Architecture de composants
La réponse de l'OMG

© Olivier Caron

Polytech'Lille

Standardisation OMG

- √ Le modèle de composants CORBA (CORBA 3.0)

Standardisation OMG

- √ Le modèle de composants CORBA (CORBA 3.0)
- √ Le processus MDA

Standardisation OMG

- √ Le modèle de composants CORBA (CORBA 3.0)
- √ Le processus MDA
- √ Programmation par aspects : alternative ou complément ?

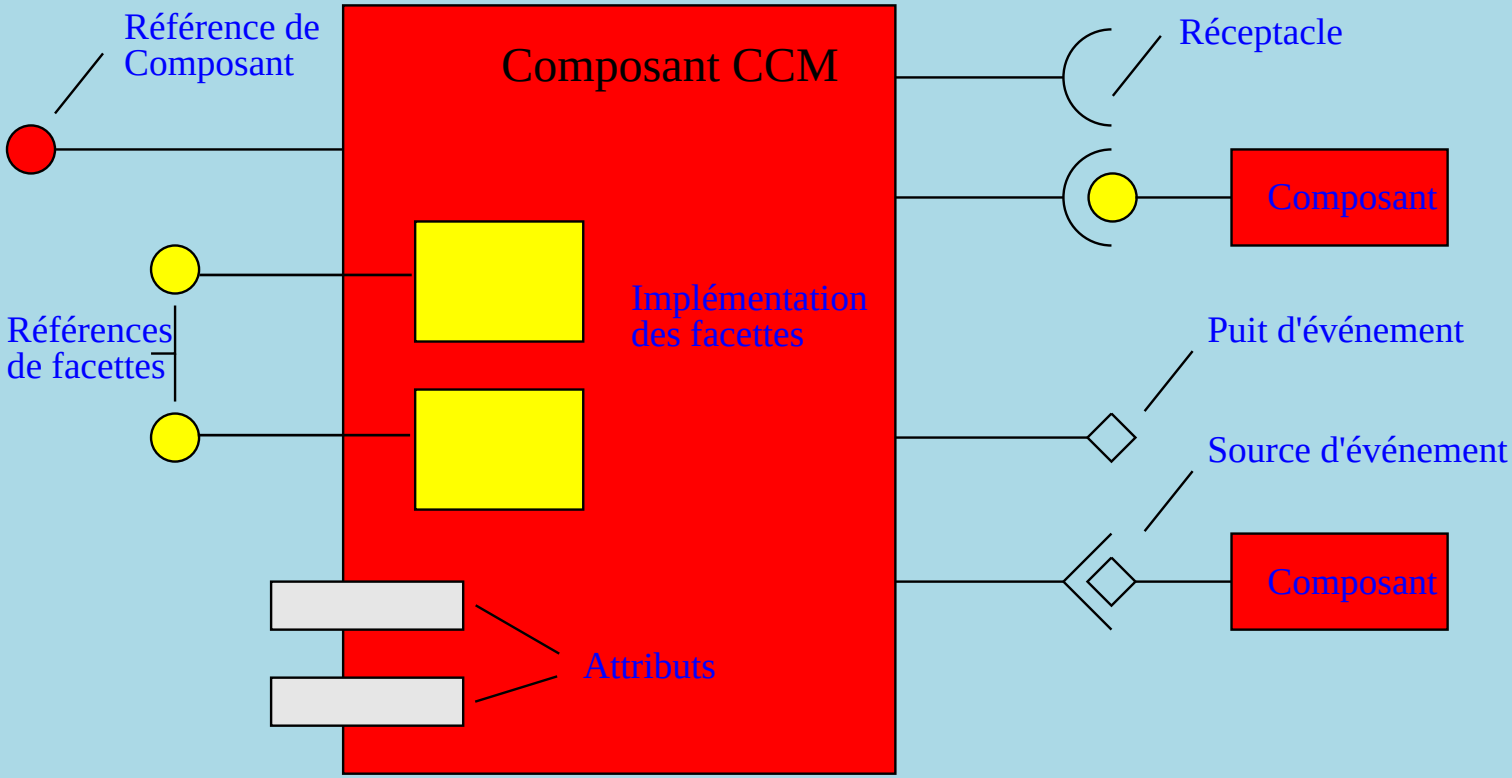
Le modèle CCM
Corba Component Model

- √ Le plus récent
- √ Le plus complet
- √ Pas forcément le plus simple
- √ Pas d'implémentation complète de cette norme (oct. 2002)

La spécification CCM

- √ Un modèle abstrait de composants
  Interface Definition Language IDL3
- √ Un framework d'implantation (CIF)
  Component Implementation Definition Language (CIDL)
- √ Un modèle de containers
- √ Des dialectes XML de conditionnement et de déploiement
- √ Des référentiels de méta-données

Un composant CCM



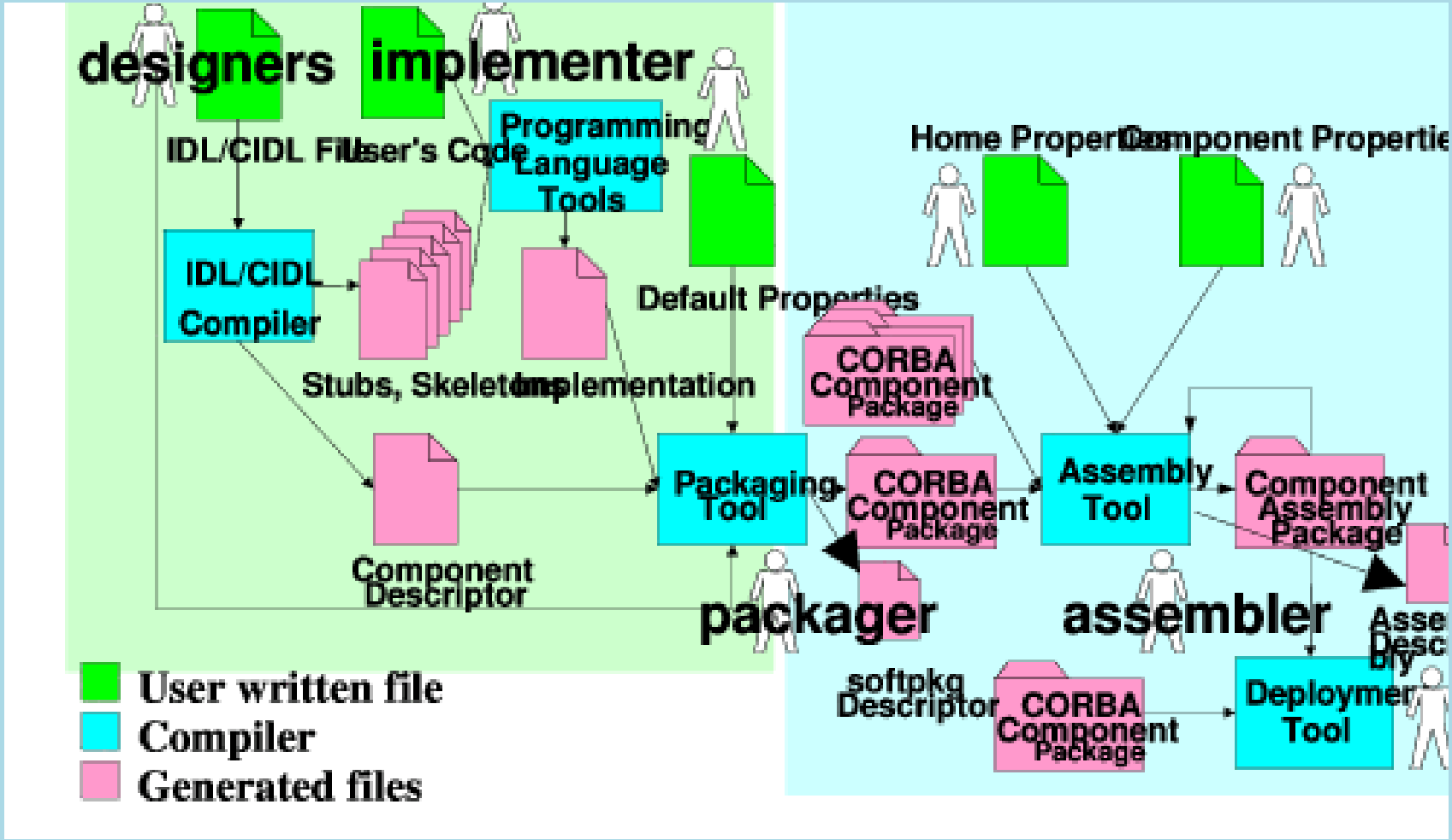
Le langage IDL3

- √ Une extension du langage de spécification IDL2
- √ Des règles de traduction de composants CORBA en objet CORBA
- √ Connection / assemblage des composants par les ports.

Exemple simple de spécification IDL3

```
module banque {
  interface IClient {
    double getSolde (in IdClient id) ;
    void crediter ( in IdClient id, in double montant) raises ...
    void debiter (in IdClient id, in double montant) raises...
  } ;
  interface IBanque {
    void virementVers (in IdClient ids, in IdClient idd, double montant)...
  } ;
  component Banque {
    provides IClient serviceClient ;
    provides IBanque serviceBanque ;
  } ;
  home BanqueHome manages Banque { } ;
```

CCM , the Big Picture



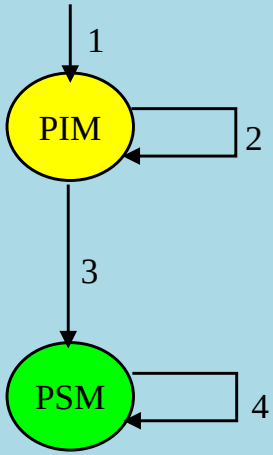
De CCM à MDA

- √ Un constat :
  Le modèle de composant unique,
  la plateforme unique
  n'existe pas et n'existera pas
- √ Comment faire en sorte d'être le moins dépendant d'une solution propriétaire (langage, modèle architectural, système d'exploitation,. . .) ?

De CCM à MDA

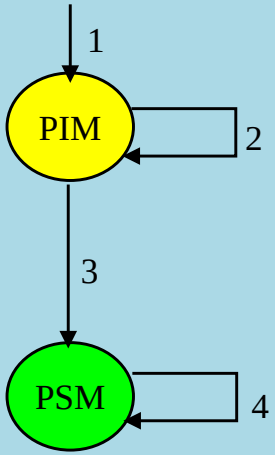
- √ Un constat :
  Le modèle de composant unique,
  la plateforme unique
  n'existe pas et n'existera pas
- √ Comment faire en sorte d'être le moins dépendant d'une solution propriétaire (langage, modèle architectural, système d'exploitation,. . .) ?
  - ▶ La solution : le processus MDA (Model Driven Architecture)

Le processus MDA



- √ PIM : Platform Independent Model (ex : UML, ODP)
- √ PSM : Platform Specific Model (ex : EJB, CORBA)

Le processus MDA



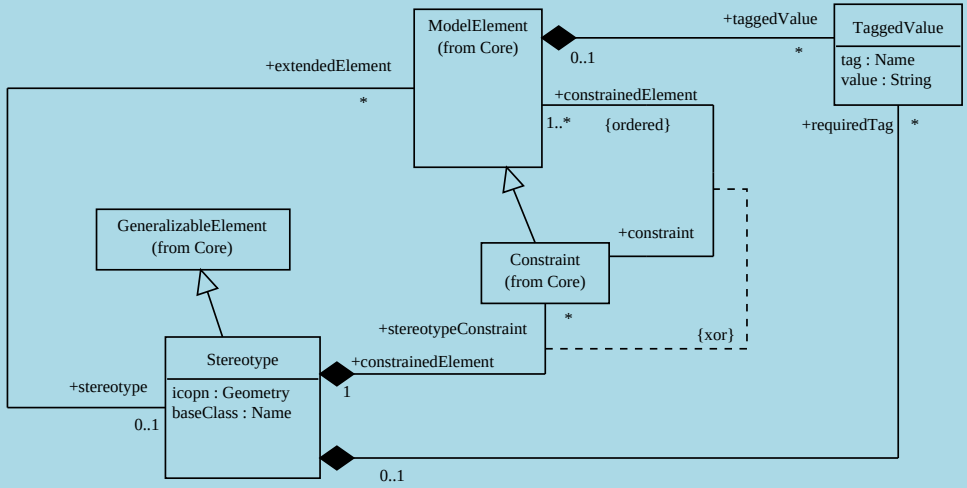
- √ PIM : Platform Independent Model (ex : UML, ODP)
- √ PSM : Platform Specific Model (ex : EJB, CORBA)
- √ Prendre en compte les aspects spécifiques à une technologie le plus tard possible

Les technologies préconisées par MDA

- √ UML (from OMG)
- √ MOF (Meta Oject facilities)
- √ profils UML
- √ OCL (Object Constraint Language)
- √ XML, XMI

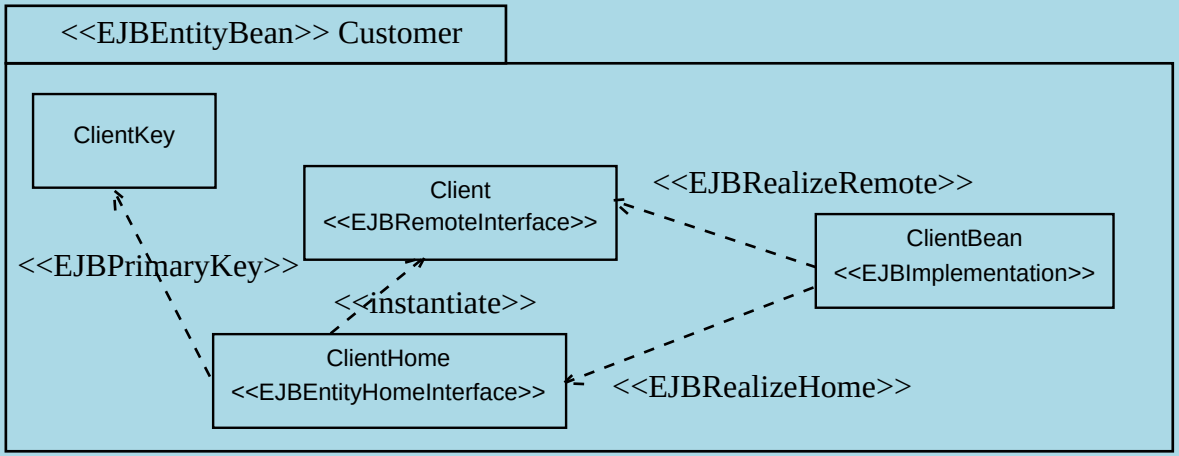
Les profils UML

- √ Mécanisme `standard` d'extension d'UML
- √ Ensemble cohérent de stéréotypes, valeur marquées (tagged value) et contraintes



Exemple profil UML

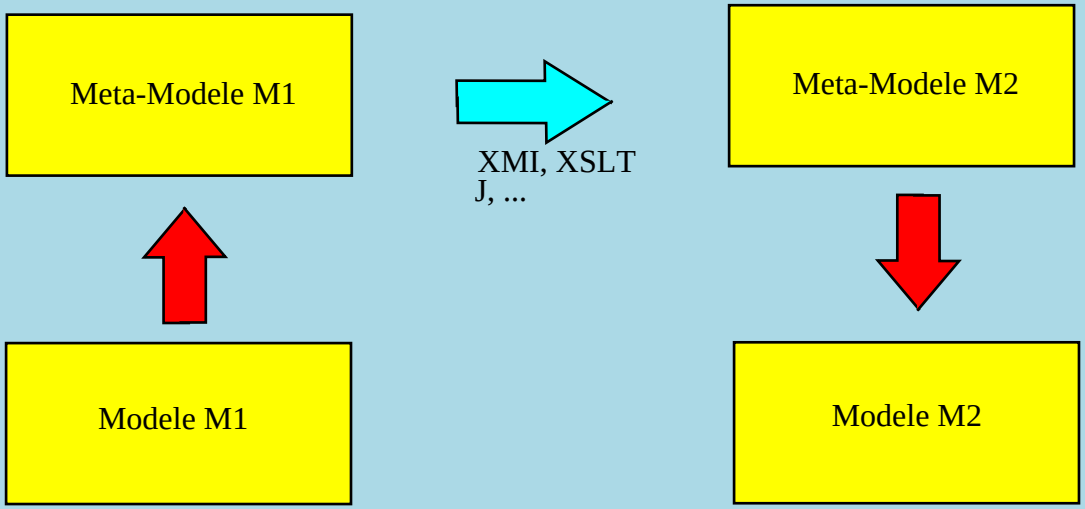
- √ Des profils standardisés existent : profil CORBA 2, profil EJB 1.1
- √ A venir : profil EJB 2.0, profil CCM



Avantages Atelier UML

- √ Spécification plus rapide d'un composant
- √ Outil intégré de vérification structure d'un composant
- √ Outil de génération de code, de transformation de modèles

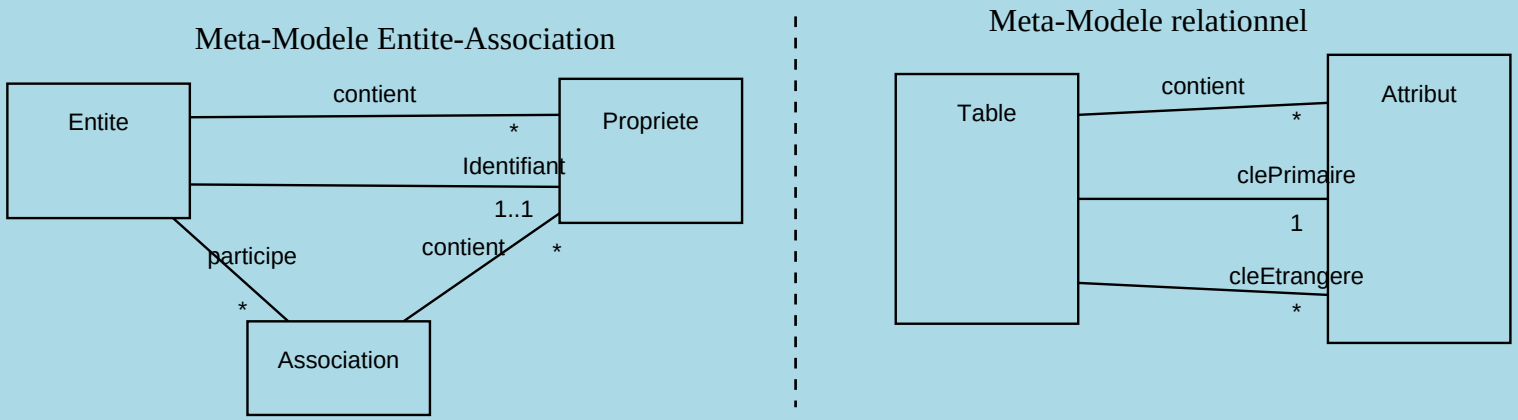
Transformation de modèles



- √ modèles XMI, règles XSL
- √ Outil Objecteering, langage objet J

Exemple transformation de modèles

√ Passage modèle entité-association vers modèle relationnel



√ Identifier les règles de traduction :
ex : Entite → Table
identifiant → clePrimaire

Conclusion

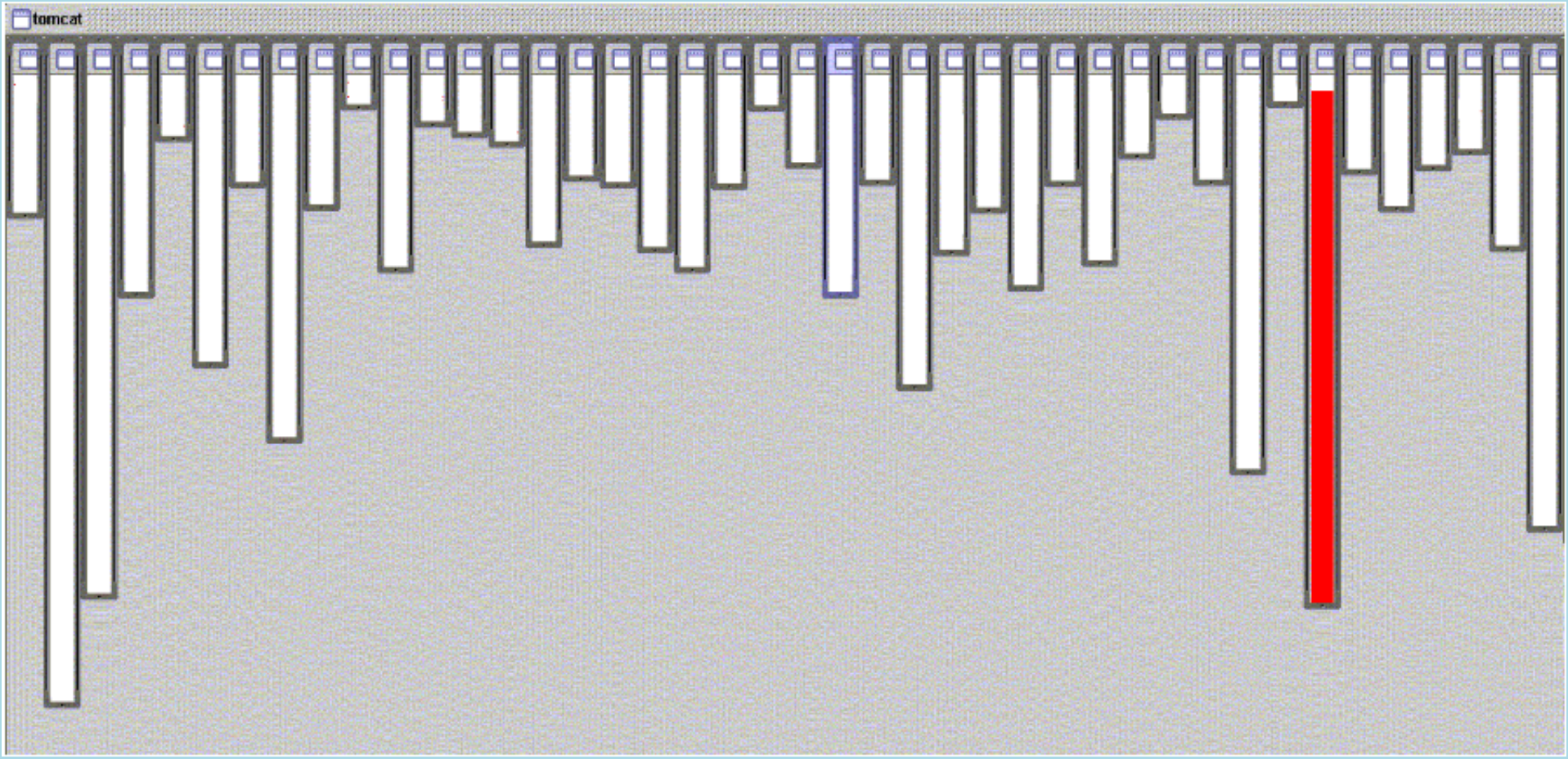
√ Encore beaucoup de choses à définir :

- Modèle de composants abstrait ? (UML 2.0 ?, autre ?)
- Trouver une méthodologie à base de composants
- Assemblage de composants : validité, tests,. . .
- ...

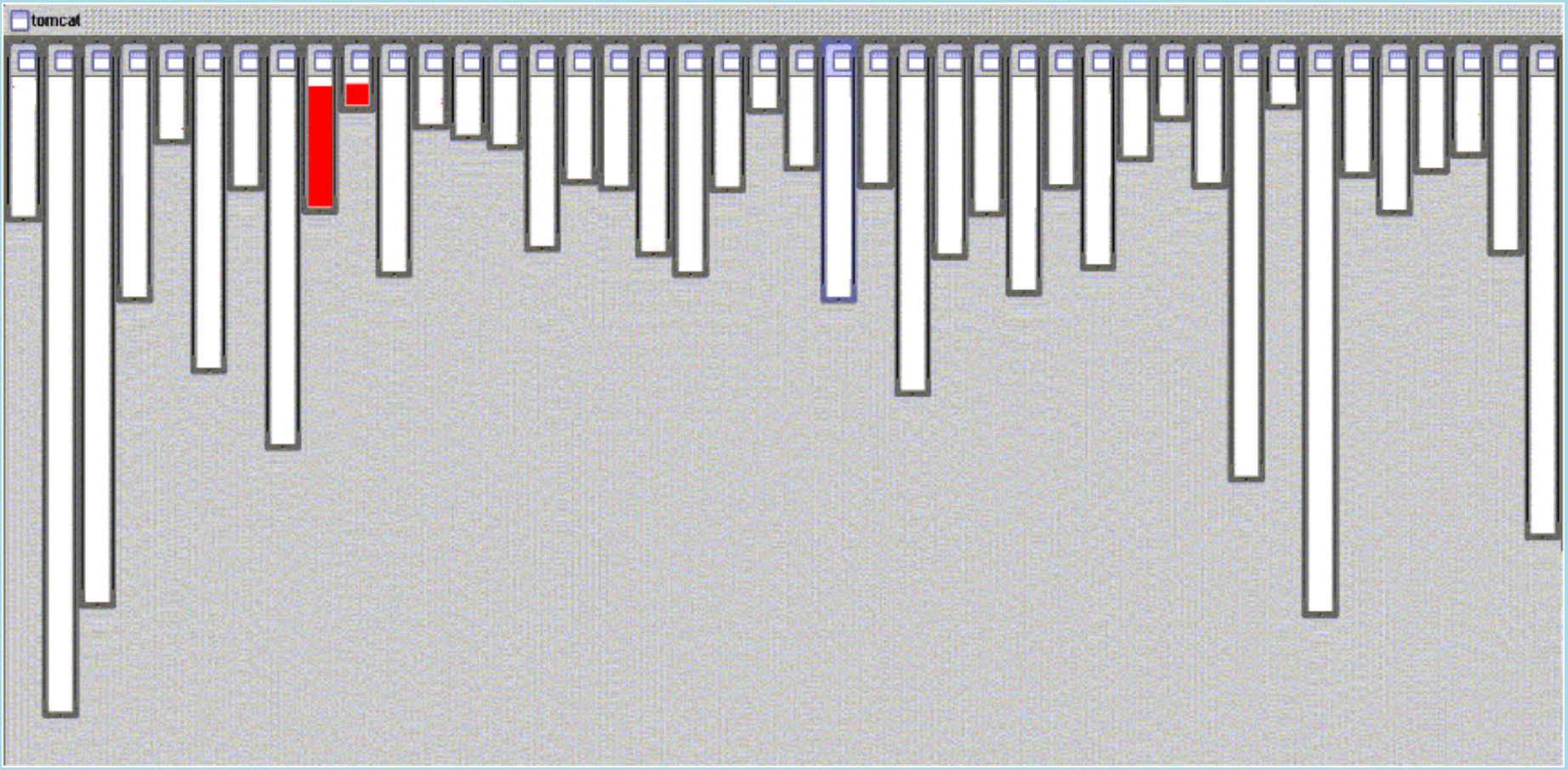
Programmation par aspects

- √ Une nouvelle manière de programmer
- √ Nouveau découpage d'une application : un programme + aspects
- √ Réutiliser des aspects pour d'autres applications : Vers des composants d'aspects

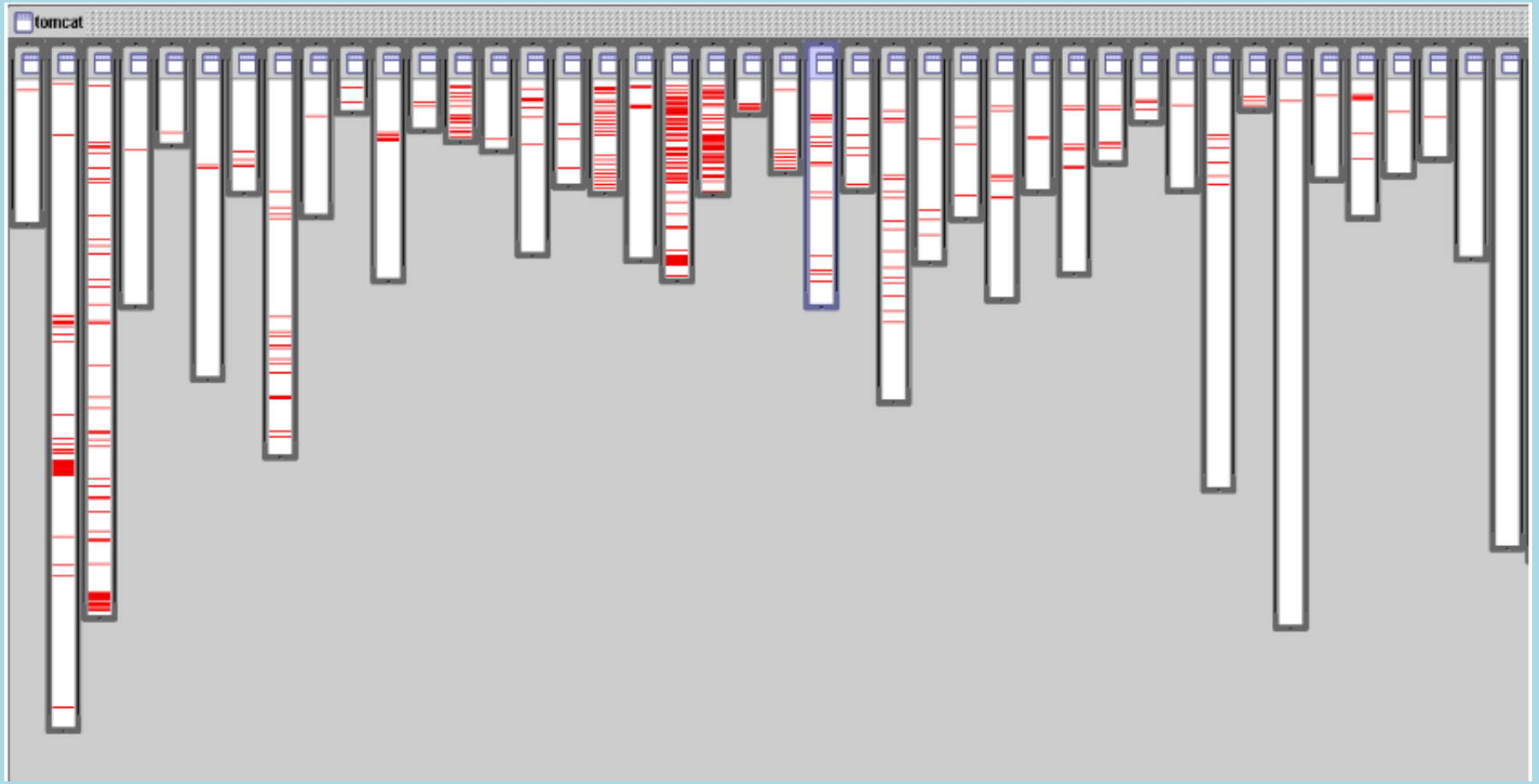
Exemple : serveur Tomcat et Parsing (1/3)



Exemple : serveur tomcat et Pattern Matching (2/3)



Exemple : serveur tomcat et Logging (3/3)



Approche AOP

- √ disposer d'un langage d'aspect
- √ définir les points de jonction
- √ Réaliser le tissage des aspects sur l'application (weaver)

Aspect J

- √ www.aspectj.org (Xerox Lab)
- √ Utilisation du langage Java
- √ compilateur statique java (ajc)

JAC

- √ Java Aspect Component (LIFL)
- √ Langage Java
- √ Assemblage dynamique d'aspects
- √ Quelques composants d'aspects : RMIAspect, . . .

JBoss 4.0

- √ Architecture interne AOP
- √ Persistance, transaction, sécurité sont des aspects
- √ Possibilité d'ajouter de nouveaux aspects !