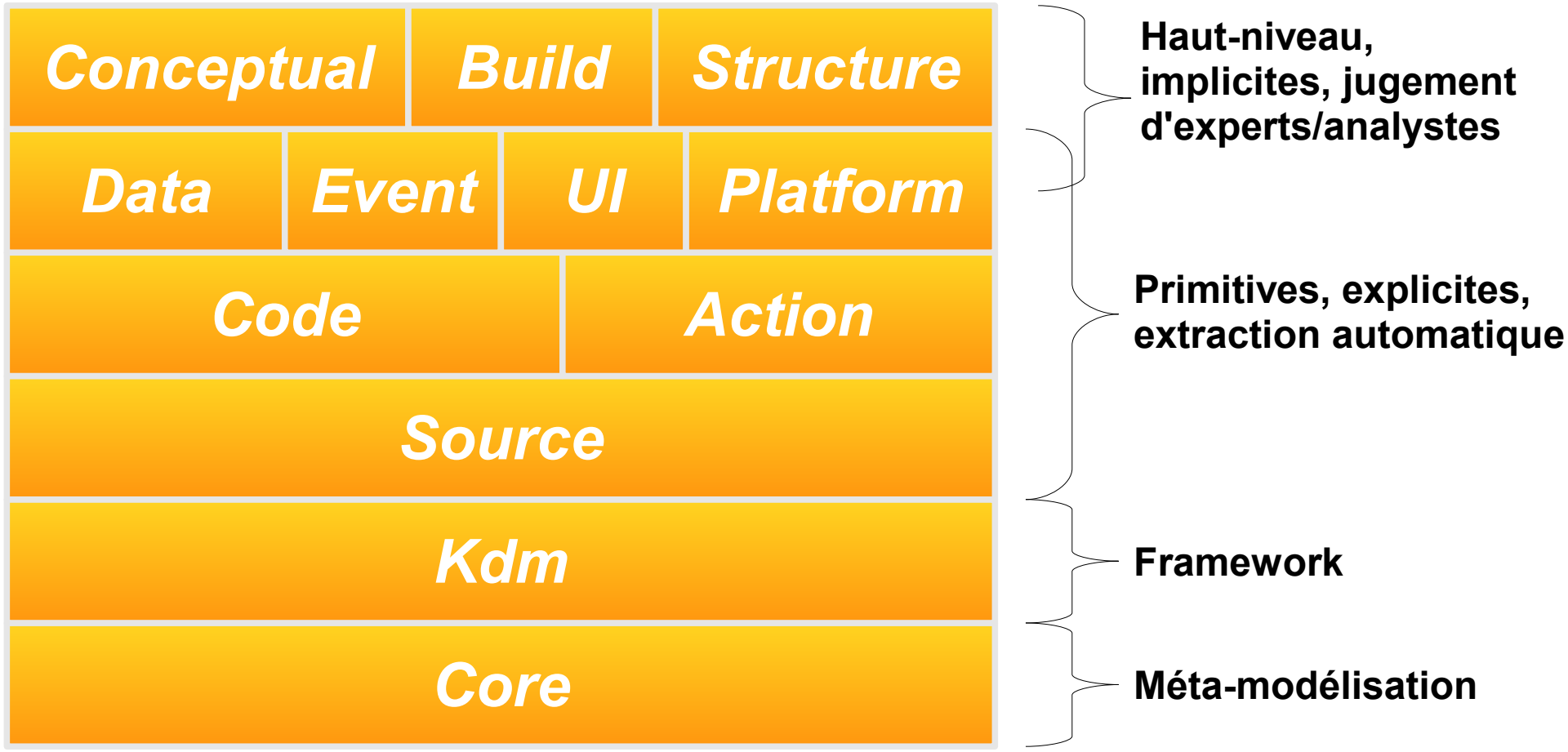
Architecture de KDM

- 12 paquetages arrangés en 4 couches



[1] aligné avec SBVR
[2] aligné avec CWM

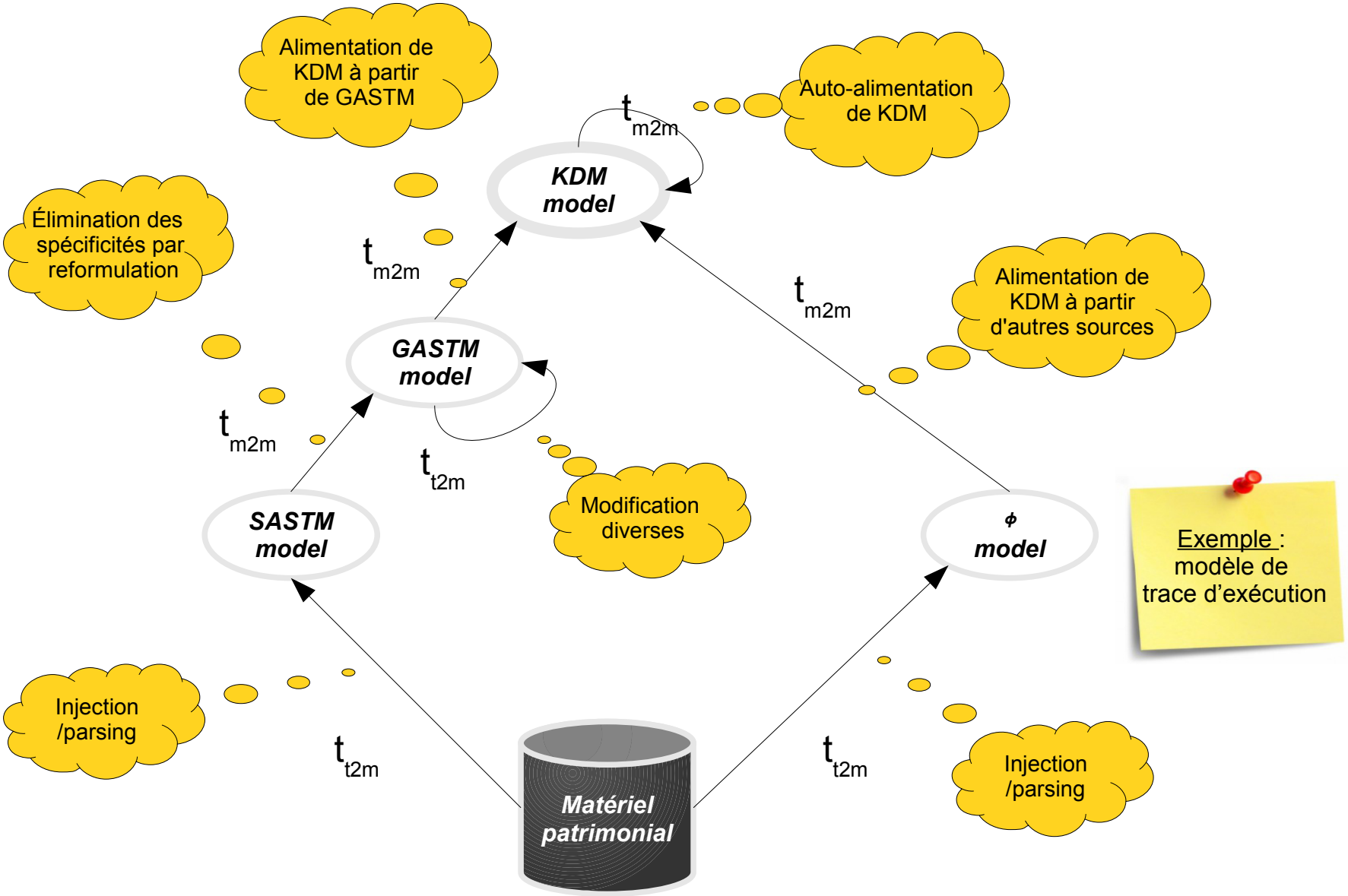
Degré d'exploitation de KDM



Le trio SASTM, GASTM, KDM

- SASTM et GASTM
  - Chaque SASTM est défini en extension de GASTM
  - Une transformation SASTM2GASTM peut être nécessaire pour éliminer les spécificités du langage
- Complémentarité de ASTM et KDM
  - ASTM (bas-niveau d'abstraction) n'est qu'un moyen d'alimenter KDM (haut-niveau d'abstraction)
  - Le paquetage `Code` de KDM n'a pas été prévu pour une modélisation du code en dessous du niveau procédure
  - ASTM est prévu pour conserver la syntaxe concrète/de surface aux niveau des nœuds de l'arbre

Reverse « ADM-compliant »

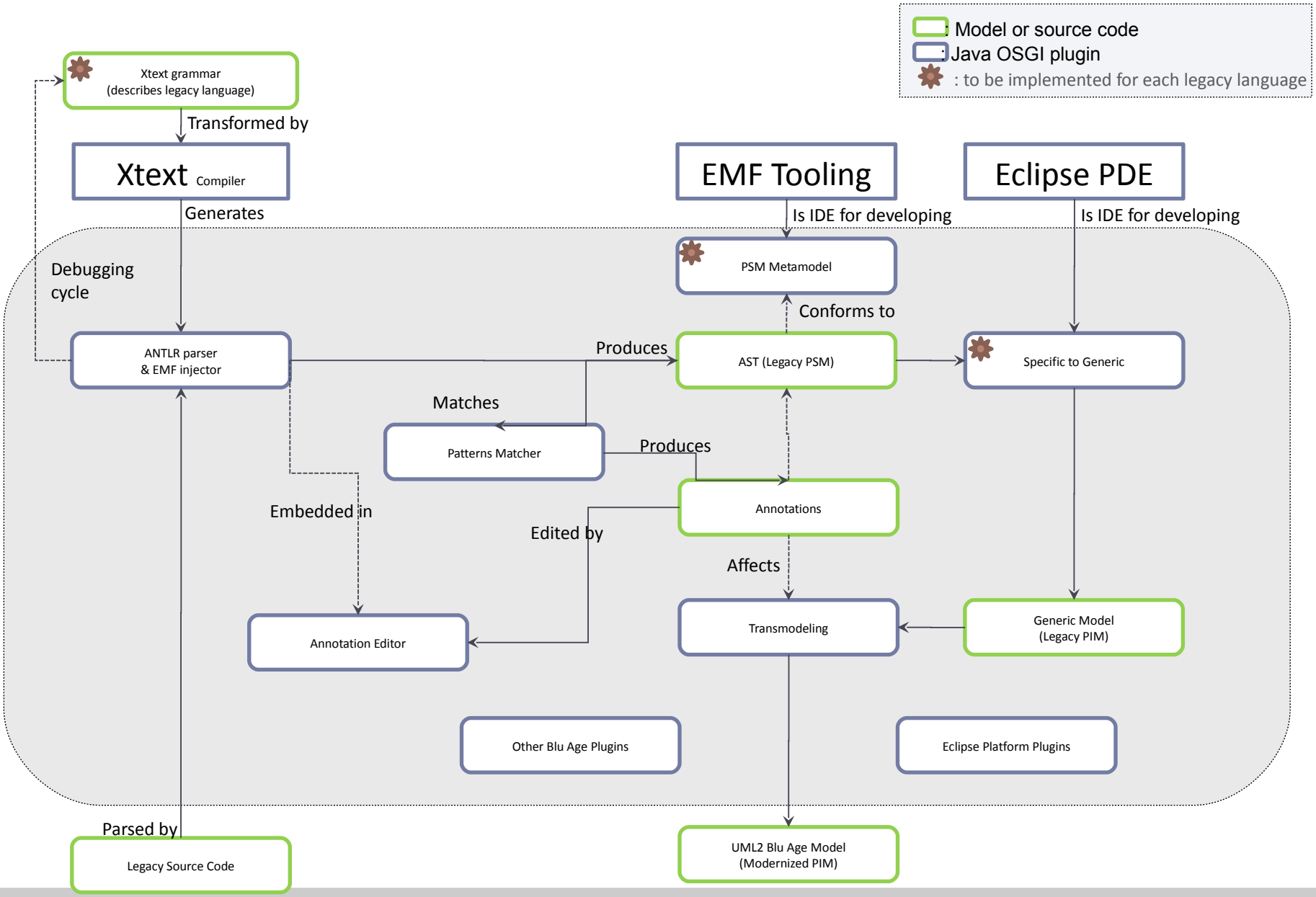


L'exemple de BluAge Reverse

Netfective BluAge®

- Module BluAge® Reverse
  - Phase de rétro-ingénierie
  - Produit des modèles UML2 en sortie
- Module BluAge® Analyst
  - Phase décisionnelle
  - Produit des modèles de «dashboard» en sortie
- Module BluAge® Design&Generate
  - Phase d'ingénierie avant
  - Prend des modèles UML2 en entrées

Architecture de BluAge® Reverse

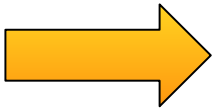


Code cobol → SASTM cobol

```
L900-CHECK-DISTRICT.

    IF ENTRY-DISTRICT     OF RPL-OUT > ZEROS
        IF ENTRY-DISTRICT OF RPL-OUT = PREV-DISTRICT OF RPL-OUT
            CONTINUE
        ELSE
            PERFORM L910-CHECK-MFLOCNBT  THRU L910-EXIT
        END-IF
    ELSE
        MOVE 1  TO SH-ENTRY-DISTRICT OF IPM-PRICE-BASIS-INCLUSIONS OF RPL-OUT
        MOVE 98 TO REPLY-CODE

        IF IP-HEADQUARTERS-USER       OF MSG-IN
        OR IP-SERVICE-CENTER-USER     OF MSG-IN
        OR IP-CONTROL-LOCATION-USER   OF MSG-IN
            MOVE MSG-MUST-ENTER-DISTRICT
              TO ERROR-MESSAGE-KEY  OF ERROR-MESSAGE-LINKAGE
        ELSE
            MOVE MSG-IP-DISTRICT-EMPTY
              TO ERROR-MESSAGE-KEY  OF ERROR-MESSAGE-LINKAGE
        END-IF
    END-IF
    .
L900-EXIT.  EXIT.
```



t_{t2m}

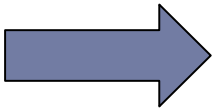
- Block Unit
  - Callable Unit L900-CHECK-DISTRICT
    - If Statement IF
      - If Statement IF
        - Continue Statement CONTINUE
        - Perform Statement PERFORM
          - Througt Expression
            - Call Expression
            - Call Expression
          - Binary Expression
            - Equal
            - Reads Of Expression
              - Ref COBOL
              - Reads Expression
              - Reads Expression
            - Reads Of Expression
        - Move Statement MOVE
          - Integer Literal 1
          - Reads Of Expression
        - Move Statement MOVE
          - Integer Literal 98
          - Reads Expression
      - If Statement IF
      - Binary Expression

```
MOVE returns cobolStatement::MoveStatement:
    name=C_Move sourceElement=(CORRESPONDING|Expression) C_TO (targetElement+=basicExpressionNoComma ','?)*
;
```

SASTM cobol → GASTM

Specific:

- Move Statement MOVE
  - Integer Literal 1
  - Reads Of Expression
    - Ref COBOL
    - Reads Expression
      - Ref COBOL
    - Reads Of Expression
      - Ref COBOL
      - Reads Expression
        - Ref COBOL
      - Reads Expression
        - Ref COBOL



t_{m2m}

Generic:

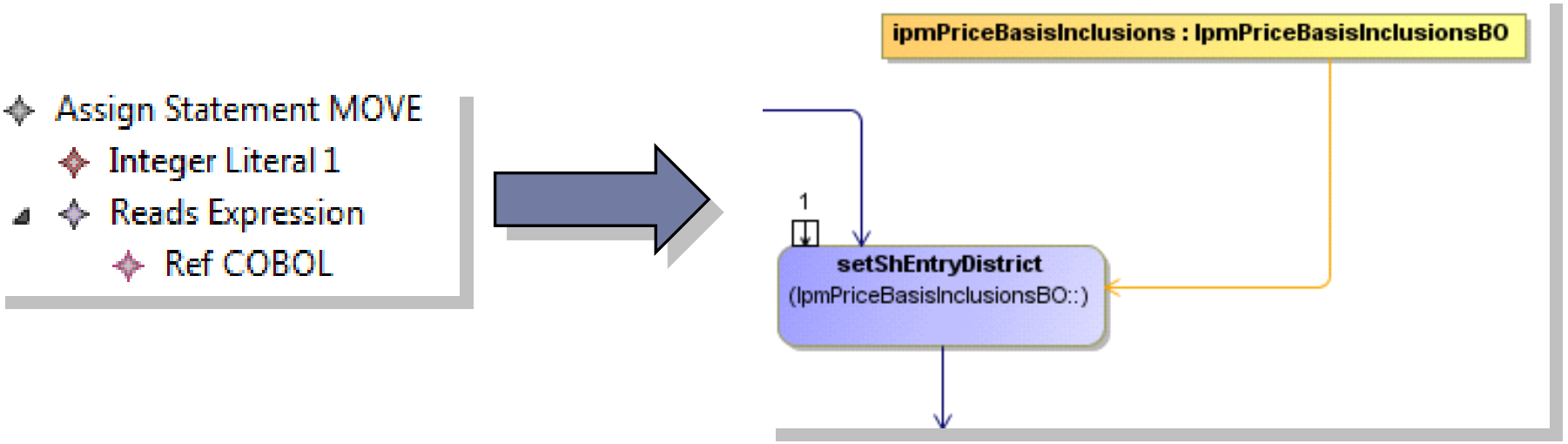
- Assign Statement MOVE
  - Integer Literal 1
  - Reads Expression
    - Ref COBOL

Autre Exemple : opérateur ternaire de Java ré-écrit en if/else

Original code:

```
MOVE 1  TO SH-ENTRY-DISTRICT OF IPM-PRICE-BASIS-INCLUSIONS OF RPL-OUT
```

GASTM→UML2 + Profil



Conclusion

La réalité de la modernisation

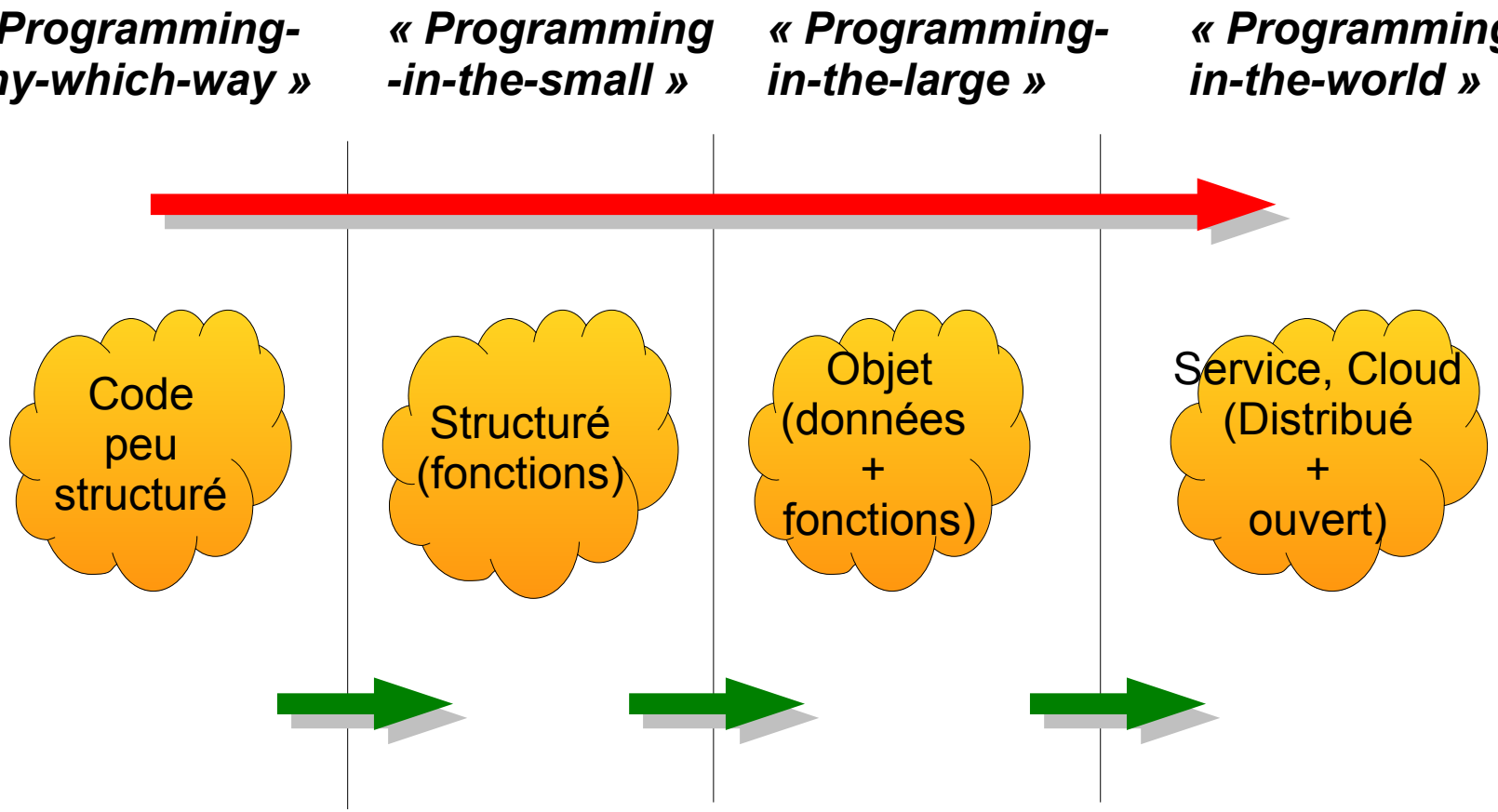
- Le rendez-vous manqué
  - Les prototypes se concentrent sur des technologies pas si veille que cela (Java, …) ou pas si répandues que cela (SmallTalk, ...)
  - Le véritable besoin de l'industrie porte sur les gros systèmes codés en COBOL (Banques) ou en C (Telecom)
- La complexité des SI « dinosaures »
  - Passage à l'échelle : millions de lignes de code
  - Abstraction : faiblesse de structuration (green screen, flat files, ...), manque de vue synthétique sur le système
  - Connaissance : totalement perdue, dure à reconstituer

Premier bilan

- Un roundtrip est nécessaire
  - Peu probable de passer du matériel patrimonial vers un modèle KDM d'un seul coup.
- Récupérer la connaissance est difficile
  - Extraction de règles métiers
  - Extraction d'architecture
  - ...
- La boite à outil ADM est sous-utilisée
  - ADM c'est bien plus que juste ASTM et KDM
  - KDM ce n'est pas que les aspects liés au code source

Le fossé des générations

- Il n'est pas raisonnable d'essayer de traverser toutes les générations d'un coup



Questions ?