Rapport

Architectures et Styles d'Architectures (X3IA080)

AHMAT issa Izzedine, GERARD Kylian

Nantes Université

Sommaire

Introduction 2
I – Spécifications 2
A) Le méta-modèle ASA (M2) 2
1 – La Configuration 3
2 – Le Component 3
3 – Les Interfaces 4
4 – Les Ports et les Roles 4
5 – Le Connector 5
6 – La Glue 5
B) Le modèle Client-Serveur (M1) 6
1 – Le Client 7
2 – Le RPC 7
3 – Le Server 7
II – Implémentation 8
A) Le méta-modèle ASA (M2) 8
B) Le modèle Client-Serveur (M1) 8
III – Conclusion 8

Introduction

Le projet d'Architectures et de Style d'Architectures (ASA), est un projet de conception et d'implémentation des trois premiers niveaux d'abstraction d'un système, à savoir l'instance du système (M0), le modèle (M1), et le méta-modèle (M2). Pour cela, nous nous sommes orientés vers une architecture C&C (Composants et Connecteurs), et la création d'un système de Client-Server.

Nous avons, dans un premier temps, spécifié notre langage C&C grâce à un méta-modèle pour définir les concepts de ce langage, à savoir le Component, le Connector, la Configuration, les Ports… Ce méta-modèle définit la syntaxe, la grammaire et les contraintes des composants de notre futur modèle.

Dans un second temps, nous avons mis au point un modèle respectant le méta-modèle précédemment défini, qui modélise le système Client-Server présenté dans le sujet : C'est-à-dire un composant Client, un composant Server constitué de 3 composants Connection Manager, Security Manager et Database, et un connecteur RPC liant le Client et le Server.
Nous avons choisi que notre système Client-Server modéliserait un système de banque, permettant à un utilisateur de créer un compte, de déposer et de retirer de l'argent.

Enfin, nous avons implémenté ces différents modèles, afin de pouvoir instancier un système Client-Server et de pouvoir interagir avec, en imaginant un cas d'utilisation simple.

I – Spécifications

A) Le méta-modèle ASA (M2)

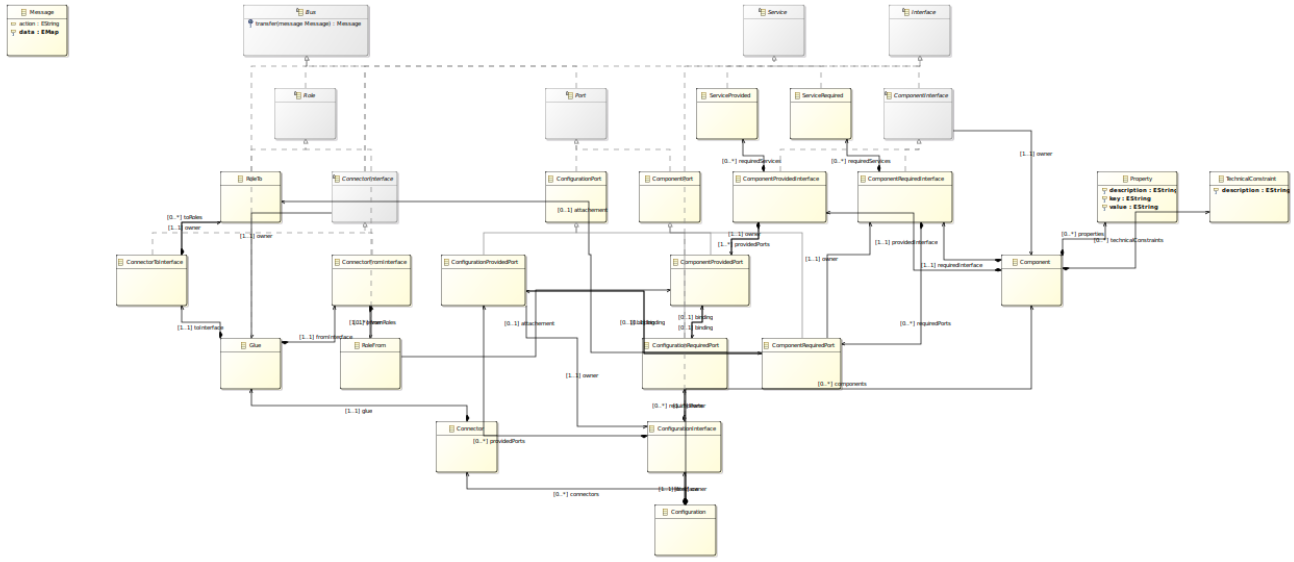

Figure 1: Représentation graphique du modèle Ecore modélisant le méta-modèle de ASA

Pour la spécification du méta-modèle ASA, nous avons choisi d'utiliser Ecore. Ce méta-modèle contient 7 composants de bases, que sont la Configuration, le Component, les Interfaces, les Ports, le Connector, la Glue et les Roles. Nous allons détailler les concepts, et les spécificités associés à chacun de ces composants. Dans une optique de visibilité, les diagrammes suivants ont été épurés de certains détails pour se concentrer sur les concepts centraux.

1 – La Configuration

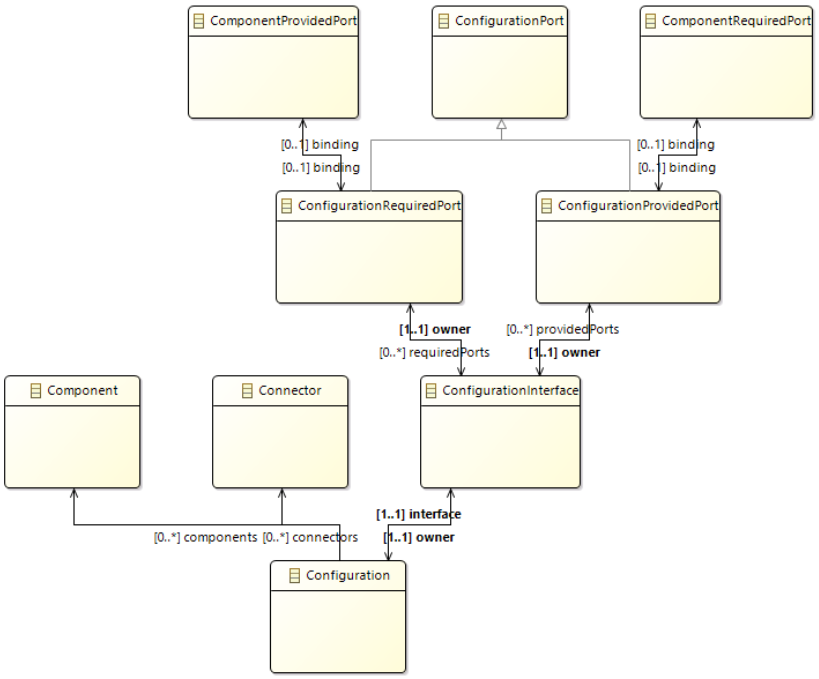

Figure 2: Représentation graphique du modèle Ecore modélisant la Configuration du méta-modèle de ASA

La configuration est le point central du modèle. En effet, une architecture possède au moins une configuration. Une configuration peut être vue comme un graphe de composants et de connecteurs relier entre eux par des liens d'attachement, accessible via une interface jouant le rôle de façade. Cela permet de faire interagir plusieurs configurations entre elles, ou de faire interagir un composant avec une configuration, comme ça sera le cas dans ce projet.

2 – Le Component

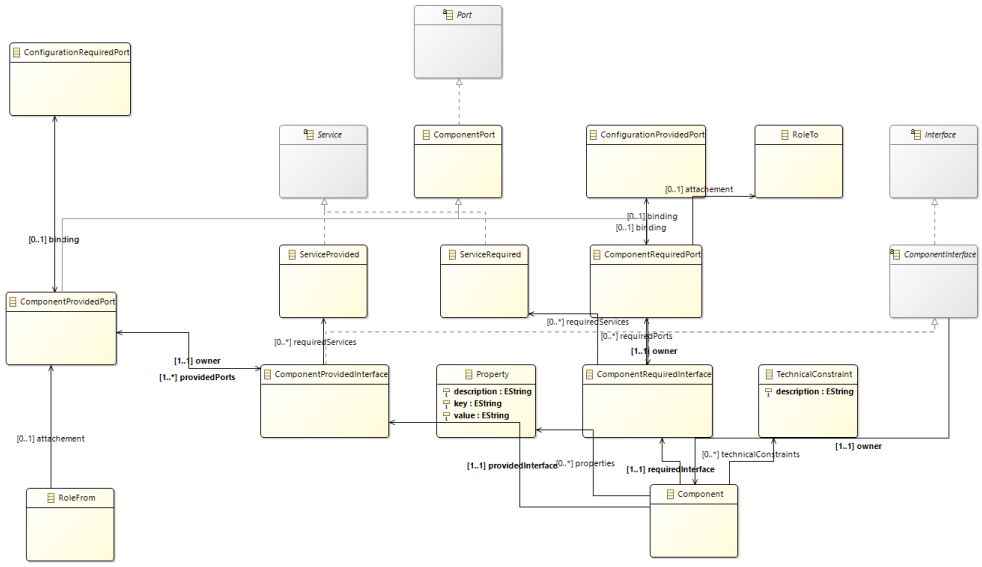

Figure 3: Représentation graphique du modèle Ecore modélisant le Component du méta-modèle de ASA

Dans l'architecture C&C, un composant est une entité autonome, possédant ses propres caractéristiques et contraintes techniques, qui encapsule une partie spécifique du système. Il expose une interface claire décrivant les services qu'il offre, et peut exposer une interface décrivant les services dont il a besoin pour fonctionner, et que les autres composants doivent lui fournir. Cela permet de favoriser leurs indépendances et leurs réutilisations. Les composants interagissent entre eux via des

connecteurs rattachés aux ports des composants. Cela permet la mise en place de mécanismes de communication ou de coordination, permettant à ces composants de collaborer pour former le système global.

3 – Les Interfaces

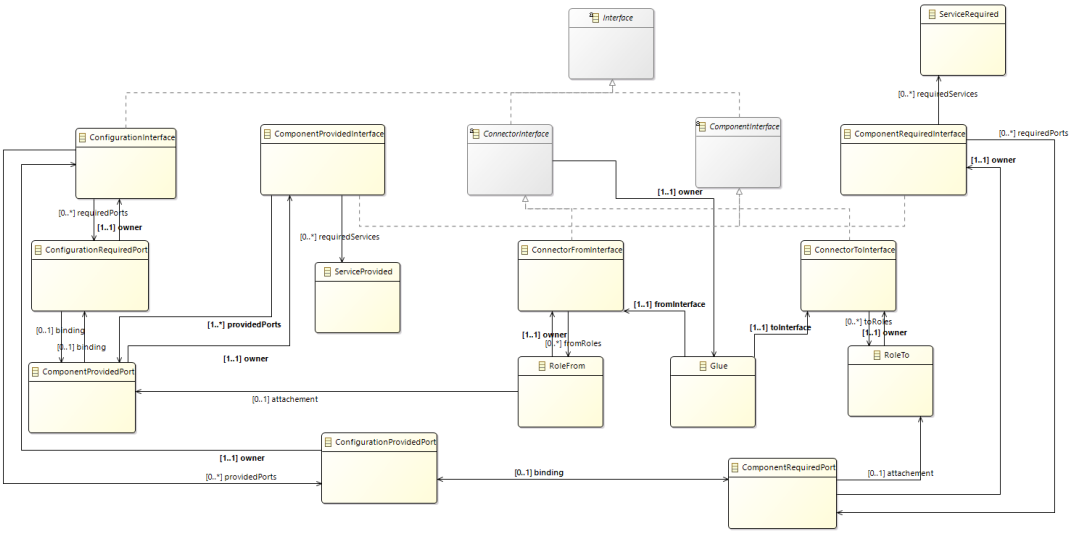

Figure 4: Représentation graphique du modèle Ecore modélisant les Interfaces du méta-modèle de ASA

Dans notre langage ASA, tous les objets autonomes possèdent des Interfaces. Elles sont les portes d'entrer et de sorties de ces objets. En effet, elles sont composées de Ports ou de Roles (pour les Connectors), qui permet aux objets de se lier entre eux, via des liens d'attachement ou de binding, sans avoir besoin d'informations sur l'objet lié, sur son fonctionnement, son implémentation… Ces Interfaces permettent donc l'indépendance, et la réutilisation de tous ces objets.
Cependant, contrairement aux Interfaces fournis par les Connectors, qui font plutôt office de zone de transit, les interfaces des Components et des Configurations jouent aussi un vrai rôle dans le fonctionnement de ces objets. En effet, ces Interfaces décrivent les services que l'objet peut fournir aux autres, mais aussi les services dont il a besoin afin de fonctionner.

4 – Les Ports et les Roles

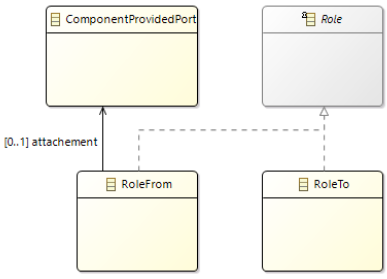

Figure 5: Représentation graphique du modèle Ecore modélisant les Roles du méta-modèle de ASA

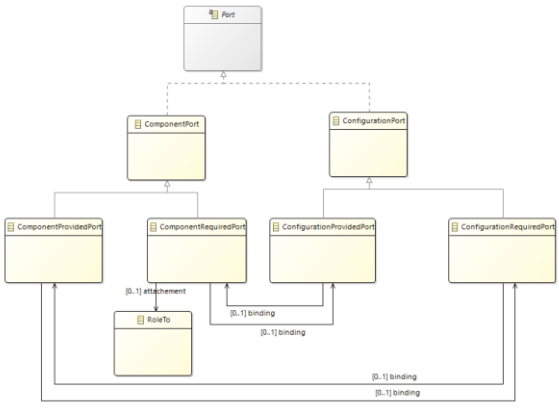

Figure 6: Représentation graphique du modèle Ecore modélisant les Ports du méta-modèle de ASA

Les Ports et les Roles sont des éléments essentiels de notre architecture. Ce sont les points d'accès aux différents composants. C'est par eux que sont connectées les Components, les Connectors ou les Configurations. Ils permettent de protéger les composants, en ne leur permettant pas de communiquer directement. Ces liaisons créent entre les Ports peuvent être de deux types. Tout d'abord, nous l'avons les Attachements, qui sont des interactions entre le port fourni d'un composant et le rôle requis d'un

connecteur, ou une interaction entre le port requis d'un composant et le rôle fourni d'un connecteur. Nous avons ensuite le Binding. Cette interaction se produit entre un composant et le Port d'une Configuration. L'interface de la configuration transmet ensuite l'information reçue sur le port au bon composant.

5 – Le Connector

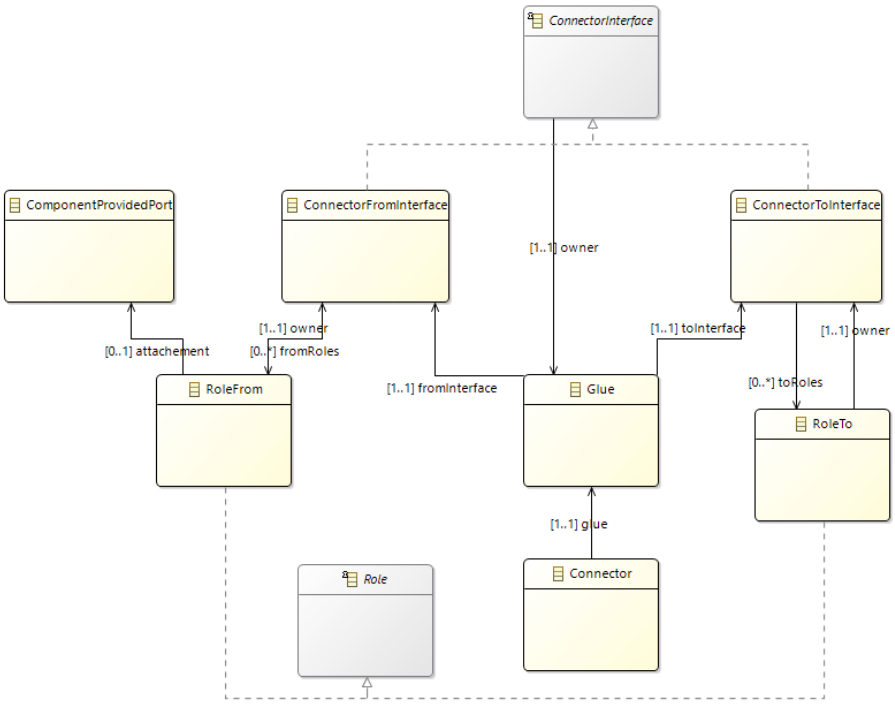

Figure 7: Représentation graphique du modèle Ecore modélisant le Connector du méta-modèle de ASA

Les Connectors sont les éléments qui permettent de mettre en relation et de coordonnées les différentes fonctionnalités offertes par tous les Components du système. Ils permettent de rendre le couplage entre les composants faible. C'est-à-dire que deux Components de système sont interconnectés d'une manière telle qu'ils n'ont pas besoin de connaître les détails internes de l'autre. Ces Connectors possèdent une Interface From et une Interface To, composée de Roles. Ces Roles ont la même fonction que les Ports pour les Components, ce sont les points d'entrés et de sorties du Connector.
Une fois entrée dans le Connector, l'information va ensuite transiter par la Glue, qui va permettre de rediriger les signaux, dans les bons Ports, entre l'entrée et la sortie du Connector.

6 – La Glue

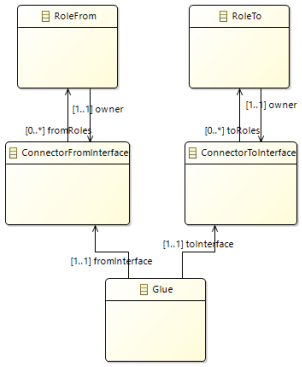

Figure 8: Représentation graphique du modèle Ecore modélisant la Glue du méta-modèle de ASA

La Glue joue un rôle crucial en assurant la cohésion et l'intégration des composants, agissant comme une sorte de substance adhésive qui maintient l'ensemble du système fonctionnel. Elle peut prendre la

forme de services, de protocoles de communication, ou de middleware, fournissant ainsi un mécanisme standardisé pour que les composants puissent échanger des informations et coopérer de manière harmonieuse.

B) Le modèle Client-Serveur (M1)

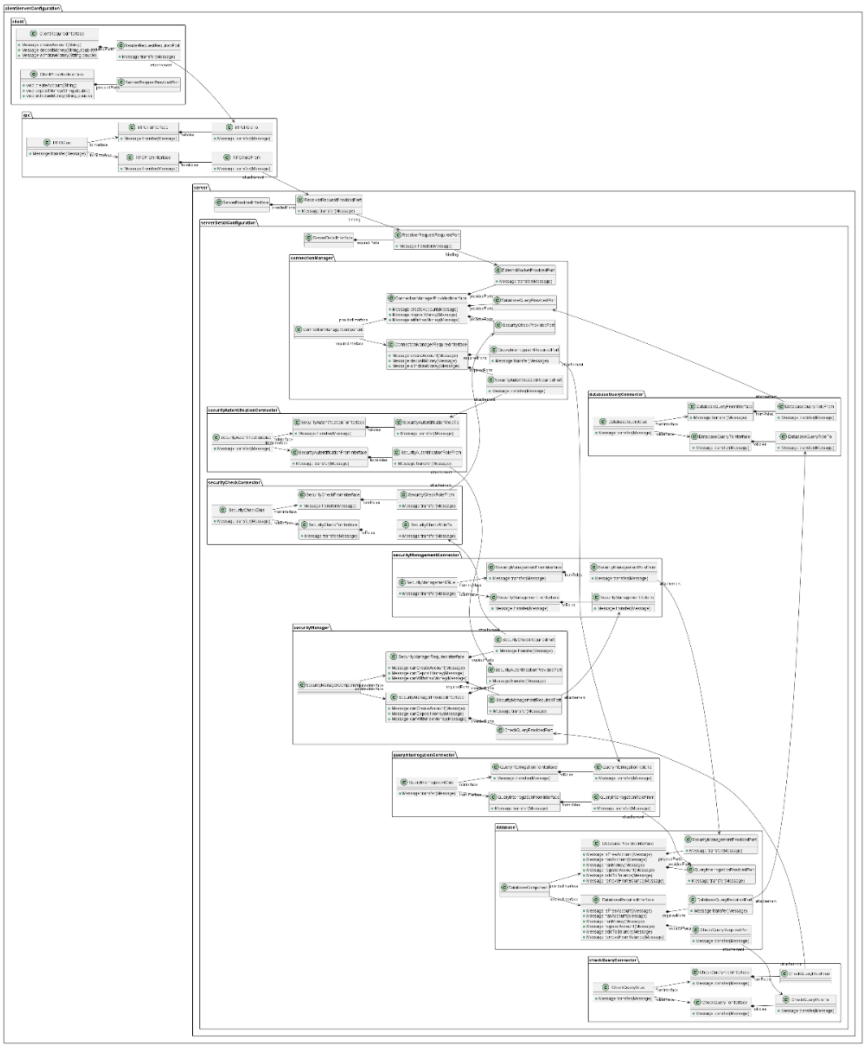

Figure 9: Diagramme de classe du modèle Client-Server

Pour la spécification du modèle Client-Server, nous avons choisi d'utiliser un diagramme de classe UML. Comme précisé dans le sujet, notre modèle contient 3 composants principaux : le Client, le RPC et le Server. Le Server lui est en réalité un Component composite contenant une configuration modélisant le Server de manière plus détaillé et complexe. Cette configuration est constituée d'un Component Connection Manager, d'un Component Security Manager et d'un Component Database relié entre eux de manière bidirectionnelle. Nous allons par la suite détailler la modélisation de chacun de ces trois composants.
Dans un souci de lisibilité, les classes de Configurations, de Components et de Connectors ont été remplacé par des packages contenant les classes liées à ces objets.

1 – Le Client

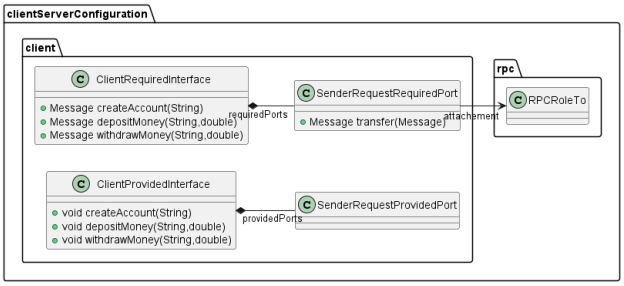

Figure 10: Diagramme de classe modélisant le Client du modèle Client-Server

Le Component Client est relativement simple. Il fournit une Interface permettant de créer un compte, de déposer, et de retirer de l'argent. Pour fonctionner, il a donc besoin d'un système permettant de stocker et de récupérer les données, mais aussi de vérifier la validité des opérations. Pour ce faire, il expose donc une Interface requise associé à ces Services. Ces Services seront fournis par le Server, nous avons donc besoin de connecter ces deux éléments à l'aide d'un Connector, le RPC.

2 – Le RPC

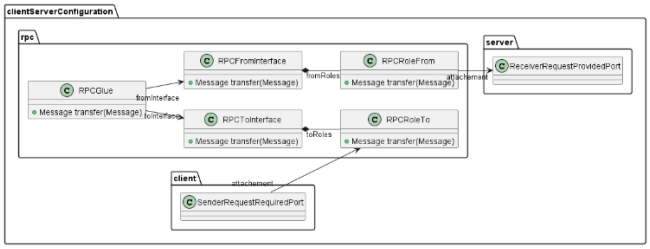

Figure 11: Diagramme de classe modélisant le RPC du modèle Client-Server

Le RPC est donc le Connector liant notre Client et notre Server. C'est par lui que les informations et les données vont transiter lors d'une action effectuée par le Client. Notre système étant relativement simple et linéaire, aucune action particulière ne sera confiée au Connector. Il se contentera de se connecter entre le port requis du Client, de récupérer les informations dans son Role To, de transférer les données vers la Glue, qui les redirigera vers le Role From, qui pourra ensuite faire sortir l'information du Connector en l'envoyant dans le Port fournis du Server

3 – Le Server

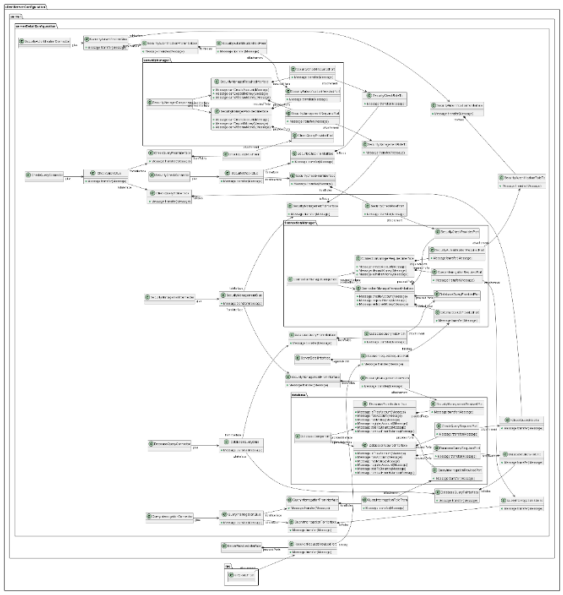

Figure 12: Diagramme de classe modélisant le Server du modèle Client-Server

Le Server est le Component le plus complexe du système. En effet, c'est un Component composite constitué d'une Configuration. L'Interface du Server joue le rôle de façade, mais le traitement sera en réalité délégué à cette Configuration interne. Cette Configuration est composée de trois Component : le Connection Manager, qui est le Composant de façade à ce sous-système, le Security Manager, qui s'occupe de la vérification des demandes du Connection Manager, et la Database qui s'occupe de stocker et de fournir les données. Chaque Component est lié entre eux de manière bidirectionnelle via des liens d'Attachement. Cependant, dans notre cas, nous n'utiliserons pas tous ces liens. Seuls les liens Security Check et Check Query, permettant de valider la validité d'une action, et DB Query permettant de stocker le résultat d'une action, seront mis à profit.

II – Implémentation

A) Le méta-modèle ASA (M2)

Le méta-modèle ASA, ayant été défini à l'aide de Ecore, nous avons décidé d'utiliser la puissance des outils fournis par EMF, à savoir la génération automatique de code. Cette approche s'est avérée extrêmement avantageuse, permettant des gains substantiels en termes de temps lors de l'implémentation du méta-modèle. Cependant, la génération automatique de code nécessitant une grande abstraction, ce qui peut rendre le code par moments excessivement complexe et peu lisible. Un défi supplémentaire réside dans l'incapacité de spécifier des comportements spécifiques lors de la création des divers objets, tels que l'association d'un port à son interface. Ces manipulations nécessitent une intervention manuelle de la part du développeur, introduisant ainsi un potentiel terrain propice aux erreurs. Cette exigence de manipulation manuelle peut entraîner une complexité accrue dans le processus de développement, tout en soulevant des préoccupations quant à la maintenabilité et à la fiabilité du code résultant.

B) Le modèle Client-Serveur (M1)

L'implémentation du modèle Client-Server a été réalisée de manière plus classique. Nous avons ainsi pu faire de véritables choix d'implémentation. Tout d'abord, nous avons décidé de ne pas créer de classe à part entière pour les services, et de les implémenter directement comme des méthodes de nos Interfaces. En effet, le système étant relativement simple, et les interfaces offrant peu de service, nous avons trouvé pertinent de réduire le nombre de classe, de faciliter et de simplifier l'accès aux opérations fournis par les Interfaces.
Nous avons aussi opté pour une instanciation statique de tous nos composants. Bien que cela fige en quelque sorte l'instanciation du système, cette approche nous a permis de simplifier la construction globale du système, notamment pour la liaison des différents éléments. En effet, du fait des connecteurs, des composants n'appartenant pas aux mêmes structures doivent être interconnecté. Cela aurait impliqué de faire transiter tous ces composants à travers les différents constructeurs ou méthode.

III – Conclusion

Ce projet nous a permis de voir une autre approche dans le développement logiciel que celle de la programmation orienté objet. Nous avons pu à travers cela découvrir un nouveau style d'architecture, à savoir le C&C. De plus, nous ce projet nous as permis de faire un lien avec le module d'ingénierie dirigé par les modèles, et de continué à expérimenter autour des concepts de modélisation, méta-modélisation et génération de code.
Cependant, nous avons trouvé que le choix d'utiliser JAVA pour la définition et l'implémentation de notre langage, n'était pas particulièrement pertinent. Cela rend la visualisation globale du système très complexe. Cette impression est aussi possiblement due à notre implémentation se voulant peut-être trop fidèle à notre méta-modèle. Par exemple le fait d'avoir créé des Connectors composé d'Interfaces, de Roles, relié à des Components via les Ports définis dans des Interfaces, au lieu de plus simplement créer une référence, via un attribut, entre l'Interface d'un Component vers celle d'un autre Component.
De plus, nous avons éprouvé des difficultés à comprendre l'intérêt d'utiliser des Connectors pour lier des Components entre eux, au lieu de directement relier leurs Ports. En effet, dans notre cas très simple, les Connectors se comportent globalement comme des passe-plats, et n'ont pas vraiment d'intérêt. Cela rajoute de la complexité au système globale. Cependant, il semblerait que ces Connectors puissent avoir différents niveaux de difficulté, allant de l'appel de méthode, à de l'ordonnancement (Lydie du Bousquet), à de la gestion d'erreur ou à de la médiation des communications…