Paris JUG

08 juillet 2008

www.parisjug.org

www.parisjug.org

Paris JUG

08 juillet 2008

Pourquoi tout le monde ne fait-il pas du MDA ?

Grégory Weinbach

Responsable de Pôle

Objet Direct

Photo

www.parisjug.org

Intervenant

- **Grégory Weinbach**
  - 18 ans en SSII
  - Technos Avancées (IA, réseaux de neurones, programmation par contraintes)...
  - ...puis Génie Logiciel et Objet
  - Pour Objet Direct depuis 10 ans
    - Architecte Java
    - Expert UML
    - Responsable du pôle MDA

Grégory Weinbach
gweinbach@objetdirect.com
http://www.objetdirect.com/
http://mdblog.fr/

Sommaire

- **Approche MD et MDA**
  - Quelques rappels
- **Le MDA au quotidien**
  - La théorie confrontée à la réalité
- **MDA pas mort**
  - Des solutions qui marchent
- **Les questions habituelles...**
  - ...et légitimes !

Pourquoi tout le monde...

- **Sur le papier le MDA c'est**
  - La proximité des utilisateurs
  - L'agilité et ses conséquences positives
  - L'indépendance vis-à-vis de la technologie
  - L'industrialisation des développements

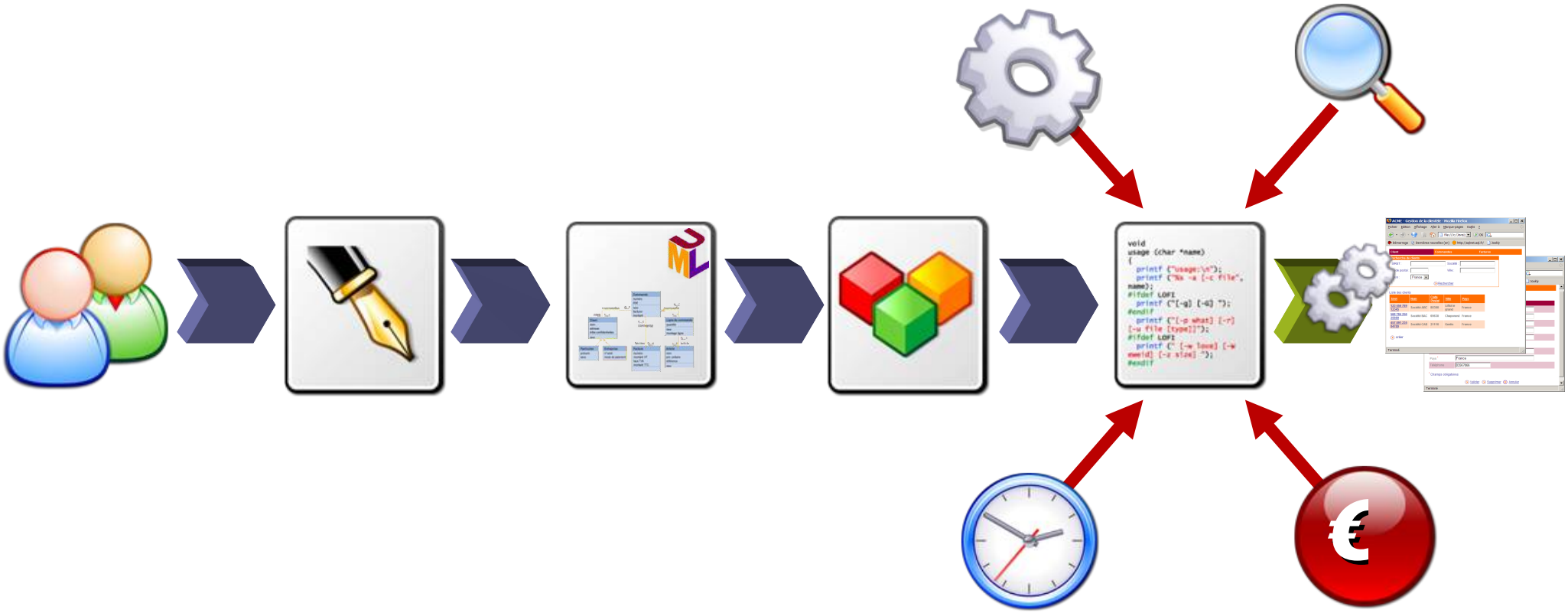
- **Alors pourquoi...**
  - ...tous les projets ne l'utilisent pas ?
  - ...toutes les SSII n'ont-elles pas pris le virage ?

Qu'est-ce que le MDA ?

- L'approche MD
- Différences vis-à-vis des approches classiques
- CIM, PIM, PSM et autres TLAs...

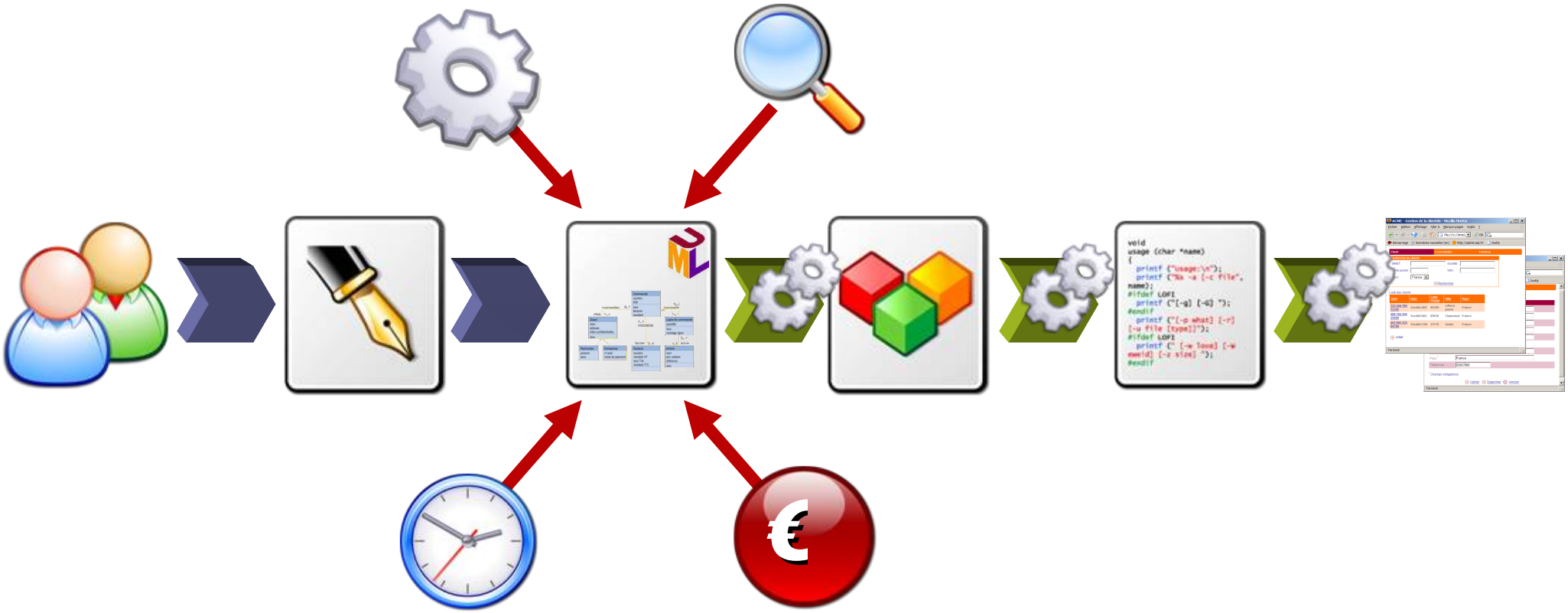
L'approche traditionnelle

- Centrée sur le code



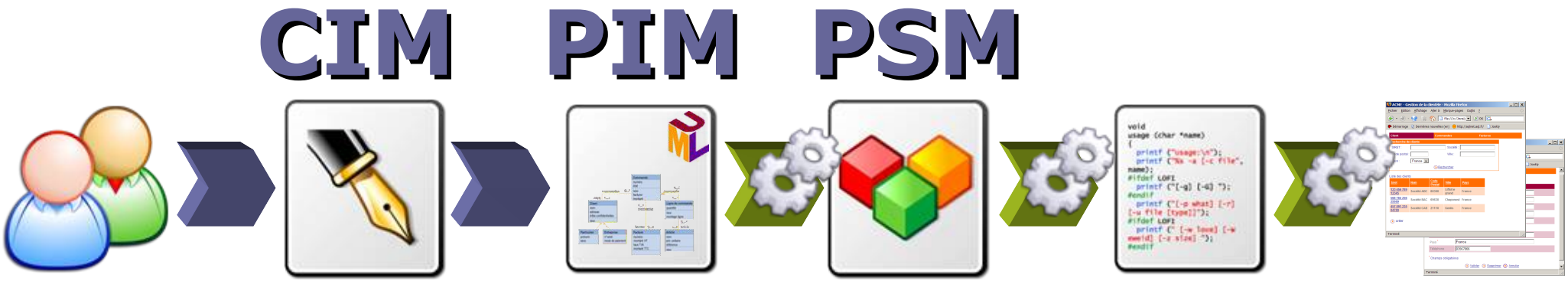
L'approche MD

- Centrée sur l'analyse fonctionnelle



MDA « standardise » le MD

- **Un vocabulaire et des outils**



Computation Independent Model = **Processus Métier**

Platform Independent Model = **Analyse Fonctionnelle Détaillée**

Platform Specific Model = **Conception Technique**

+ MOF, QVT, OCL...

Les points clés

- **L'énergie est mise dans l'analyse**
  - Et non dans le code
- **La complexité fonctionnelle est concentrée dans le PIM**
  - La complexité technique est dans le framework cible ET dans les transformateurs/générateurs
- **Le PIM est strictement fonctionnel**
  - C'est le « code source » de l'application
- **Le cycle de développement est très court**
  - Les utilisateurs sont souvent sollicités

Le MDA au quotidien

- Le PIM livrable ultime : théorie et réalité
- Le Graal de la plate-forme
- Les nouveaux rôles

Le PIM livrable ultime !

- **Théoriquement le PIM c'est**
  - Un modèle strictement fonctionnel
  - Un modèle Objet
    - Structures ET comportements
  - Un modèle de domaine et un modèle applicatif
    - Entités ET Services, IHM, Transactions...
  - Le code source de l'application
    - Cible = 100%

La réalité (1)

- **Le PIM anémique**
  - Un modèle de Domaine réduit au modèle de la base de données
  - Un diagramme d'état pour les enchaînements d'écran
  - Quelques règles de gestion modélisées
    - Diagramme d'activités ou de séquence

- **~30% de la complexité fonctionnelle est adressée !**

La réalité (2)

- **Le PIM obèse**
  - La plate-forme interdit la personnalisation du code généré
  - Tout est modélisé en UML
  - Chaque élément de code a une contrepartie dans le modèle

- **Pas de compression de la complexité fonctionnelle**
- **Autant d'efforts (plus !) que de développer l'application**

La réalité (3)

- **Le PIM « PSM »**
  - Modèle technique
    - « WebService », « PrimaryKey », « DAO », « EJBSession »...
  - Pas d'indépendance fonctionnelle/technique
  - Pas de transformation
  - Chaque élément de code a une contrepartie dans le modèle

- **Pas de compression de la complexité technique**

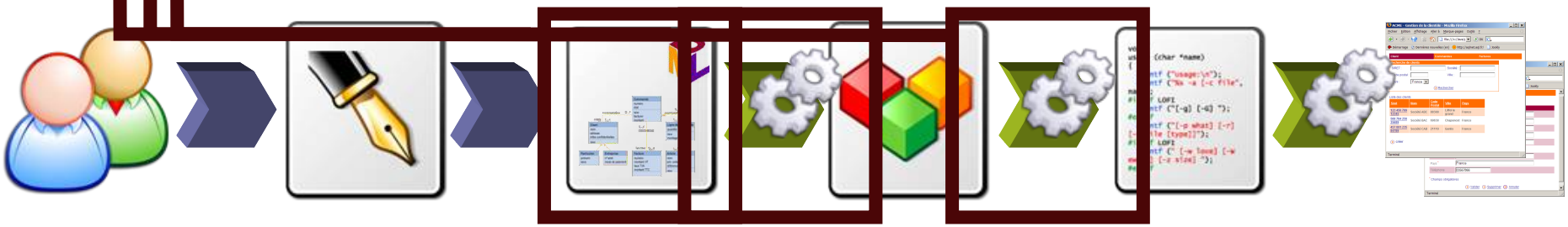
La réalité

- **PIM anémique**
  - Pas de comportement, pas de modèle applicatif
- **PIM obèse**
  - Modèle fonctionnel ultra-détaillé, lourd et coûteux
- **PIM PSM**
  - Génération de code : pas fonctionnel, pas agile

- **Ne respectent ni l'esprit ni la lettre du MDA !**

Le Graal de la plate-forme

- **Trois types de besoins**
  - Modélisation : production des « sources » MD
  - Transformations « model to model »
  - Transformations « model to code » : génération ou « templating »



Attention à la confusion !

- **Entre « outil MDA » et « plate-forme MDA »**
  - Outil MDA =
    - modeleur UML ou atelier DSL
    - + moteur de transformation
    - + outil de templating
  - Plate-forme MDA = Outil MDA instancié
    - Outil MDA
    - + sémantique de modélisation (profil UML ou DSL)
    - + architecture technique cible
    - + transformateurs
    - + générateurs

MDA les nouveaux rôles (1)

- **MOE outils – Architectes**
  - Met son savoir faire dans l'outillage
  - Spécialisation forte
  - Travail d'éditeur d'outils dans une cellule spécialisée
    - Recycle les « super développeurs » prêts à abandonner le fonctionnel

MDA les nouveaux rôles (2)

- **MOAD – Analystes**
  - Très bons modélisateurs objet
  - « Développent » le fonctionnel
  - Maîtrisent la chaîne de transformation

MDA les nouveaux rôles (3)

- **MOE - Développeurs projet (il en reste !)**
  - Personnalisation du code généré
  - Production du reliquat de code non généré

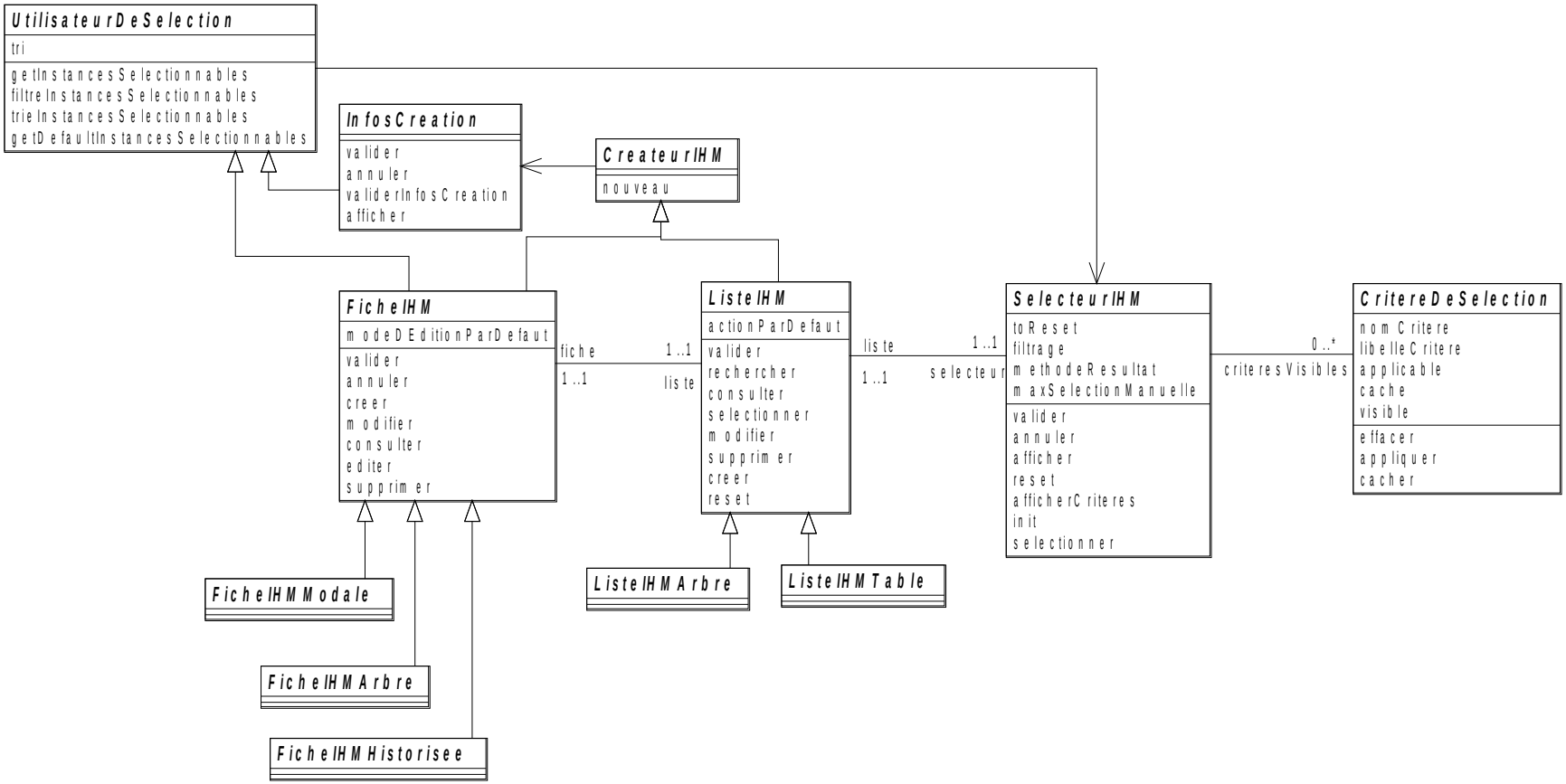
MDA, pas mort !

- **Les problèmes posés**
  - Comment modéliser une application ?
  - Comment construire un PIM efficacement ?
  - Comment décrire les comportements ?

Modéliser une application

- **Construire un « framework applicatif »**
  - = Spécifier et implémenter les composants applicatifs « de base »
    - Ex : Menu, Sélecteur multi-critères, Liste paginée, Fiche détail, Wizard de création…
    - Pour chaque composant type :
      - Quel est son comportement utilisateur ?
      - Quels sont les services qu'il fournit ?
      - Comment peut-il interagir avec les autres composants ?

Exemple



Modéliser une application

- **Framework applicatif :**
  - Définit le comportement standard d'une (des) application(s)

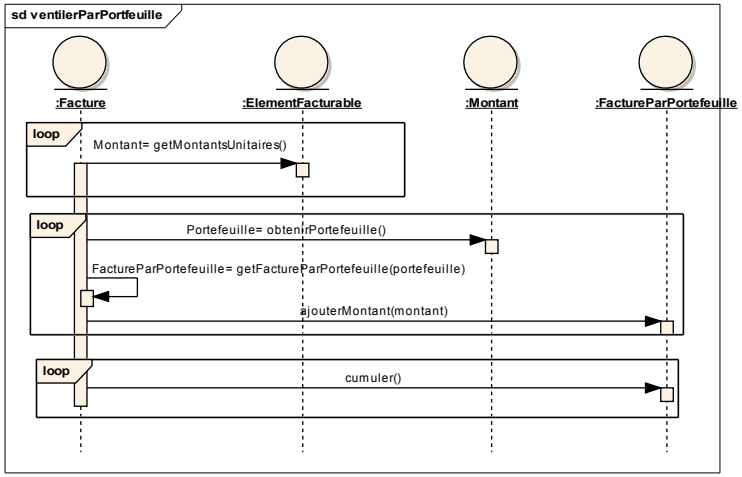
- **Besoin indépendant de l'approche MD !**
- **Dans une approche MD :**
  - Le framework applicatif définit complètement les éléments du PIM
  - Le profil UML ou le DSL correspondant structure la modélisation

Intelligente la plateforme ?

- **Le framework applicatif fournit de nombreux comportements par défaut**
  - Un acte de modélisation est très riche en informations
  - On procède par exceptions (Aggressive Defaulting)
- **Le PIM compresse la complexité**
- **La plate-forme MDA la restitue**

Décrire les comportements

- **Deux approches possibles**
  - UML : Modéliser le contenu des méthodes



  - ASL : Implémenter le contenu des méthodes

```
MontantsUnitaires := [];
for Element in Self.elementsFacturables do
  {
    Element << getMontantsUnitaires(Montants);
    MontantsUnitaires =+ Montants;
  };

for Montant in MontantsUnitaires do
  {
    Montant << obtenirPortefeuille(Portefeuille);
    Self << getFactureParPortefeuille(Portefeuille,FactureParPortefeuille);
    FactureParPortefeuille << ajouterMontant(Montant);
  };

/* calcul des cumul de facturation par portefeuille */
for FactureParPortefeuille in Self.facturesParPortefeuille do
  {
    FactureParPortefeuille << cumuler;
  };
```

- **Modéliser le contenu des méthodes**
  - Les outils = les diagrammes dynamiques d'UML
    - Diagramme de séquence, d'activités, de vue d'ensemble des interactions…
  - Si on veut pouvoir produire le code cible, les modèles doivent être sémantiquement équivalents à du code
    - Il faut être complet et extrêmement rigoureux

UML (2)

- **Ces diagrammes sont un bon support de dialogue dans les cas simples mais :**
  - Ils n'ont pas été conçus pour être « compilés »
  - Ils sont très difficiles à maintenir
  - Les modeleurs sont souvent incomplets ou trop laxistes

ASL

- **Les diagrammes dynamiques d'UML s'appuyent sur une syntaxe abstraite : l'*Action Semantics Language* (ASL)**
  - Définit les actions possibles sur un modèle UML et leur sémantique

- **L'OMG ne spécifie pas de syntaxe concrète**
  - Pas de « langage de programmation » ☹

ASL (2)

- **Objet Direct utilise une implémentation d'ASL : *D.Script***
  - S'appuie directement sur un modèle MOF
  - Supporte une véritable sémantique objet
  - Exécutable tel quel
  - Compilable dans plusieurs langages cibles (Java, C#, VB…)
  - Implémenté dans l'outil D.OM
  - Vers un « UML exécutable »

Comportements et PIM

- **Deux axes d'utilisation**
  - Analyse et prototypage fonctionnels
    - Rendre le PIM exécutable indépendamment de la plate-forme technique cible
    - Simuler l'application pour travailler très tôt avec les utilisateurs et/ou les experts métier
  - Développement MD
    - Minimiser le code métier écrit dans le langage cible
    - Etendre le processus MD à l'ensemble de la complexité métier

Les questions habituelles

- « Doit-on autoriser la modification manuelle du code ? »
- « Que se passe-t-il si je modifie le code généré ? »
- « Comment travailler en itératif si le code généré est modifié ? »
- « Dois-je gérer le code généré en configuration ou seulement les modèles ? »
- « Quel pourcentage de code est-il raisonnable de générer ? »
- « Quels gains peut-on attendre de l'approche »

« Autoriser les retouches »

- **Postulat : la plate-forme MD ne génère pas tout**
  - Le processus outillé « full-MD » n'existe pas
  - …sauf usine à gaz
- **Deux approches :**
  - Modification ou spécialisation du code généré
- **Modification**
  - cf. question suivante ☺
- **Spécialisation**
  - Code final ≠ d'un code manuel : sa structure dépend du processus MD
  - Pas toujours possible (JSP, SQL…)

«Modifier le code généré ? »

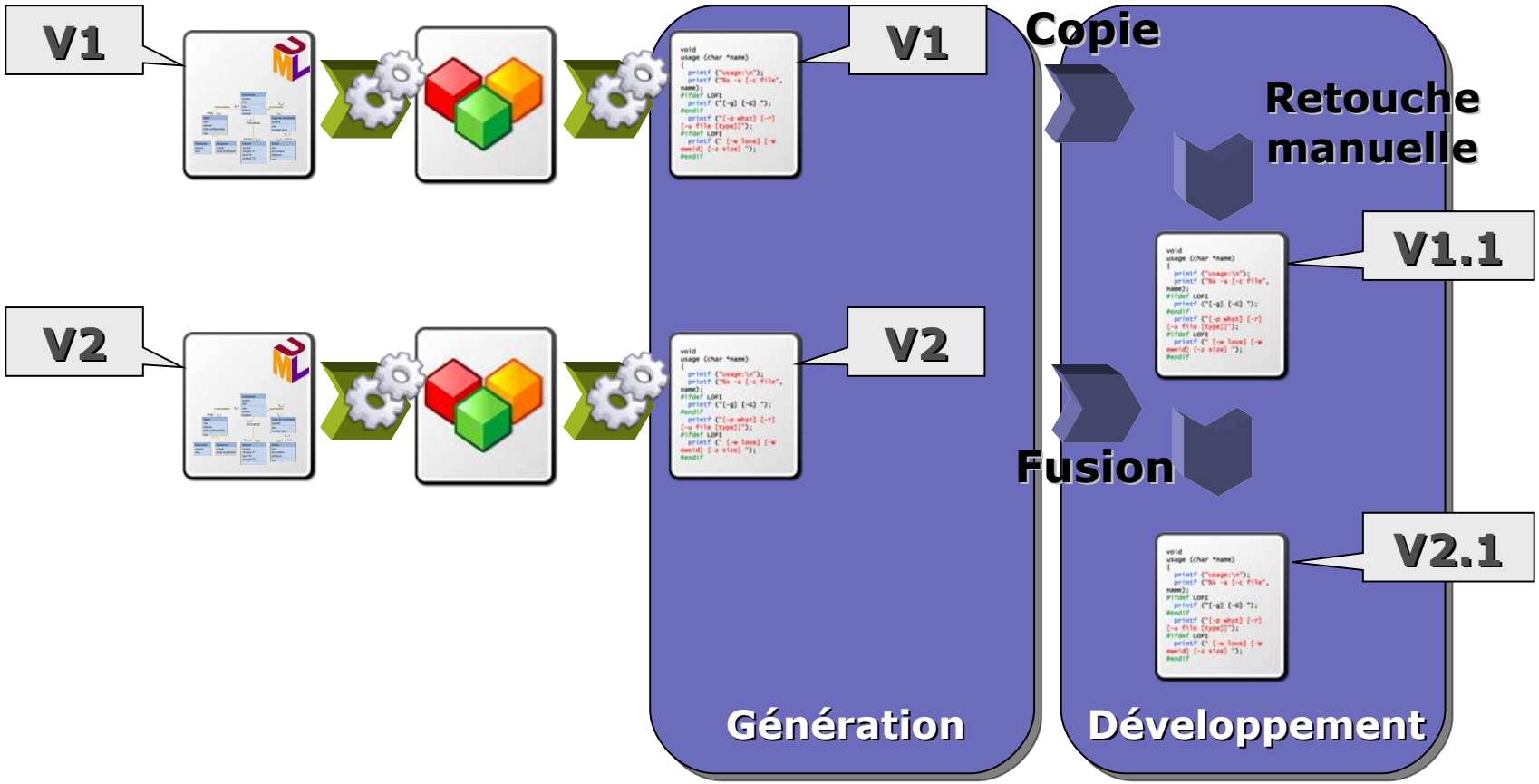
- **Le « round-trip » PSM ⇔ code est supporté par certains outils**
- **Deux approches**
  - Commentaires signifiants : code illisible et fragile
    - plug-in pour l'IDE indispensable
  - Analyse syntaxique : code lisible « standard »
    - mais sensible au renommage !

- **Mais vraiment utile si on a aussi un « round-trip » PIM ⇔ PSM**
  - Les outils les plus puissants le permettent
  - En pratique jamais mis en œuvre car nécessite d'écrire pour chaque transformation, sa transformation inverse !

- **Dans tous les cas : finalement pas très utilisable ☹**

« Travailler en itératif ? »

- L'approche « tout automatique » n'est donc pas réaliste
- L'approche pragmatique (celle qui marche !)



« Code généré en conf ? »

- **Dans l'approche pragmatique vue précédemment on doit gérer en configuration**
  - Les modèles
  - Le code retouché
- **Mais aussi, évidemment**
  - Les profils UML
  - Le framework applicatif
  - Les transformateurs/générateurs

« % de code générable ? »

- **100% de la « colle technique »**
  - Code sans valeur ajoutée produit habituellement par copier/coller
- **De 60 à 100% du code métier**
  - 100% du code « structurel »
  - Les 40% d'ajustement correspondent aux méthodes métier et aux services applicatifs
- **De 0 à 100% des IHM**
  - Le plus étant le mieux !
- **En dessous de 50% ça n'est pas du MDA !**
  - C'est du simple « templating » : utilisation de wizards
  - On n'est pas « centré sur les modèles », mais « centré sur le code »

Retours d'expérience (1)

- **Des projets réussis dans des domaines métiers très variés**
  - CIG Petite couronne (SNAPI)
  - Snecma moteurs (Explore)
  - Péchiney (Tigra)
  - Ministère de l'équipement (BDCL)

Retours d'expérience (2)

- **Très gros gains constatés sur**
  - La qualité
  - Les performances
  - L'évolutivité fonctionnelle
  - La pérennité
  - L'acceptabilité des applications
  - La productivité

Conclusion

- **Pourquoi tout le monde ne fait-il pas du MDA ?**
- **...parce que pratiquement personne n'en fait !**

- **Il faut généraliser**
  - Les frameworks applicatifs riches
  - Les langages concrets associés à UML
- **Et former des analystes !**

Grégory Weinbach
gweinbach@objetdirect.com
http://www.objetdirect.com/
http://mdblog.fr/

Questions / Réponses

www.parisjug.org

Sponsors

Merci de votre attention!

www.parisjug.org

Licence

http://creativecommons.org/licenses/by-nc-sa/2.0/fr/