IBM Software Group

Développement à base de modèles avec IBM Rational Rose XDE

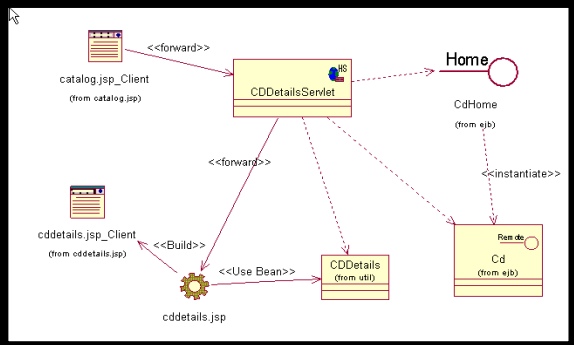
Jeudi 13 mai 2004 – Planet Hollywood

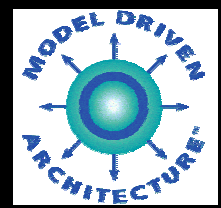
Rational. software

e business on demand software

Agenda

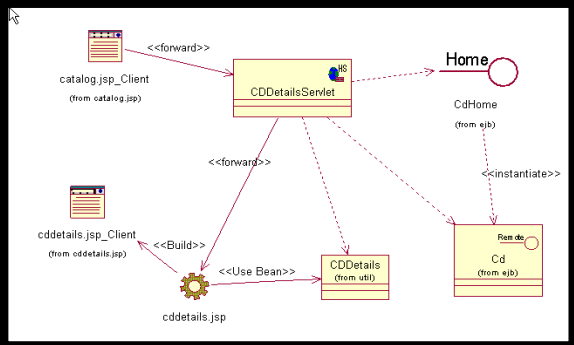
- Développement piloté par le métier
- De l'intérêt de modéliser : Quels bénéfices ?
- Les différentes approches de modélisation
- MDA, MDD, UML 2.0 et toutes ces choses ….
- Les solutions IBM pour le développement à base de modèles
- Les bénéfices : retour clients et analyses de marché

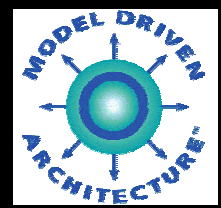




Agenda

- Développement piloté par le métier
- De l'intérêt de modéliser : Quels bénéfices ?
- Les différentes approches de modélisation
- MDA, MDD, UML 2.0 et toutes ces choses ….
- Les solutions IBM pour le développement à base de modèles
- Les bénéfices : retour clients et analyses de marché





Le développement logiciel : *une clé pour l'entreprise …*

… pour mieux contribuer aux résultats

Réactive | **Adaptable** | **Focalisée sur son métier** | **Robuste**

… pour automatiser et intégrer les processus métier

Automatise et intègre les processus métier

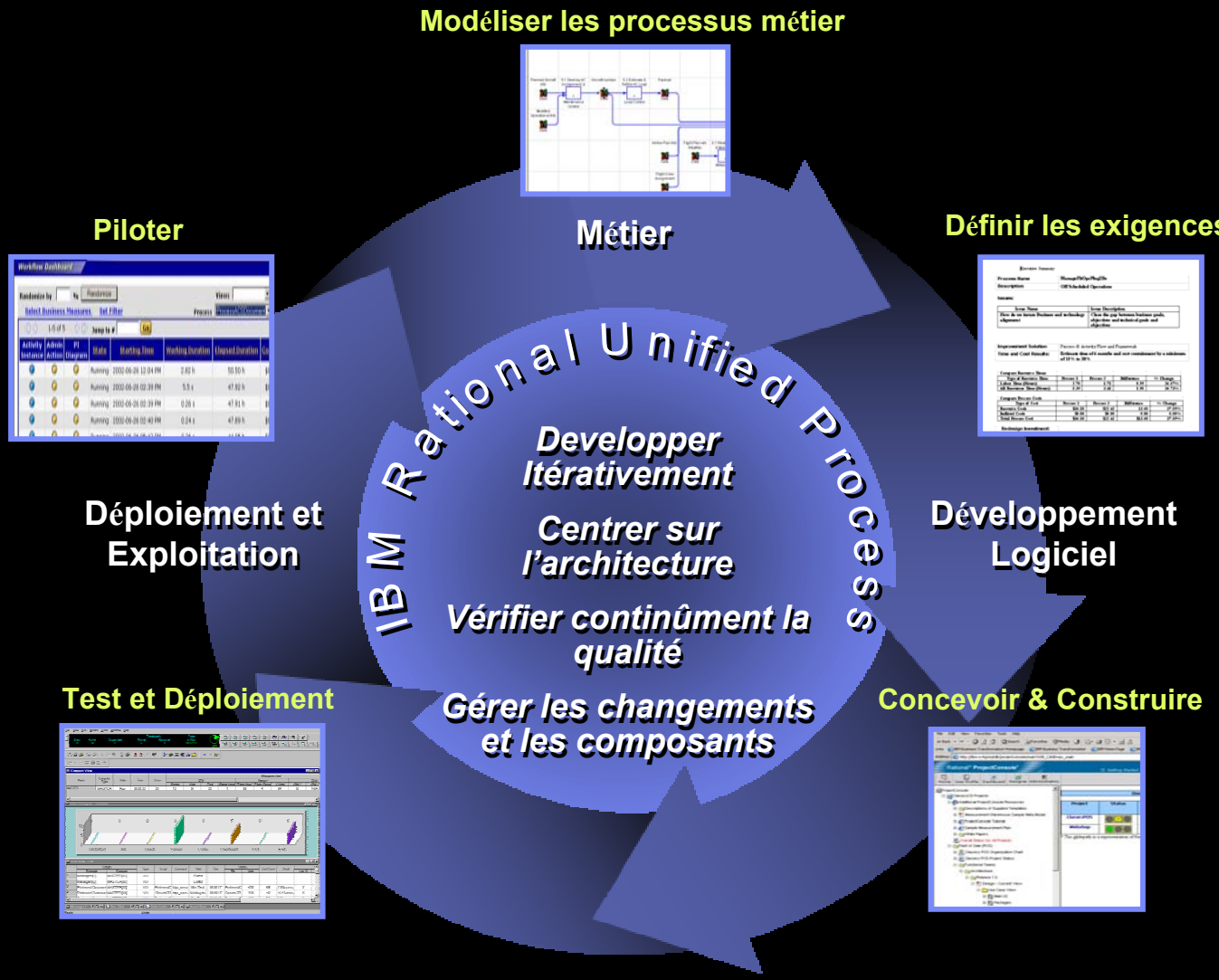
S'adapte *rapidement* aux évolutions métier | **Crée un avantage stratégique** | **Fiables, évolutifs et plus faciles à gérer**

Améliorer sa capacité à développer du logiciel …

La valeur plus vite

Processus itératif | **Focalisation sur l'architecture** | **Vérification continue de la qualité** | **Gestion des changements et des composants**

Processus de développement piloté par le métier



La plate-forme de développement logiciel




Analyste | Architecte | Développeur | Testeur

Modélise, simule et pilote les processus métier

Modélise les applications et les données

Construit visuellement les programmes, et génère le code

Conçoit, crée, et exécute les tests

Mobile et Embarqué

DB2 Lotus Tivoli WebSphere

Chef de Projet

- Applique un processus commun
- Suit l'avancement du projet
- Gère les exigences
- Gère les changements et les composants
- Gère la qualité

Serveurs IBM et autres

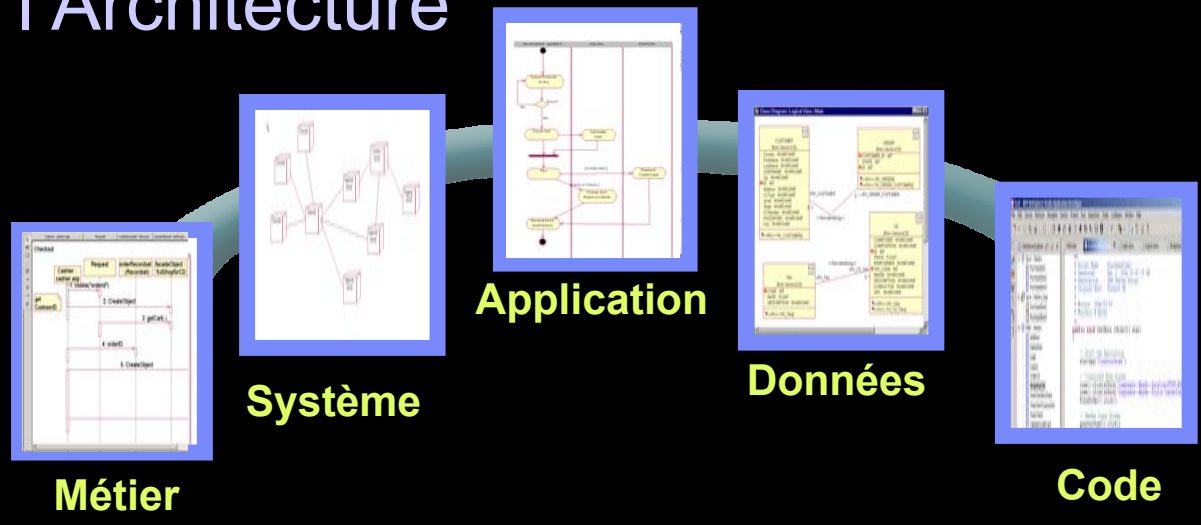
Partenaires

TEAM UNIFYING PLATFORM

Microsoft | *Standards Ouverts* | *Services IBM*

Focalisation sur l'Architecture

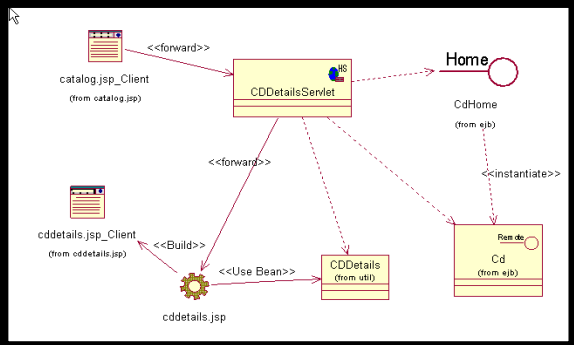
- ✓ Utiliser des architectures à base de composants et de services
- ✓ Développer rapidement
- ✓ Réutiliser des composants

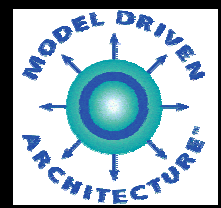


Applications métier	**S'adaptent *rapidement* aux évolutions métier**	**Créent un avantage stratégique**	**Fiables, évolutives et plus faciles à gérer**
Focalisation sur l'architecture pour …	**Concevoir pour le changement**	**Réduire la complexité ; travailler au bon niveau d'abstraction**	**Garantir l'intégrité et la qualité finale de l'architecture**

Agenda

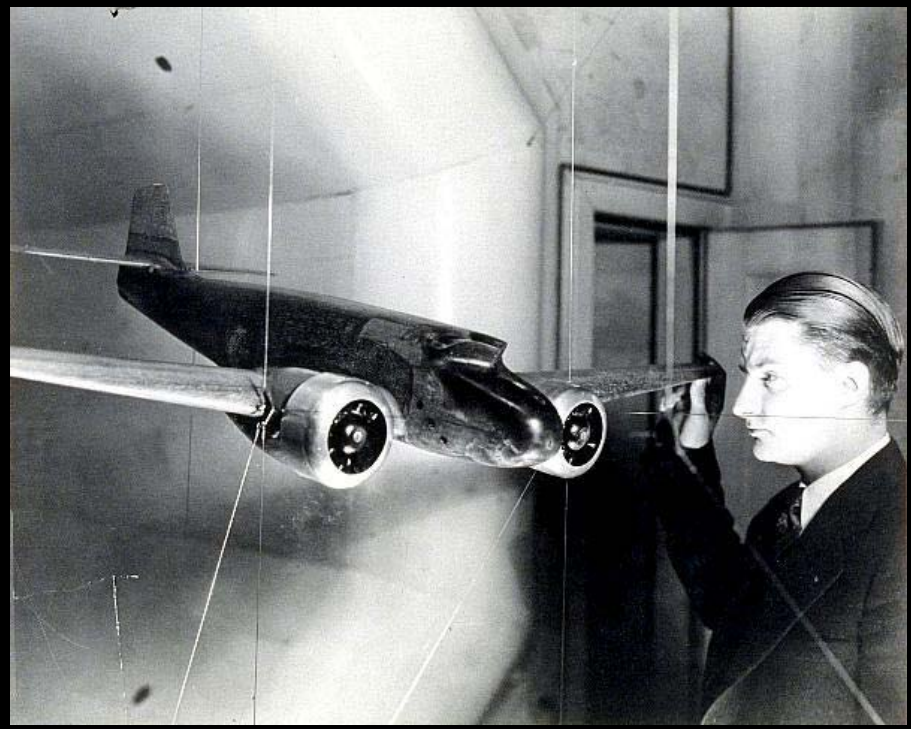
- Développement piloté par le métier
- De l'intérêt de modéliser : Quels bénéfices ?
- Les différentes approches de modélisation
- MDA, MDD, UML 2.0 et toutes ces choses ….
- Les solutions IBM pour le développement à base de modèles
- Les bénéfices : retour clients et analyses de marché





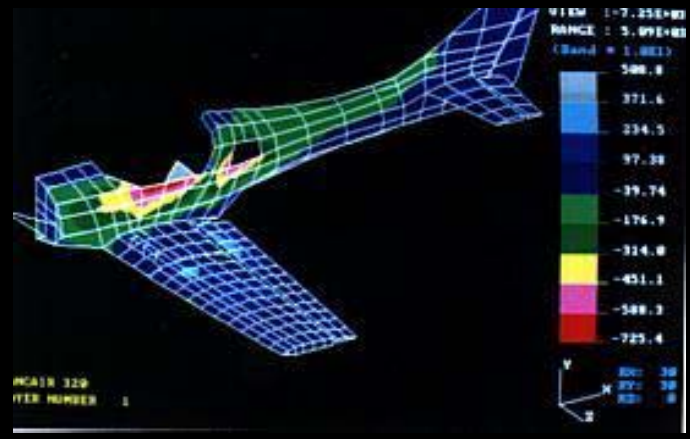
Les modèles dans l'ingénierie du logiciel

- Principe aussi *ancien* que l'ingénierie elle-même
- Une manière *historique* de réduire les risques



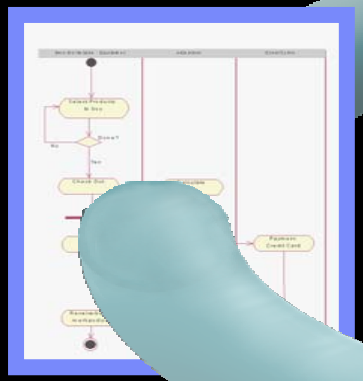
Pourquoi modéliser ?

- Simplifier le développement
- Communiquer sans ambiguïté
- Comprendre la structure
- Élever le niveau d'abstraction
- Formaliser le problème et sa solution
- Piloter la fabrication finale
- Séparer le *quoi* du *comment*
- Mieux analyser l'impact
- S'assurer très tôt d'un certain niveau de qualité

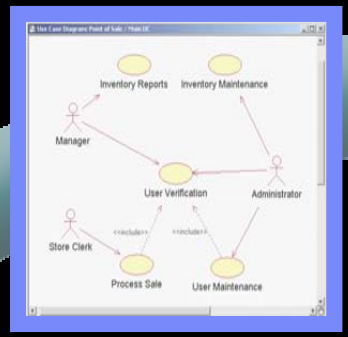


Que modéliser ?

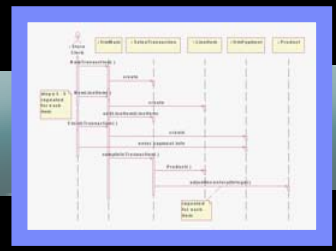
Modélisation métier



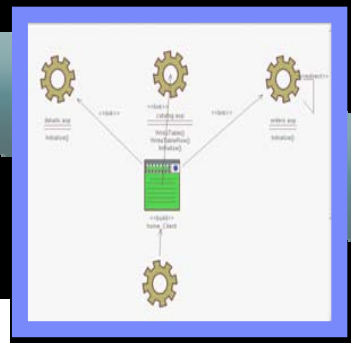
Modélisation des exigences



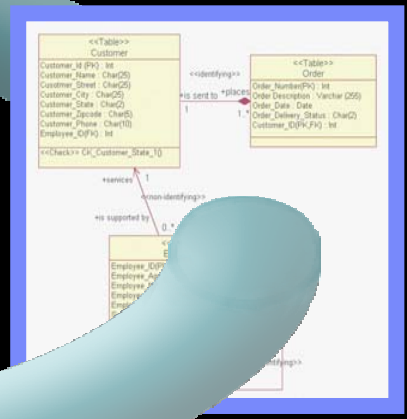
Modélisation de l'application



Modélisation du web



Modélisation des données



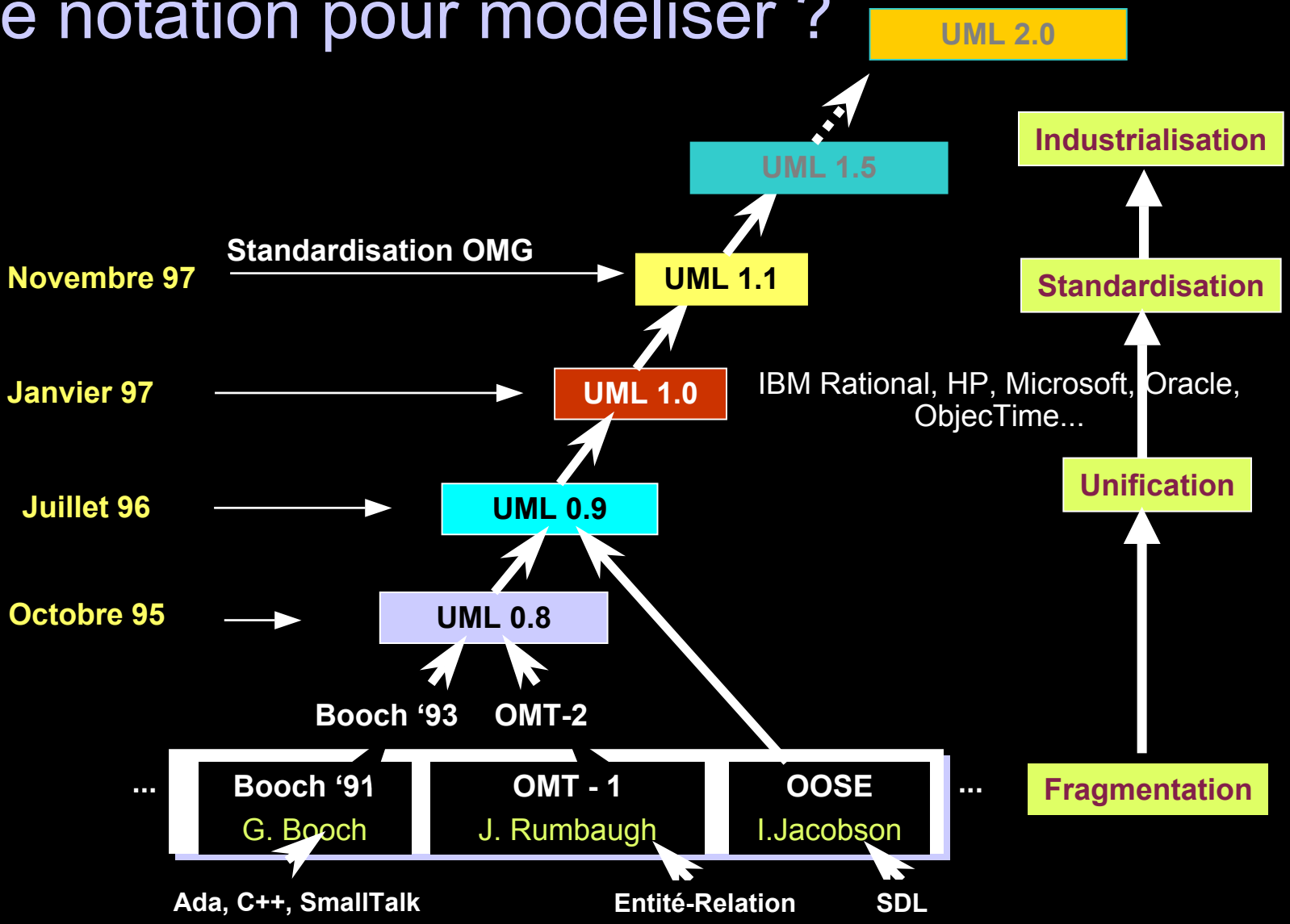
UNIFIED MODELING LANGUAGE

UNIFIED!

*Ne pas oublier la modélisation libre

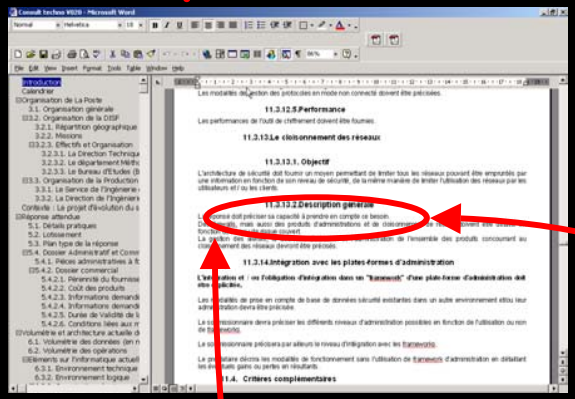
Un langage unique pour l'équipe

Quelle notation pour modéliser ?

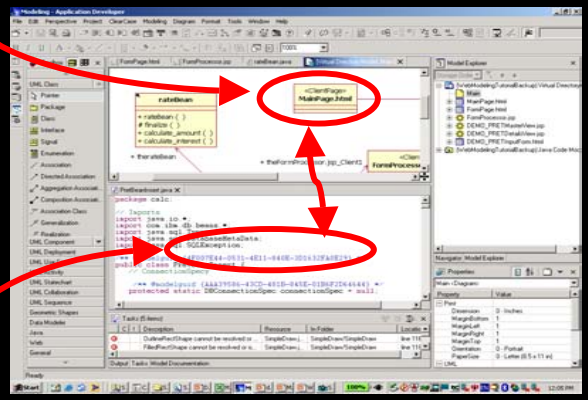


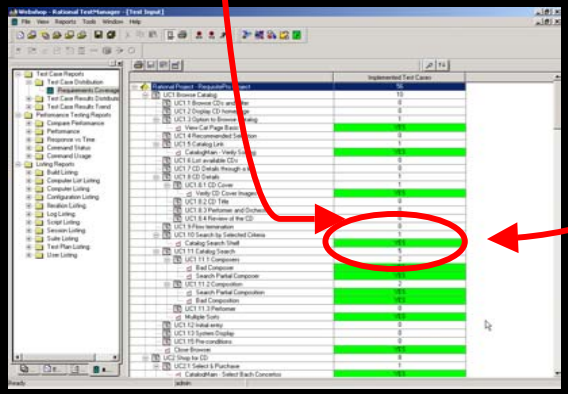
Quel positionnement dans un cycle de vie ?

La spécification



La conception visuelle



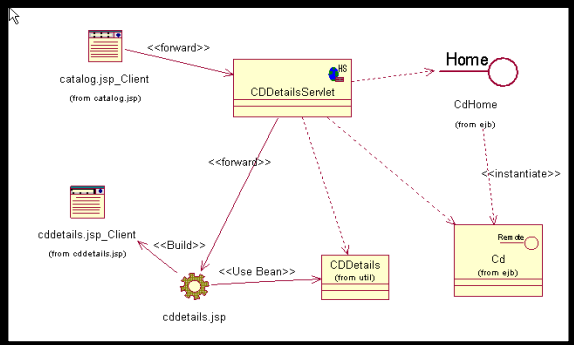


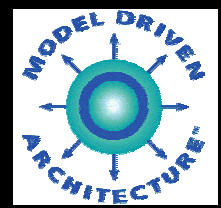
Le code source

La plan de test

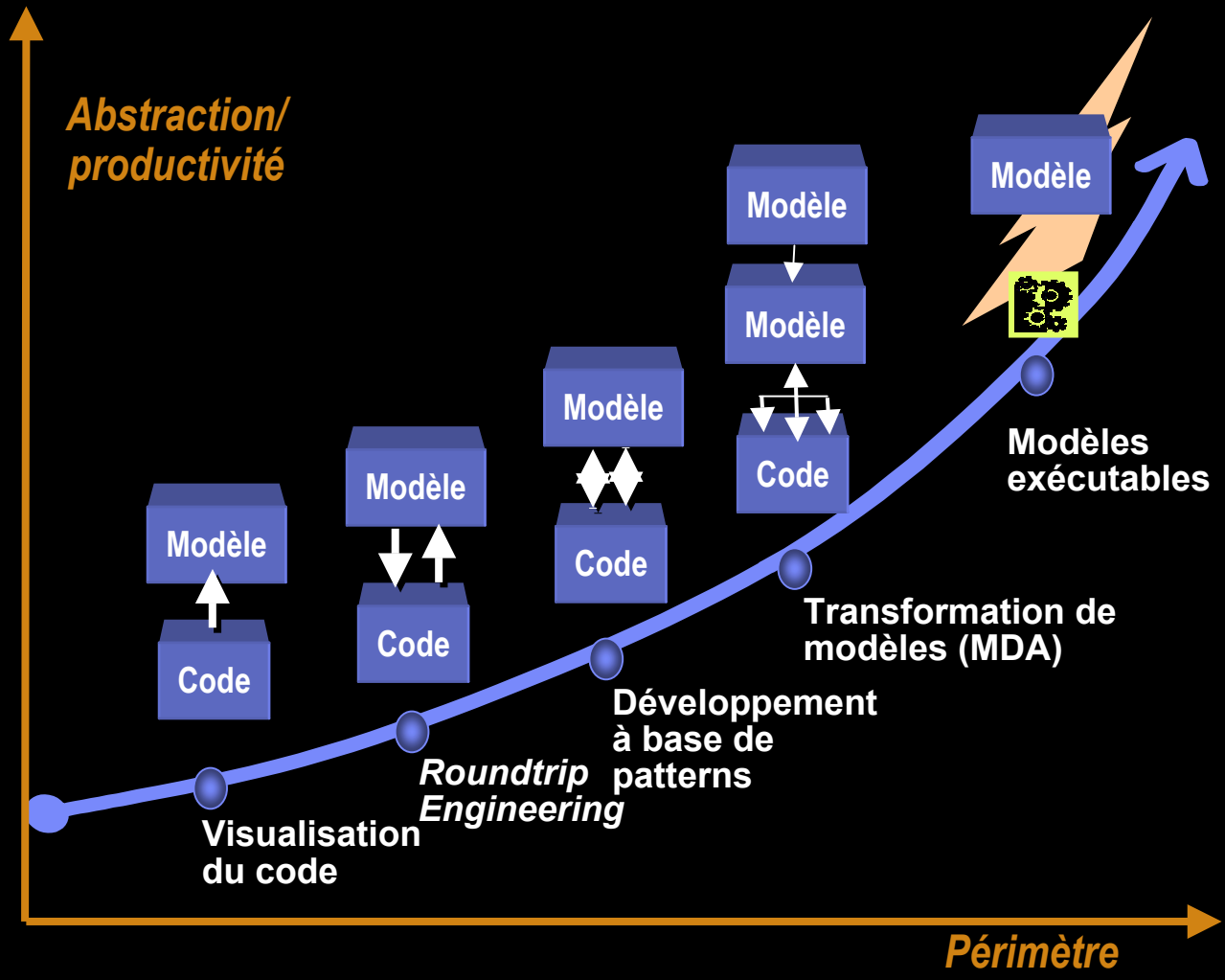
Agenda

- Développement piloté par le métier
- De l'intérêt de modéliser : Quels bénéfices ?
- Les différentes approches de modélisation
- MDA, MDD, UML 2.0 et toutes ces choses ….
- Les solutions IBM pour le développement à base de modèles
- Les bénéfices : retour clients et analyses de marché





Les différents paradigmes de modélisation

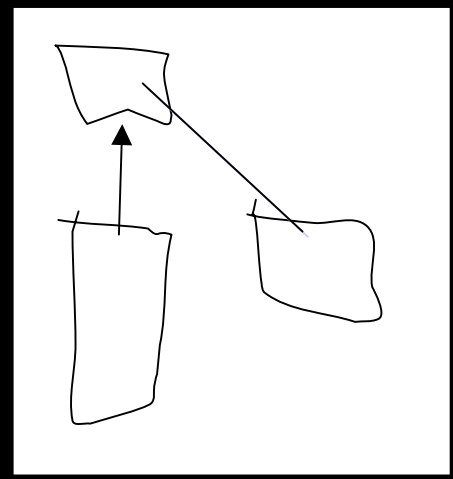


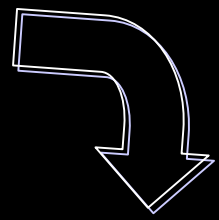
UML adapté à plusieurs:

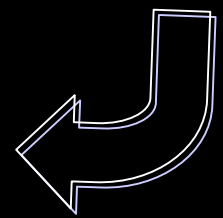
- Langages de développement
- Environnement d'exécution
- Niveaux de compétence
- Paradigmes de modélisation

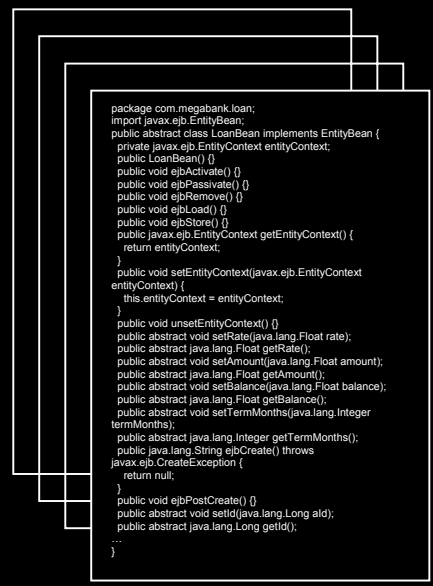
Niveau 0: Manuel

- Modélisation ad-hoc
  - En général non-outillée
  - Basé sur le papier et le crayon avec une transformation manuelle vers l'implémentation
- Difficile de maintenir la relation entre le modèle et l'implémentation
- Encore utile car :
  - Aide à réfléchir lors de la confrontation à un problème complexe
  - Vous permet de visualiser et communiquer des documents détaillés





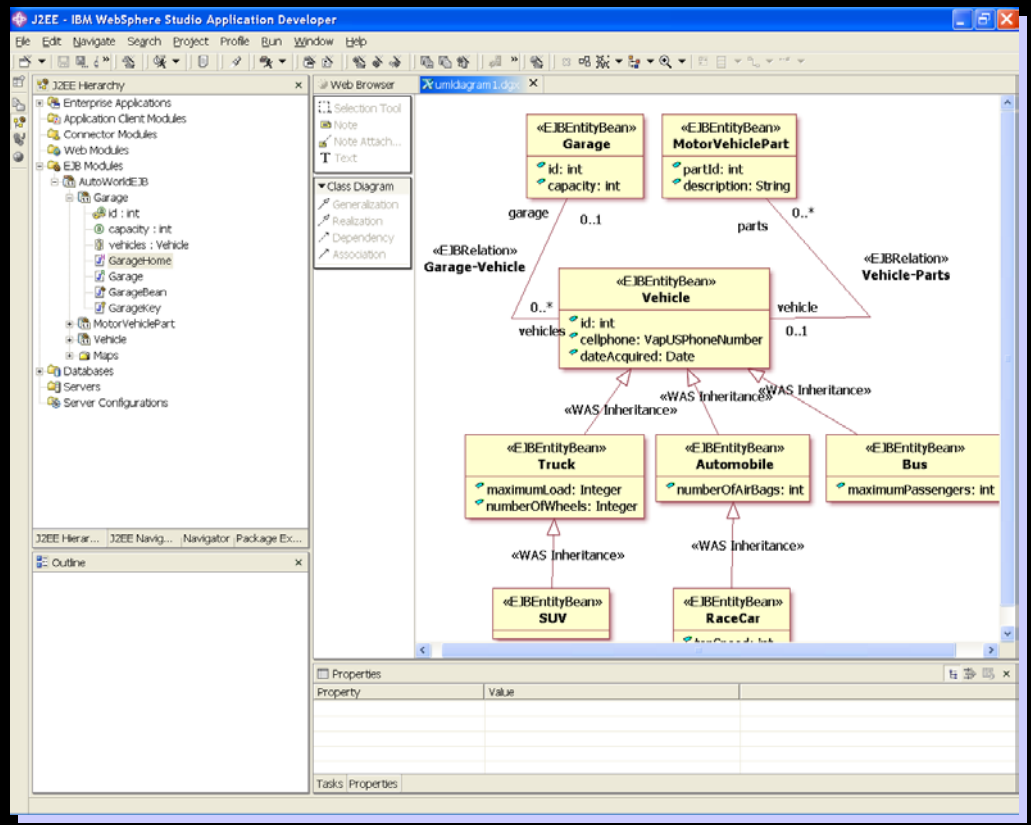




```
package com.megabank.loan;
import javax.ejb.EntityBean;
public abstract class LoanBean implements EntityBean {
  private javax.ejb.EntityContext entityContext;
  public LoanBean() {}
  public void ejbActivate() {}
  public void ejbPassivate() {}
  public void ejbRemove() {}
  public void ejbLoad() {}
  public void ejbStore() {}
  public javax.ejb.EntityContext getEntityContext() {
    return entityContext;
  }
  public void setEntityContext(javax.ejb.EntityContext
entityContext) {
    this.entityContext = entityContext;
  }
  public void unsetEntityContext() {}
  public abstract void setRate(java.lang.Float rate);
  public abstract java.lang.Float getRate();
  public abstract void setAmount(java.lang.Float amount);
  public abstract java.lang.Float getAmount();
  public abstract void setBalance(java.lang.Float balance);
  public abstract java.lang.Float getBalance();
  public abstract void setTermMonths(java.lang.Integer
termMonths);
  public abstract java.lang.Integer getTermMonths();
  public java.lang.String ejbCreate() throws
javax.ejb.CreateException {
    return null;
  }
  public void ejbPostCreate() {}
  public abstract void setId(java.lang.Long aId);
  public abstract java.lang.Long getId();
  ...
}
```

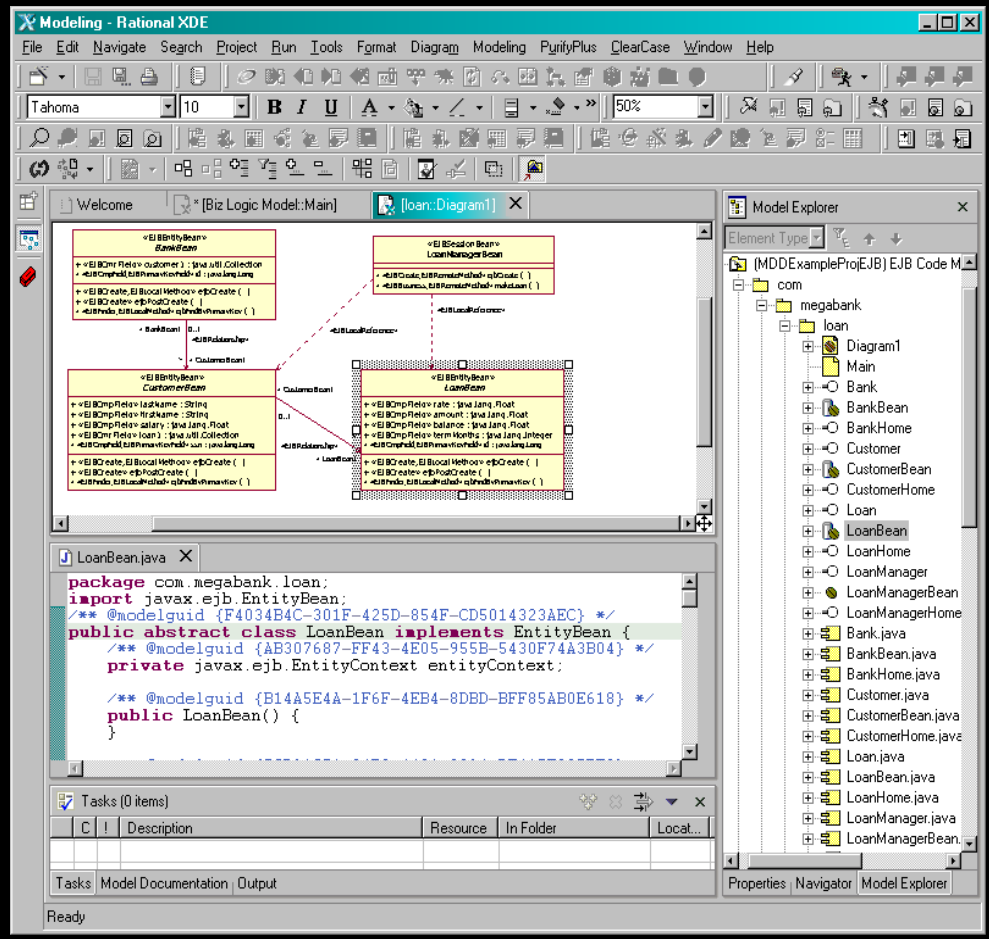
Niveau 1: Visualisation

- La plupart du temps : une photo du code
- **Objectif**
  - Documenter
  - Naviguer
  - Comprendre
  - Évaluer le couplage
  - Réutiliser



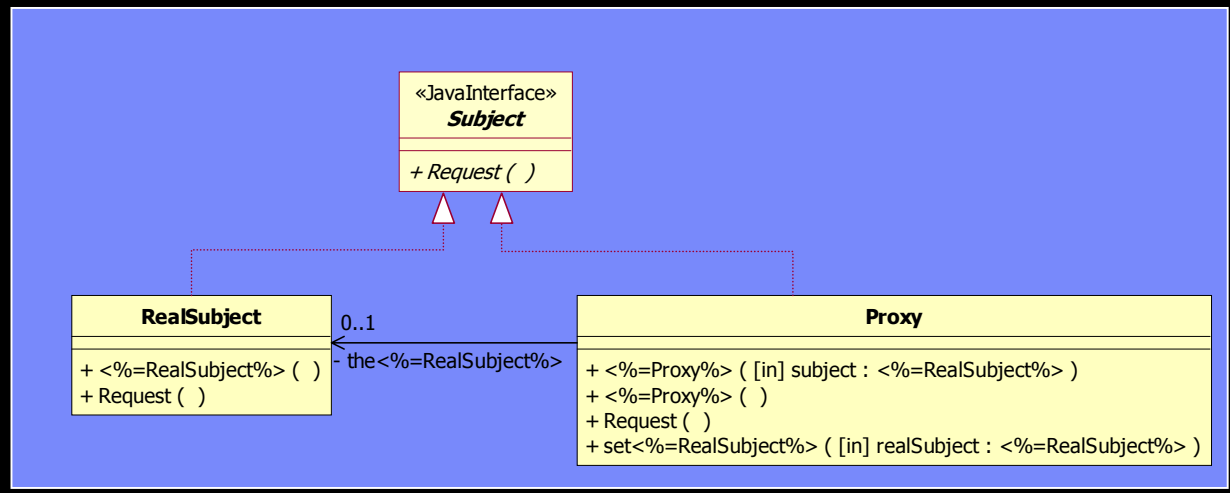
Niveau 2: Synchronisation

- Modèle et code restent synchronisés (1 - 1)
  - Automatique
  - Manuel
- Avec résolution des conflits
- Liberté d'éditer le code ou le modèle



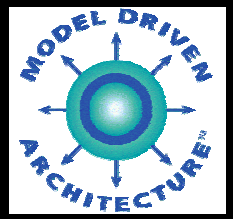
Niveau 3: Patterns

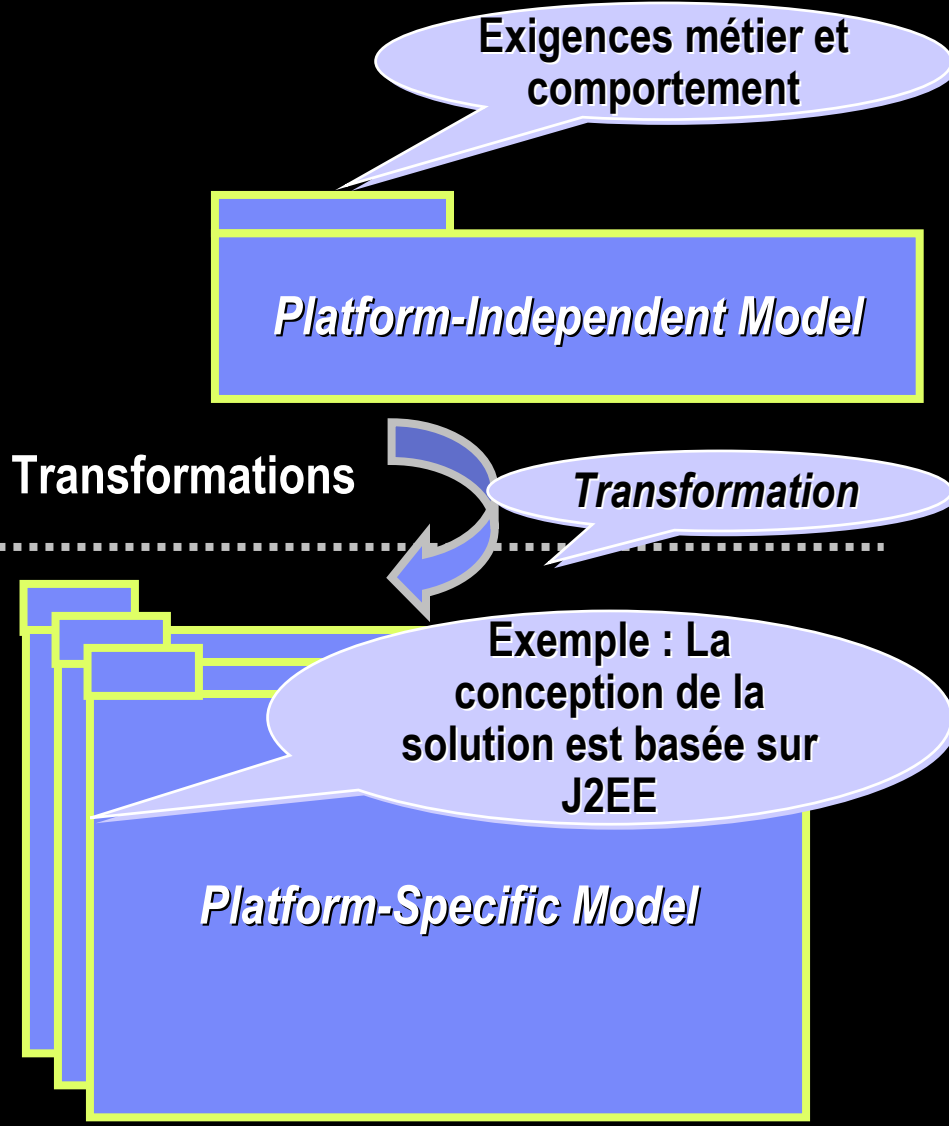
- *“A pattern provides a common solution to a problem in a specific context” (Grady Booch)*
- Facilite la réutilisation de savoir-faire éprouvé
- Le pattern se paramètre pour être applicable
- Spécifique au contexte et au domaine d’application
  - GoF Patterns
  - IBM e-business



Niveau 4: Transformation de modèles

Les investissements faits sur les *PIM* sont protégés des changements technologiques

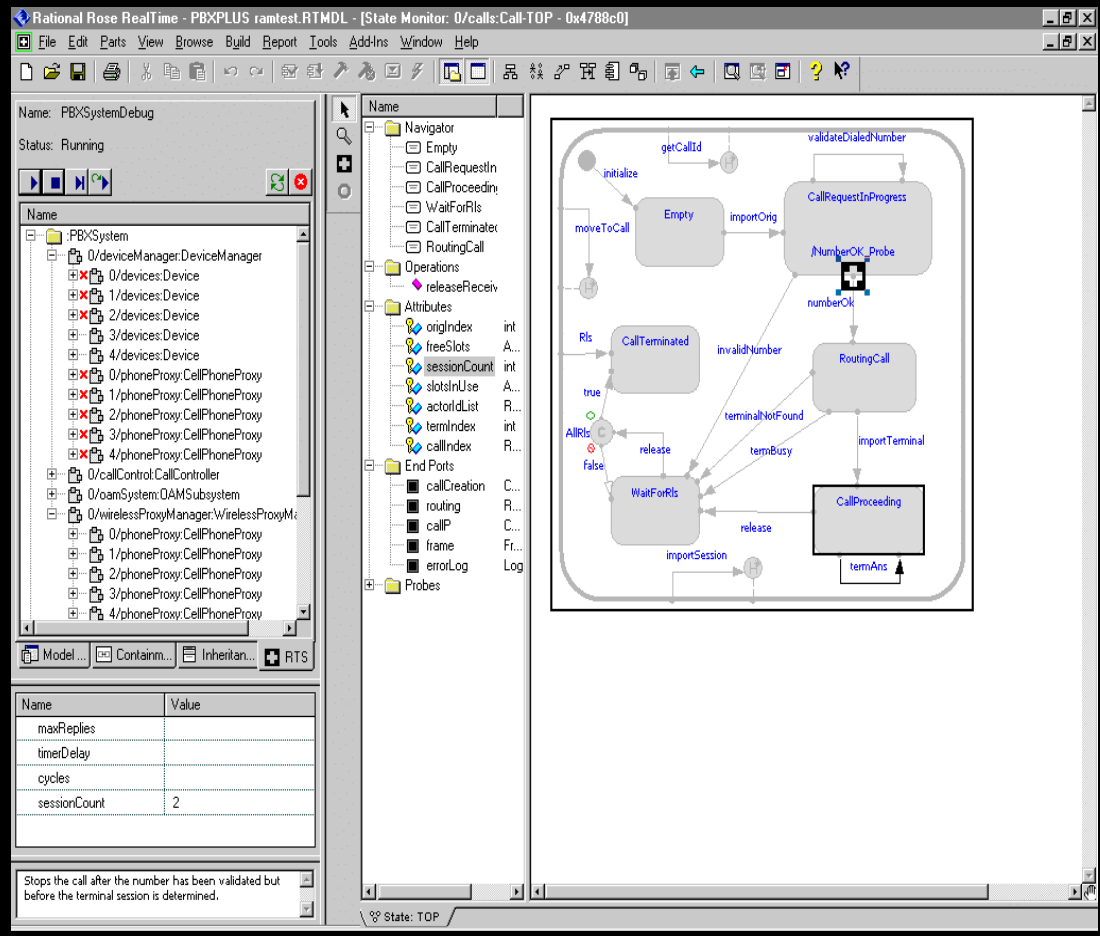




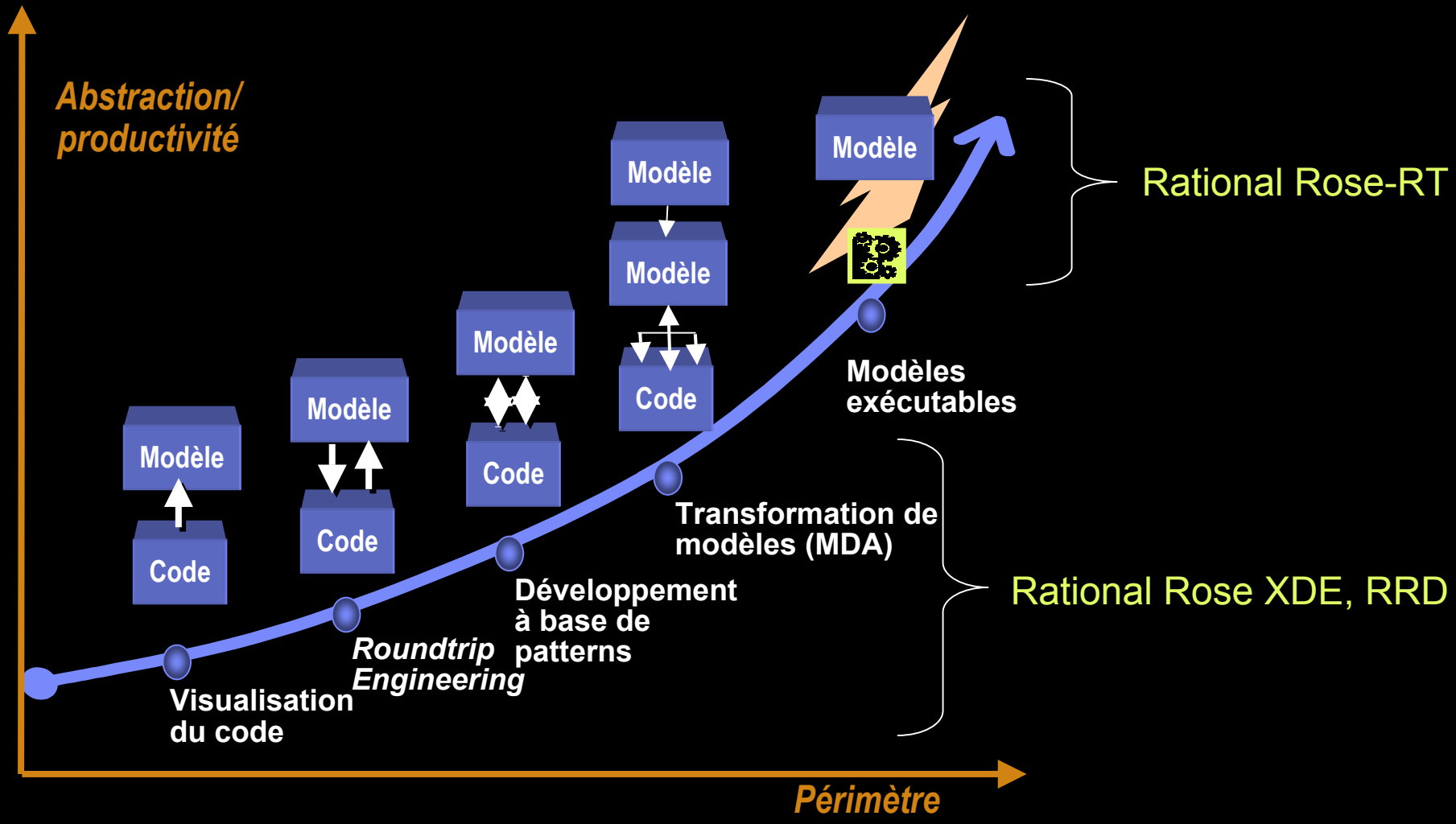
Le PIM est transformé afin de produire des PSM spécifiques de l'implémentation.

Niveau 5: Modèles exécutables

- Les modèles eux-mêmes sont exécutables
- La modélisation est précise et non-ambiguë, un minimum de code est requis
- Les utilisateurs ne travaillent la plupart du temps qu'avec des modèles
- Moins d'emphase sur les transformations de modèles
- Le modèle est compilé afin de produire du code (Rose-RT) exécutable

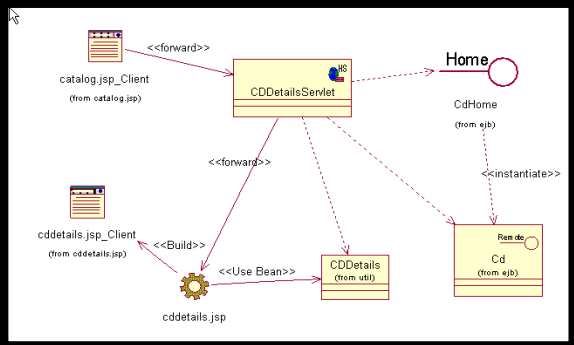


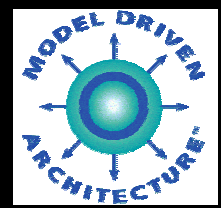
Les différents paradigmes de modélisation



Agenda

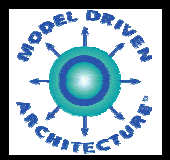
- Développement piloté par le métier
- De l'intérêt de modéliser : Quels bénéfices ?
- Les différentes approches de modélisation
- MDA, MDD, UML 2.0 et toutes ces choses ….
- Les solutions IBM pour le développement à base de modèles
- Les bénéfices : retour clients et analyses de marché





MDA : Model Driven Architecture

- Une initiative de l'OMG visant à définir un ensemble de standard pour supporter le développement à base de modèles
  - Un standard pour la transformation de modèles
  - Un standard pour versionner les modèles
  - Un standard pour l'échange de modèles
  - Des langages de modélisation et des profils (UML, MOF, CWM) pour des domaines d'application particuliers
- Une conséquence de l'état de maturité des outils et des langages
- UML 2.0 est la pierre angulaire de Model Driven Architecture



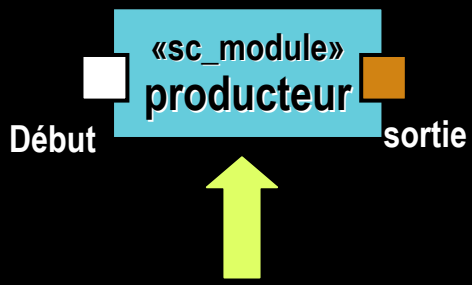
Développement à base de modèles ?

Une approche du développement logiciel ou la production principale est composée de modèles au lieu de programmes

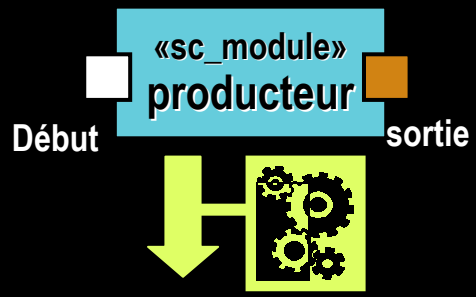
(1) ABSTRACTION

(2) AUTOMATISATION

Le domaine des langages de modélisation (i.e., UML)




```
SC_MODULE(producer)
{sc_inslave<int> in1;
int sum; //
void accumulate (){
sum += in1;
cout << "Sum = " <<
sum << endl;}
```




```
SC_MODULE(producer)
{sc_inslave<int> in1;
int sum; //
void accumulate (){
sum += in1;
cout << "Sum = " <<
sum << endl;}
```

Le domaine des outils:

- Génération automatisée de code
- Exécution de modèles
- Vérification formelle
- Génération et exécution de test

- L'OMG a lancé l'initiative *Model-Driven Architecture* (MDA) afin de fournir un ensemble de standards pour faire du *développement à base de modèle (MDD)*.

Développement à base de modèles ?

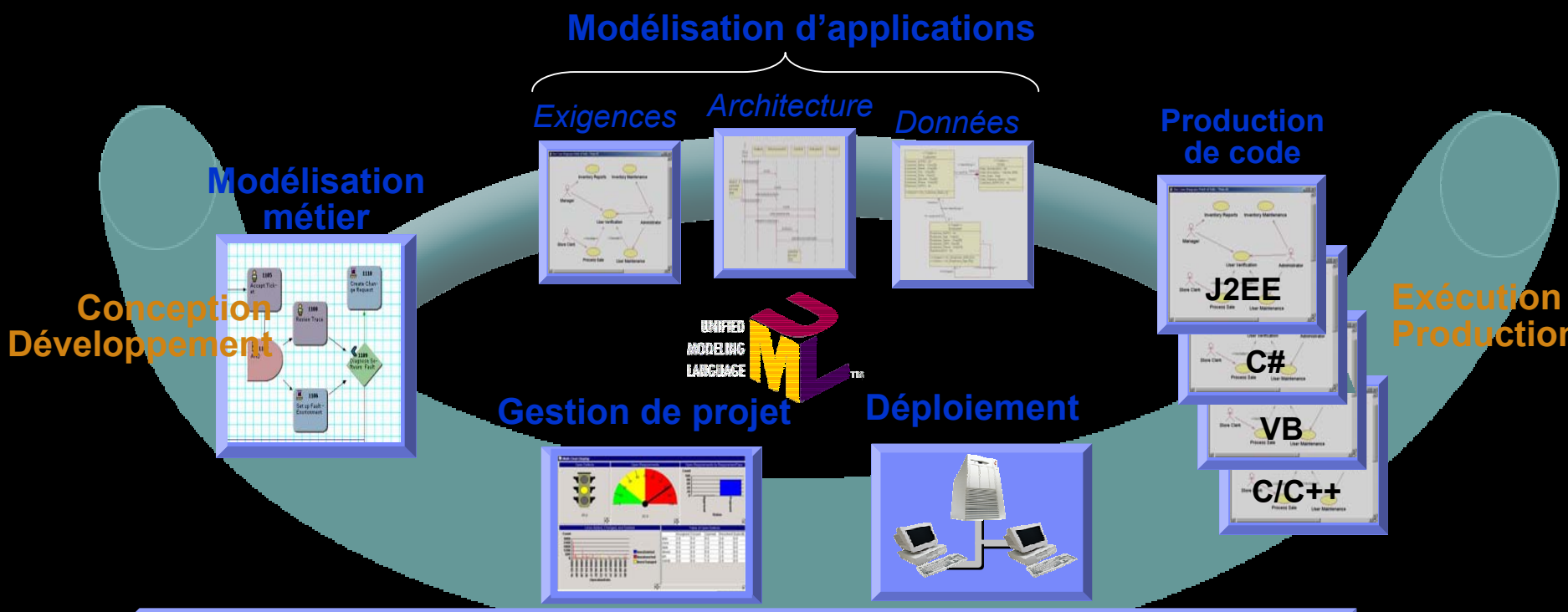


Plate-forme MDD ouverte

- Eclipse, EMF, XMI et API publiques
- Présentation simple adaptée à chaque rôle
- Interopérabilité et traçabilité assurée par une représentation interne commune basée sur la modélisation

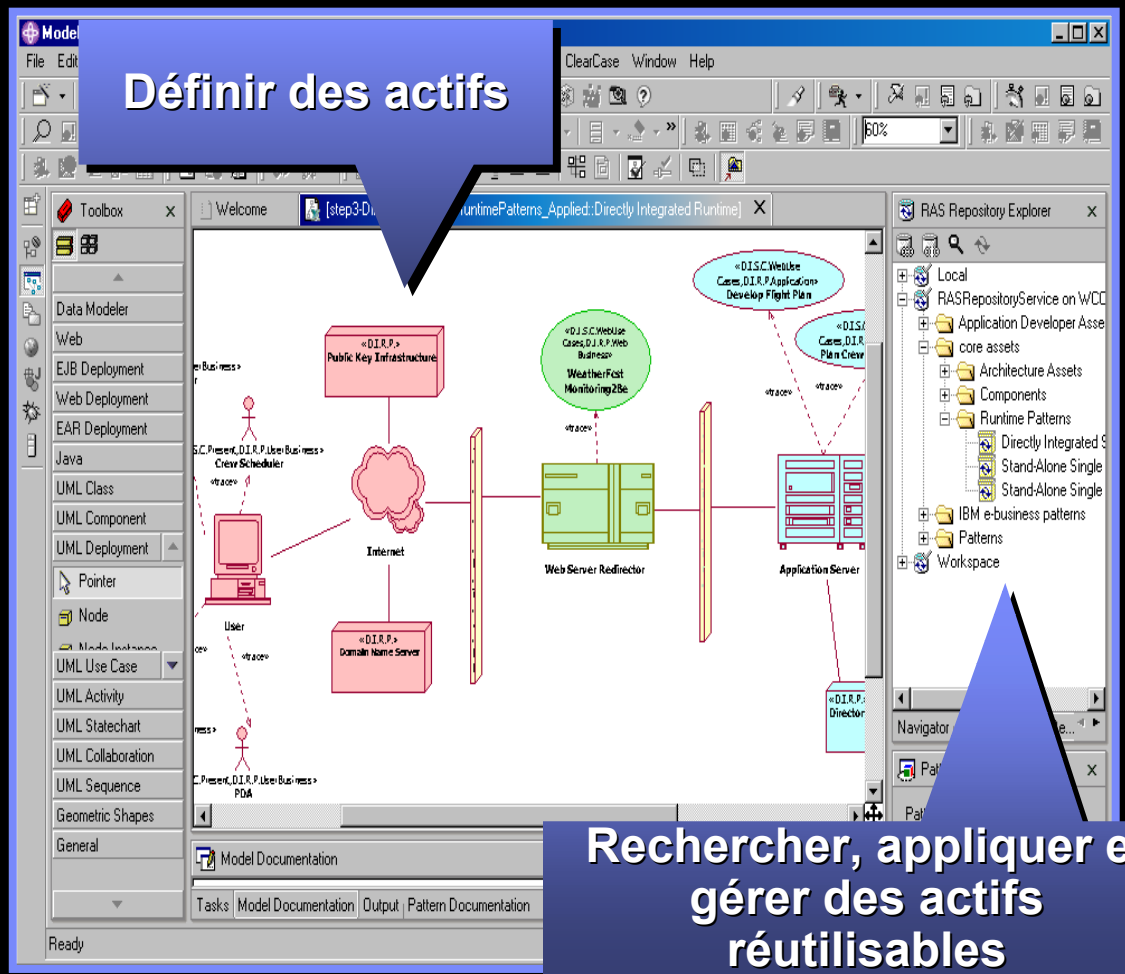
Extensions par les utilisateurs

Outils tiers

L'OMG adopte *Reusable Asset Specification* (RAS)

Du savoir-faire réutilisable au services des projets

- Résultats de l'initiative confiés à l'OMG.
- Standard pour décrire, archiver et réutiliser des actifs.
- Basé sur UML
- Consortium RAS intègre l'OMG
- Un actif : composant, Web Services, …..



UML 2.0: Un langage pour supporter MDD/MDA

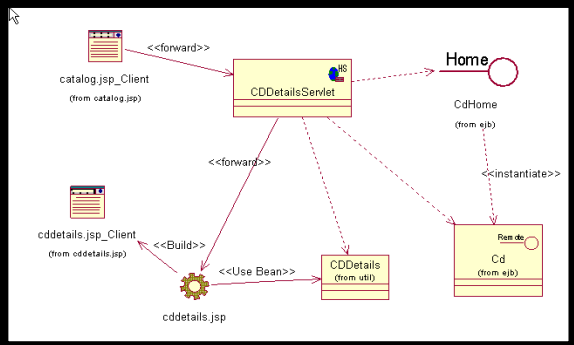
- La 1ère révision majeure depuis 1997
- Planning pour la standardisation d'UML 2.0 :
  - Adoption préliminaire: Juin 2003
  - En cours de finalisation
  - Prévision de sortie : Fin 2004
  - Bran Sélic, chairman du group UML 2.0
- Les composantes de UML 2.0
  - Infrastructure – *UML internals*
  - Superstructure – *User-level features*
  - OCL – *Object Constraint language*
  - *Diagram interchange standard*

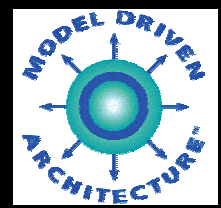
UML 2.0: Quoi de neuf ?

- Une sémantique plus précise
  - Afin de supporter MDD (génération de code, modèles exécutables, vérification formelle, test, etc.)
- Mieux adapté pour la modélisation de grands logiciels ou systèmes complexes
- De nouvelles constructions spécifiques pour modéliser l'architecture
- Mieux adapté à la modélisation du domaine
  - Processus métier
- Support amélioré pour les nouvelles technologies
  - Services Web
  - Développement à base de composants
  - Etc.
- Compatibilité ascendante

Agenda

- Développement piloté par le métier
- De l'intérêt de modéliser : Quels bénéfices ?
- Les différentes approches de modélisation
- MDA, MDD, MOF, UML 2.0 et toutes ces choses ….
- Les solutions IBM pour le développement à base de modèles
- Les bénéfices : retour clients et analyses de marché





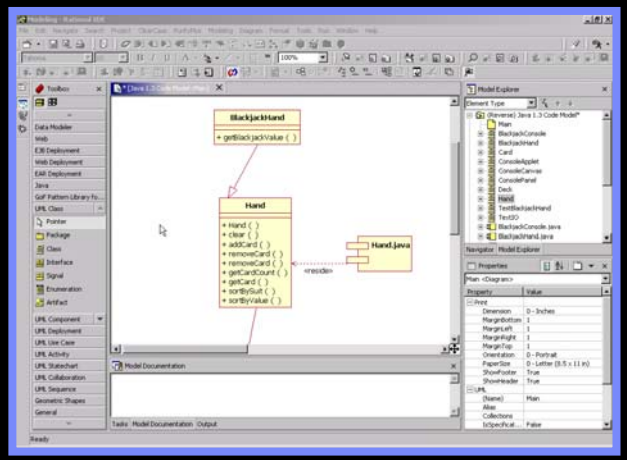
Les défis du développement logiciel

- La limitation des environnements de développement traditionnels
  - Manque d'automatisation, d'ouverture et d'intégration
- Développer du code de qualité est difficile
  - J2EE reste complexe
  - Reutiliser est difficile
  - Intégrer des applications est consommateur de temps
  - La mise au point du code reste souvent encore artisanale.
- Développer du logiciel reste toujours une activité complexe
  - Interopérabilité, distribution, plate-formes, systèmes d'exploitation
  - Mécanismes : Transactions, persistence, sécurité, ...
- **Conséquences:**
  - Echecs projets : livraison en retard, dysfonctionnement, ne répondant pas aux besoins des utilisateurs
  - Les échecs des projets de développement impacte le métier de l'entreprise

La modélisation simplifie les développements

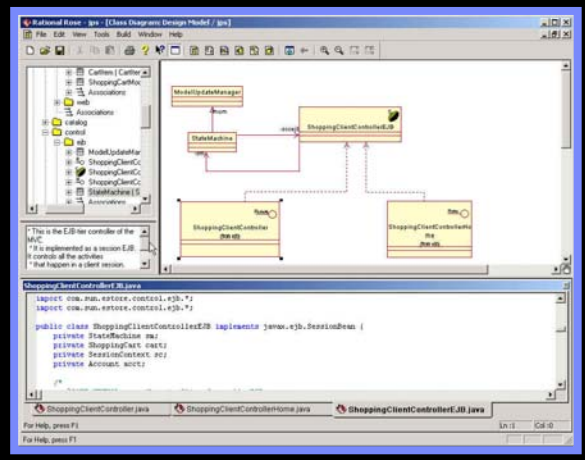
Développement à base de modèles visuels

IBM Rational Rose XDE Developer



IBM Rational XDE Developer

- ✓ Modélisation UML dans l'IDE
- ✓ Synchronisation modèle / code
- ✓ Visualisation de l'exécution
- ✓ Moteur de pattern



IBM Rational Rose

- ✓ Modélisation UML + IDE
- ✓ Numéro 1 depuis 7 ans
- ✓ Compatible avec XDE
- ✓ Support multi-language

La modélisation simplifie les développements

Résoudre les problèmes de développement

- Décupler les capacités de votre environnement
- Démarrer rapidement les projets
- Simplifier le développement J2EE
- Développer du code de qualité
- Faciliter le développement en équipe



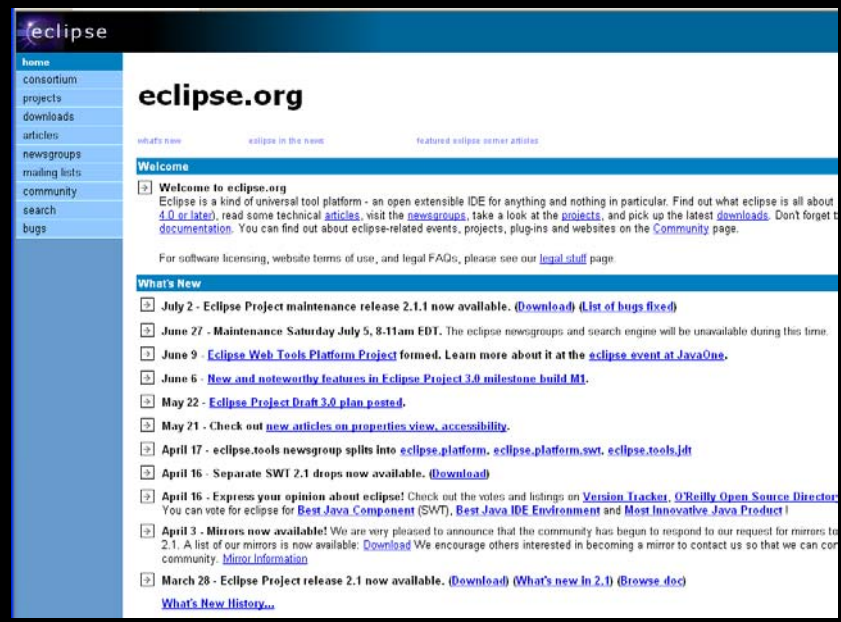

Résoudre les problèmes de développement

- Décupler les capacités de votre environnement
- Démarrer rapidement les projets
- Simplifier le développement J2EE
- Développer du code de qualité
- Faciliter le développement en équipe



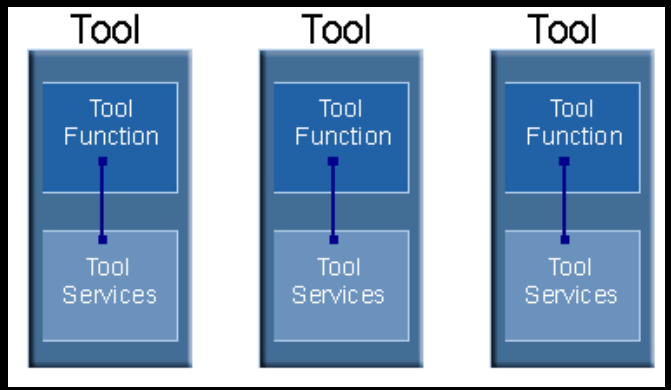

Le projet Eclipse

- **Qu'est ce ?**
  - Une plate-forme d'intégration
  - Un ensemble de composants réutilisables pour implémenter des outils et des environnements de gestion du cycle de vie du logiciel
- **IBM a contribué pour $40M en logiciel et R&D**
- **Croissance de la communauté**
  - +10 Millions de téléchargements
  - 25 sites miroirs dans 12 pays
  - Plus de 200 projets Open source
    - www.eclipse-workbench.com
    - eclipse-plugins.2y.net



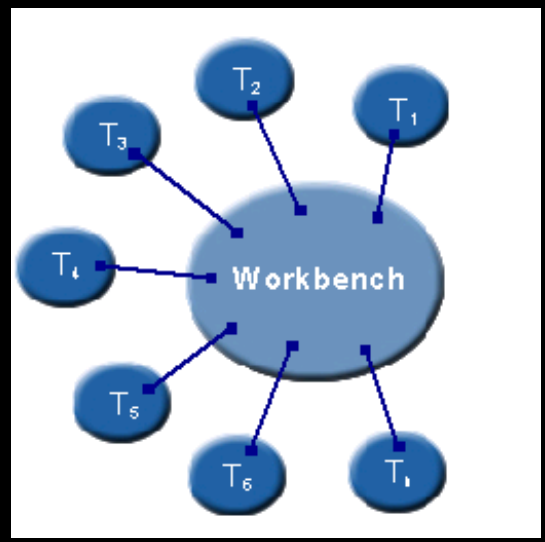
L'apport essentiel d'Eclipse

Un changement fondamental pour les outils de développement



Avant

- Intégration difficile,
- Différentes interfaces,
- Référentiels multiples,
- Gestion compliquée, …

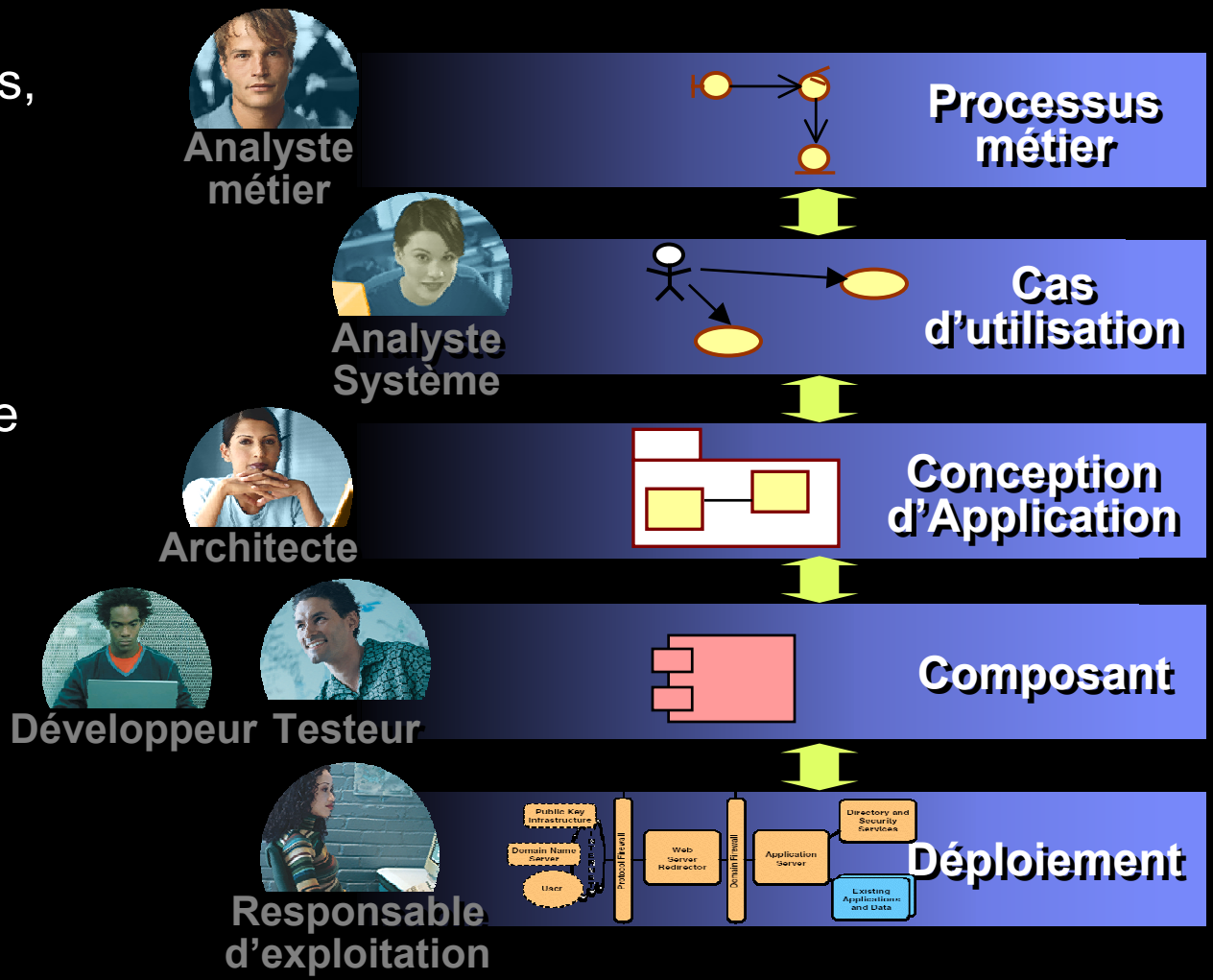


Maintenant

- Intégration plus facile
- Même expérience utilisateur
- Une seule gestion
- Un seul référentiel
- Évolution plus rapide

L'accès à l'information basé sur les rôles

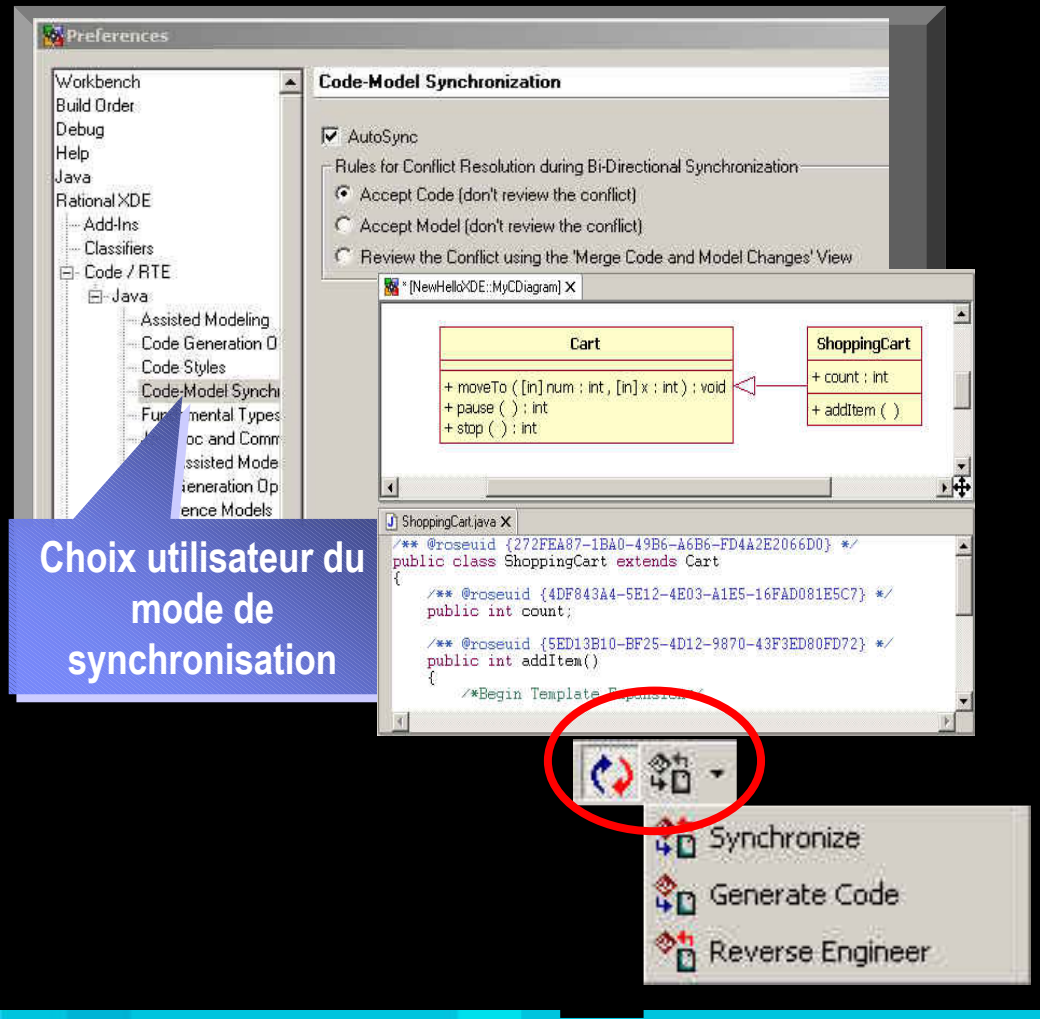
- Perspectives multiples, optimisées pour chaque rôle, sur des données communes
- Traçabilité complète sur tout le cycle de vie
- Maximise la productivité individuelle et de l'équipe
  - Eclipse Meta-model Framework (EMF)
  - Team API



Décupler les capacités de votre environnement

Synchronisation modèle/code : A vous de choisir !

- Liberté de faire les changements sur le modèle, le code ou les deux
- Synchronisation automatique
- Synchronisation manuelle
- Résolution des conflits
- Refactoring supporté



UN ENVIRONNEMENT INTEGRE

VISUALISATION DU CODE

Résoudre les problèmes de développement

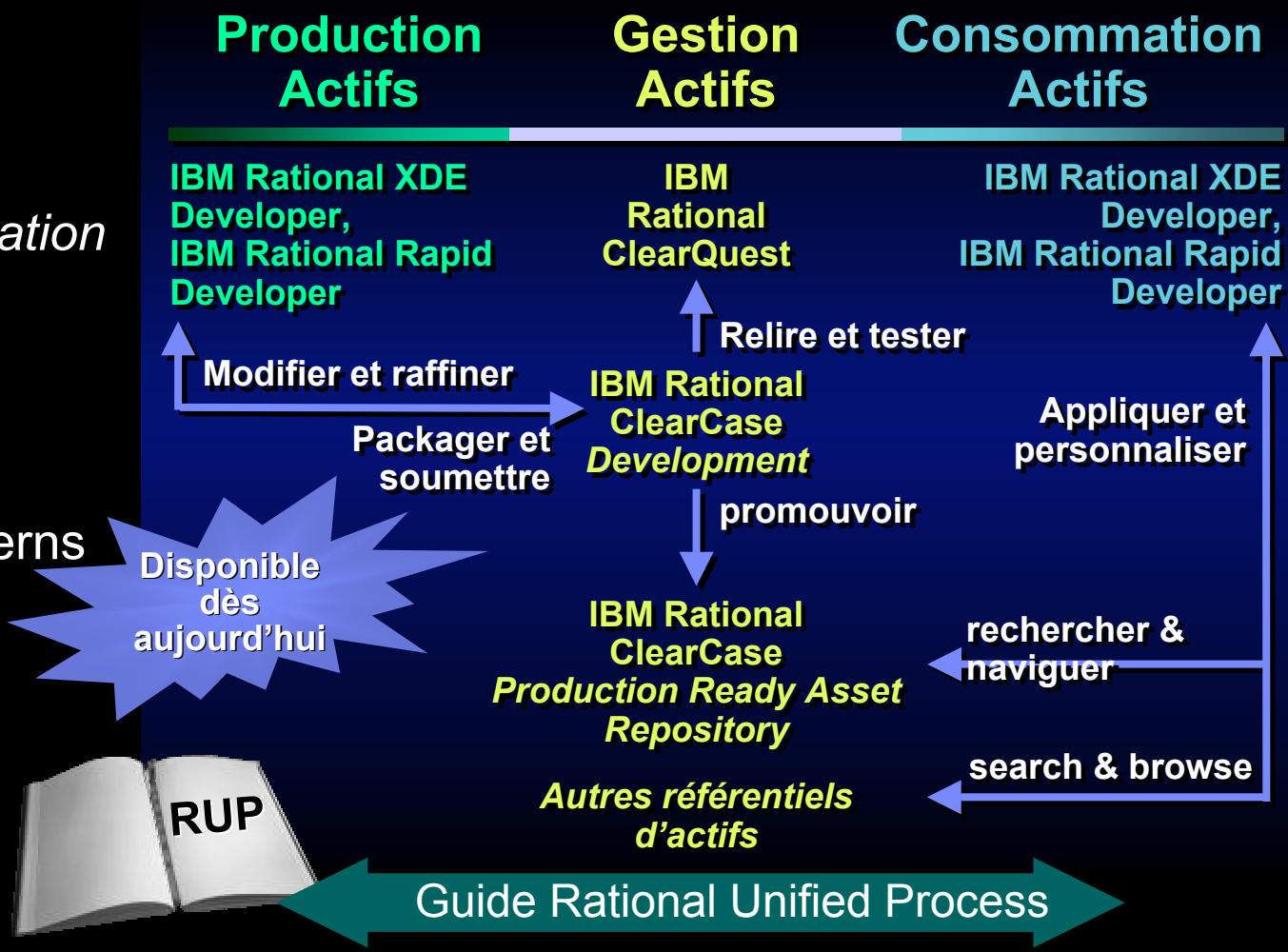
- Décupler les capacités de votre environnement
- Démarrer rapidement les projets
- Simplifier le développement J2EE
- Développer du code de qualité
- Faciliter le développement en équipe




Développement à base d'actifs

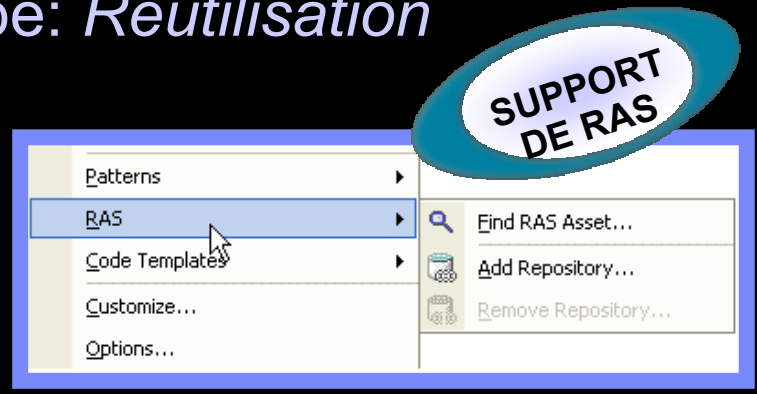
Du savoir-faire réutilisable au services des projets

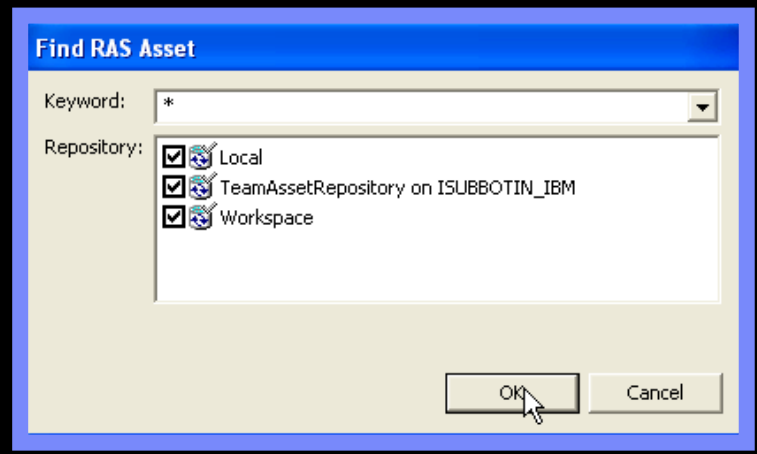
- Standard
  - MDA
  - *Reusable Asset Specification*
- Outillé
- Guidé
- Actifs
  - Business Patterns
  - Frameworks
  - Composants
  - *Templates*

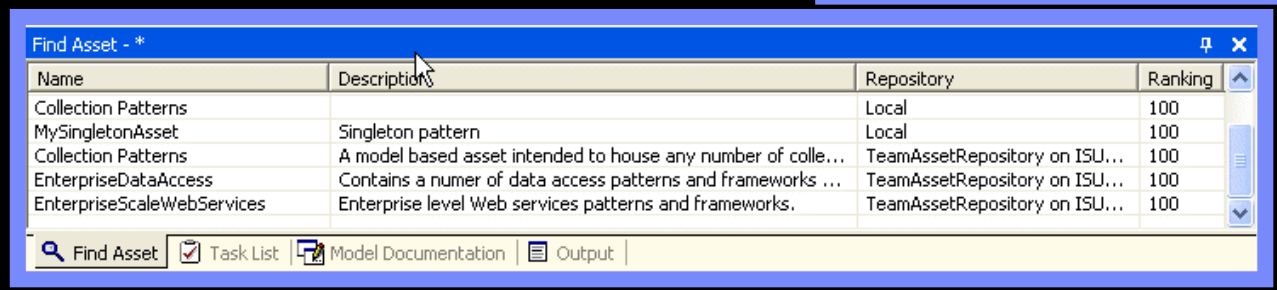


Améliorer le Développement en Équipe: *Réutilisation*

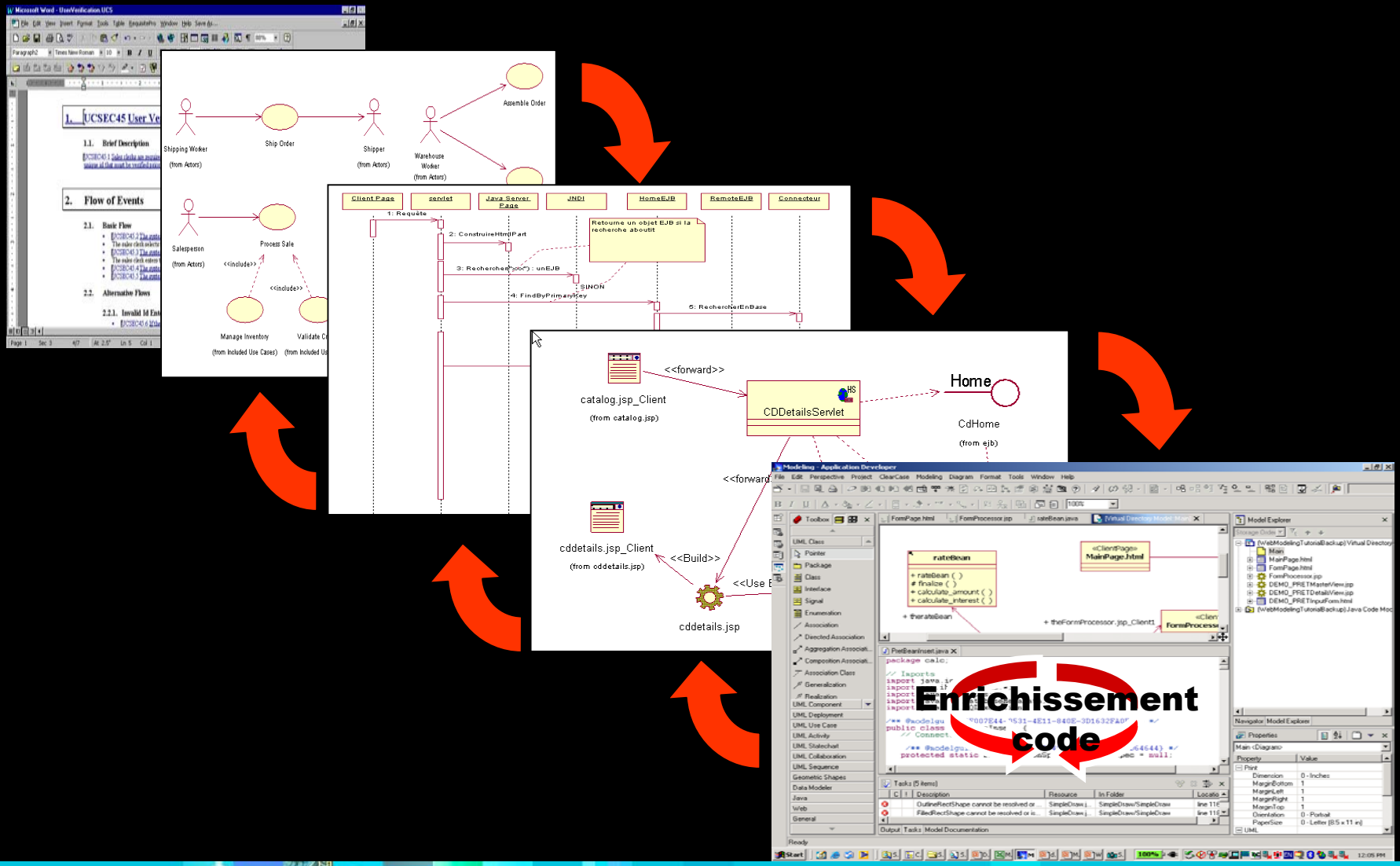
- Mettre en place la réutilisation
  - Référentiel d'actifs logiciel
    - Stockage sur site web / accès http
  - Référentiel local ou distant
    - Recherche par môts-clés
  - Actifs : Patterns, Code Templates
  - Basé sur Reusable Asset Specification (RAS)







Des cas d'utilisation aux composants

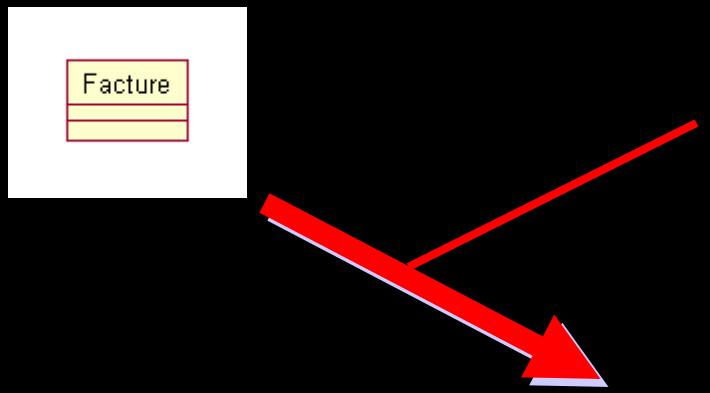


Qu'est-ce qu'un Pattern ?

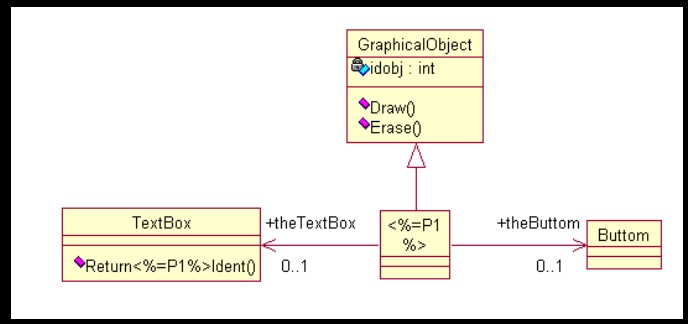
" A pattern provides a common solution to a problem in a specific context"
Grady Booch

" A design pattern describes the problem, a solution to the problem consisting of a general arrangement of objects and classes, when to apply the solution, and the consequences of applying the solution" Gamma et. al.

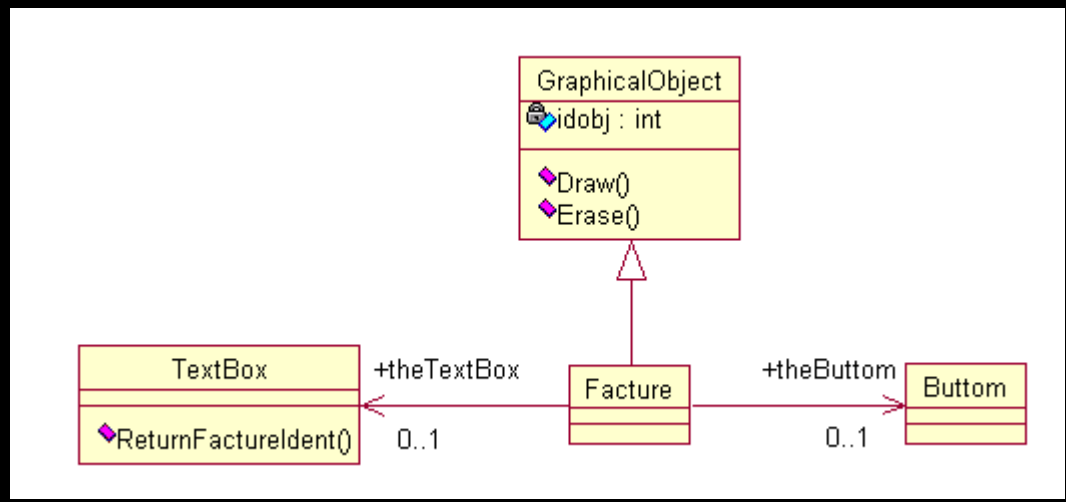
Obtenir une architecture évolutive et réutilisable



Pattern



Réutilisation de savoir-faire



APPLICATION DES PATTERNS

Résoudre les problèmes de développement

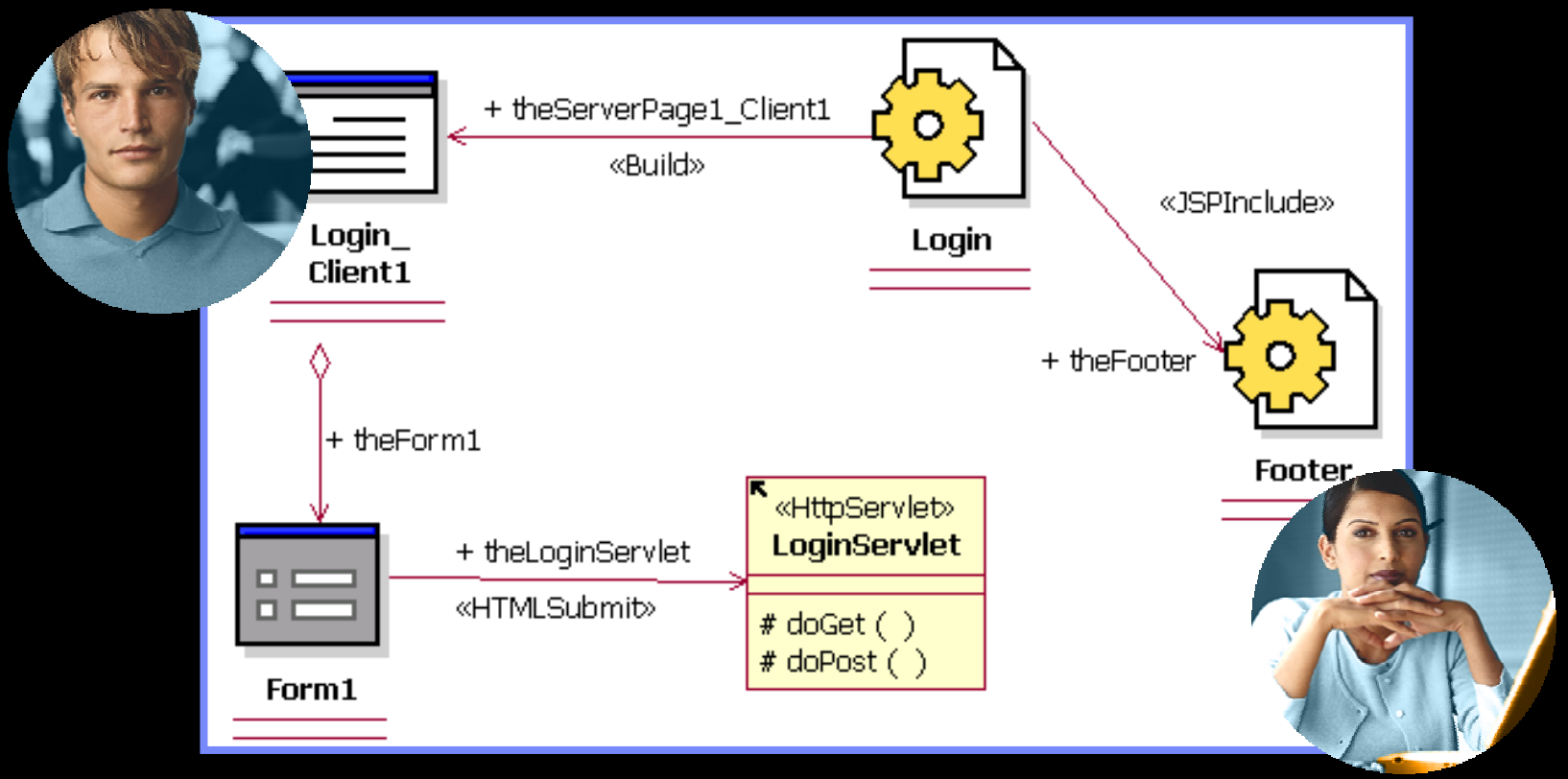
- Décupler les capacités de votre environnement
- Démarrer rapidement les projets
- Simplifier le développement J2EE
- Développer du code de qualité
- Faciliter le développement en équipe




Décupler les capacités de votre environnement

Conception J2EE

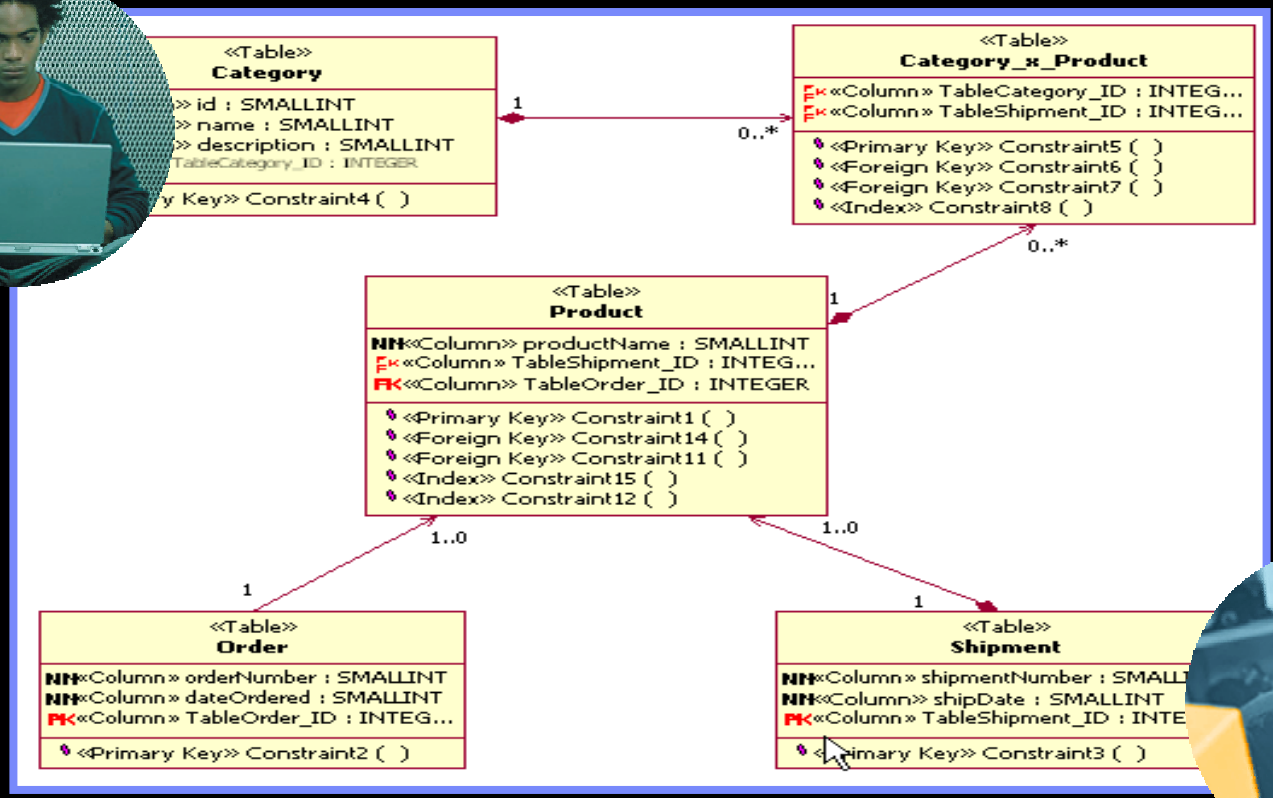
- Génération de code automatisée pour une architecture J2EE incluant les composants EJBs, servlets, JSPs, HTML, etc.



Décupler les capacités de votre environnement

Conception de base de données

- Concevoir et construire votre système avec un seul outil.



LA CONSTRUCTION D'UNE APPLICATION J2EE

Résoudre les problèmes de développement

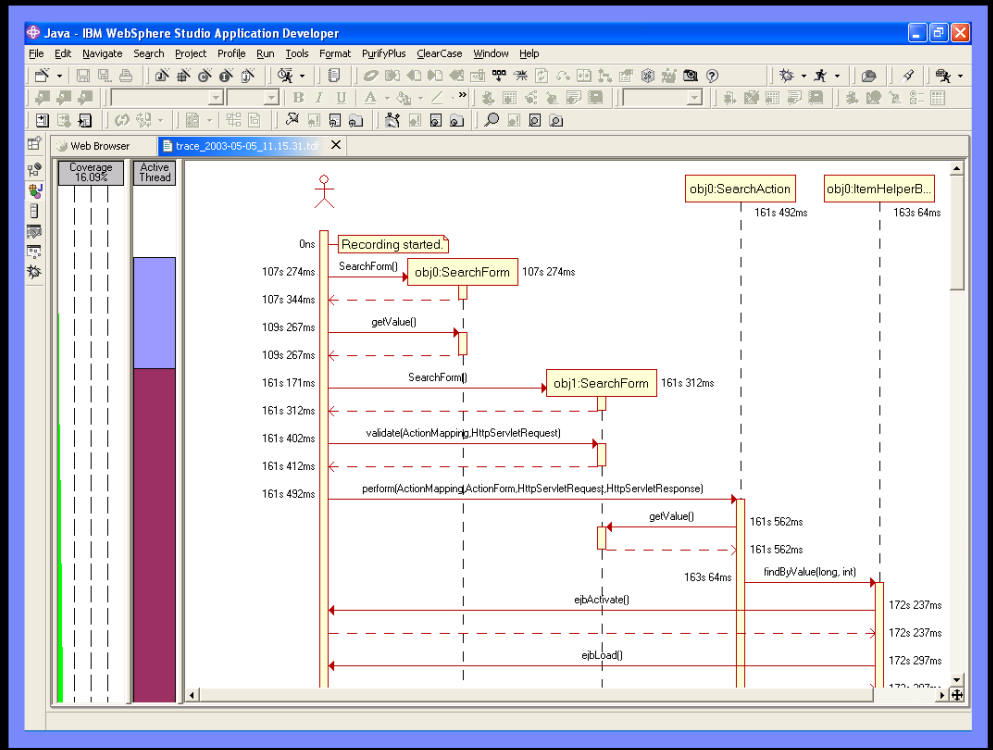
- Décupler les capacités de votre environnement
- Démarrer rapidement les projets
- Simplifier le développement J2EE
- Développer du code de qualité
- Faciliter le développement en équipe




Développer du code fiable

*Trace Visuelle **

- **Faciliter la réutilisation et la rétro-documentation**
  - Diagramme de séquence crée lors de l'exécution du code
- **Améliorer les capacités de mise au point du code**
  - Visualisation du flot de contrôle, des informations temporelles, de l'utilisation de la mémoire et du taux de couverture dans une unique vue.
- **Comprendre plus vite le code**
  - Enregistrer le comportement réel.
  - Aucune modélisation préalable n'est nécessaire.



** Disponible avec XDE Developer Plus Edition*

TRACE VISUELLE

Résoudre les problèmes de développement

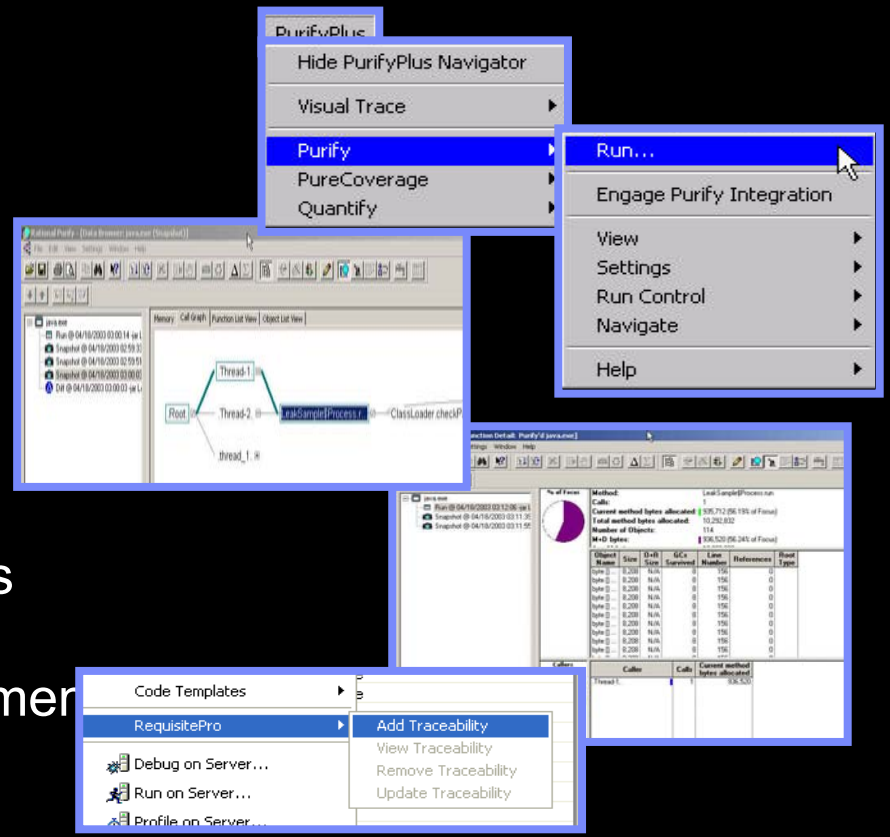
- Décupler les capacités de votre environnement
- Démarrer rapidement les projets
- Simplifier le développement J2EE
- Développer du code de qualité
- Faciliter le développement en équipe




Faciliter le développement en équipe

Intégrations

- IBM Rational PurifyPlus
  - Purify, PureCoverage, Quantify
  - Améliore la mise au point
- IBM Rational SoDA
  - Production automatique de documentation
- IBM Rational RequisitePro
  - Traçabilité exigences <-> modèles UML
  - Comprendre l'impact des changemer d'exigences sur le modèle

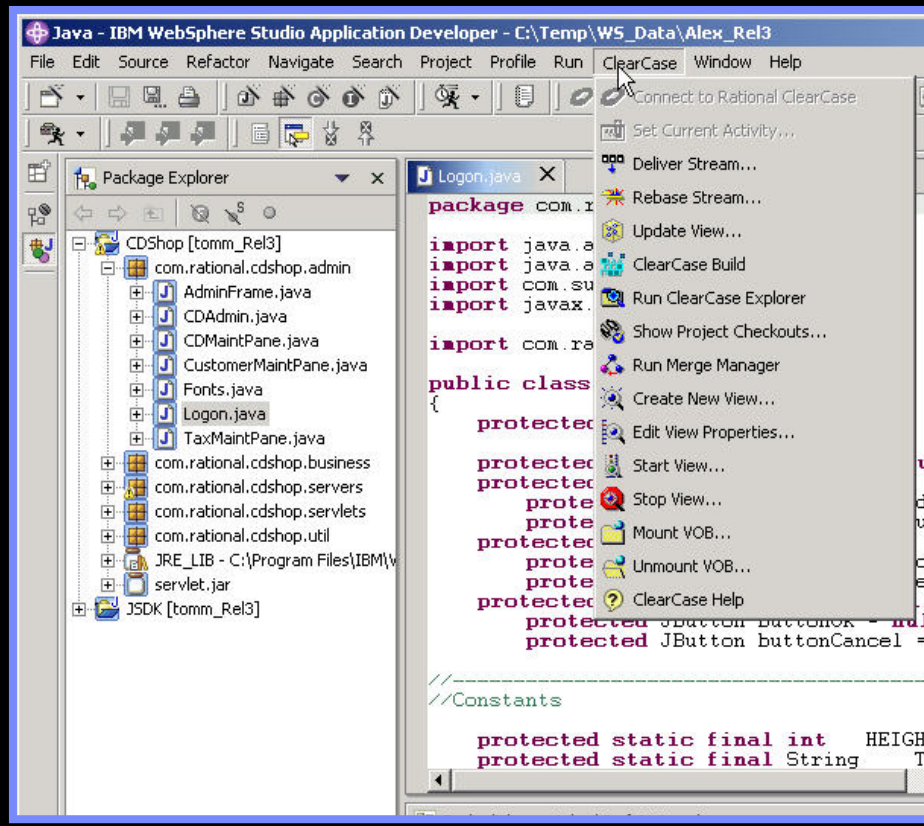


Note: XDE Developer Plus Edition seulement

Faciliter le développement en équipe

Gestion de configuration et des changements

- Support IBM Rational ClearCase
  - Intégration transparente
  - Cohérence garantie avec la fonction de *refactoring*
  - Aucune perte d'informations
  - Icônes visuelles
  - Comparaison / Fusion de modèles



TRAVAILLER EFFICACEMENT EN EQUIPE

IBM Rational Rose XDE : Bénéfices ?

- **La garantie du respect des standard**
  - Règles de programmation
  - UML, J2EE, XML, ….
- **Gagner du temps en développement**
  - La production automatisée de code
  - Réutilisation d'actifs de type modèles
- **Minimiser le risque d'erreur**
  - Une démarche pour modéliser une architecture flexible
  - Moins de stress pour les développeurs
  - Moins de détails d'implémentation à traiter
- **Réduire les coûts de maintenance**
  - Analyse d'impact
  - Traçabilité

La plate-forme de développement logiciel




Analyste | Architecte | Développeur | Testeur

Modélise, simule et pilote les processus métier

Modélise les applications et les données

Construit visuellement les programmes, et génère le code

Conçoit, crée, et exécute les tests

Mobile et Embarqué

DB2 Lotus Tivoli WebSphere

Chef de Projet

- Applique un processus commun
- Suit l'avancement du projet
- Gère les exigences
- Gère les changements et les composants
- Gère la qualité

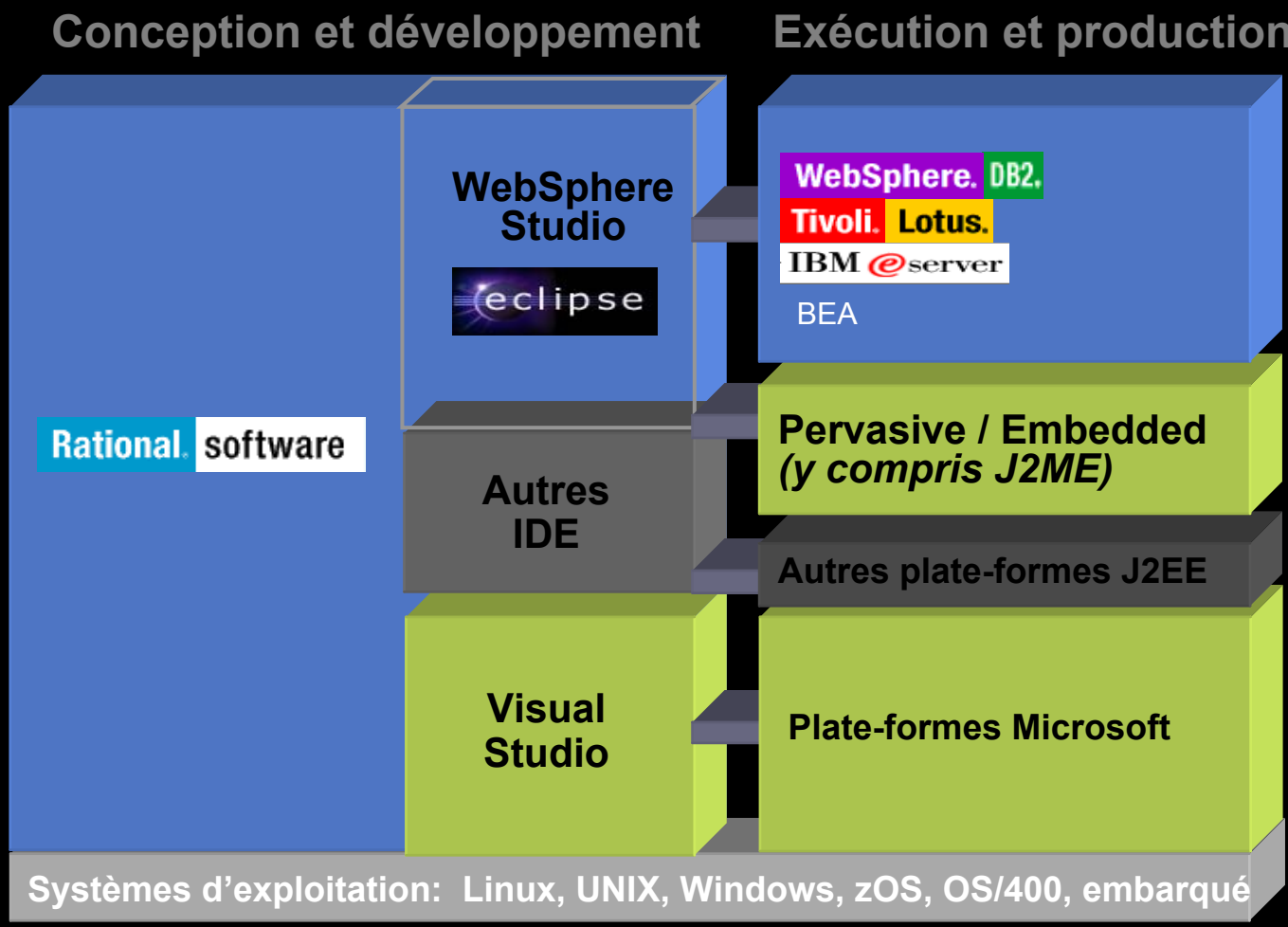
Serveurs IBM et autres

Partenaires

TEAM UNIFYING PLATFORM

Microsoft | *Standards Ouverts* | *Services IBM*

IBM Rational: Intégration dans un monde hétérogène



Eclipse permet une intégration plus forte


Analyste


Architecte


Développeur


Testeur

Extensions Utilisateur

ECLIPSE

- Interfaces basées sur les rôles
- Modèles communs d'intégration sur tout le cycle de vie
- Partagé cohérent des informations pour tous

3rd Party ISV Tools


Chef de Projet

Gestion de configuration logicielle

Processus et gestion de projet

TEAM UNIFYING PLATFORM

Exemple témoignage client (projet 1)

- Une compagnie mondiale du diagnostic médical
- Mission : logiciels d'analyses médicales automatisés pour les laboratoires
- Développement à base de modèles avec **Rational Rose**
- Utilisation combinée de tous les produits Rational
- Méthodologie : Rational Unified Process
- Couche métier
  - 250 objets métiers dont 200 persistants générés
- Application automatique de patterns de conception
  - 1200 composants générés par transformation des objets métier
- Architecture
  - 60% des composants réutilisables dans un nouveau projet (code + modèle)
- Développement
  - Fonction de sécurité : 75% temps gagné / produit précédent
  - Tests de non régression, 50% de gain de temps

Exemple témoignage client (projet 2)

« … avant, tout le monde était plutôt multi-casquette, aujourd'hui, les rôles liés aux nouvelles technologies (fonctionnel, architecte, analyste objet, DBA,) ont pu être mieux définis grâce à la matière que nous avons trouvée dans RUP »

« … le temps moyen de correction d'une anomalie a été améliorée considérablement, pour une même évolution fonctionnelle, nous sommes passé de 5 jours à ½ journée entre l'ancien et le nouveau système grâce à l'ensemble des outils de la suite Rational »

Chef de Projet – Éditeur de logiciel Pharmaceutique


Merci