Components and Architectures

Master ALMA 2eme année

TP Composant : Implémentation HADL

Encadrant : M.Oussalah

A.Marguerite
R.Rincé

Université de Nantes
2 rue de la Houssinière, BP92208, F-44322 Nantes cedex 03, FRANCE

Table des matières

1 Introduction

1.1 Présentation du sujet

L'objectif de ce projet/TP est de réaliser un protocole de transfert de messages entre deux machines avec des listes des canaux de communication *First In First Out*. La démarche proposée par l'énnonce propose les étapes suivantes :

1. Modéliser ce protocole avec l'outil Roméo.
2. Construire le graphe des marquages.
3. Spécifier en logique temporelle :
   le protocole peut toujours revenir à l'état initial.
4. Expliciter les propriétés vraies.
5. Programmer une implémentation :
   Une machine est pilotée par l'utilisateur, l'autre par le programme.
6. Tests :
   rendre observable les différentes transitions.

2 Implémentation du M2

2.1 Détails de la modélisation

Construisons un méta-modèle basé sur l'entité `Composant` :

- Un `Composant` possède des `Propriétés`. Ces dernières peuvent être fonctionnelles (`PropFonc`) ou non fonctionnelles (`PropNonFonc`).
- Il possède deux `InterfaceComposants` : une pour les `Services` requis, une pour les `Services` fournis.
- Une interface est aussi constituée de `Ports` (fournis ou requis).
- Chaque port est associé à un et un seul `Service`, mais un `Service` possède plusieurs `Ports`.

Afin d'effectuer la communication entres composants, il est nécessaire d'introduire la notion de connecteur :

- Un `Connecteur` a deux `InterfaceConnecteurs`.
- Chaque `InterfaceConnecteur` possède deux `Rôles` : un rôle requis `RoleR` et un rôle fourni `RoleF`.
- Chaque rôle d'une interface est relié par un lien `attachement` à son port correspondant (*c.a.d* : un rôle requis à un port requis et un rôle fourni à un port fourni).
- Les deux interfaces d'un connecteur sont reliés par deux `Glus`, chaque `Glu` reliant un rôle requis au rôle fournis opposé.

La `Configuration` est un composant composé de `Composants` :

- Elle possède deux `InterfaceConfigs`. Ces dernières hériant d'`InterfaceComposant`.
- Il est possible de relier un `PortR` de l'`InterfaceConfig` à un `PortR` d'un composant de la configuration (et réciproquement pour le `PortF`) grâce à un lien `binding`.
- Un connecteur peut être aussi une `Configuration`.

Corrections du M2 après évaluation Suite aux remarques émises lors de l'évaluation, nous avons apporté les corrections suivantes :

1. Expliciter un lien direct entre `Connecteur` et `Glu`.
2. Appliquer un pattern *composite* à l'entité `Connecteur`.
3. Refactoring de la hiérachie entre `Composant`, `Connecteur` et `Configuration`. Désormais `Composant` et `Connecteur` sont au « même niveau » par rapport à `Configuration`.

2.2 Diagramme M2

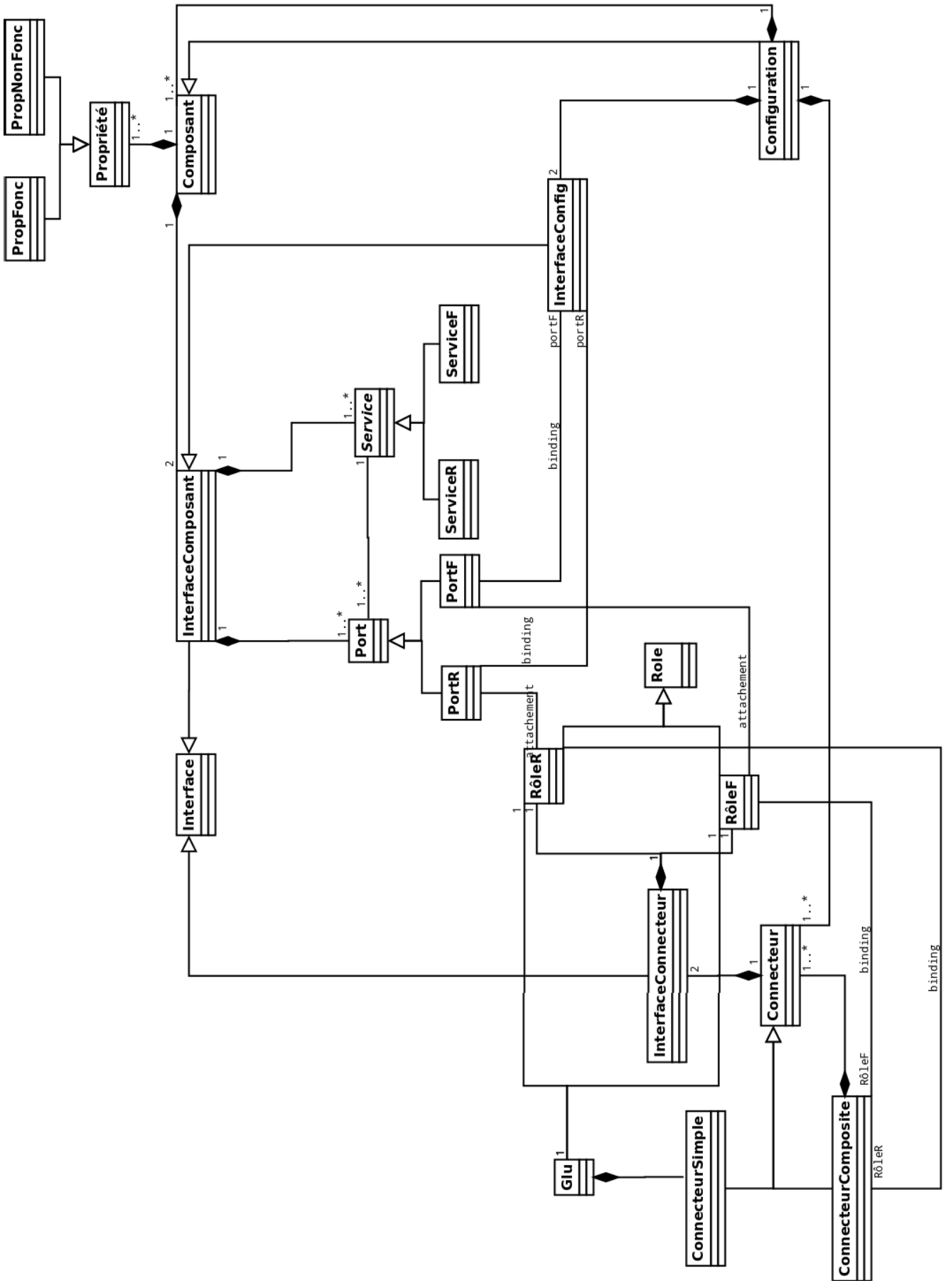

Figure 2.1 – Méta-Model (M2)

3 Implémentation du M1

Dans ce chapitre nous définirons une application *Clients-Serveur* en implémentant le méta-modèle proposé au chapitre 2.

3.1 Démarche modélisation M1

L'énoncé du projet/TP nous propose un serveur avec les fonctionnalités suivantes :

- `ConnexionManager` : Gestion des connexions.
- `Database` : Gestion de la base de données.
- `SecurityManager` : Gestion d'un système de sécurité.

Un composant étant par définition la spécification d'une fonctionnalité, chacune des fonctionnalités précedement listées est un `Composant`. Chaque service de chaque composant a son port requis relié au rôle requis du connecteur (et inversement pour les ports et rôles fournis).

Les liaisons entre les différents composants du serveur et le client sont shématisées dans la figure 3.1.

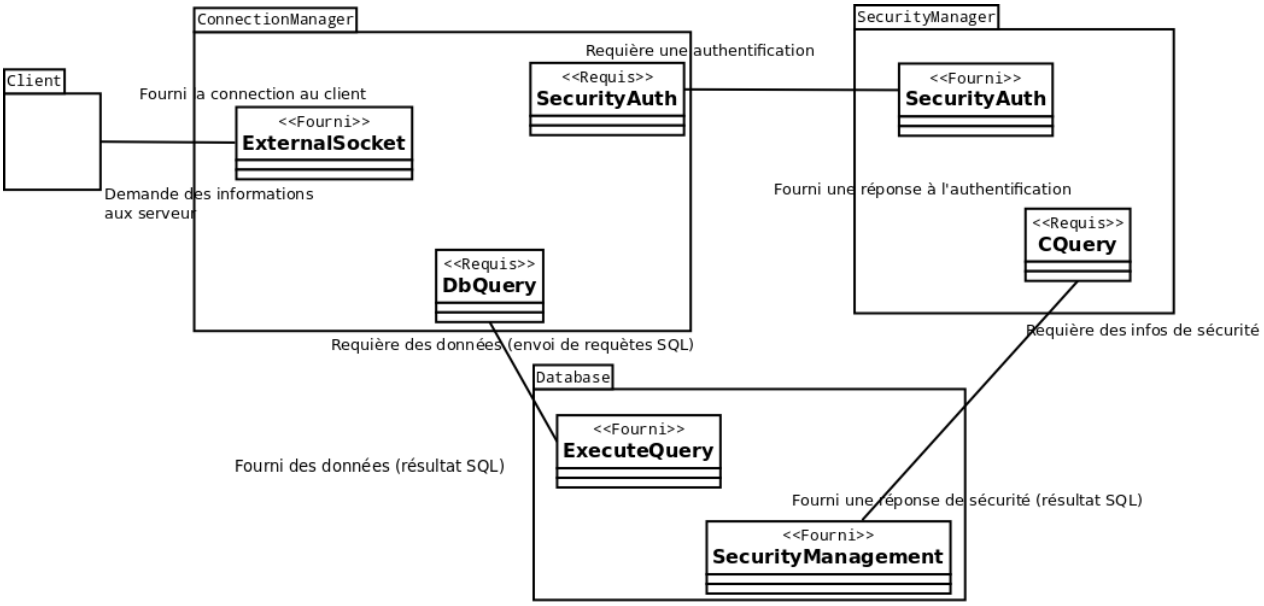

FIGURE 3.1 – Description du Serveur

.

Nous retrouvons une version formelle de ce shéma dans la figure 3.2 pour la partie « serveur » puis dans la figure 3.3 pour la partie « client » ainsi que sa liaison avec le serveur.

Corrections du M1 après évaluation Dans le M1 intialement proposé, on pouvait noter l'absence de *composant serveur*. Pour palier à ce problème nous avons apporté les modifications suivantes :

1. Création d'un `Composant Serveur`.
2. Mise en place d'un lien `binding` entre cette configuration et le `Composant serveur`.
3. Liaison `attachement` entre les rôles du `Connecteur` RPC et le `Composant serveur`.

Ces modifications sont visibles sur la figure 3.3.

3.2 Côté Serveur

3.2.1 Déscription des composants serveur

ConnexionManager

Ce composant possède les services suivants :
- `ServiceFExternalSocket` : fournit une connexion au(x) potentiel(s) client(s).
- `ServicerDDBQuery` : requière une réponse de la base de données (requête SQL).
- `ServiceRSecurityManager` : requière une validation de l'authentification.

Database

Ce composant possède les services suivants :
- `ServiceFSecurityManagement` : fournit une réponse à la requête de sécurité.
- `ServicerFExecuteSQL` : fournit une réponse à une requête du client (par le Connexion-Manager).

SecurityManager

Ce composant possède les services suivants :
- `ServiceFSecurityAuth` : fournit une validation de la demande de connexion.
- `ServicerSecurityManager` : requière une réponde de sécurité de la base de données (requête SQL).

3.2.2 Diagramme M1 côté serveur

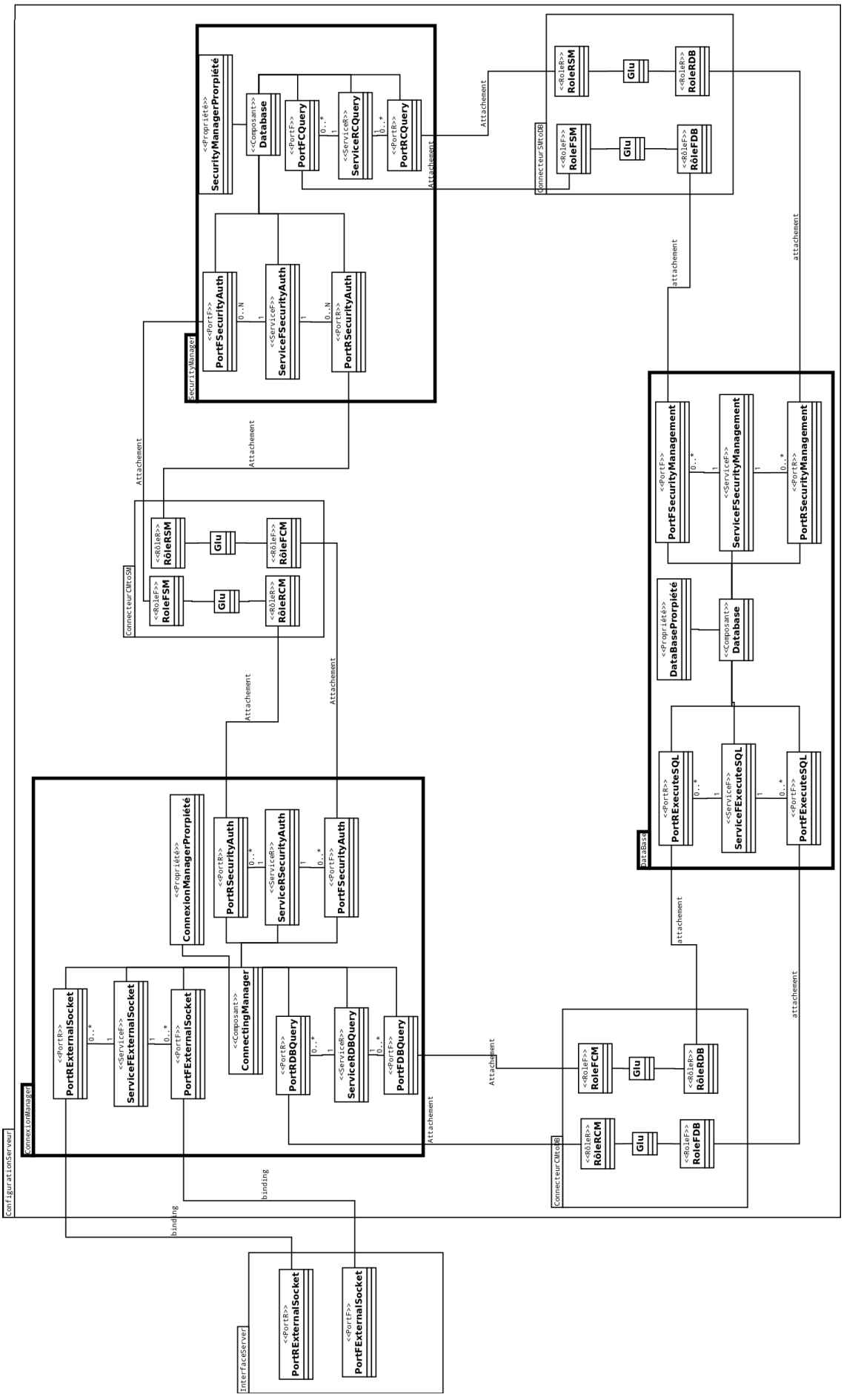

FIGURE 3.2 – Model (M1) côté serveur

3.3 Côté Client

3.3.1 Description du Client

Le client est un composant permettant à un utilisateur de demander une connexion au serveur à partir d'un login et d'un mot de passe. Une fois connecté, le client peut interroger la base de données du serveur.

Le composant client possède le service :

- `ServiceRConnexionRPC` : requière une connexion au serveur.

3.3.2 Diagramme M1 côté client

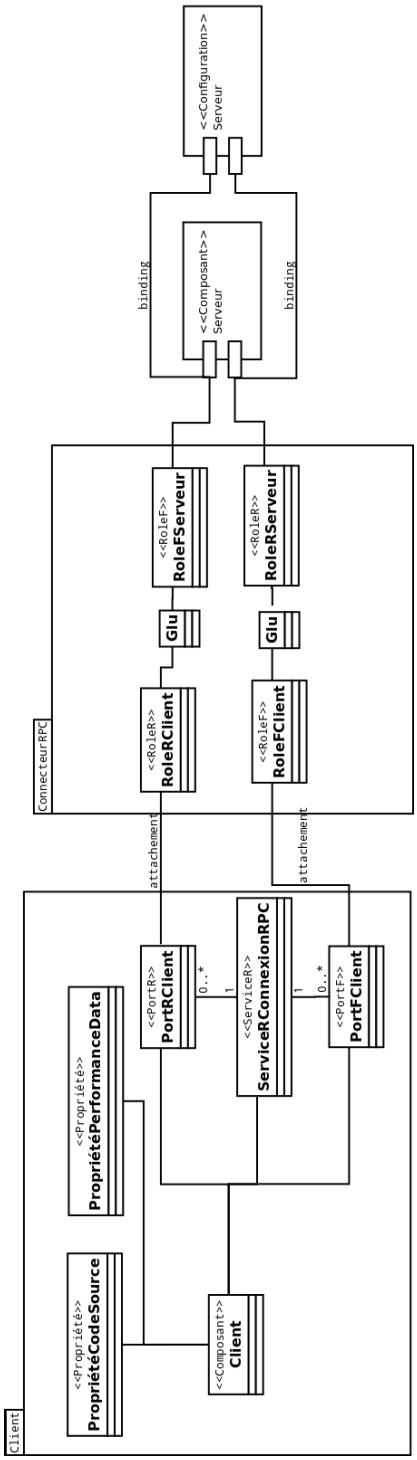

FIGURE 3.3 – Model (M1) côté client

4 Conclusion

À ce stade, la modélisation de notre méta-modèle (*cf.* chapitre 2) nous a permis de générer un modèle pour notre application *Clients-Serveur* (*cf.* chapitre 3). Par la suite il sera nécéssaire de passer à l'étape 3 présentée au chapitre 1. C'est à dire d'implémenter le méta-modèle M2 en JAVA.

La création de l'application *Clients-Serveur* repose sur l'héritage des classes du méta-modèle. Enfin l'instanciation concrète de ses classes achèvera l'implémentation de nottre application.

Table des figures