Architecture et Styles d'architecture

Rapport de projet COSA

Duclos Romain & Jehanno Clément

Master 2 ALMA
Université de Nantes
12/2018

I - Introduction

Ce rapport présente notre travail pour le projet d'architecture et styles d'architecture autour de COSA. COSA pour Component Oriented Structured Architecture est un langage de modélisation. Il permet de construire des modèles orientés autour d'une architecture par composant. D'un point de vue architectural, il se situe au niveau M2 (meta-modèle).

Dans ce projet, nous devions définir le langage COSA à l'aide de Adoxx ou de Eclipse EMF/ecore. Puis nous devions instancier une architecture Client-Serveur se basant sur COSA (niveau M1). Enfin nous devions proposer une implémentation du Client-Serveur (code, niveau M0).

Dans ce rapport nous allons d'abord présenter notre définition de COSA, puis nous décrirons notre Client-Serveur avant de passer au code. Enfin nous terminerons par une conclusion avec les perspectives d'améliorations.

II - Le meta-modèle COSA

Le meta-modèle COSA permet de décrire des systèmes de Composant. Nous avons plusieurs entités : des composants, des configurations, des connecteurs, des ports, des services, etc. Pour décrire COSA, nous avons décidé d'utiliser Adoxx.

COSA

Le meta-modèle COSA est réparti en différents éléments architecturaux. Dans cette section nous allons revenir sur ces éléments et détailler leurs interactions.

D'après le cours, un **composant** est "*une unité de composition qui spécifie une ou des fonctionnalités. Ils fournissent des services et requièrent des besoins. Indépendamment, Ils peuvent être déployés et composés avec d'autres composants.*"
Ils peuvent donc être n'importe quel élément du système. Ils seront reliés aux autres composants par le biais d'une interface qui mettra à disposition des ports et des services.

Le **connecteur** représente les interactions entre les composants. Nous pourrons dire qu'il connecte deux composants. Cette connexion se fait par le biais d'interfaces qui vont proposer des rôles. Le lien entre les interfaces est fait grâce à la glue.

La relation entre les composants et les connecteurs ne se fait pas directement entre leurs interfaces. Elle passe par des **attachements** qui eux, vont lier les deux interfaces de nos objets.

La **configuration** est l'élément qui va regrouper nos composants et nos connecteurs. Cette configuration aussi possède une interface qui va nous permettre de lier notre configuration au reste du système par le biais de ports (requis et fournis).

Finalement, afin de relier des ports entre eux (entre ceux de la configuration et un composant par exemple), notre configuration contiendra des **bindings**.

Adoxx

Adoxx est un logiciel gratuit permettant de développer ses propres outils de modélisation. Ne connaissant pas cet outil nous avons décidé de l'utiliser dans ce projet afin de le découvrir. De plus Adoxx est en soi plus "puissant" que EMF puisqu'il nous permet de créer notre propre langage de modélisation (tandis que EMF se base sur UML).

COSA dans Adoxx

Adoxx permet dans le Development Toolkit de décrire ses outils de modélisation dans une bibliothèque formée de *Classes* et de *Relations*. Nous pouvons aussi avoir des "*Superclasses*" dont les autres peuvent dépendre (sorte d'héritage en UML/Objet). Nous avons donc défini notre COSA de cette façon :

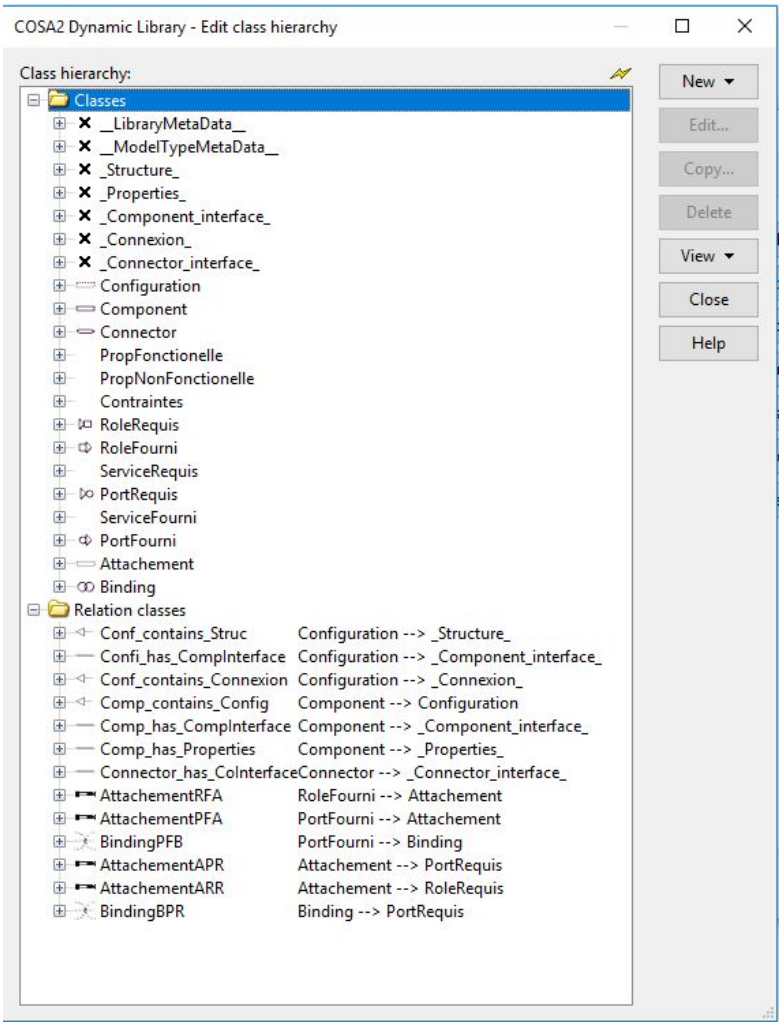


Figure 1 : Les classes et relations de la librairie COSA dans Adoxx

Disponible en annexe : COSA sous la forme d'un diagramme Classes-Relations.

- Les *Classes*

Chaque "élément" de COSA se trouve être une *Classe*. Cela nous a permis de regrouper certains éléments dans des familles (**Structure** par exemple, définie comme abstraite). Nous pouvons alors les relier entre eux à l'aide de *Relations* (les relations ne se faisant qu'entre *Classes*).

Nous avons décidé de représenter les **Attachement** et les **Binding** comme des *Classes.* Bien que ces éléments semblent être de simples liens entre des **Ports**/**Services** et des **Rôles**, nous avons décidé de les représenter comme des éléments à part entiers connus d'une **Configuration** qui les contient. Nous avons préféré nous dire que ces **Attachement** et ces **Binding** soient représentés par des classes bien définies, cela nous semblait plus correct que de les laisser comme de simple lien, laissée à l'interprétation du modeleur.

- Les *Relations*

Enfin, les *Relations* permettent de relier les *Classes* entre elles. Nous avons plusieurs types :

- Les **contains**, pour une **Configuration** qui peut contenir des **Structures** (ou un **Composant** contenant une **Configuration**).
- Les **has** pour relier les interfaces aux **Structures** qui leur sont associées, c'est en fait une "affectation" d' un **Port**/**Role**/**Service** à une **Structure**.
- Enfin les relation liées aux **Attachement** et aux **Binding** permettant de connecter pour de bon des **Ports**/**Rôles**/**Services** entre eux.

Les relations étant définies comme allant d'une *Classe* à une autre, nous avons pu restreindre certaines connexions (interdire qu'un **Rôle** soit connecté à un **Rôle** avec un **Attachement** par exemple). Mais aussi en "généraliser" d'autres (**InterfaceComposant** avec le **Composant**).

- Cardinalités, GraphRep

Adoxx permet de définir des cardinalités dans les classes. Nous avons créé plusieurs restrictions sur certaines classes en lien avec certaines relation. Par exemple, un **Port** ne peut être relié qu'à un seul **Attachement**. Les cardinalités sont visibles dans le diagramme Classes-Relations de COSA disponible en annexe.

GraphRep est tout simplement l'attribut graphique des éléments que nous avons défini.
Adoxx propose un langage pour définir les représentations graphiques. Ici, nous avons fait nos propres dessins en nous inspirant un peu de COSABuilder.

III - Modèle Client-Serveur

Une fois notre bibliothèque opérationnelle, nous pouvons utiliser le Modeling Toolkit pour créer des modèles basés sur ce le model type (ici COSA) que nous venons de créer. dans ce projet, nous devions faire une instanciation de COSA au travers d'un système Client-Serveur.

Décrit en COSA, le Client-Serveur donne ceci :

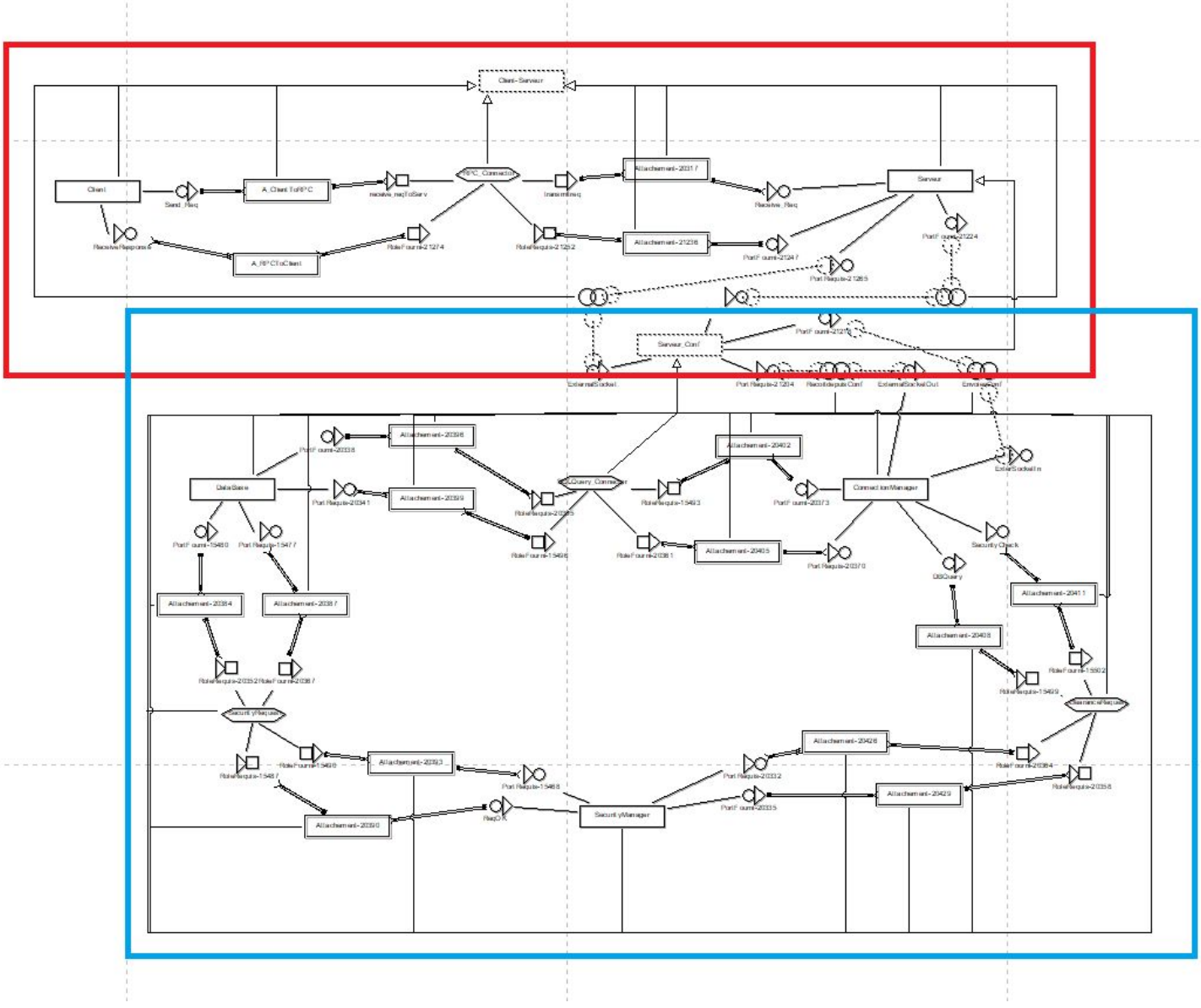


Figure 2 : Le client-serveur en COSA

Nous distinguons deux parties, la configuration client-serveur (en rouge) et la configuration du serveur (en bleu). Revenons sur ces deux parties.

Configuration Client-Serveur

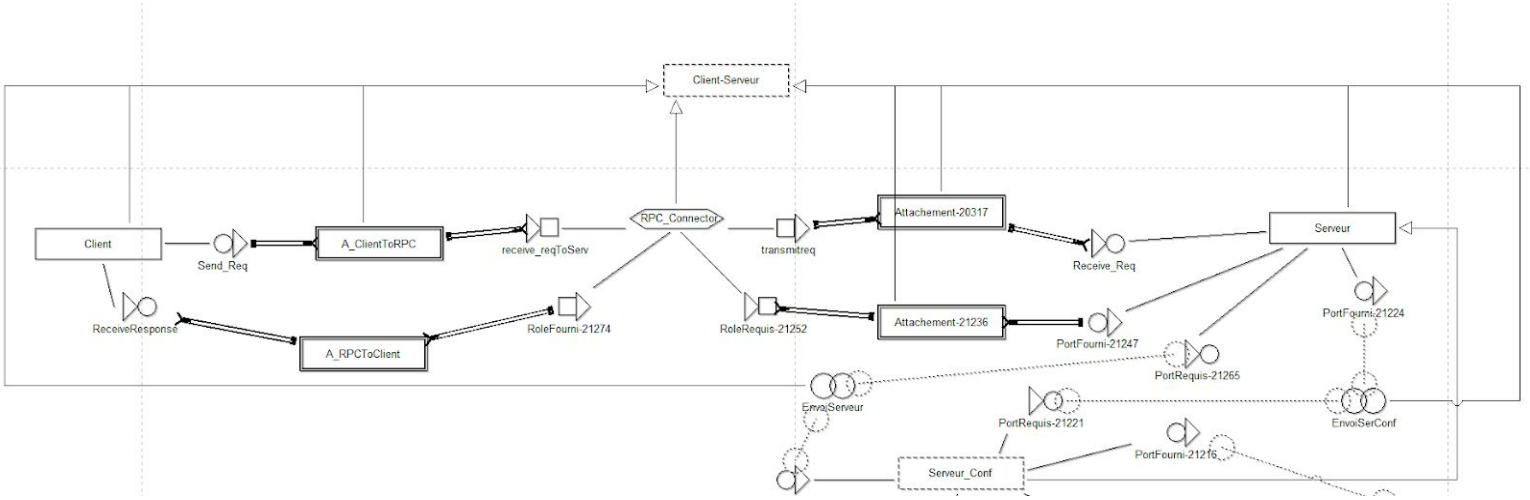


Figure 3 : Configuration Client-Serveur sur notre modèle Adoxx

Dans cette section nous modélisons l'interaction entre un client et un serveur.

Le Client est relié au Serveur via un RPC Connecteur. Chaque liaison est faite par des attachements sur les bons ports et rôles respectifs. Notons bien que chaque composant a deux liens (le port fourni et le port requis) puisque nous voulons avoir des entrées et des sorties.
Le connecteur quand à lui possède 4 rôles pour faire le lien entre les deux composants.
Le serveur est relié à sa configuration par des bindings (pour l'entrée d'un message et la sortie d'un message).

Configuration Serveur

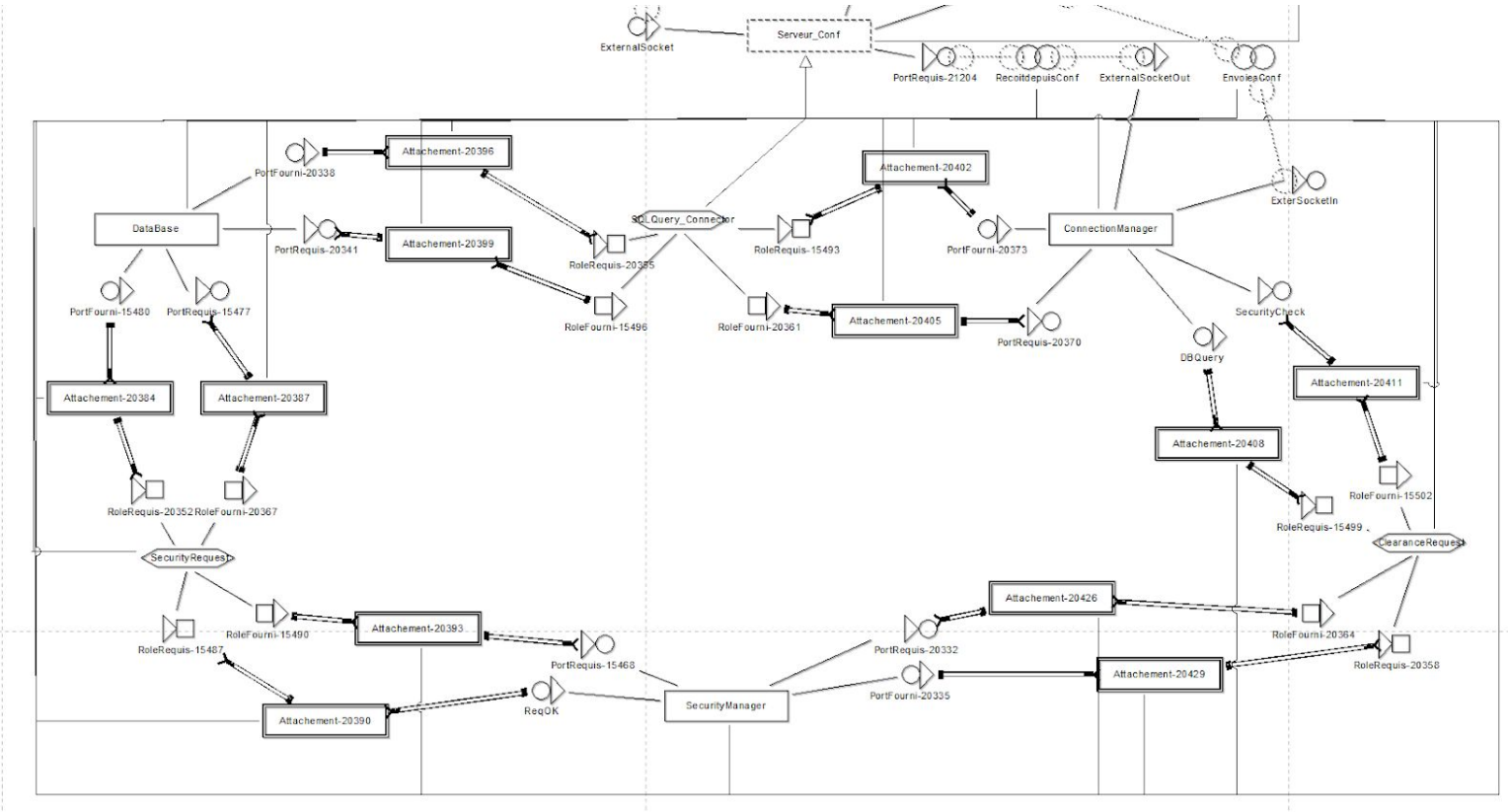


Figure 4 : Configuration Serveur sur notre modèle Adoxx

La configuration du serveur renferme toute la complexité du modèle. C'est ici que nous allons traiter la requête du client.
Prenons un peu d'abstraction sur ce modèle complexe afin de le voir plus simplement.
Il faut remarquer 3 choses : notre base de données, notre gestionnaire de connexion ainsi qu'un gestionnaire de sécurité. Chacun de ces composants est relié à l'autre via le même système que sur la configuration client-serveur :
composant <-> attachement <-> connecteur <-> attachement <-> composant

Bien qu'elle semble complexe, la configuration est plutôt simple. Chaque composant a un port fourni et un port requis, que nous allons relier au bon rôle du bon connecteur, par le biais d'un attachement.
Pour conclure sur cette configuration, nous avons besoin de bindings afin de lier l'entrée de la configuration à notre gestionnaire de connexion. C'est lui qui va s'occuper de recevoir la requête et de la gérer correctement

IV - Client-Serveur en Java

Adoxx ne permettant pas de créer du code à partir d'un modèle nous avons dû l'implémenter nous-même. Nous avons décidé de l'implémenter en Java. En effet, nous avons préféré utiliser un langage objet de haut niveau pour instancier le Client-Serveur étant donné que COSA a été créé sous la forme de Classes/Relations. Cela nous semblait plus naturel de repasser par des classes.

Comme nous partons de rien, il ne sera pas nécessaire pour nous de créer des classes abstraites "composant", "connecteur", etc. Ce qui va nous permettre de grandement simplifier notre code. Attention, ce n'est pas pour autant que notre code sera fait n'importe comment puisque celui-ci doit absolument rester conforme au modèle.

Pour ce faire nous avons donc pris chaque composant de COSA comme une classe à part entière. Les éléments binding, attachement, rôle et port sont directement des classes. En fait leur fonctionnement est toujours le même, le code ne change pas d'un port à l'autre. Ainsi, nous avons créé une classe pour chacun de ces éléments. Puis nous les composons dans les autres éléments (connecteur, composant,...) part des attributs nommés (port1, port2 du type Port par exemple).
A l'inverse, les connecteurs, les configurations et les composant ont pour nous des buts et des comportements bien précis, donc nous avons décidé d'en faire des classes "nommées" à part entière (un Client, un Serveur, etc.)

Afin de préserver le concept des boîtes noires, nous ne voulions pas que nos ports connaissent d'autres éléments de la configuration. Nous avons dû penser notre code de sorte à ce qu'il respecte le modèle et notre principe des boîtes noires. Chaque port contient donc un buffer qui va contenir le message qu'il doit transmettre. Nos ports sont connus par les attachements, c'est eux qui vont faire passer le message. L'attachement va donc prendre un message sur un port fourni pour l'envoyer vers un rôle requis (ou l'inverse suivant l'ordre dans lequel on est). Il va donc prendre le message dans le buffer de la source pour le mettre dans le buffer de la cible.

Cette transition est effectuée via une méthode *transmettre()*. Pour le bon déroulement du programme il est fondamental que la configuration soit omnisciente et donc qu'elle connaisse tous ses éléments. Comme notre COSA est conçu ainsi, c'est parfait. La configuration va donc nous permettre de relier tous nos éléments entre eux dans le bon ordre. Par la suite, le client pourra envoyer un message. Ici, nous utilisons le desing pattern Observer. Suite à ce message, le client va notifier l'observeur. Notre client ne connaît rien d'autre que l'observeur (et ses ports, évidemment) il peut donc mettre un message dans son port de sortie et dire à l'observeur "J'ai un message, traite le moi". L'observeur va ensuite se charger d'acheminer le message à bon port.

Voici un schéma UML pour résumer nos explications :

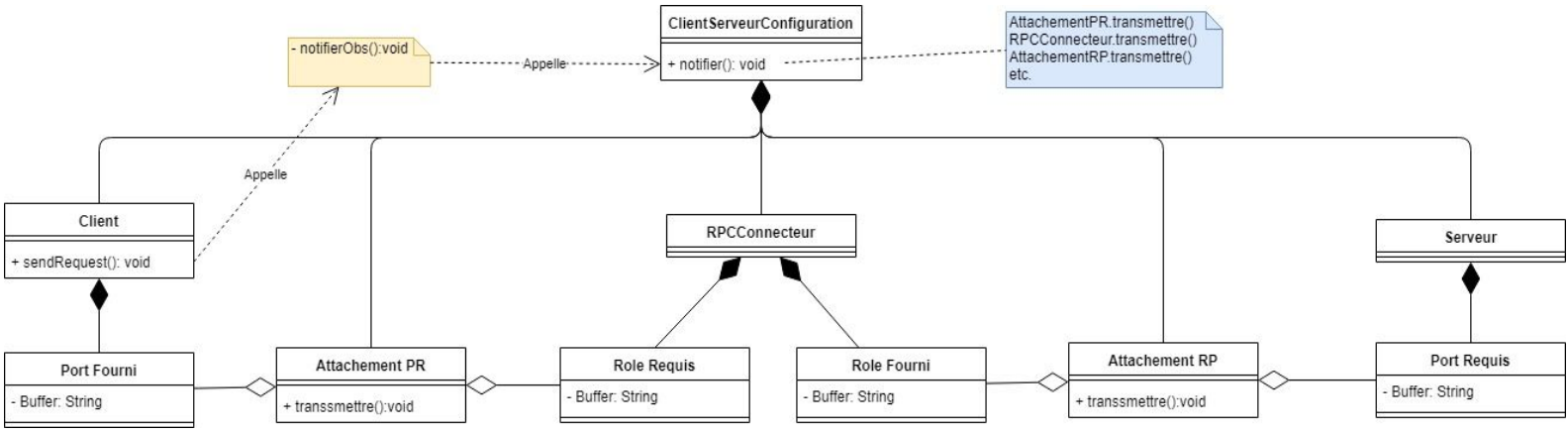


Figure 5 : UML du client-serveur en Java

V - Conclusion

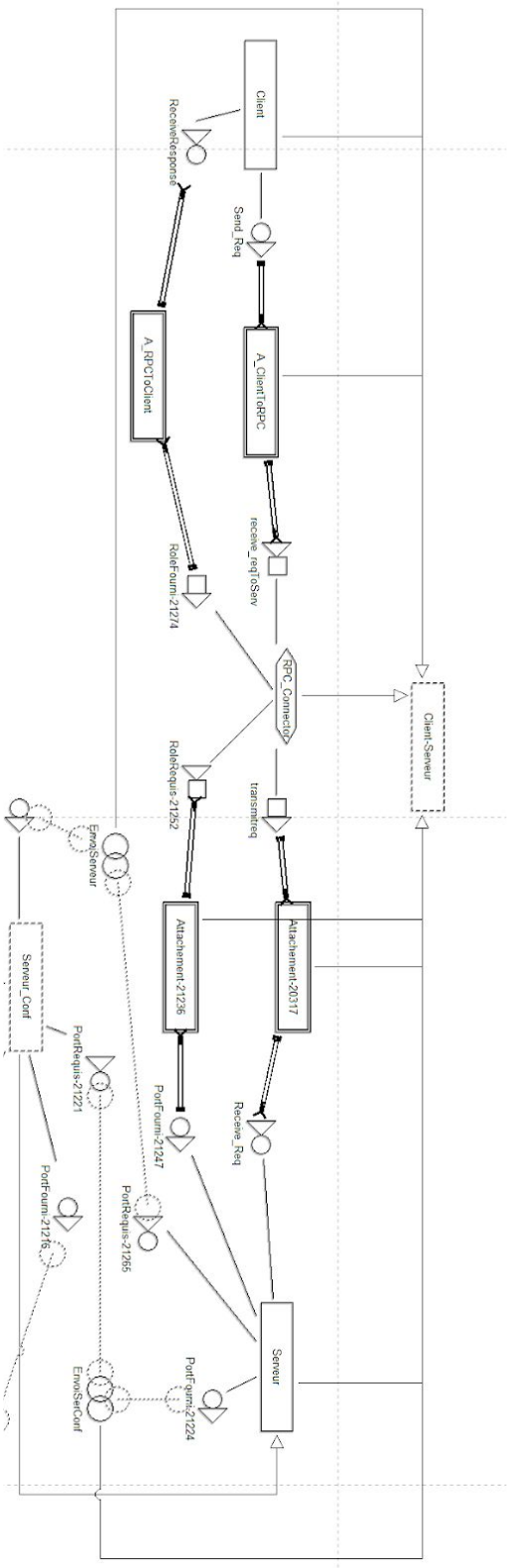
Dans ce rapport nous avons résumé notre approche pour implémenter un modèle client serveur en Java en partant de la définition d'un Méta-modèle jusqu'à l'implémentation du code en passant par la réalisation du modèle.

Adoxx est un outil très puissant puisqu'il se place à un niveau d'abstraction très haut, en effet il nous permet de définir des meta modèles. Cependant nous noterons qu'il n'existe pas encore de moyen de générer du code à partir de celui-ci ce qui nous pousse donc soit à le faire, soit à migrer vers EMF.

Nous avons eu beaucoup de difficulté à prendre Adoxx en main ce qui nous a donc laissé moins de temps pour créer notre implémentation du client serveur.
Toutefois, dès lors que nous avons compris Adoxx et le meta-modèle COSA nous avons été beaucoup plus rapides que si nous étions passés par EMF. Nous pensons toutefois que des améliorations seraient possibles voire nécessaires. Notre COSA peut souffrir de quelques lacunes. Notamment au niveau du code, il se peut que certains choix d'implémentation ne soient pas optimisés. Nous retiendrons tout de même qu'une fois la prise en main d' Adoxx faites, nous avons pu créer notre modèle Client-Serveur et le code extrêmement rapidement et simplement ce qui est un plus.

Annexe

Configuration Client-Serveur



Configuration Serveur

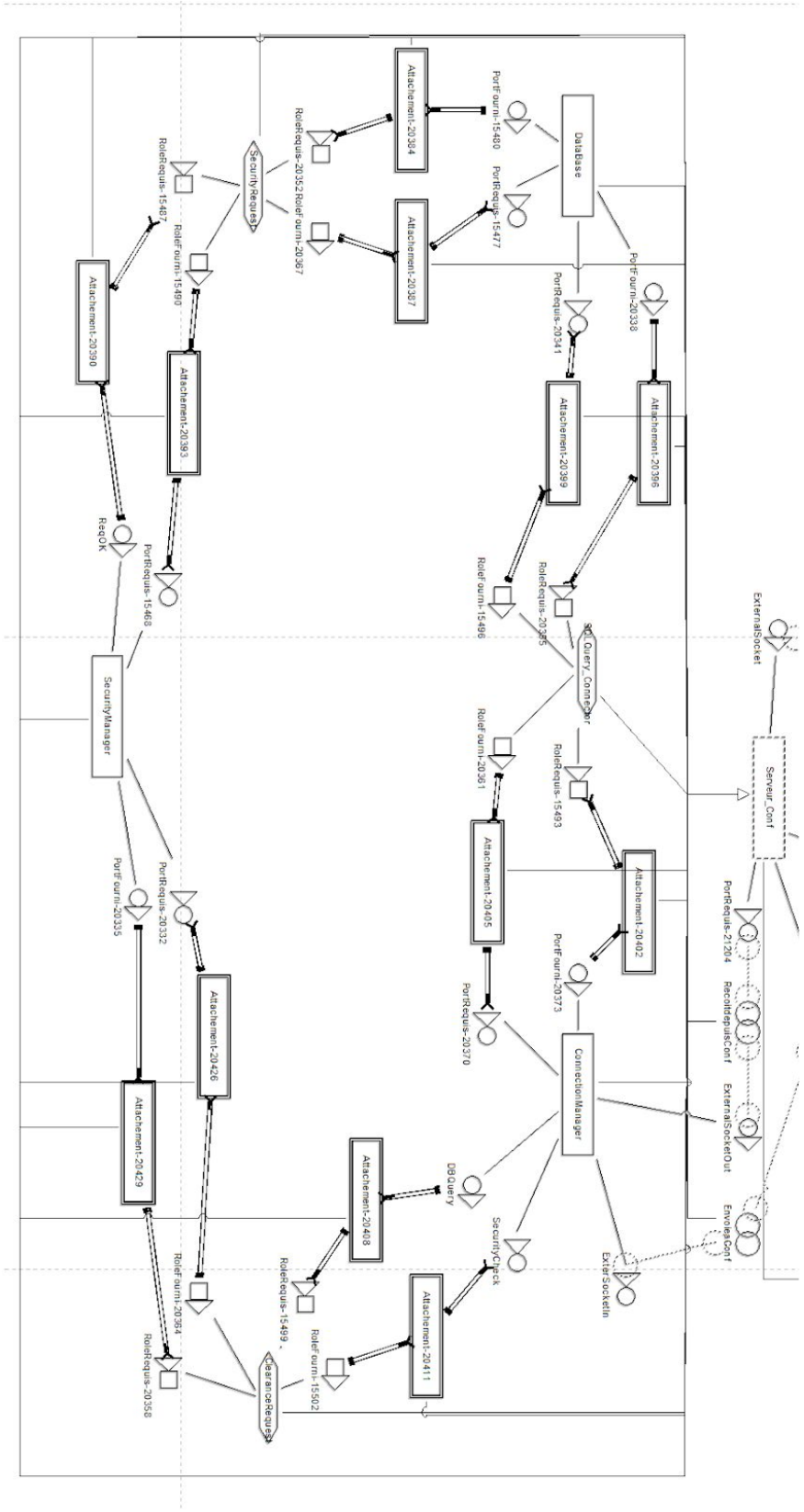


Diagramme Classe-Relation de COSA