Rapport Projet ASA

Architectures et Styles d’Architectures

par Nicolas Bitaillou et Nihel belhadj kacem

Introduction

Notre projet consiste à mettre en place un langage d'architecture et à l'utiliser pour définir de manière concrète une structure Client/Serveur. Dans une première étape, nous avons élaboré le méta-modèle (M2), qui spécifie précisément la syntaxe, la grammaire et les contraintes inhérentes aux composants du modèle à venir. Par la suite, nous avons développé un modèle M1 en se basant sur le méta-modèle M2, détaillant ainsi la configuration du système Client/Serveur. Cette configuration comprend un composant Client, un composant Serveur composé de trois entités distinctes (database, connection manager et security manager), ainsi qu'un connecteur RPC reliant le Client et le Serveur.

Pour mettre notre démarche à l'épreuve, nous avons concrétisé les modèles M1 et M2 en utilisant l’outil EMF et plus particulièrement Ecore afin de visualiser les schémas des différents méta-modèles et générer les différentes classes Java. Cela permet de simplifier le processus d'instanciation des différents composants de notre architecture. L'ensemble de ces étapes représente une phase cruciale dans la validation et le déploiement de notre architecture Client/Serveur.

Spécification du Méta Modèle M2

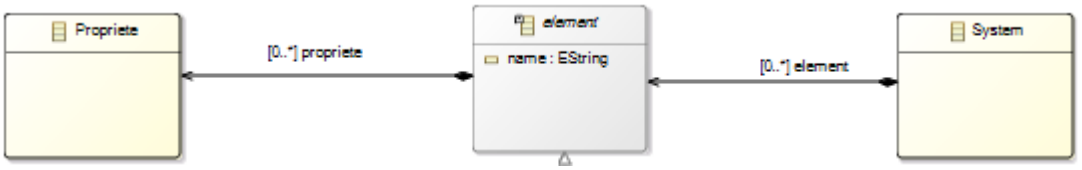
Contenue du Méta modèle

Le projet comporte 8 grandes éléments composants le méta modèle:

- Système
- Element
- Composant
- Interface
- Configuration
- Port et service
- Connecteur
- Attachements et Bindings

Le cœur de notre architecture réside dans le système, qui agit comme le point de départ et le conteneur pour tous les éléments qui le composent. La super-classe "Element" est implémentée par chaque objet du système, tels que les composants, connecteurs, liens,

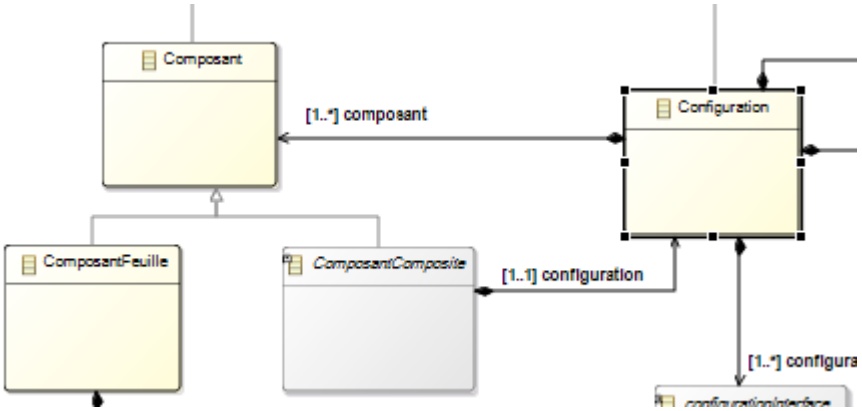
et configurations. Cette classe permet notamment de factoriser l'attribut "Propriétés" commun à ces divers éléments.



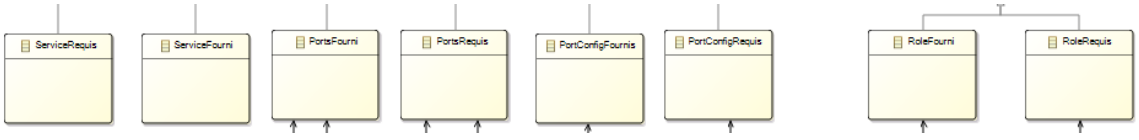
Les composants, conformément à leur définition, représentent des unités de composition spécifiant une ou plusieurs fonctionnalités. Ils disposent d'interfaces pour la communication, peuvent être conteneurs d'autres composants (avec une configuration associée), et sont différenciés en composants composites (via un pattern composite) et composants non composites, appelés feuilles.

Les interfaces, agissant comme des façades, sont définies pour les composants, connecteurs et configurations, chacune comprenant des ports (ou rôles pour les connecteurs). Les ports requis/fournis servent d'entrées/sorties entre composants, permettant l'accès à des services représentant des fonctionnalités.

Les configurations des composants composites définissent leur composition, similaire à un système autonome. Elles contiennent des composants, connecteurs, liens, et peuvent avoir plusieurs interfaces pour la communication avec d'autres éléments du système.



Les connecteurs établissent des liens entre deux composants, reliant les ports requis d'un composant aux ports fournis d'un autre (et vice versa). Nous utilisons des connecteurs explicites, pouvant également contenir d'autres connecteurs via un pattern composite. La "glue" d'un connecteur permet de mapper et de transformer les entrées/sorties.



Les attachements et bindings, prenant la forme de liens, connectent les composants et les connecteurs. Les attachements lient le rôle d'un connecteur au port d'un composant, tandis que les bindings relient le port d'un composant au port d'une configuration. Ces liens, matérialisés par des références (port ou rôle), facilitent le mappage des éléments au sein du système.

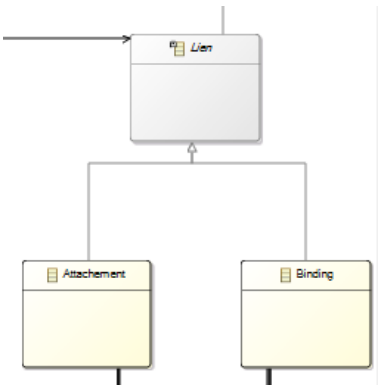
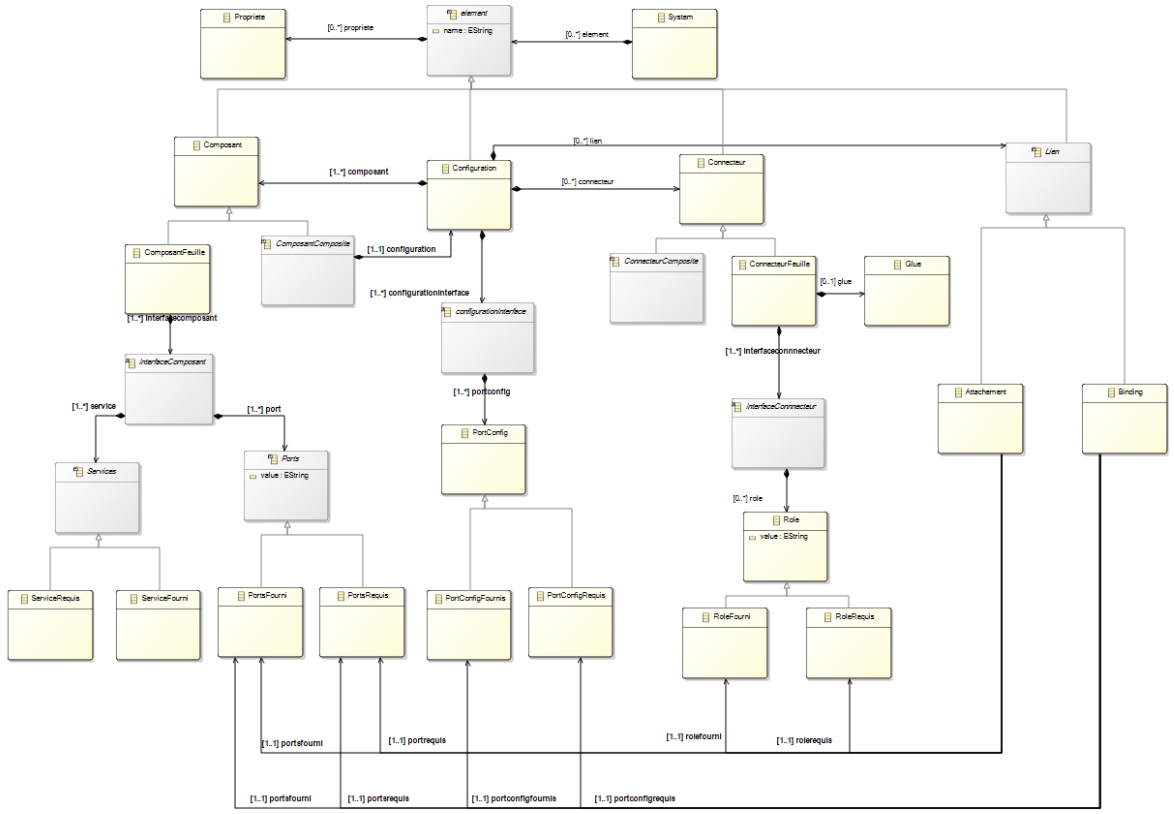


Schéma global du Méta Modèle M2



Spécification du Méta Modèle M1

Grandes parties du Modèle M1

Concernant le modèle M1, nous avons mis en œuvre les composants de l'architecture Client/Serveur tels qu'indiqués dans l'énoncé, en nous appuyant sur le méta-modèle M2 établi précédemment. À titre d'exemple, le serveur est représenté par un composant composite comprenant trois composants feuilles : le gestionnaire de connexion (Connection Manager), le gestionnaire de sécurité (Security Manager) et la base de données (Database). Chacun de ces éléments est étiqueté avec la classe à laquelle il appartient dans le méta-modèle M2. Pour chacun des trois liens internes entre les

composants du serveur, nous avons créé un connecteur, accompagné de deux attachements permettant de relier les ports des composants à ceux du connecteur.

M1 Client/Serveur

Schéma Global:

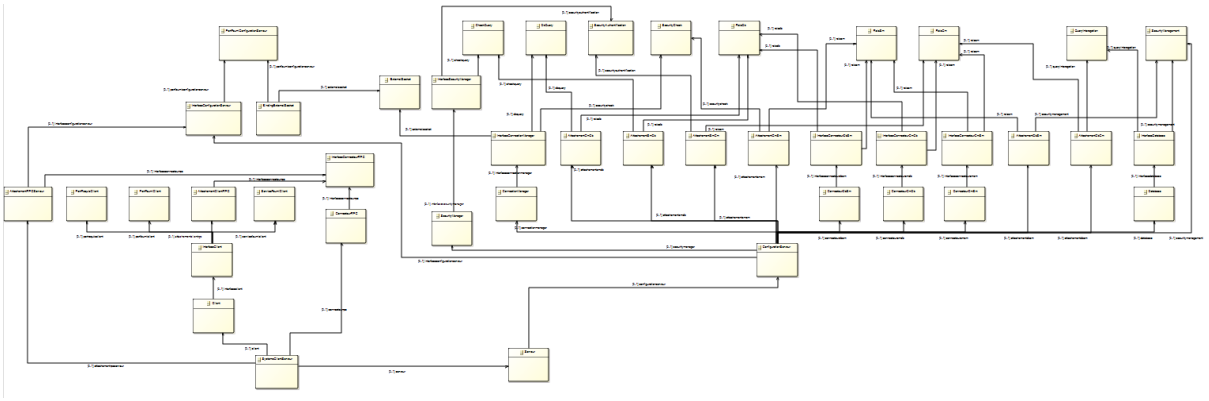


Schéma Client:

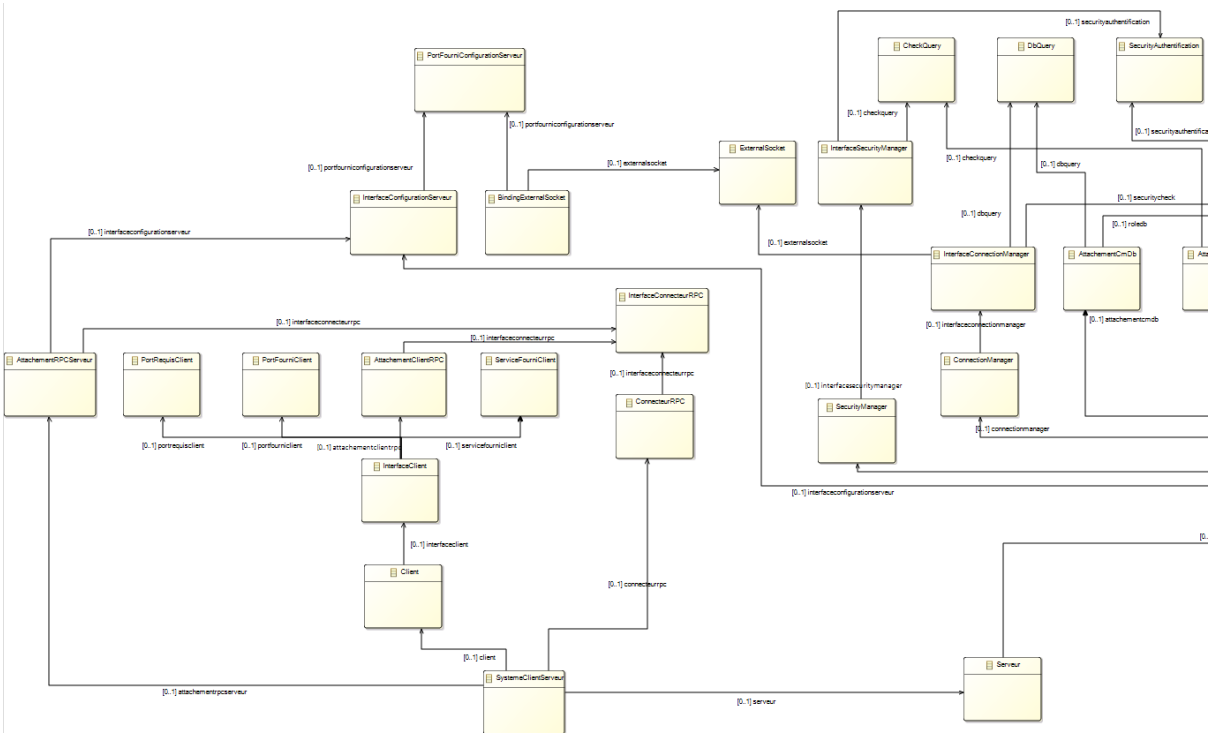


Schéma Serveur global:

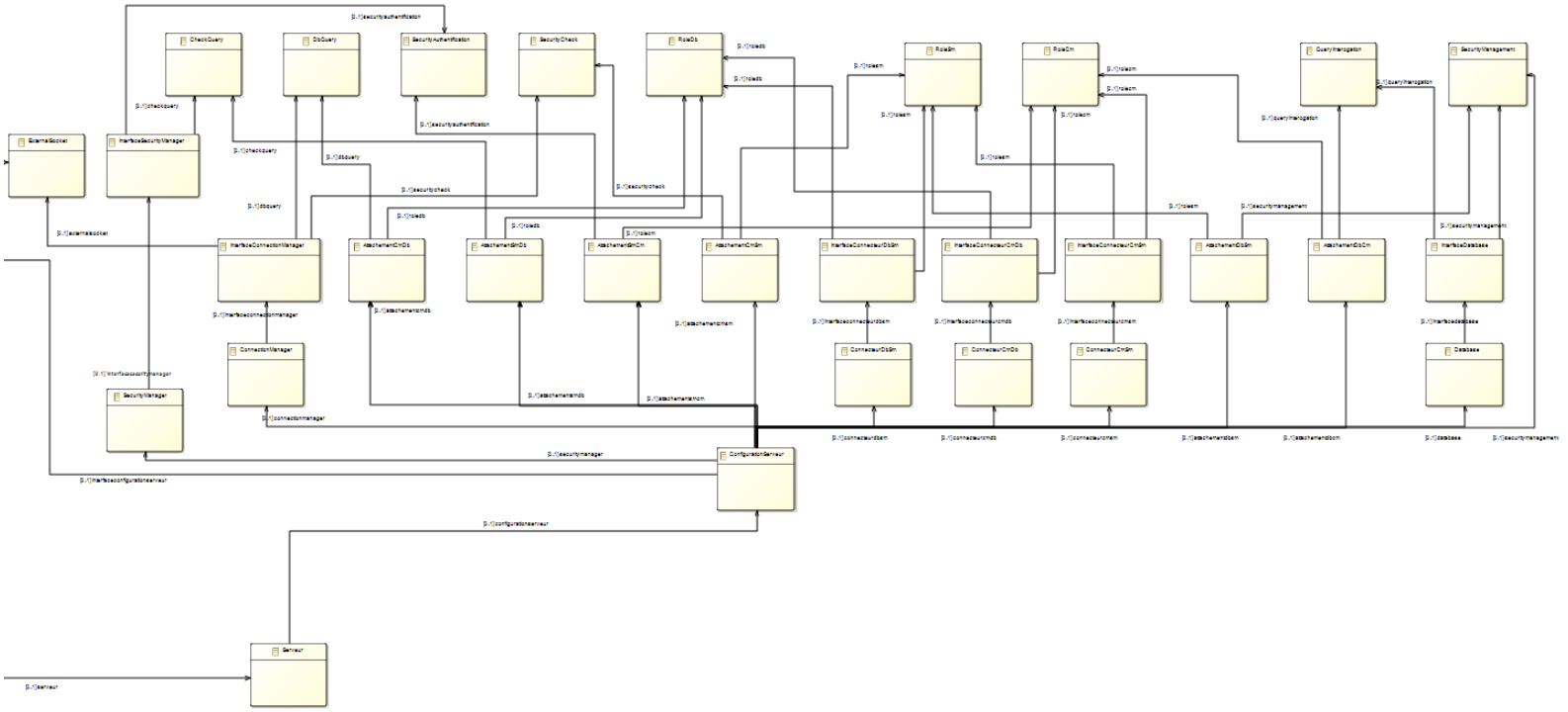


Schéma Configuration Serveur partie gauche:

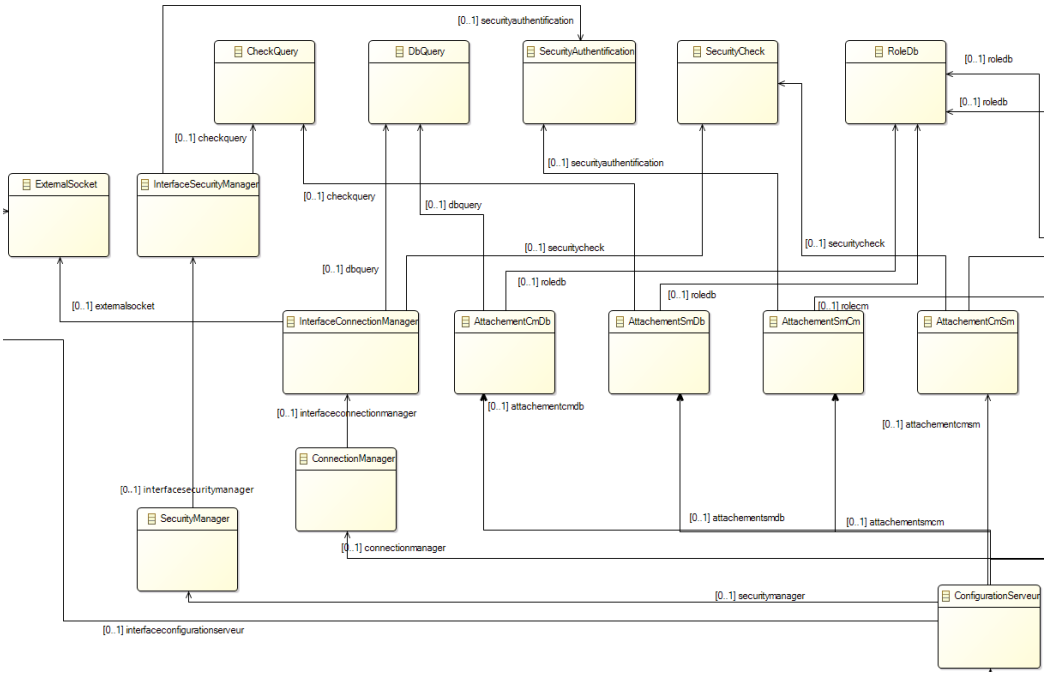
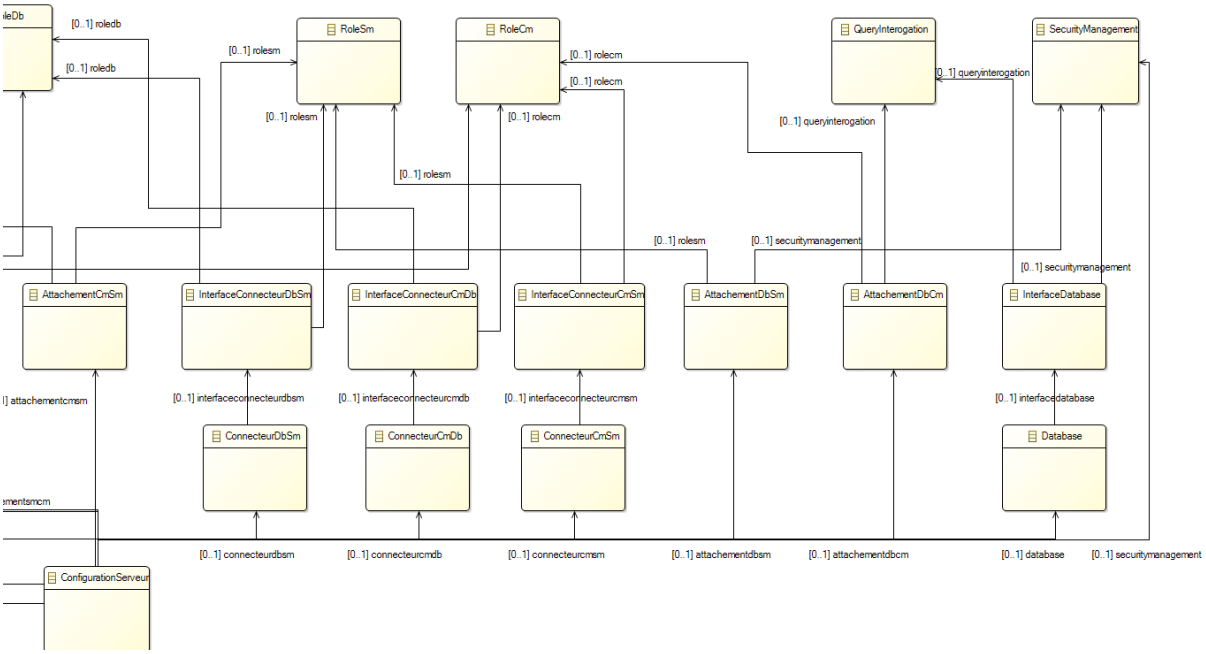


Schéma Configuration Serveur partie droite:



Implémentation M0

Afin de produire le code Java du M0 nous avons utilisé les modèles M1 et M2 générés par EMF. Le méta-modèle M2 engendre des classes métiers, lesquelles sont ensuite utilisées dans le modèle M1. Par conséquent, si le M2 représente un méta-modèle d'architecture et le M1 décrit l'architecture d'un système Client/Serveur, le M0 représente une instanciation spécifique de ce système Client/Serveur. Ainsi, dans le M0, chaque composant du M1 est converti en une classe qui étend un composant du M2.
Voici un exemple de Main afin d'implémenter le Client/Serveur:

```java
package main;
import modelM1.impl.*;

public class main {
	public static void main (String[] args) {
		SystemeClientServeurImpl system = new SystemeClientServeurImpl();


		ClientImpl client = new ClientImpl();
		InterfaceClientImpl interfaceClient = new InterfaceClientImpl();
		PortRequisClientImpl portClient = new PortRequisClientImpl();
		portClient.setValue("valeur");
		interfaceClient.setPortrequisclient(portClient);
		client.setInterfaceclient(interfaceClient);

	   // Creation serveur
		ServeurImpl serveur = new ServeurImpl();
		ConfigurationServeurImpl configServeur = new ConfigurationServeurImpl();
		InterfaceConfigurationServeurImpl interfaceConfigServeur = new InterfaceConfigurationServeurImpl();
		PortFourniConfigurationServeurImpl portFourniConfigServeur = new PortFourniConfigurationServeurImpl();
		interfaceConfigServeur.setPortfourniconfigurationserveur(portFourniConfigServeur);
		configServeur.setInterfaceconfigurationserveur(interfaceConfigServeur);

	   //Creation connecteur RPC
		ConnecteurRPCImpl rpc = new ConnecteurRPCImpl();
		InterfaceConnecteurRPCImpl interfaceRPC = new InterfaceConnecteurRPCImpl();
		RoleFourniImpl roleClient = new RoleFourniImpl();
		RoleRequisImpl roleServeur = new RoleRequisImpl();
		interfaceRPC.roles.add(roleServeur);
		interfaceRPC.roles.add(roleClient);


		// Creation des attachement
		AttachementRPCServeurImpl attachRPCServeur = new AttachementRPCServeurImpl();
		attachRPCServeur.setRoleRequis(roleServeur);
		AttachementClientRPCImpl attachRPCClient = new AttachementClientRPCImpl();
		attachRPCClient.setRoleFourni(roleClient);
		attachRPCClient.setPortRequis(portClient);

	   //Ajout des composant au systÃ¨me
		system.setComposantFeuilleClient(client);
		system.setComposantCompositeServeur(serveur);
		system.setConnecteurrpc(rpc);
		system.setAttachementrpcclient(attachRPCClient);
		system.setAttachementrpcserveur(attachRPCServeur);
	}
}
```

Conclusion

Ce projet visant à définir un langage d'architecture et à l'appliquer pour concevoir une architecture Client/Serveur a suivi une méthodologie bien structurée en deux étapes distinctes. Tout d'abord, nous avons élaboré le méta-modèle M2, qui a établi les bases en définissant la syntaxe, la grammaire et les contraintes des composants du futur modèle. Ensuite, nous avons développé le modèle M1 en conformité avec le méta-modèle M2. Ce dernier représente de manière précise le système Client/Serveur, avec un composant Client, un composant Serveur comprenant les éléments essentiels (database, connection manager et security manager), et un connecteur RPC établissant la liaison entre le Client et le Serveur.

En phase finale, nous avons utilisé un outil de génération de code afin de créer une implémentation M0 du modèle M1, permettant ainsi d'instancier les différents composants du système. Dans l'ensemble, ce travail nous a permis de mettre en pratique nos connaissances en méta modèle et de l’outil EMF pour réaliser une implémentation d’un client/serveur du début à la fin. Nous pouvons cependant noter la complexité lier à l’utilisation d’EMF et à la difficulté de relier les différents modèles entre eux.