MDD/MDA : Génération d'applications avec IBM Rational Software Architect

Jean-Pierre Schoch – jp.schoch@fr.ibm.com

Agenda

- *Le développement logiciel aujourd'hui*
- *Les technologies orientées-objet et UML*
- *Les approches MDD/MDA*
- *Un exemple concret : génération d'une application Web à partir d'UML*
- *Conclusion*

Agenda

- ***Le développement logiciel aujourd'hui***
- *Les technologies orientées-objet et UML*
- *Les approches MDD/MDA*
- *Un exemple concret : génération d'une application Web à partir d'UML*
- *Conclusion*

Taux de réussite des projets de développement

Rapports du « Standish Group »	Année	Taux de réussite (1)
Premier rapport « Chaos »	1994	16 %
« Extreme Chaos »	2001	28 %
Dernier rapport « Chaos »	2003	31 %

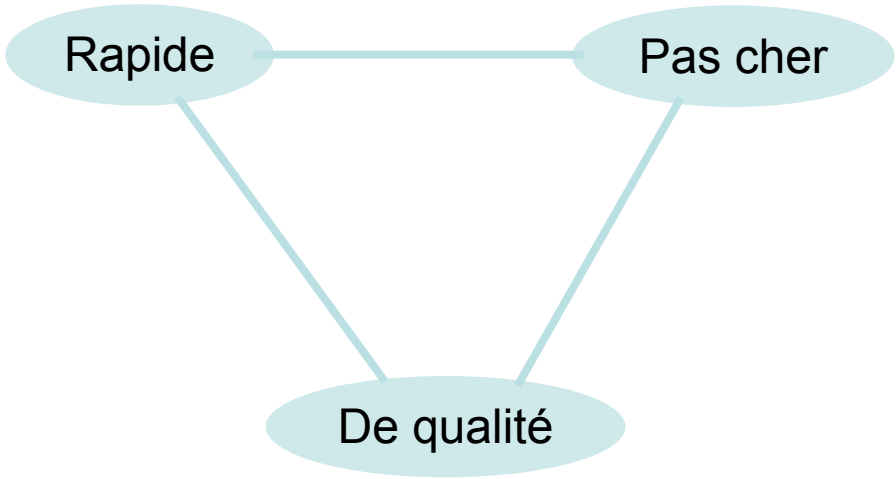
(1) Réussite = projet délivré à temps, dans les budgets et répondant aux besoins des utilisateurs

Au début du 21e siècle, la maîtrise des développements logiciels reste largement aléatoire !

Challenges du développement logiciel

- *Développer …*
  - rapidement,
  - à moindre coût,
  - un logiciel de qualité
    - performant
    - facile à maintenir
    - facile à faire évoluer

Le développement peut être :



Choisissez-en deux !

Quelques solutions actuelles ...

- *Gestion des exigences avec les cas d'utilisation*
  - Améliore l'adéquation du logiciel au « vrai » besoin
  - Nécessite un changement d'attitude des « spécificateurs » et des analystes
- *Développement itératif*
  - Améliore le contrôle et le suivi des projets de développement
  - Nécessite un changement d'attitude de toute l'organisation
- *Technologies orientées-objet*
  - Améliorent la qualité du logiciel développé
  - Souvent mal appliquées malgré une large adoption et le succès d'UML

Les solutions actuelles sont nécessaires mais pas suffisantes.

Agenda

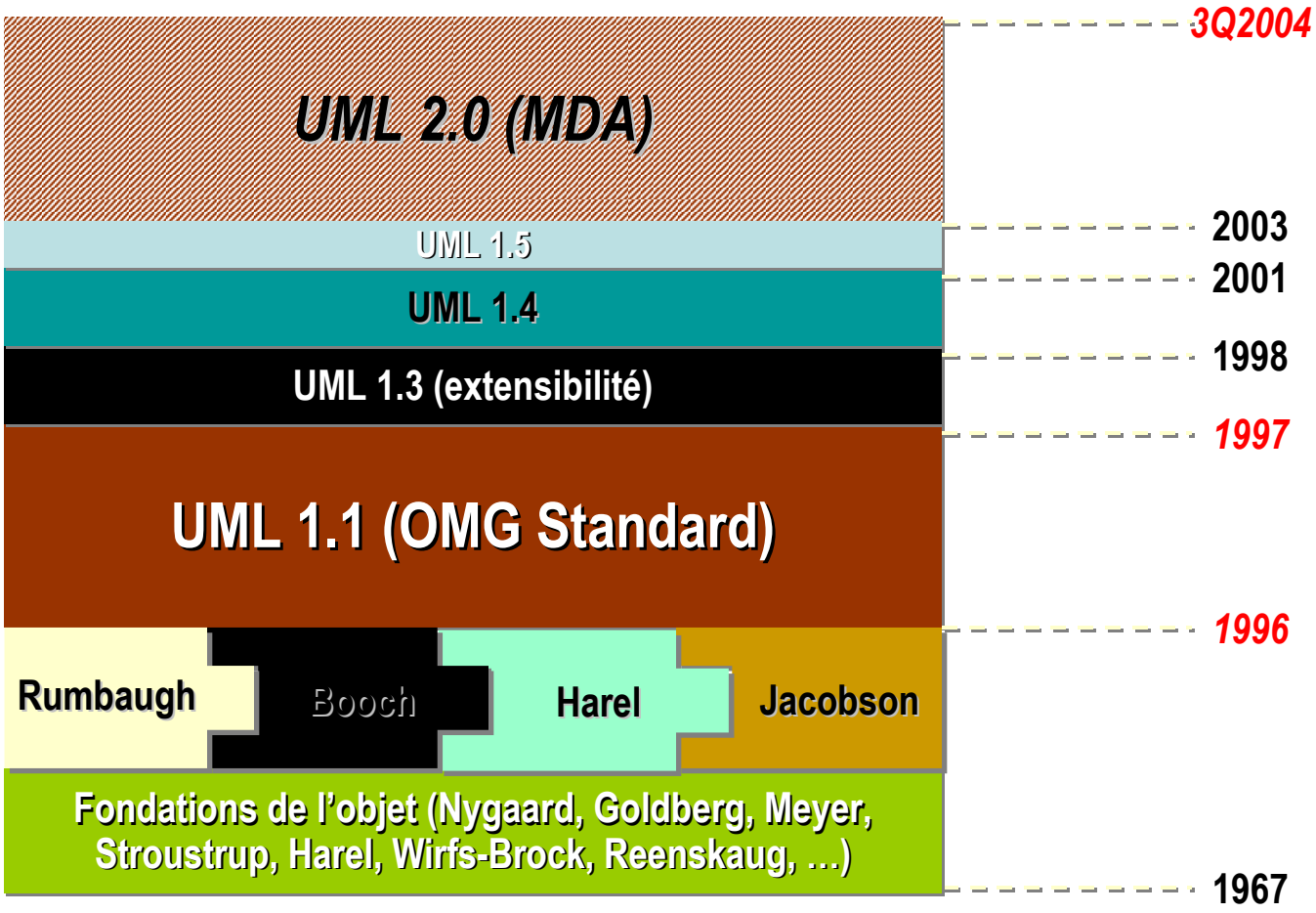
- *Le développement logiciel aujourd'hui*
- ***Les technologies orientées-objet et UML***
- *Les approches MDD/MDA*
- *Un exemple concret : génération d'une application Web à partir d'UML*
- *Conclusion*

Fondements des techniques orientées-objet

- *Abstraction*
  - Simplification de la réalité
- *Encapsulation*
  - Séparation des interfaces et des implémentations
- *Polymorphisme*
  - Capacité à traiter un ensemble d'objets de classes différentes, comme s'ils étaient de la même classe
- *Applications*
  - Design Patterns
  - Plate-forme J2EE
  - Service-Oriented Architecture (SOA)

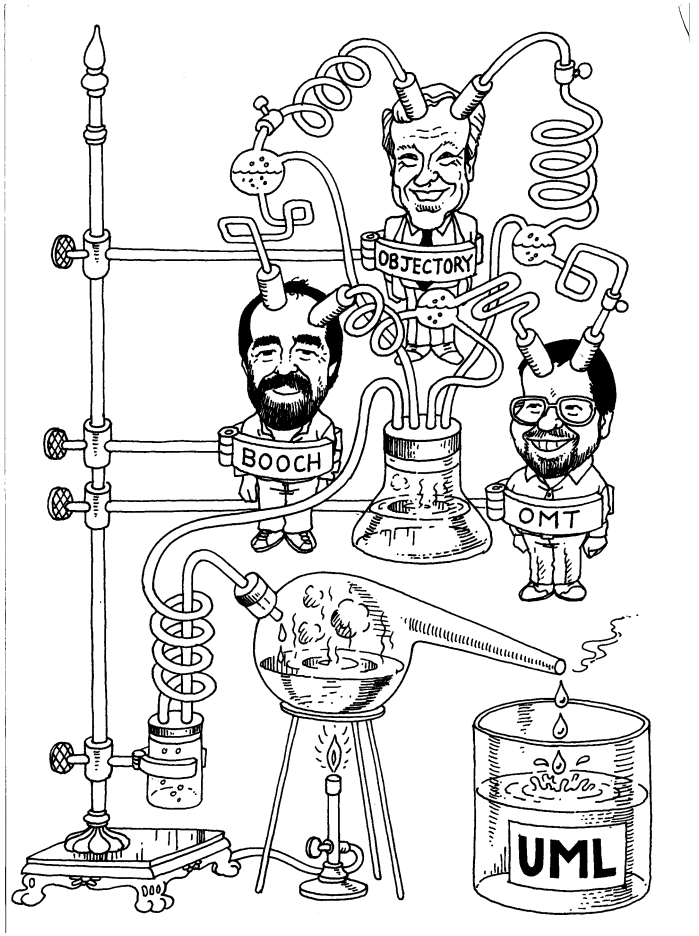
Les bons principes d'ingénierie ne se démodent jamais.
(Grady Booch)

Evolution de UML (Unified Modeling Language)



Pourquoi UML 2 ?

- *Première révision majeure d'UML*
- *Nécessité de modifier le standard pour mieux traiter :*
  - Les exigences MDD (Model-Driven Development) :
    - Précision
    - Génération de code
    - Capacité à exécuter
  - Les « gros » systèmes logiciels

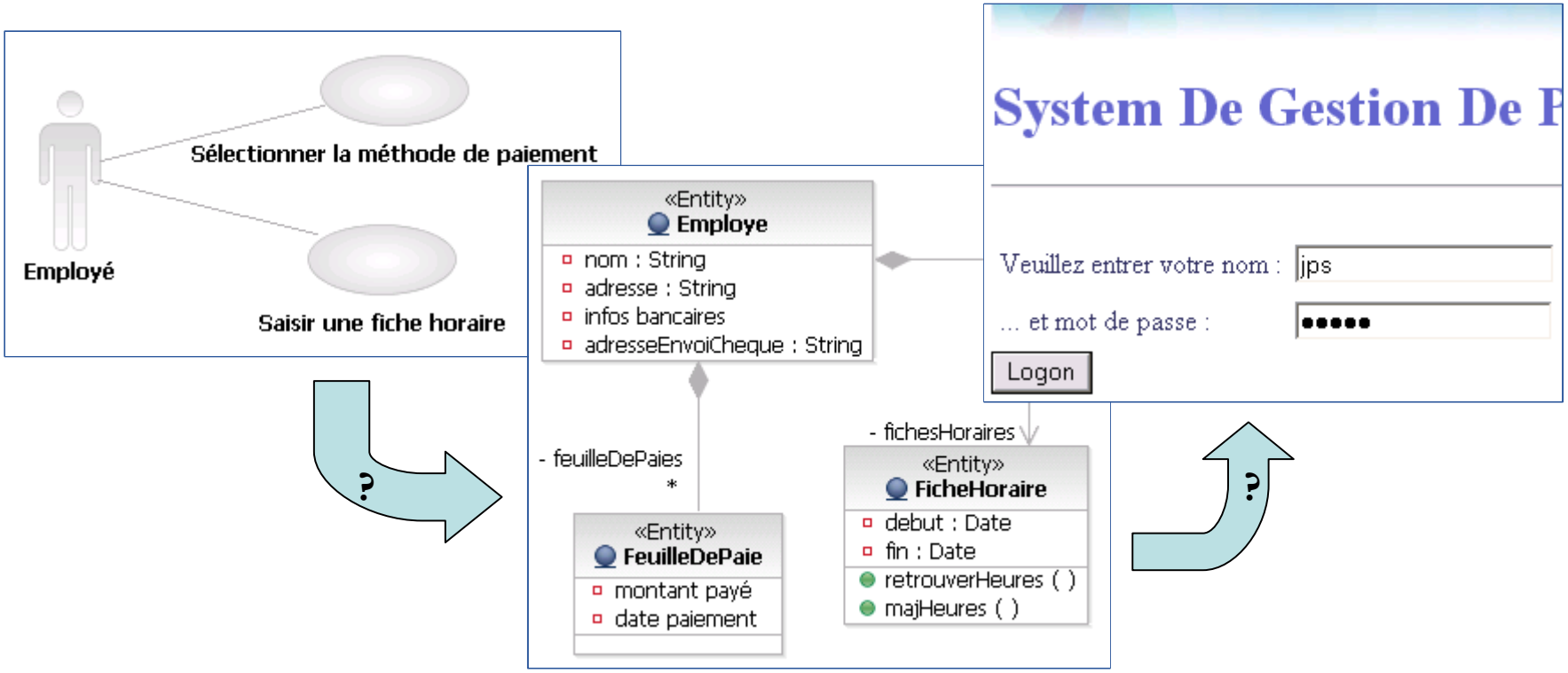


Agenda

- *Le développement logiciel aujourd'hui*
- *Les technologies orientées-objet et UML*
- **Les approches MDD/MDA**
- *Un exemple concret : génération d'une application Web à partir d'UML*
- *Conclusion*

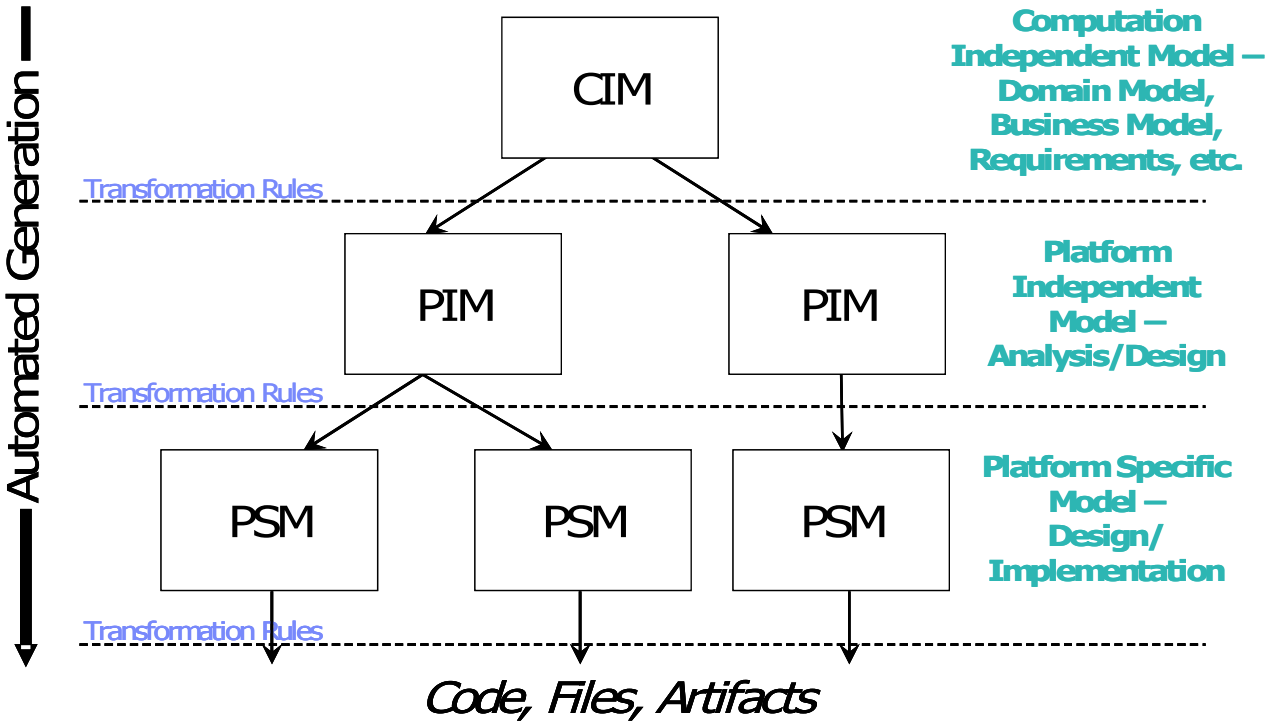
Model-Driven Development (MDD)

- *Une évolution naturelle des technos OO*
- *L'encapsulation de la logique métier dans des modèles*
- *L'utilisation de ces modèles pour automatiser le développement d'applications, la génération de code, le test et la maintenance*



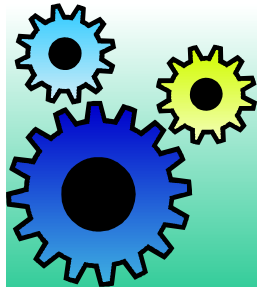
Model-Driven Architecture (MDA)

- *Un « style architectural »*
- *Une initiative OMG*



Transformations

- *Transformations*
  - entre modèles et/ou
  - de modèle à code
- *Les transformations encapsulent :*
  - Les implémentations techniques
  - Les meilleures pratiques
- *Mettent en œuvre des profils UML*
  - Profil = collection de stéréotypes
  - Stéréotypes = pour étendre/spécialiser UML pour des domaines et/ou des applications spécifiques

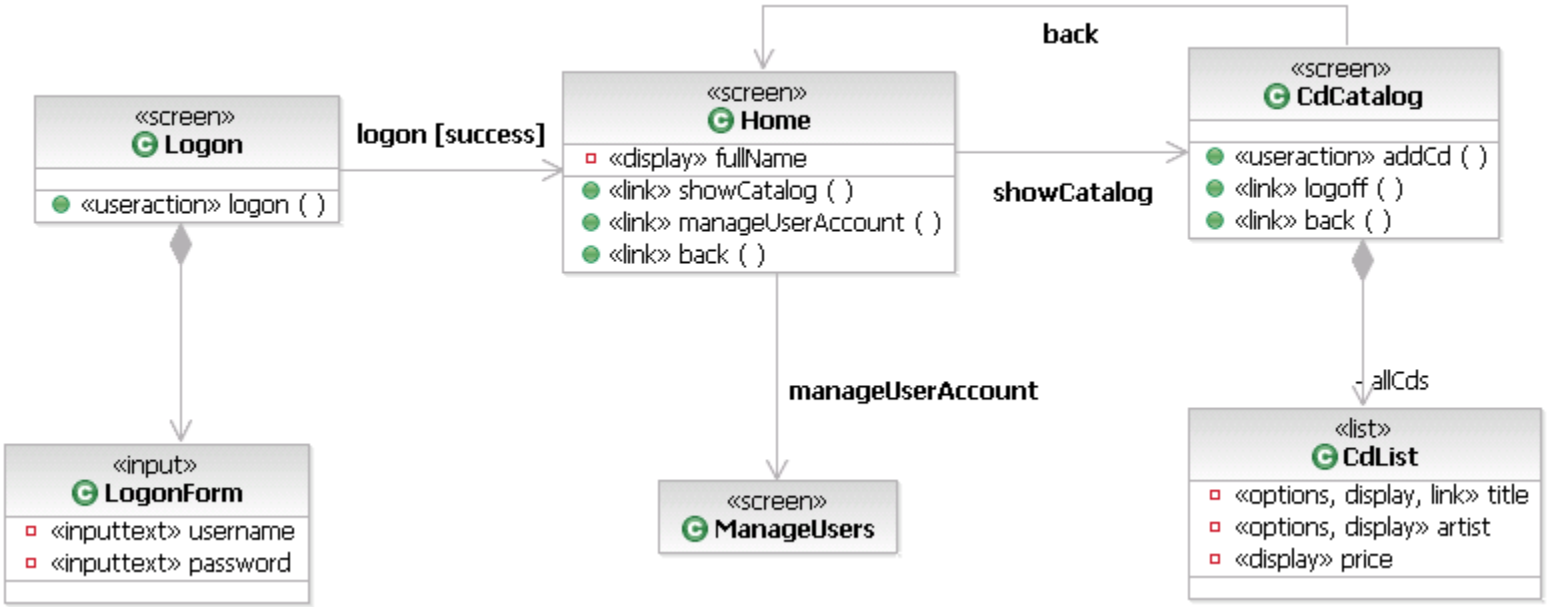


Agenda

- *Le développement logiciel aujourd'hui*
- *Les technologies orientées-objet et UML*
- *Les approches MDD/MDA*
- **Un exemple concret : génération d'une application Web à partir d'UML**
- *Conclusion*

Modélisation UML de l'IHM

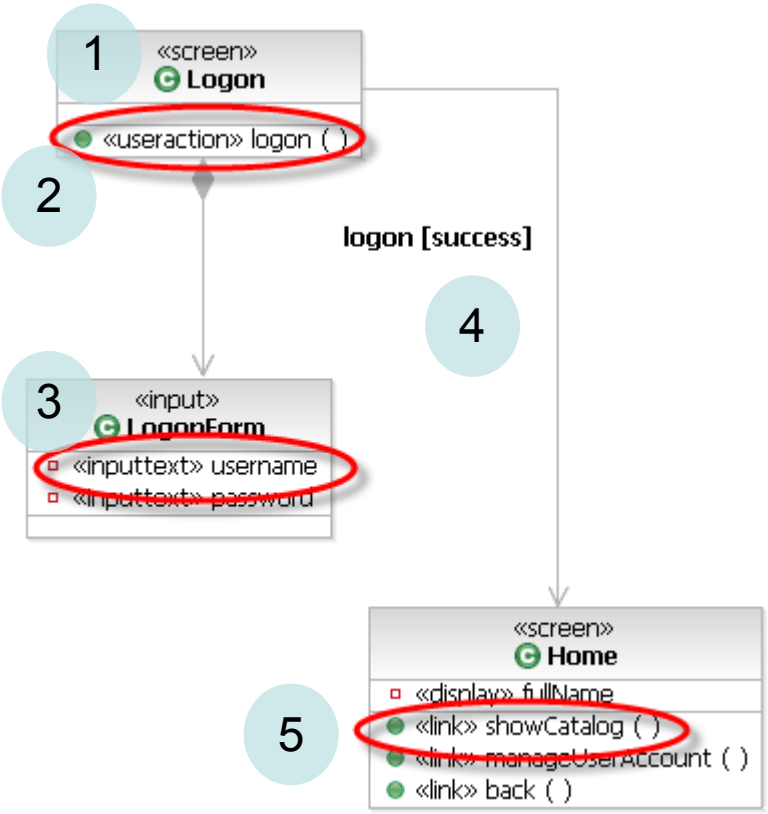
- *Modélisation UML du contenu des écrans et de la navigation entre écrans (User eXperience Modeling) :*

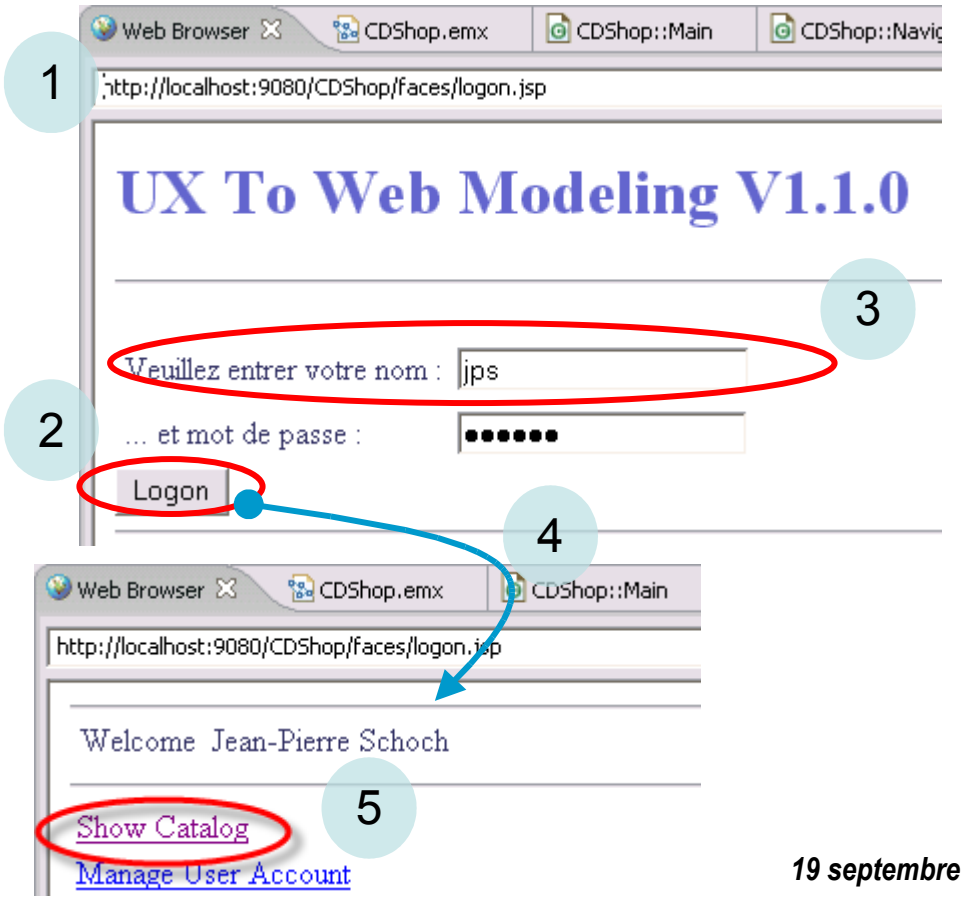


- *Et si on pouvait générer les écrans à partir du modèle, …*
- *Et l'accès à la base de données … ?*

Transformation « UX To Web »

- *Utilise JSF pour la couche présentation*
  - Mapping entre les classes UML et les écrans de l'IHM : profil UXModeling
- *Utilise Hibernate pour la couche persistence (classes « entité » ou « beans »)*
  - Mapping entre les classes UML (« entité ») et les tables : profil DataModeling





Démonstration

UX To Web Modeling V1.1.0

- *Présentation du modèle UML*
- *Génération et exécution de l'application Web*
- *Modification du modèle UML (logique métier)*
- *Modification de la génération de code (template JET)*

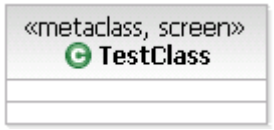
Comment ça marche … en quelques mots

- *La transformation utilise deux règles de transformation :*
  - La règle d'extraction des données du modèle UML :
    - Résultat stocké sous forme d'une représentation interne
    - Par exemple, une classe stéréotypée <<screen>> et les données associées sont stockés dans une structure « JSP »
  - La règle de génération de code :
    - Retrouve les données à partir de la représentation interne
    - Applique les templates JET de génération de code (diapo suivante)

- *Cette séparation signifie qu'il est très facile d'étendre la transformation :*
  - Par exemple, à partir de la même représentation interne d'une « JSP », la règle de génération peut appliquer un template Struts ou un template JSF.

Templates JET (Java Emitter Templates)

- *Un template par type de fichier à générer (par exemple : page JSP, fichier XML pour la configuration de Hibernate, classe Java) : déclarations du « langage » ciblé + tags Java (en rouge dans l'exemple)*
- *Une classe de génération compilée « en ligne » pour chaque template, avec un opération « generate » qui prend un paramètre représentant l'objet à générer*




```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stereotypedClass name="TestClass">
   <stereotype name="UXModeling2::screen">
   </stereotype>
   <stereotype name="Basic::Metaclass">
   </stereotype>
</stereotypedClass>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<$
   Class cl = (Class) argument;
   if (cl.getAppliedStereotypes() != null) {$>
<stereotypedClass name="<$= cl.getName()$>"><$
      for (Iterator j = cl.getAppliedStereotypes().iterator(); j.hasNext();) {
         Stereotype st = (Stereotype) j.next();$>
   <stereotype name="<$= st.getQualifiedName()$>"><$
   </stereotype><$
      }$>
</stereotypedClass><$
    }
$>
```

Agenda

- *Le développement logiciel aujourd'hui*
- *Les technologies orientées-objet et UML*
- *Les approches MDD/MDA*
- *Un exemple concret : génération d'une application Web à partir d'UML*
- **Conclusion**

- *Expertise technique directement capturée dans les transformations au lieu d'être documentée sous forme de guides de projet ou d'être « redécouverte » encore et encore…*
- *Pas que du code : beaucoup des « artefacts » logiciels autres que le code dérivés complètement ou partiellement à partir des modèles*
- *Parce que les modèles de haut niveau ne sont pas pollués par les détails d'implémentation, il est (beaucoup) plus facile de gérer les changements de technologie. En particulier :*
  - Il est possible d'essayer différentes idées avant de faire un choix définitif
  - Les « mauvaises » décisions sont facilement changées
- *L'approche MDD est particulièrement rentable quand elle est appliquée au niveau d'une organisation : le ROI des transformations augmente chaque fois qu'elles sont réutilisées…*

MDD/MDA marque une nouvelle ère du développement logiciel : Son industrialisation