Nouvelles technologies pour automatiser le développement

Déductions

J.M. Vanel - 2009-05

Appliquer l'intelligence artificielle au génie logiciel
Modélisation, moteurs de règles
Multi-modèles, multi-plateformes
Ergonomie et logique

Vision

- Appliquer l'Intelligence Artificielle au développement logiciel
- « Model once, run everywhere »
  « logiciel durable »
- Réduire la fracture des plateformes et des langages
  - Les infrastructures ne sont que des commodités, qui doivent être interchangeables

Objectif à long terme

- Augmenter la flexibilité des logiciels
- Diminuer les coûts de développement
- Maîtriser le développement logiciel
- Améliorer l'ergonomie

Le besoin des clients

- Vous développez des logiciels spécifiques,
  - Essentiels pour le service offert
- Besoin de réactivité
- Impossible de sous-traiter
- Ne pas être lié à une architecture technique
- Maîtrise et traçabilité dans la chaîne de production

Développement Logiciel: le constat

- coûteux, d'une façon non prévisible
- non flexible
- non réutilisable
- les modèles et règles métier sont cachés parmi le code d'infrastructure
- non traçable: spécifications & implémentations non reliées
- non amical pour l'utilisateur

Non !

Notre réponse aux besoins des clients

- Basée sur des modèles précis du métier (ontologies et règles)
- Indépendante d'une architecture technique
- Utilise des techniques éprouvées de l'Intelligence Artificielle: systèmes experts, logique de description

Notre réponse aux besoins: outils

- Environnement de développement à base de règles (EulerGUI)
- 3 moteurs d'inférence (Java-Drools, Prolog, Python-CWM)
- lit des modèles métiers dans plusieurs formats : UML, OWL, RDF(S), SQL, eCore
- Exemple d'utilisation: générateur de formulaires
  - Règles logiques inhérentes ==> ergonomie supérieure

Moteurs de règles: avantages

- Une règle isolée a un sens
- Le moteur enchaîne les règles, pas le programmeur
- + efficace que la programmation classique
- Capture bien l'expertise métier
- Traduisible depuis et vers le langage naturel
- Les « systèmes experts « ont fait leurs preuves

Règles: exemples

S'il y a une commande, il y a une demande de paiement associée :

```
{ ?C a Order } =>
{ ?C hasPaymentRequest ?PR }.
```

Si le porteur a eu un incident de paiement, alors la commande est annulée:

```
{ ?C hasHolder ?H .
  ?H hasPaymentIncident ?PI } =>
{ ?C hasStatus cancelled }.
```

Règles: exemples 2

If one customer has Ordered items in same Domain but different Category than the last items, propose this different Category to customer:

```
{ ?CUSTOMER :hasOrderedItem ?ITEM .
  ?ITEM :domain ?DOMAIN .
  ?ITEM :category ?CATEGORY .
  ?ITEM2 :belongsTo :lastItems .
  ?ITEM2 :domain ?DOMAIN .
  ?ITEM2 :category ?CATEGORY2 .
  ?CATEGORY2 log:notEqualTo ?CATEGORY .
} => {  :proposal :category ?CATEGORY2 } .
```

Modélisation: pourquoi ?

- Si on y réfléchit, une application ce n'est que:
  - un modèle de données,
  - des règles (validation, présentation, formulaires, ...),
  - et de l'infrastructure (Web , IHM, bases de données, ...)
- Mais couramment on aboutit à un code source où tout cela est mélangé ==> maintenabilité et réutilisabilité faibles

Analyse de coûts

- Coûts cachés:
  - plus de 50% du coût logiciel est en maintenance
  - Délais longs: le « time to market » est pourtant essentiel
- Rigidités : ajouter des fonctionnalités ou changer de plateforme sont difficiles ==> on fait de nouveaux développements
- Interopérabilité difficile entre logiciels ==> on ressaisit des données
- services informatiques liés à la TMA en France en 2008 : 2 milliards € ; SOA : 700 millions

Points forts et avantages

- « Model once, run everywhere »
  « logiciel durable »
- Les infrastructures ne sont que des commodités, qui doivent être interchangeables
- Open Source
- Basé sur des standards reconnus : RDF, OWL, N3, Drools
  - UML, eCore, SQL lus en entrée

Modélisation facilitée – UML textuel

```
Person<|---Customer---Buying<|---ServiceBuying
                              <|---MerchandiseBuying

HelpRequest-----------Buying
date: Date
explanation: String
```

Modélisation facilitée – Anglais contrôlé

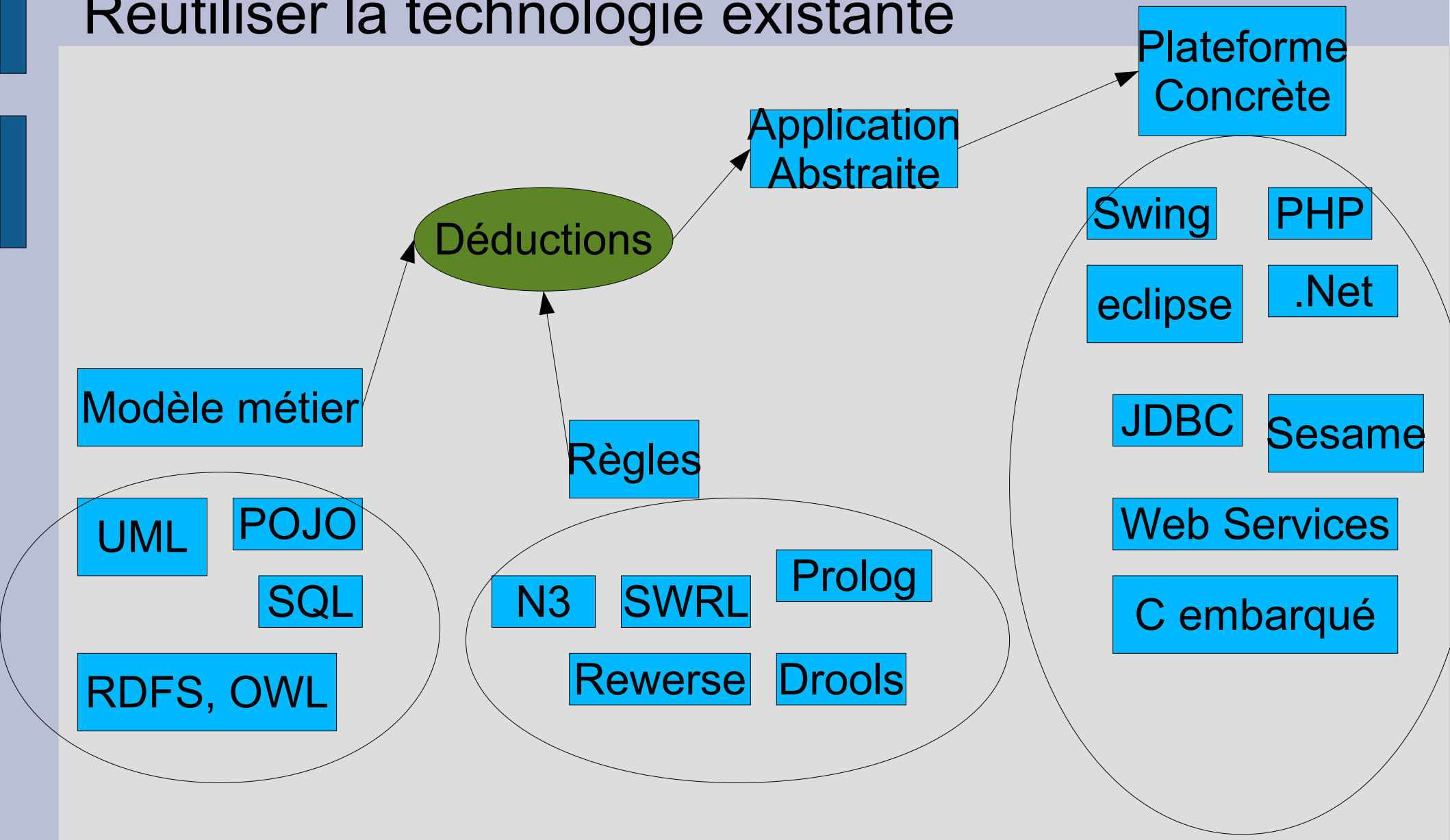
Every card is a means of a payment .

Le modèle formel est récupéré à partir de la phrase:

```
Card ---|> MeansOfPayment
```

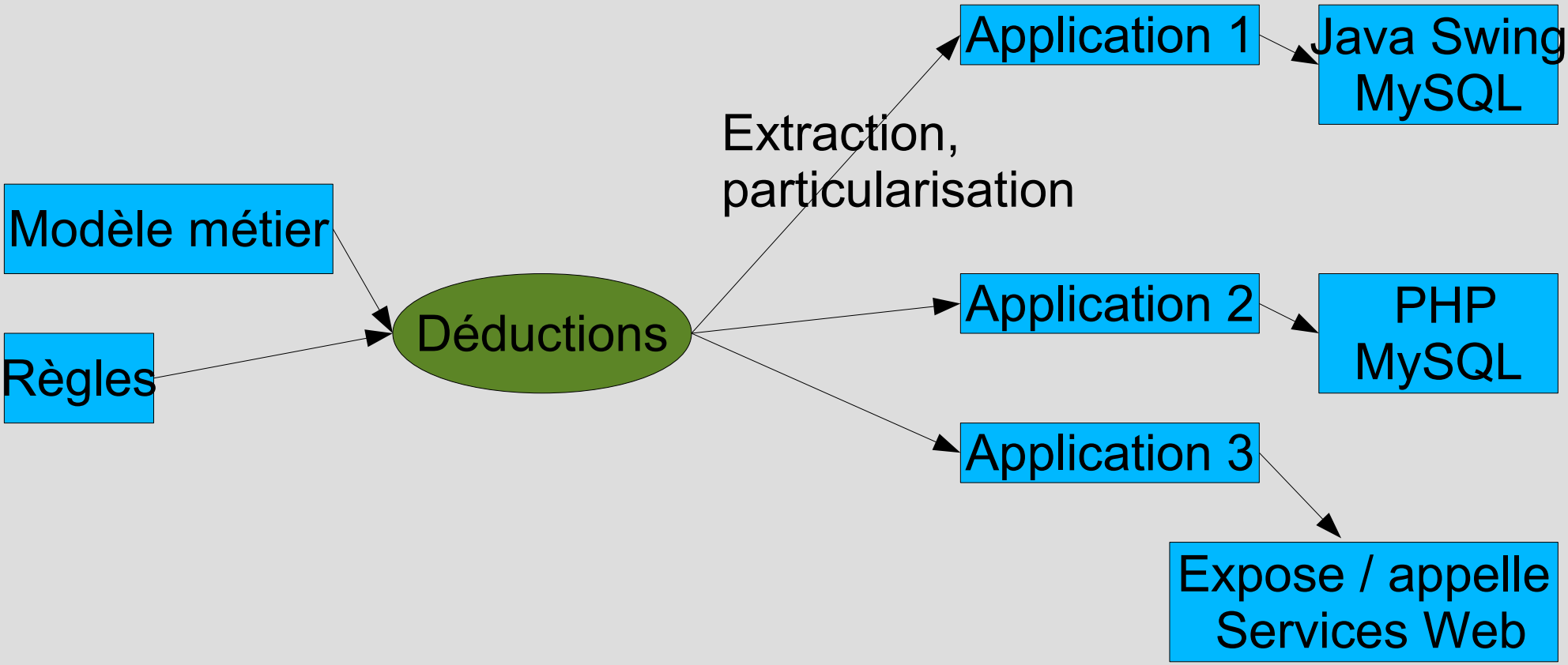
Le flux d'information (vu par le concepteur)

Réutiliser la technologie existante



Cohérence et réutilisation

Réutiliser les modèles existants



Générateur de formulaire

- Les techniques courantes des générateurs d'application sont limitées:

Générateur X	Déductions
Couteux en technicité	Permet de générer 100% d'une application à partir du modèle et des règles
Formats spécifiques ou propriétaires en entrée	S'appuie sur une modélisation riche, mais standard : OWL du W3C
plateforme spécifique en sortie	Conception multi-plateforme
Coûts de licence	Open Source

- OWL est mieux adapté qu'UML pour modéliser les concepts métier
- OWL est basé sur la logique de description

Le Web Sémantique

- Fondement solide pour les modèles métier
  - Standards d'échange: RDF, OWL, N3
  - Milliers de modèles disponibles sur plusieurs moteurs de recherche : Swoogle
  - Au contraire UML est complexe, peu interopérable, sans modèles disponibles
- Le WS est pensé pour exposer des données riches sur le Web; nous le réutilisons comme socle pour l'ingénierie du logiciel

Problèmes avec UML

- Modélisation par et pour les développeurs
- Cependant le rapport entre un modèle UML et le code source est complexe
  - Il existe des variantes plus ou moins proches de la programmation
  - Certaines notations déterminent directement le source, d'autres sont descriptives
- L'OMG a toujours mal promu ses technologies

Ouvrir l'Open Source

- Des milliers d'applications et de composants disponibles, mais ...
- Pour en intégrer un, il faut souvent des jours de lecture de code;
- Déductions vise à formaliser l'interface entre un composant et l'application, et à automatiser l'intégration

Mesures à prendre

- Impliquer la direction
- Recenser, améliorer, créer des modèles
  - Utiliser toutes les sources, même textuelles
- Expliciter les règles : niveaux générique, entreprise, applicatif
- Formation sur la modélisation, les technologies basées sur les règles
- Choisir un projet pilote pour commencer avec la technologie « Déductions »