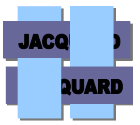
ICAR'03

École d'été sur les Intergiciels et sur la Construction d'Applications Réparties

Au cœur d'OpenCCM

Philippe Merle
Project Jacquard (INRIA et LIFL)
http://www.lifl.fr/~merle



Plan

- Le projet OpenCCM
- Les briques généralistes
- La chaîne de production des composants
- L'outillage de conditionnement et d'assemblage
- L'infrastructure répartie de déploiement
- L'outillage d'administration
- Le plate-forme d'exécution
- Conclusion

Le projet OpenCCM

Les objectifs du projet OpenCCM

- **Devenir la 1ière implantation de référence du CCM**
  - Couverture complète du modèle de composants CORBA 3.0
  - Validation, contribution et évolution de la spécification OMG
- **Fournir une plate-forme CCM ouverte**
  - Logiciel libre et indépendance vis-à-vis ORBs
  - Flexible, extensible et adaptable pour faire du CCM++
- **Offrir un terrain d'expérimentation en R&D**
  - Projets INRIA Sardes, Paris, Jacquard, …
  - Cadena Kansas University, …
- **Logiciel libre LGPL - consortium ObjectWeb**
  - *http://openccm.objectweb.org/*

Historique du projet OpenCCM

- 1998 - 2000 : suivi et étude de la proposition CCM
  - 1iers exposés et cours : [Car99], [Ecoop00], etc.
- 2000 - 2001 : expérimentation et prototypage
  - 1ier prototype du compilateur OMG IDL 3.0 et d'un « micro-runtime »
  - OpenCCM 0.1 (01/01) : 1ière version sur site LIFL
  - OpenCCM 0.2 (03/01) : bugs et améliorations
- 2002 - 20xx : projet logiciel libre hébergé par ObjectWeb
  - V.0.4 (07/02) : 1ière version sur site ObjectWeb
  - V.0.5 (12/02) : compilateur CIDL/PSDL, générateur XMI UML, déploiement XML
  - V.0.6 (03/03) : nouvelle chaîne compilation / génération, runtime CIDL
  - V.0.7 (07/03) : service PSS, outil C&A, infrastructure déploiement, browser, …
  - V.0.8 (10/03) : service trader, stabilisation et améliorations diverses
  - *OpenCCM 1.0 (juin 2004) : couverture complète du CCM*

L'état d'avancement du projet OpenCCM

- Implantation partielle du CCM
  - Chaîne de compilation OMG IDL, PSDL et CIDL
  - Outillage de conditionnement et d'assemblage
  - Infrastructure de déploiement
  - Support d'exécution (composants Session et service PSS)
  - Outillage d'administration
  - Diverses démonstrations (~ 7)
- Non encore couvert
  - Support à l'exécution, i.e. conteneurs
    - Générateur CORBA Component Descriptor
    - Conteneurs pour composants Service, Process et Entity
    - Majeure partie des interfaces des conteneurs
    - Intégration des services dans les conteneurs
      - Persistance, transaction, sécurité, notification

Le logiciel libre OpenCCM

- **Entièrement écrit en Java**
  - SUN JDK 1.2.x, 1.3.x et 1.4.x
  - Portabilité, maintenance et support
  - Linux, Solaris, Windows et Windows CE pour PDA
- **Construit au dessus de CORBA 2.4 (et +)**
  - ORBacus 4.x
  - OpenORB 1.2.1, 1.3.0, 1.3.1, 1.4.0
  - Borland Enterprise Server 5.0.2 et 5.2
- **Utilise d'autres logiciels libres**
  - ObjectWeb : Apollon et Monolog
  - Enhydra : Zeus
  - Apache : Ant, Log4j, Velocity et Xerces
  - Sun Microsystems : JavaCC
  - LIFL : JIDLscript
- **Prototype d'un plug-in pour Eclipse**

Les ressources Internet du projet OpenCCM

- **Site Web : http://openccm.objectweb.org**
  - Information et documentation
- **Forge : http://forge.objectweb.org/projects/openccm**
  - Fichiers releases, CVS, suivi des bugs et des tâches
- **Liste public : openccm@objectweb.org**
  - Aide aux utilisateurs, rapports de bugs et de tâches
  - ~ 130 inscrits
- **Liste team : openccm-team@objectweb.org**
  - Animation équipe de développement (surtout LIFL)
- **Commits CVS : openccm-commits@objectweb.org**
  - Suivi de tous les commits CVS

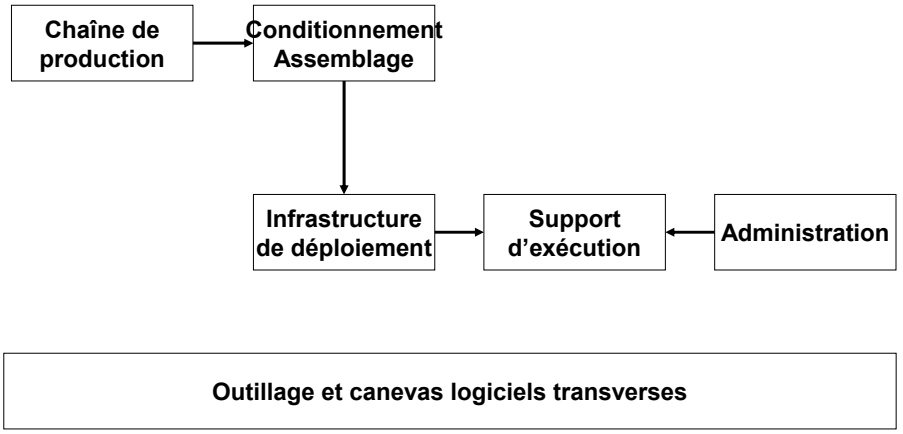
La communauté OpenCCM

- **Nombreux utilisateurs / évaluateurs**
  - De toutes origines, i.e. universités et industriels
  - Beaucoup de rapports de bugs, de demandes d'informations, etc.
  - Projet Cadena – Kansas University
  - THALES – Perco/CCM
  - Lucent – projet IST COACH – test interactif et observation de composants
  - Intracom – projet IST COACH – canevas de gestion d'éléments de réseaux
- **Peu de contributeurs**
  - Principalement équipe au LIFL
  - Tran Huynh - THALES - générateur XMI
  - Mike Gratsas - Bank of Lithuania - portage BES
  - Lucent - IST COACH - observation et test interactif de composants

Les sponsors du projet OpenCCM

- **Projet INRIA Jacquard**
  - Création en juin 2003
- **Contrat ITEA OSMOSE**
  - Juillet 2003 – juin 2005
- **Contrat IST COACH**
  - Avril 2002 – mars 2004
- **ACI GRID RMI**
  - 2002 - 2003
- ***Contrat RNTL IMPACT***
  - *01/2002 – 06/2003 (fini)*
- **Contrat RNRT COMPiTV**
  - 2002 – 2003
- ***Contrat RNRT CESURE***
  - *1999 –2001 (fini)*

La plate-forme OpenCCM



Les briques généralistes

Les briques généralistes

- Le canevas `cmdline`
  - Analyse de lignes de commande
- L'application pré processeur
  - Intégration de pré processeurs C/C++
- L'utilisation Enhydra Zeus
  - Génération de code Java pour XML
- L'outil Launcher
  - Démarreur d'applications Java
- Le canevas Browser
  - Navigateur graphique personnalisable
- L'outil Apollon
  - Générateur de composants graphiques d'édition XML

Le canevas `cmdline`

- De nombreux programmes dans OpenCCM
  - Compilateurs, générateurs, éditeurs XML, démons, IHM, etc. nécessitant
    - Uniformisation de l'utilisation des programmes
      - Lignes de commande, options, aide, etc.
    - Automatisation de l'analyse des arguments de la ligne de commande
      - Contrôle, stockage des arguments, etc.
- Canevas `cmdline`
  - Des interfaces Java
    - `Application`, `Usage`, `CommandLine`, diverses `Option`
  - Une implantation de base
    - Options obligatoires, facultatives, flags, 1 ou n arguments, etc.
    - Analyse des arguments selon les options d'une ligne de commande
    - Affichage de l'aide en ligne des applications
  - Package `org.objectweb.util.cmdline`

Une illustration du canevas `cmdline`

```
CommandLine cmd = new DefaultCommandLine("preprocessor",
   "file", "Preprocess a file.");
// Ajouter des options générales.
cmd.addOption(new DefaultOptionHelp(…));
cmd.addOption(new DefaultOptionVersion(…));
cmd.addOption(new DefaultOptionFlag(…));
// Ajouter des options spécifiques du pré processeur.
cmd.addOption(new OptionCPP(…));
cmd.addOption(new OptionP(…));
cmd.addOption(new OptionDNAME(…));
cmd.addOption(new OptionUNAME(…));
cmd.addOption(new OptionIDIR(…));

String[] args = cmd.parseArguments(main_args);
```


L'application pré processeur

- **Nombreux compilateurs dans OpenCCM**
  - OMG IDL - OMG PSDL - OMG CIDL
  - Devant utilisés un pré processeur C/C++
- **Application pré processeur**
  - Uniformisation accès à des pré processeurs C/C++
  - Wrapper pour exécuter pré processeurs externes (`/usr/bin/cpp`)
  - 1 pré processeur écrit en Java
    - Amélioration de celui fourni par ObjectWeb Jonathan
  - Construit sur le canevas `cmdline`
    - Des options spécifiques *-I path*, *-Dname*, *-Dname=def*, etc.
  - Package `org.objectweb.util.cpp`

L'utilisation d'Enhydra Zeus

- **Nombreuses DTD XML dans OpenCCM**
  - Quatre DTD XML définies dans le CCM
  - Quelques DTD spécifiques (launcher, browser)
- **Besoin de représentations mémoire typées**
  - Binding Java = classes Java représentant éléments XML
  - ~ arbres XML typés Vs arbres génériques DOM
- **Enhydra / ObjectWeb Zeus**
  - Générateur de binding Java
  - A la compilation : DTD XML → Zeus → classes Java
  - A l'exécution : fichiers XML ←→ objets Java ← code applicatif
  - http://zeus.enhydra.org
- **Utilisation systématique de Zeus dans OpenCCM**
  - Génération systématique du code OpenCCM relatif à XML

Les motivations pour l'outil *Launcher*

- **De nombreux programmes dans OpenCCM**
  - Compilateurs, générateurs, éditeurs XML, démons, IHM, etc.
- **Démarrage nécessite la configuration d'informations techniques pour lancer une JVM**

```
java -Djava.compiler=NONE -DMyProperty=Value
      -Xbootclasspath=MyJavaRuntime.jar
      -cp MyArchive.jar
      MyPackage.MyClass MyArguments
```

  - Le nom de la classe principale
  - La liste des chemins d'accès aux classes et archives Jar
  - Les propriétés Java et arguments spécifiques aux applications

Les motivations pour l'outil *Launcher*

- **Non simple pour l'utilisateur final**
  - → encapsuler le lancement JVM par des scripts shell
  - Différents pour Unix, Windows, Windows CE, etc. !
  - Les petits équipements PDA n'ont pas toujours un interpréteur shell !
- **Seulement accès à des fichiers locaux (ou NFS)**
  - Comment référencer une archive Java stockée sur un serveur Web ?
- **Difficile de surcharger les classes du runtime JVM (archive rt.jar)**
  - Nécessaire pour remplacer JDK ORB par un « vrai » ORB
- **Une seule fonction principale**
  - Lancer plusieurs démons dans la même JVM
  - Economiser le nombre de JVM lancée

L'outil *Launcher*

- **Description des applications Java via une notation XML dédiée**
  - Le nom de la classe principale
  - Les URL des archives Java à utiliser
  - Les propriétés Java à définir
  - Les arguments par défaut de la ligne de commande
- **Modularité et réutilisation de descriptions XML launcher**
  - Importation de descripteurs XML Launcher
  - Référencement des éléments importés

MonApplication.xml

```
<launcher>
  <include url="file:Middleware.xml">
  <run id="main"
       mainclassname="MonPackage.MaClasse"
       classpath="MonClassPath"
       properties="MesProprietes"
       arguments="MesArguments"/>
  <classpath id="MonClassPath">
    <path ref="DefaultMiddlewareClassPath"/>
    <path url="http://www/MonApplication.jar"/>
  </classpath>
  <properties id="MesProprietes">
    <property name="MaPropriete" value="…"/>
    <property ref="DefaultMiddlewareProperties"/>
  </properties>
  <arguments id="MesArguments">
    <argument ref="DefaultMiddlewareArguments"/>
    <argument value="Mes Arguments par Défaut"/>
  </arguments>
</launcher>
```

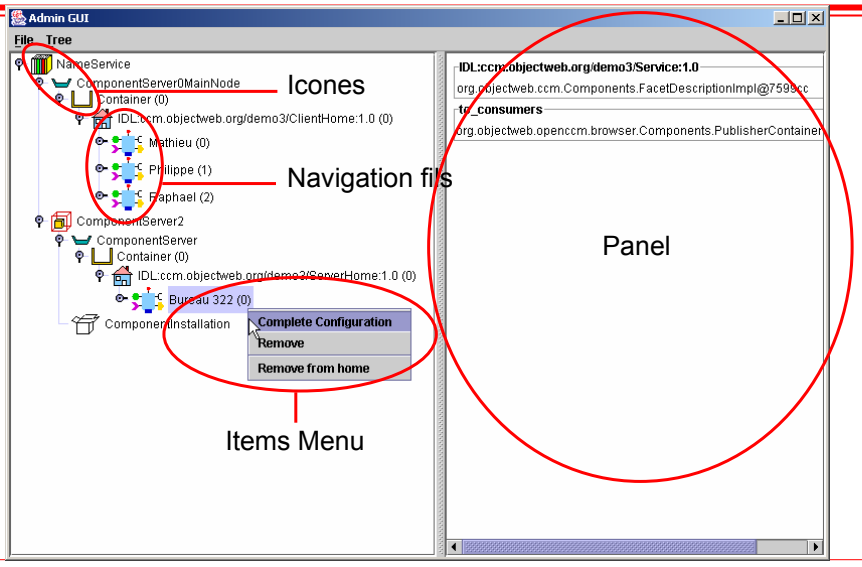

L'outil *Launcher*

- **Démarrage des applications Java**
  - `Launcher.bat MonApplication.xml`
  - Chargement des descripteurs via binding Zeus
  - Interprétation des objets du binding Zeus
  - Lancement d'une (ou plusieurs) classe(s) principale(s) dans
    - Le même espace mémoire
    - Des threads différents
    - Des chargeurs de classes isolés (URLClassLoader étendu)
    - Recherche dans archives avant recherche dans runtime Java
- **Conditionnement**
  - 1 script par OS, archive Java ~ 80Ko, + parser XML SAX
  - Package `org.objectweb.util.launcher`

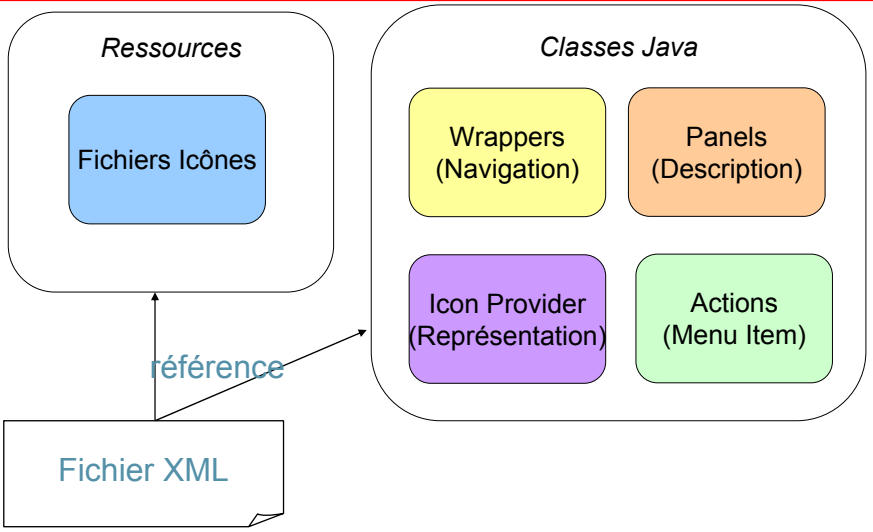
Le canevas *Browser*

- **Nombreuses IHM dans OpenCCM**
  - E.g. outils de conditionnement et d'assemblage, navigateur OpenCCM, etc.
  - Volonté d'uniformisation des IHM
  - Pouvoir assembler IHMs pour former de nouveaux outils
- **Canevas *Browser***
  - Navigateur graphique personnalisable par plug-ins
  - Plug-ins décrits en XML et programmés en Java
  - Interfaces simples entre Navigateur ←→ Plug-ins
  - Package `org.objectweb.util.browser`
- **Personnalisation pour Fractal et OpenCCM**

Que configurer dans le canevas *Browser* ?



La structure d'un plug-in pour le canevas *Browser*



Exemple de configuration le canevas *Browser*

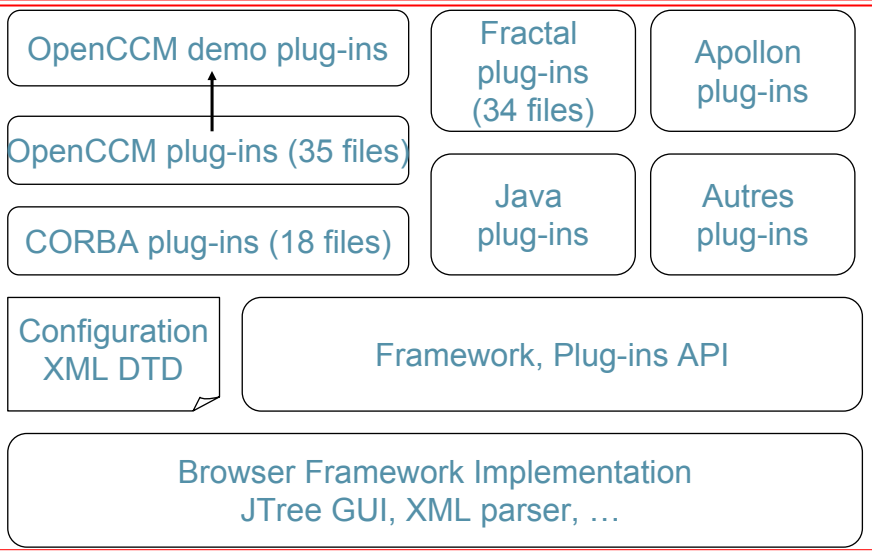
```
<browser>
 ...
  <node java-class="org.omg.CosNaming.NamingContext">
    <wrapper java-class="YourContextJavaClass" />
    <icon kind="icon-file" value="file:YourIconFile"/>
    <panel java-class="YourPanelJavaClass" />
    <menu>
      <item label="Bind an object"
            java-class="YourActionJavaClass" />
      <item label="Unbind" java-class="YourJavaClass"
            tree-child-visible="true" />
    </menu>
  </node>
...
 </browser>
```

- **Description XML, orientée type et supportant l'héritage**

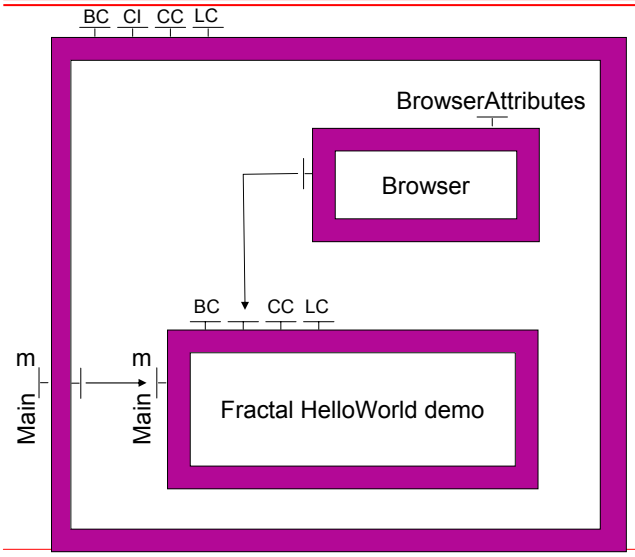
Exemple de code plug-in du canevas *Browser*

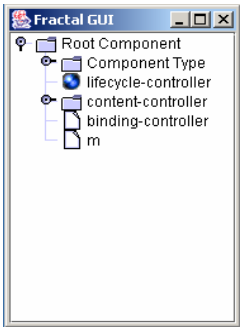
```
public class ConfigurationCompleteAction
  implements MenuItem
{
   public int getStatus(TreeView t)
   { return ITEM_VISIBLE; }

   public void actionPerformed(TreeView t) throws Exception
   {
     CCMObject object = (CCMObject)t.getCurrentValue();
     object.configuration_complete();
   }
}
```


Un vision architecturale du canevas *Browser*



La personnalisation Fractal du canevas *Browser*





Conclusion sur le canevas *Browser*

- **Indépendant des technologies**
  - Possibilité de développer des plug-ins pour toute technologie Java
- **Non intrusif**
  - Aucun changement dans les technologies encapsulées
- **Hautement configurable**
  - Description plutôt que programmation
- **Composable**
  - Création de nouveaux navigateurs par assemblage de plug-ins
- **Perspectives**
  - Bientôt disponible comme un module indépendant du CVS OpenCCM
  - Intégration d'autres langages pour développer les plug-ins
    - IDLscript, Python, composants Kilim, composants Fractal, etc.
  - Bibliothèque de plug-ins : Web services, Grid, JDBC, JNDI, etc.

L'outil Apollon

- **Nombreux éditeurs XML pour OpenCCM**
  - Quatre DTD XML définies dans le CCM
  - Quelques DTD spécifiques (launcher, browser)
- **Outil Apollon**
  - Générateur d'éditeurs XML graphiques personnalisables
  - Construit au dessus de Enhydra Zeus
  - Produit des composants Swing manipulant objets Zeus
  - Configuration IHM via le canevas *Browser*
  - Url : http://forge.objectweb.org/projects/apollon

Apollon : d'une XML DTD à son IHM

- **XML partout => besoin d'éditeurs XML personnalisables**

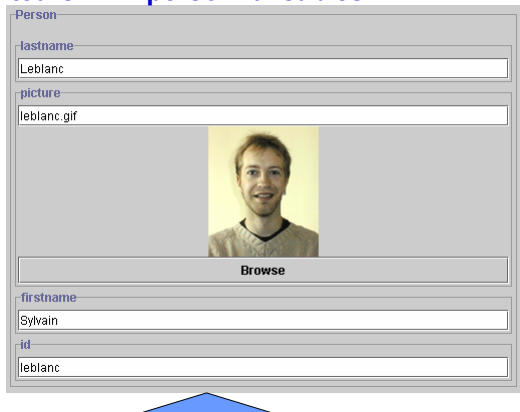
Une DTD XML …

```
<!ELEMENT person
      (address
      | phone)*>
<!ATTLIST person
      id ID #REQUIRED
      firstname CDATA #IMPLIED
      lastname CDATA #IMPLIED
      picture CDATA #IMPLIED
>
```

Zeus

Objets Java

Person
AddressList
PhoneList
id lastname
firstname picture

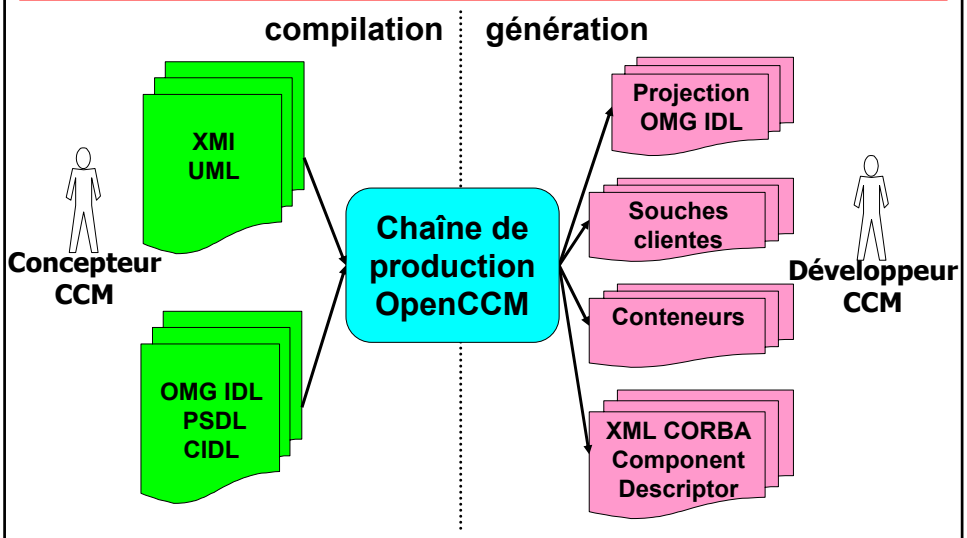


Apollon

model

La chaîne de production des composants

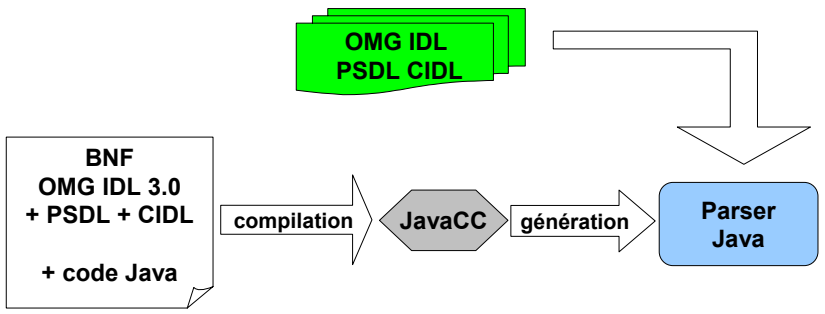
La chaîne de production de composants CCM



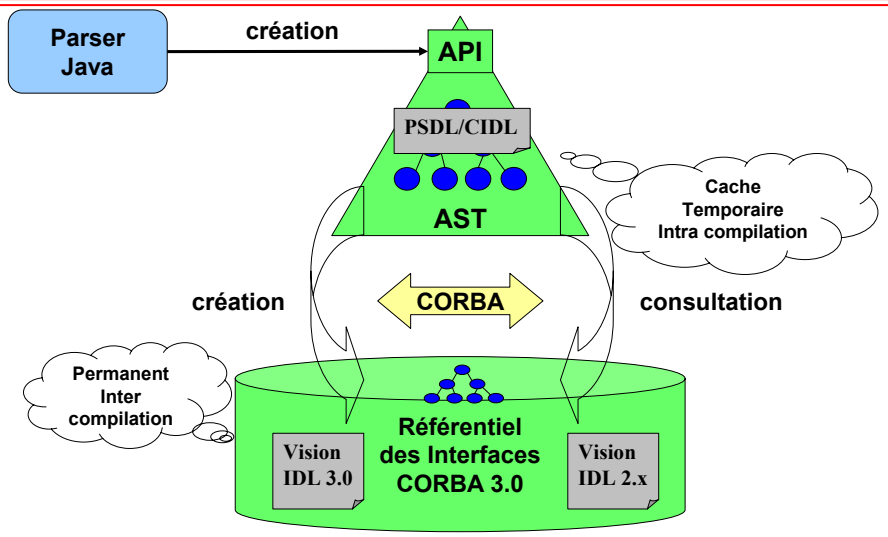
Le chaîne de production OpenCCM

- **Compilateurs**
  - CORBA 3.0 Interface Definition Language (OMG IDL)
  - OMG Persistent State Definition Language (OMG PSS)
  - OMG Component Implementation Definition Language (OMG CIDL)
  - XMI UML CCM
- **Cœur = contrôle sémantique**
  - Arbre de Syntaxe Abstraite (AST)
  - Référentiel des Interfaces conforme CORBA 3.0
- **Générateurs**
  - Projection OMG IDL 2.x cliente et serveur
  - Conteneurs Java et squelettes Java pour CIDL
  - Templates Java pour implantation des composants
  - Pretty-printers OMG IDL 3.0, PSDL et CIDL
  - XMI UML CCM (contribution Thalès)

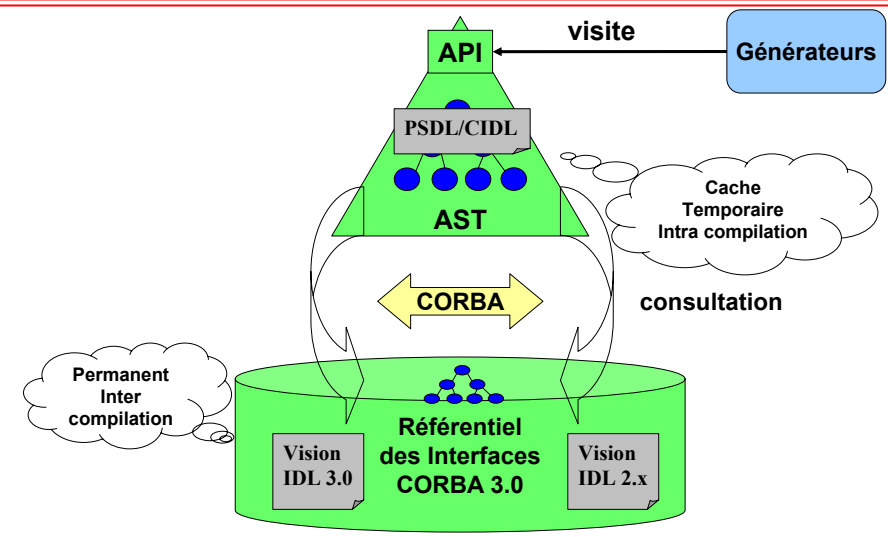
Le compilateur OMG IDL - PSDL - CIDL



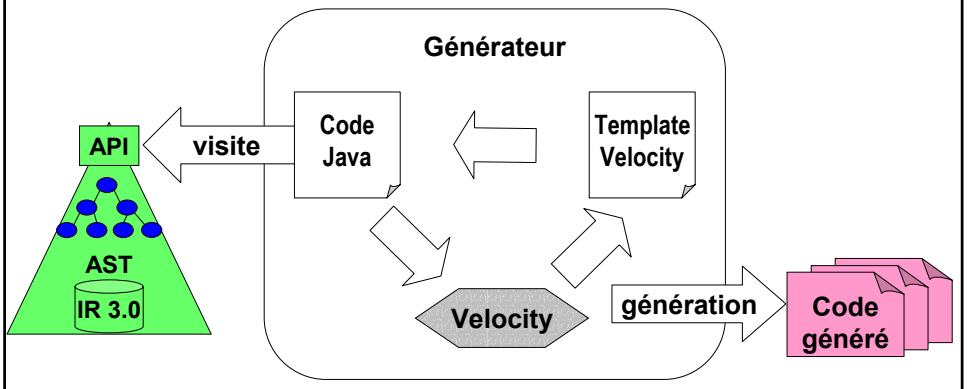
L'Arbre de Syntaxe Abstraite et le Référentiel des Interfaces



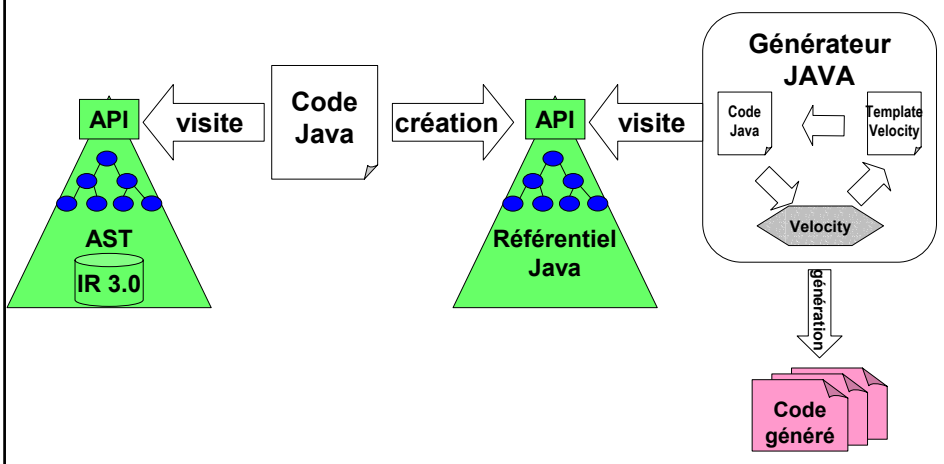
L'Arbre de Syntaxe Abstraite et le Référentiel des Interfaces



La structure des générateurs OMG IDL – PSDL - CIDL

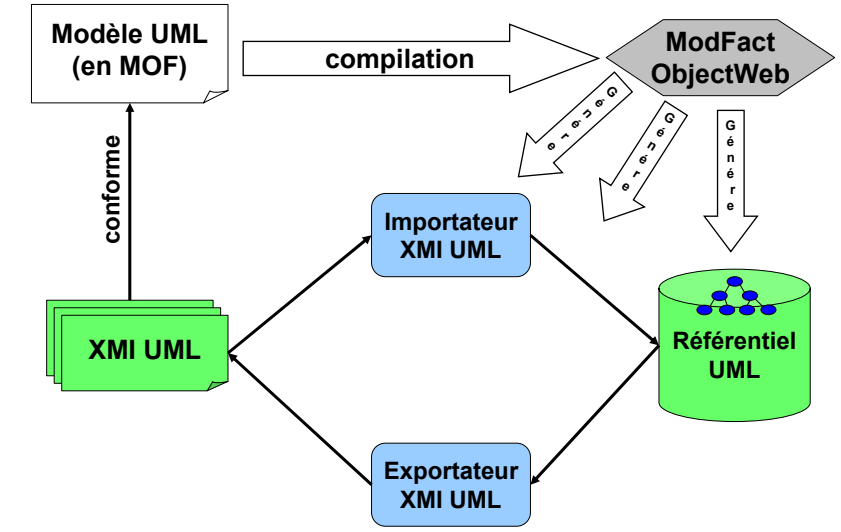


La structure des générateurs de code Java

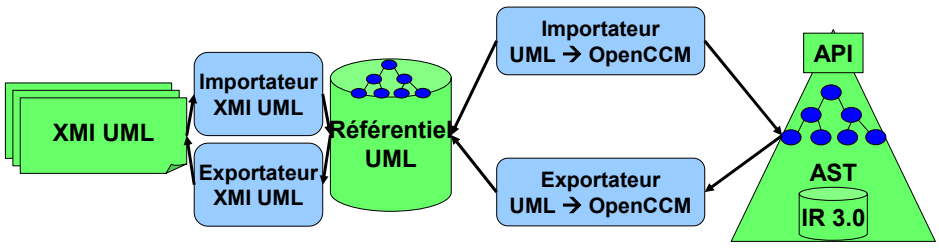


Appliqué à CIDL → Java ; PSDL → Java ; …

La compilation XMI UML

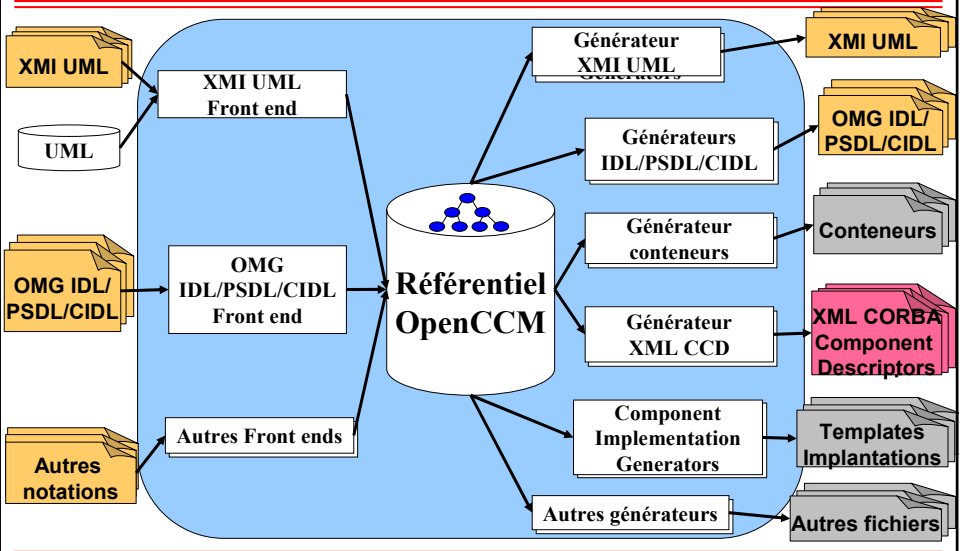


La compilation XMI UML



- **Application de l'approche MDA**
  - Importateur et exportateur = transformations MDA
  - Aujourd'hui codées manuellement
  - Demain exprimées en MOF QVT → génération automatique ou interprétation

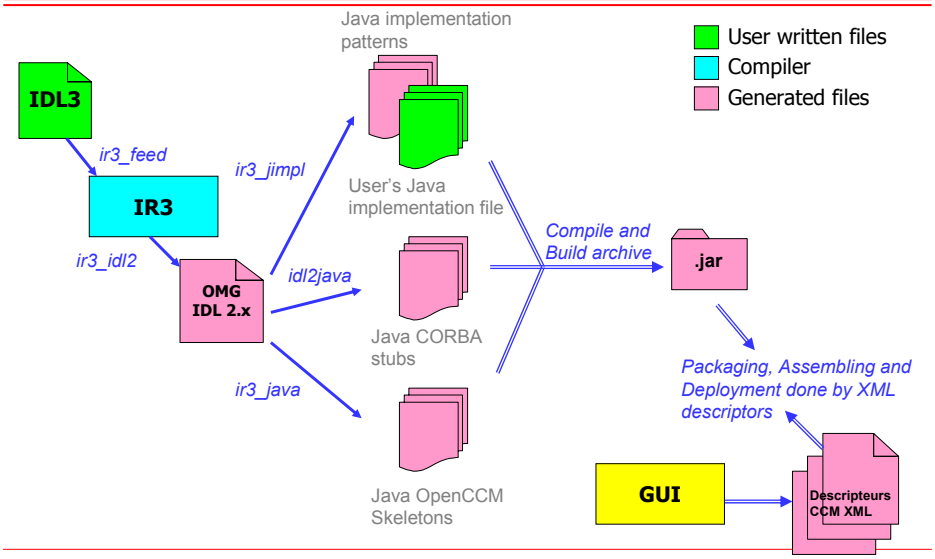
Une architecture modulaire et extensible

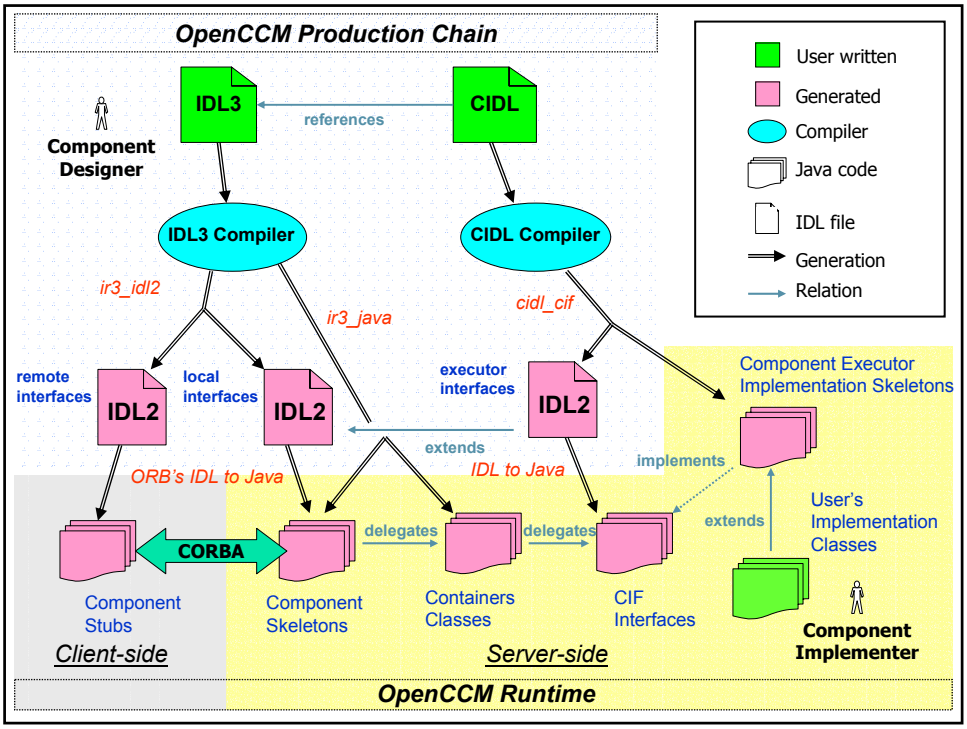


Les commandes utilisateur de la chaîne de production

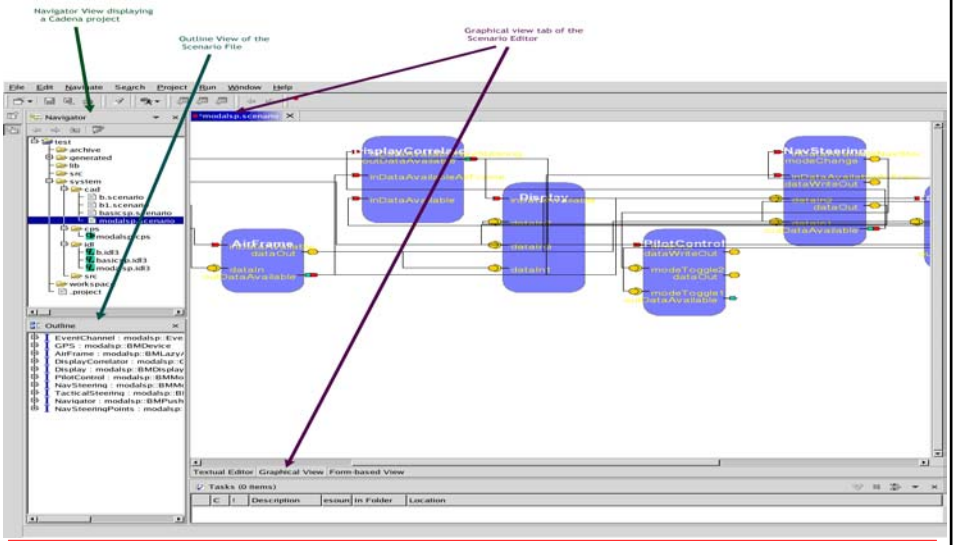
- **Principales**
  - `ir3_start` : démarrer le référentiel des interfaces
  - `ir3_stop` : arrêter le référentiel des interfaces
  - `ir3_feed` : charger un fichier OMG IDL dans le référentiel
  - `ir3_idl2` : générer projection OMG IDL depuis référentiel
  - `ir3_java` : générer conteneurs Java depuis référentiel
  - `psdl_java` : générer code Java pour définitions OMG PSDL
  - `cidl_cif` : générer code Java pour définitions OMG CIDL
- **Utilitaires**
  - `idl3_check` : compiler un fichier OMG IDL
  - `ir3_destroy` : détruire définition dans référentiel
  - `ir3_jimpl` : générer templates d'implantation Java
  - `ir3_idl3` : générer OMG IDL 3.0 depuis référentiel
  - `ir3_xmi` : générer XMI UML depuis référentiel
  - `cidl` : pretty-printer OMG CIDL
  - `psdl` : pretty-printer OMG PSDL

Le processus de production





L'atelier Cadena
Plug-in Eclipse + OpenCCM

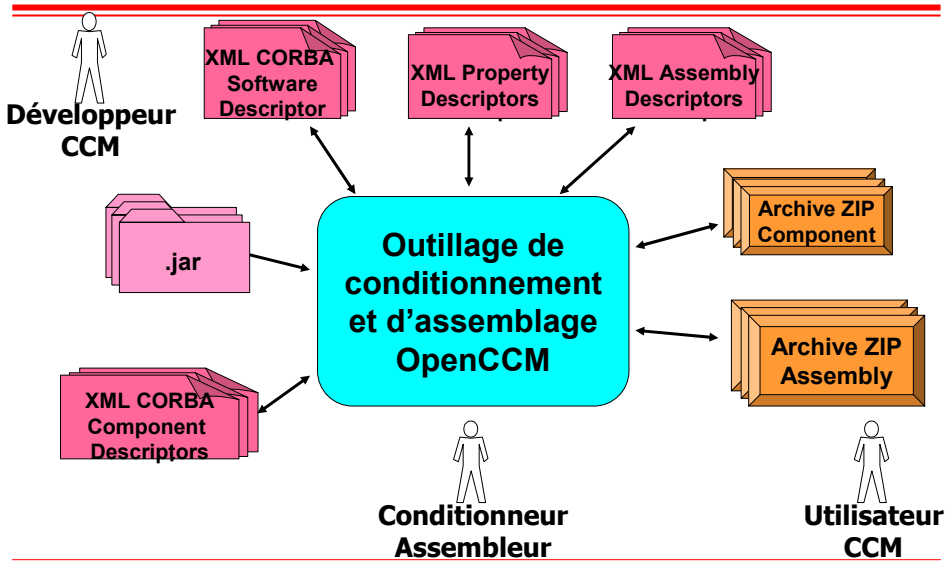


Perspectives pour la chaîne de production

- **Réutilisée dans THALES Perco/CCM et atelier Cadena**
- **Avant OpenCCM 1.0**
  - Générateur templates Java depuis CIDL
  - Générateur XML CORBA Component Descriptors (.ccd)
  - Stabilisation et documentation des API
  - Résolution des bugs dans l'implantation
  - Fourniture d'un unique compilateur / générateurs intégré
- **Après OpenCCM 1.0**
  - Appliquer systématiquement démarche MDA pour générer la chaîne automatiquement
  - Modèles spécifiés en MOF → génération référentiels et XMI DTD
  - Transformations entre modèles spécifiés en MOF QVT → compilation et/ou interprétation des transformations

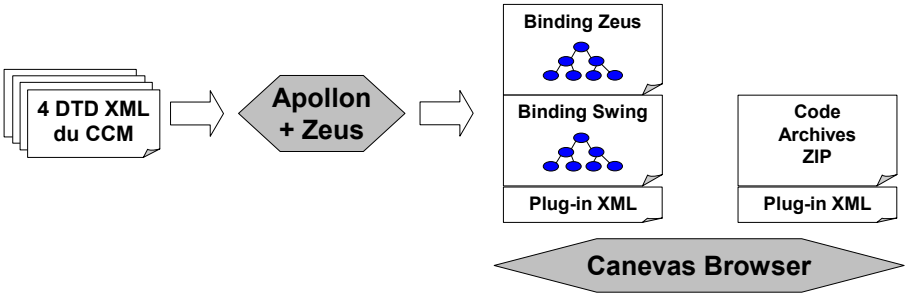
L'outillage de conditionnement et d'assemblage

L'outillage de conditionnement et d'assemblage



L'outillage de conditionnement et d'assemblage

- **Une interface graphique Swing**
  - Composants graphiques d'édition XML générés avec Apollon à partir des 4 DTD XML du CCM
  - Manipulation d'archives ZIP
  - Plug-ins pour le canevas Browser



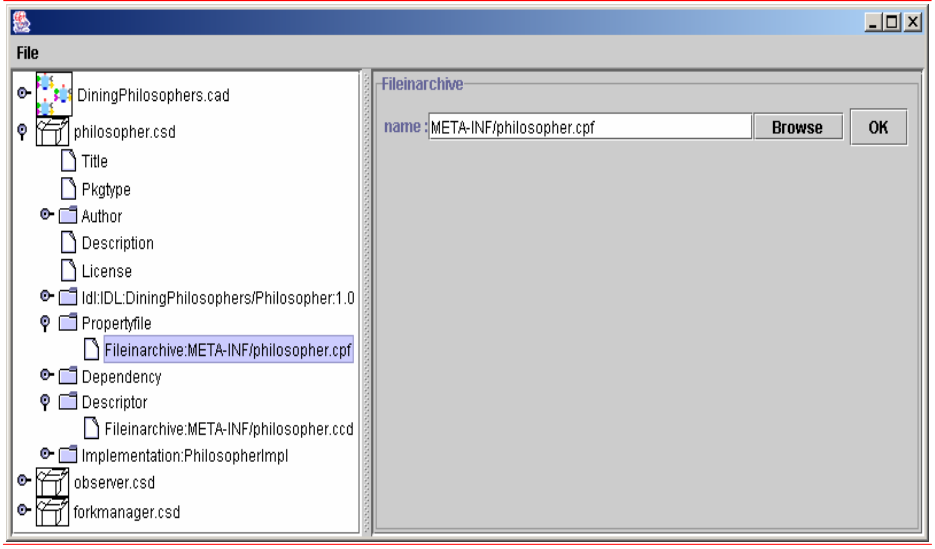
L'outillage de conditionnement et d'assemblage

- **Un canevas pour construire des outils de conditionnement et d'assemblage CCM**
  - Navigateur adaptable via canevas Browser
  - Composants d'édition XML adaptables via générateur Apollon
  - Intégrable dans d'autres applications Swing

➔ **Utilisateurs peuvent construire leur propre outillage de conditionnement et d'assemblage**

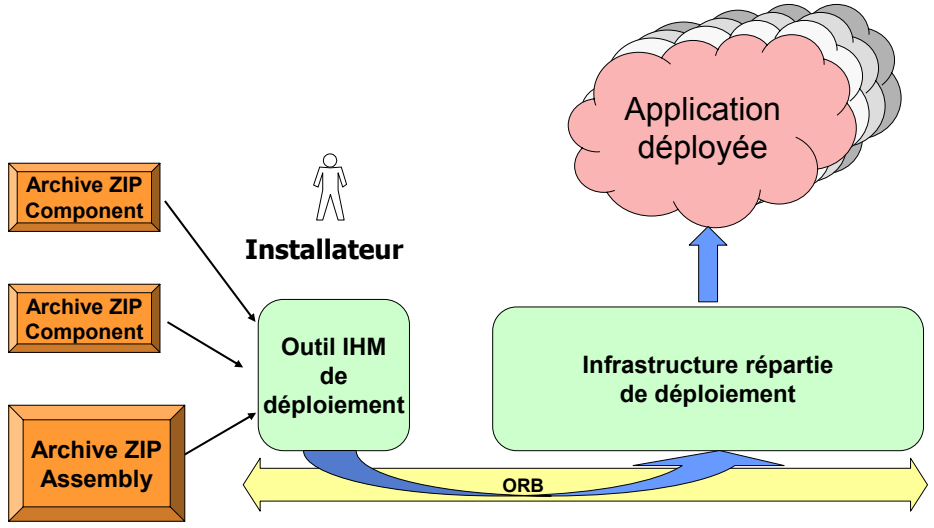
- **OpenCCM fournit une instance prête à l'emploi**

Une instance de l'outillage de conditionnement et d'assemblage



L'infrastructure répartie de déploiement

L'infrastructure répartie de déploiement



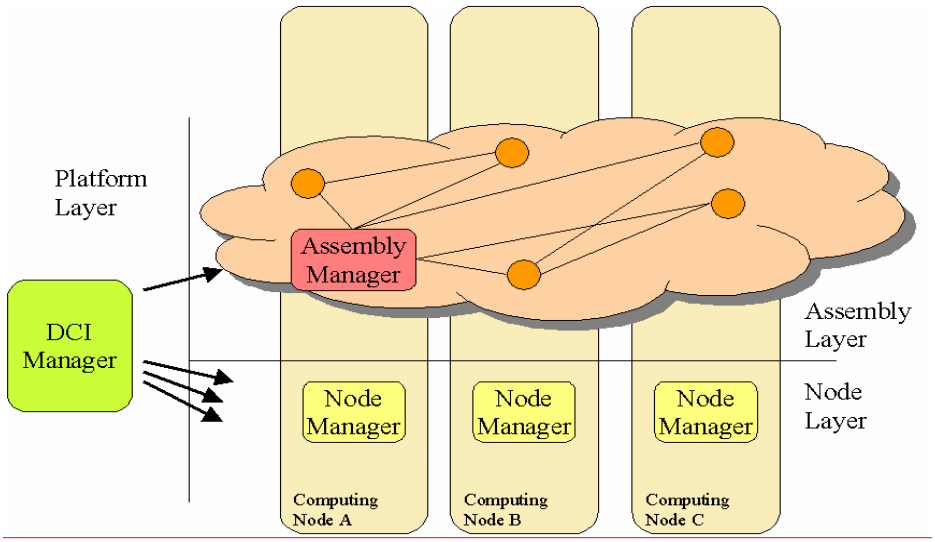
L'infrastructure répartie de déploiement

- **Outil IHM de déploiement**
  - Actuellement une console texte
  - Plus tard une IHM permettant d'assigner les composants aux sites
- **L'infrastructure répartie de déploiement**
  - Définie dans le projet IST COACH – Distributed Computing Infrastructure (DCI)
  - Réalisée en composants CCM dont les facettes sont les interfaces de déploiement définies dans la spécification CCM
  - Hébergée dans des conteneurs CCM ➔ déploiement avec propriétés non fonctionnelles comme persistance, transactions, sécurité, etc.
- **Déploiement transactionnel intégré dans OpenCCM**
  - Si panne durant déploiement alors retour à l'état avant déploiement

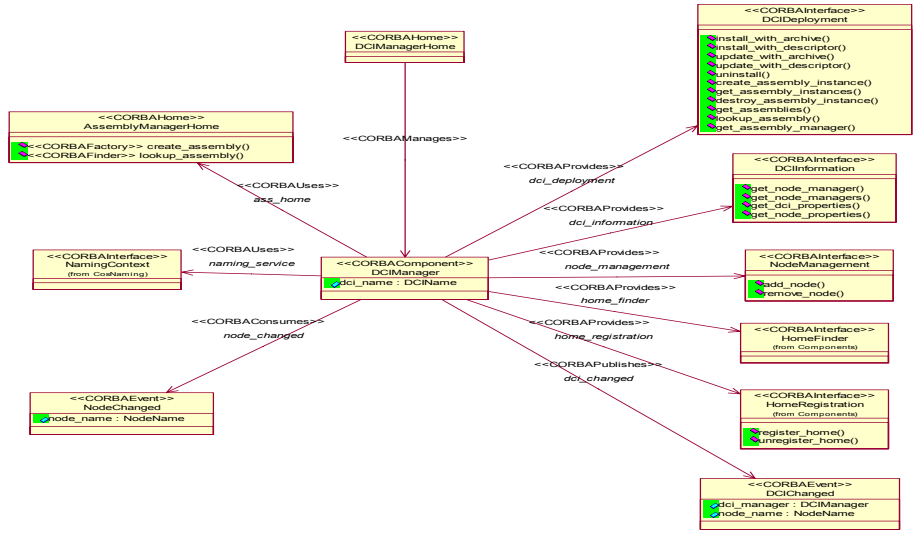
Les composants de l'infrastructure répartie de déploiement

- `DCIManager` = un domaine de déploiement
  - Facettes `AssemblyFactory` et `HomeFinder`
  - Autres ports
- `NodeManager` = une machine virtuelle
  - Facettes `ComponentInstallation` et `ServerActivator`
  - Autres ports
- `ComponentServerManager` = un serveur de composants
  - Facette `ComponentServer`
  - Autres ports
- `ContainerManager` = un conteneur de composants
  - Facette `Container`
  - Autres ports
- `AssemblyManager` = un assemblage déployé
  - Facette `Assembly`
  - Autres ports
- Autres ports pour administration et supervision

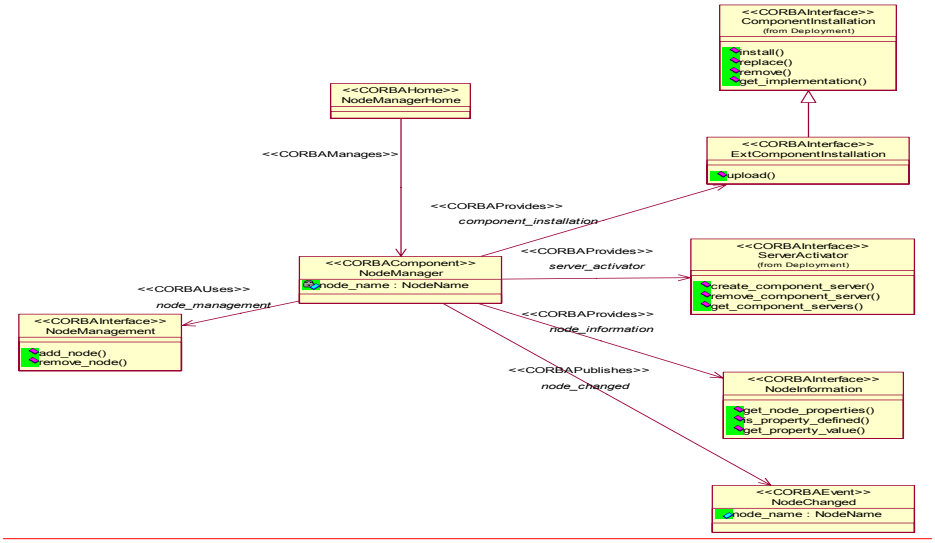
COACH Distributed Computing Infrastructure



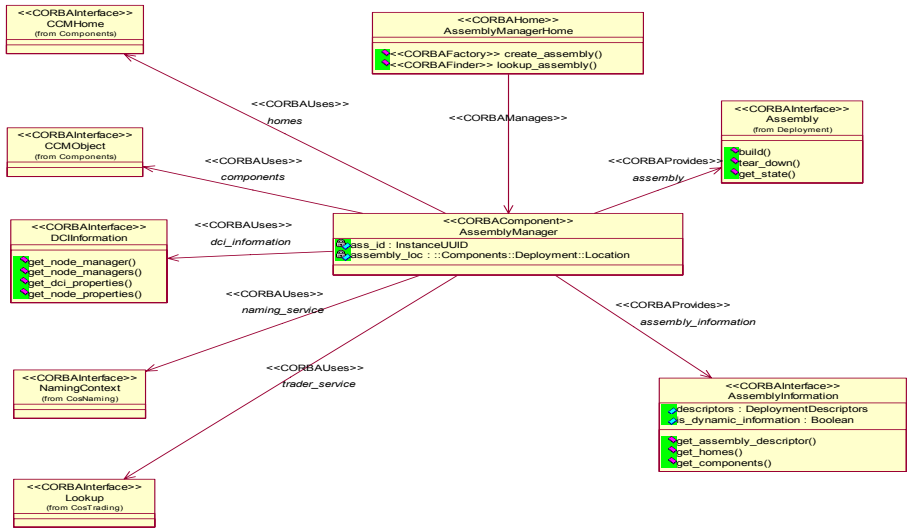
Le composant DCIManager



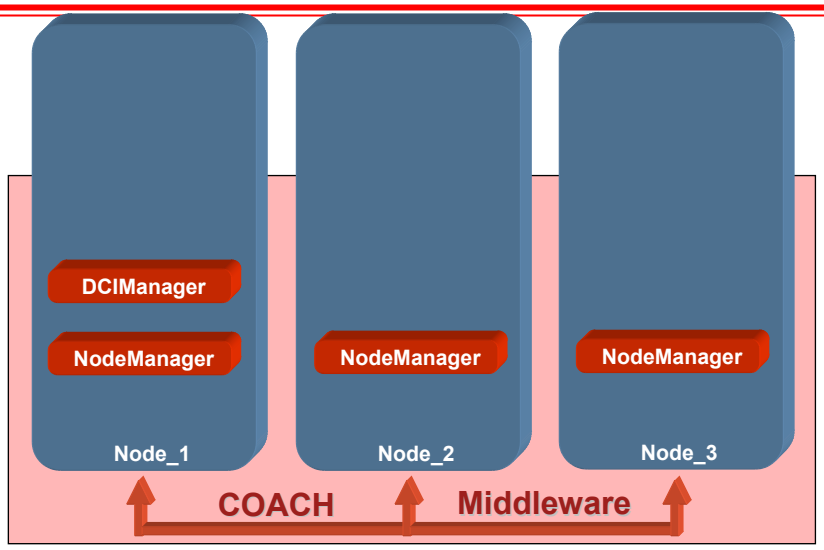
Le composant NodeManager



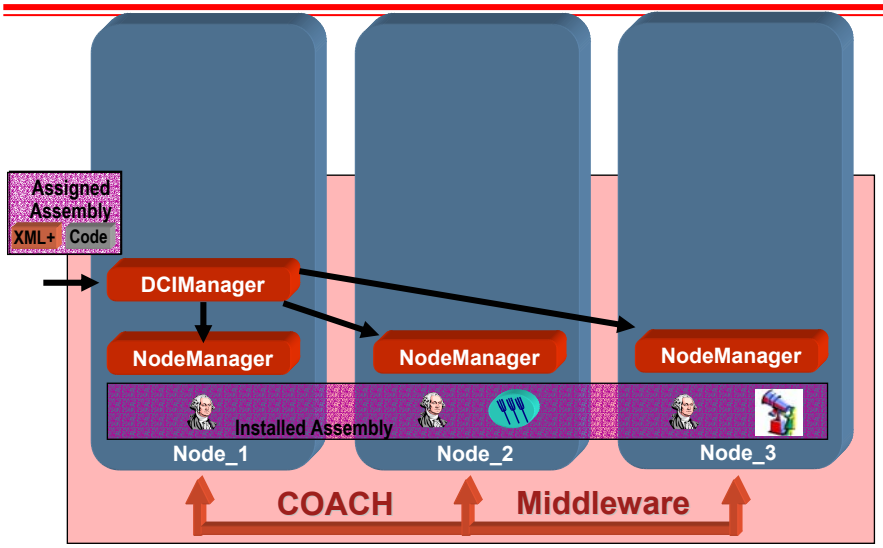
Le composant AssemblyManager



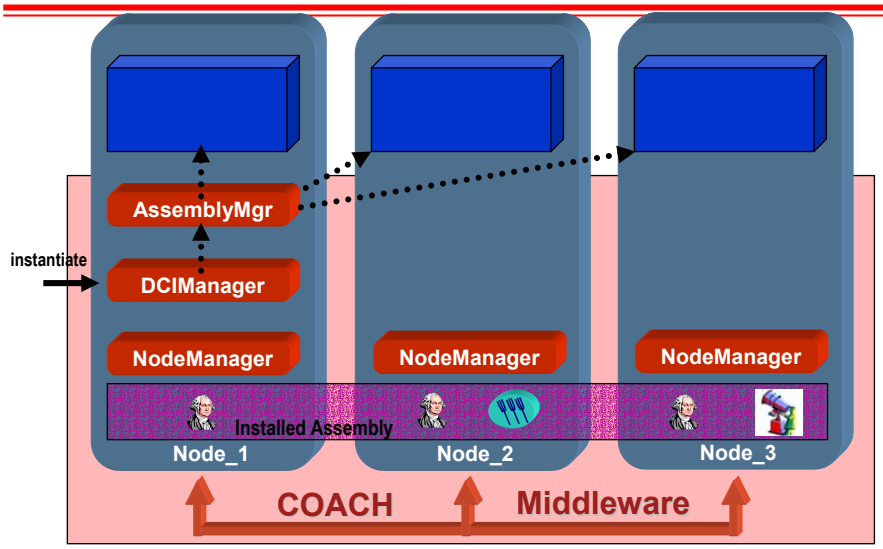
Les composants permanents de l'infrastructure DCI



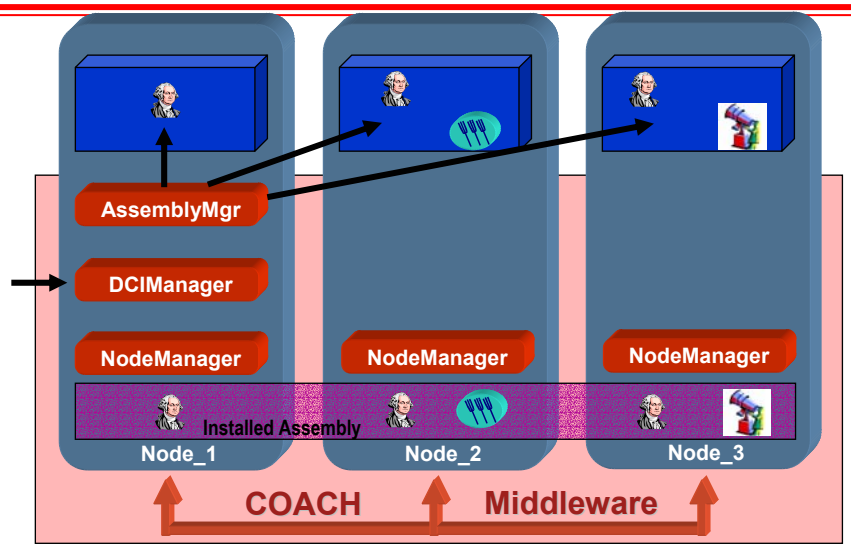
L'installation des composants



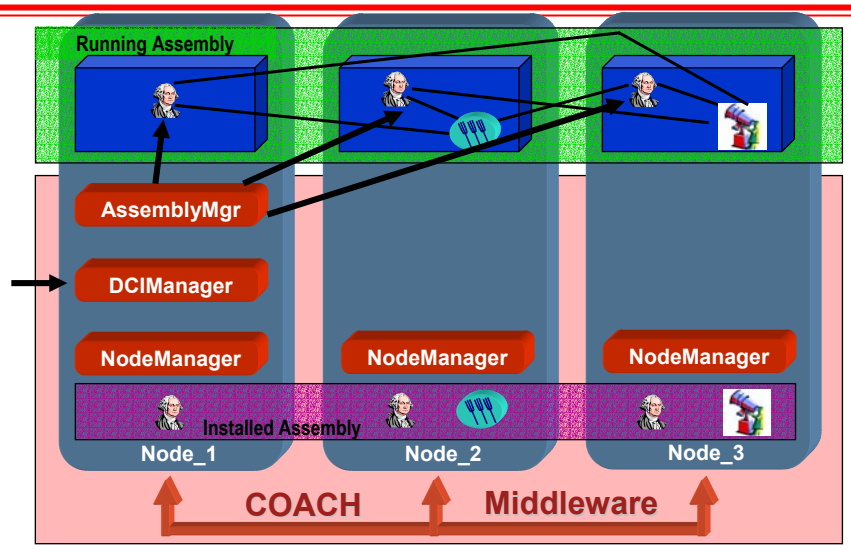
L'instanciation des assemblages



L'instanciation des assemblages



L'interconnexion et le démarrage des composants



Perspectives

- **Etudier d'autres propriétés extra fonctionnelles pour le déploiement**
  - Sécurité pour autoriser ou non les déploiements
  - Répartition de charge
- **Composite = Assemblage + Composant**
  - Assemblage offre une interface de composants
  - Exportation et importation des ports de l'assemblage

L'outillage d'administration

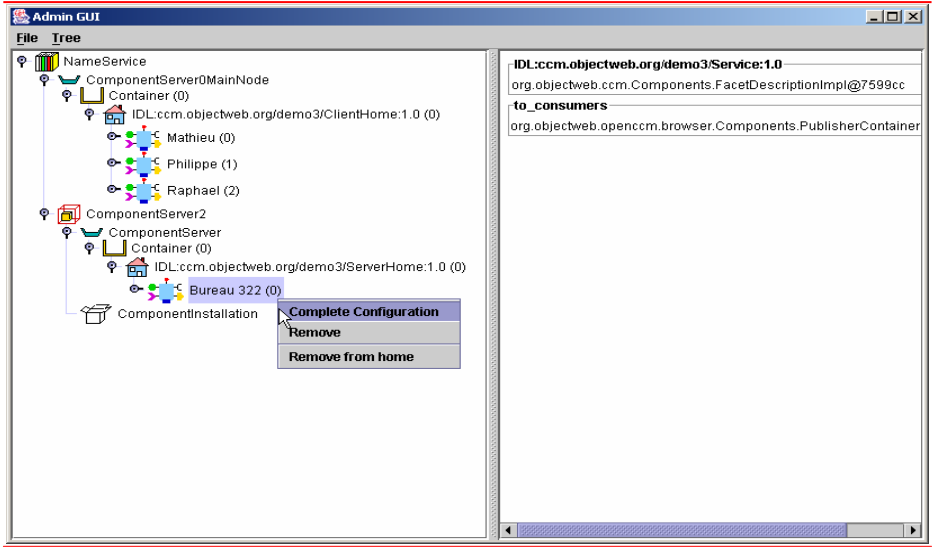
L'outillage d'administration OpenCCM

- La console de navigation et de contrôle
  - Personnalisation du canevas Browser
- L'infrastructure d'observation et de test interactif
  - Développée par Lucent dans projet IST COACH
  - Disponible à partir OpenCCM 0.8

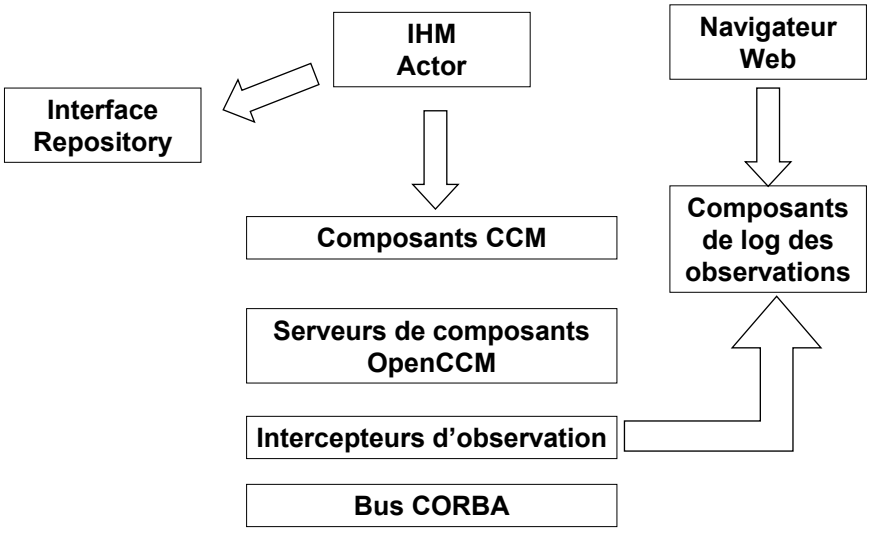
La console de navigation OpenCCM

- Plug-in pour canevas Browser
- Offre navigation et menus de contrôle pour
  - Référentiel des Interfaces
  - `CCMHome` et `CCMObject`
  - API de déploiement CCM + DCI
  - Services
    - Nommage - `CosNaming`
    - Courtage - `CosTrader`
    - Transaction – `CosTransactions`
    - Notification – `CosNotification`
  - …

Une instance du navigateur OpenCCM



L'infrastructure COACH d'observation et de test



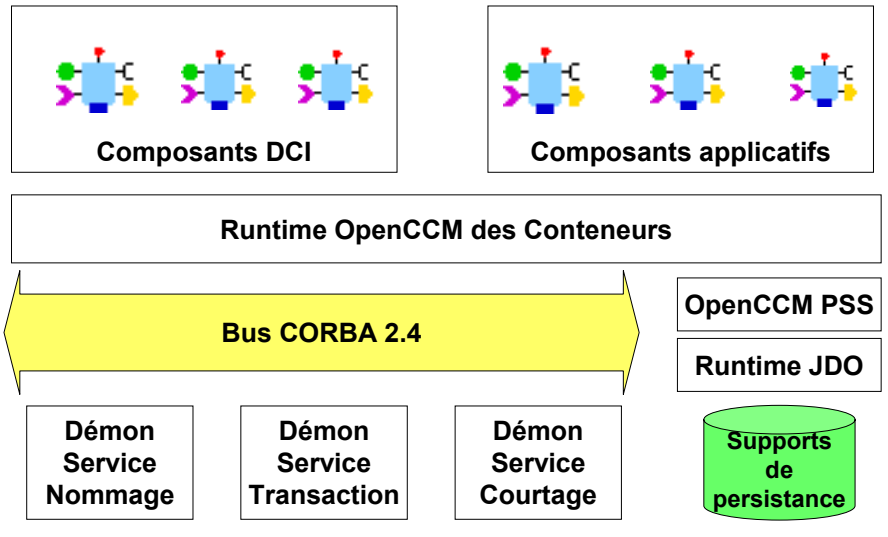
Captures écran prochainement

La plate-forme d'exécution

La plate-forme d'exécution

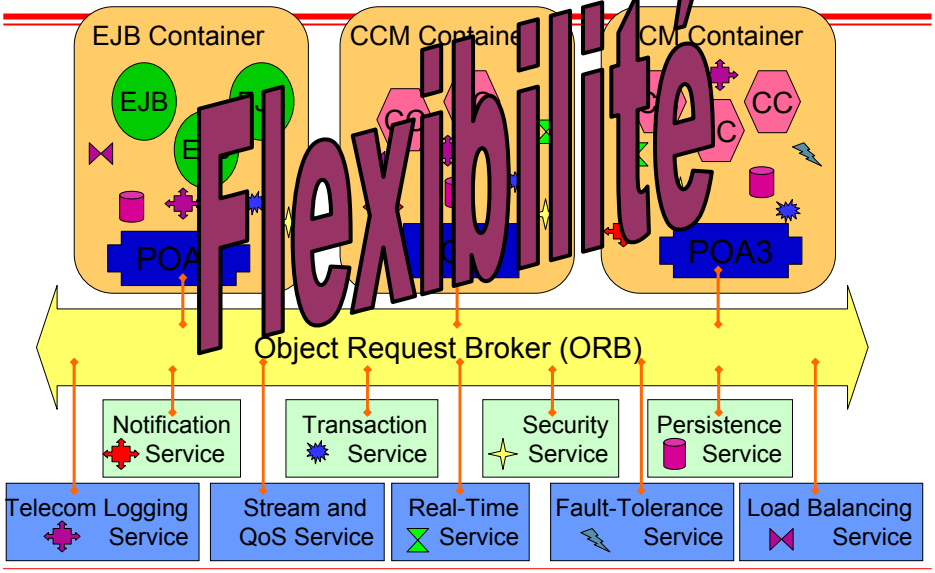
- Construite au dessus d'un bus CORBA 2.4 et des services fournis
- Des scripts pour démarrer / arrêter les services
  - Nommage, Transaction, Courtage
- Une bibliothèque runtime
  - Ensemble de classes héritées par le code conteneur généré
- Une implantation du service de persistance CORBA
  - Construite au dessus de JDO
  - Actuellement produit Kodo, prochainement ObjectWeb Speedo

La plate-forme d'exécution



A COMPLETER

A la recherche du Graal : les conteneurs ouverts



Conclusion

Conclusion

- **Le projet OpenCCM**
  - Une plate-forme CCM pour concevoir, développer, conditionner, assembler, déployer, exécuter et administrer des applications réparties à base de composants CORBA
  - Une plate-forme logiciel pour construire des plates-formes CCM
  - Un logiciel libre LGPL hébergé par le consortium ObjectWeb
- **La partie conteneur d'OpenCCM est à compléter !**
  - Générateur CORBA Component Descriptor
  - Conteneurs pour composants Service, Process et Entity
  - Majeure partie des interfaces des conteneurs
  - Intégration des services dans les conteneurs
    - Persistance, transaction, sécurité, notification
  - Recherche en cours sur les conteneurs ouverts

Les trois axes de recherche OpenCCM - projet INRIA Jacquard

- **Ouverture de la chaîne de production**
  - Vers des modèles de plus haut niveau (UML 2.0, EDOC, DSL, ADL)
  - Une plate-forme CCM prête pour démarche MDA
  - ➔ Assembler de nouvelles chaînes de production
- **Fiabilisation et flexibilite de l'infrastructure de déploiement**
  - Courtage de composants sur l'étagère avec TORBA/TOSCA
  - Déploiement avec propriétés extra fonctionnelles transactionnel, sécurité, répartition de charge, QoS, …
  - Composite = composant + assemblage + glue
  - ➔ Assembler l'infrastructure de déploiement
- **Structures d'accueil adaptables**
  - Conteneurs ouverts et connecteurs
  - Calibration des petits équipements aux mainframes
  - Construction de services, e.g. transaction GOTM
  - ➔ Assembler les structures accueils

Essayer et adopter OpenCCM

http://openccm.objectweb.org