Projet HADL

Anthony Caillaud Manoël Fortun

4 décembre 2010

Table des matières

1 Introduction

Dans le cadre de notre module Composants dans lequel nous avons pu étudier les modèles à composants et leurs fonctionnements, nous avons du modéliser et écrire notre propre langage d'architecture logicielle à composants. Ce projet ne consistait pas seulement en la définition d'un langage d'architecture à composants, mais aussi en l'implémentation en langage objet d'un moteur d'architecture à composants. Ce rapport présente donc le travail que nous avons pu effectuer sur cet HADL(home architecture definition langage). Dans un premier temps, le travail de modélisation des concepts composants (meta-modélisation) ainsi que l'implémentation du modèle vous sera présenté. Ensuite, l'application du modèle à composants (modélisation) vous sera expliquée à travers l'exemple du système Client/Serveur. Enfin, nous détaillerons le meta-meta-modèle que nous avons pu définir pour notre HADL.

2 Description du méta-modèle HADL

2.1 Définitions

L'architecture logicielle à base de composants est consitutés de trois principaux éléments : des composants, des configurations et des connecteurs.

2.2 Eléments principaux

Un composant est une unité de composition qui spécifie une ou plusieurs fonctionnalités. Celui-ci peut être déployé indépendemment, il s'agit de la brique élémentaire.

Une configuration est un ensemble de composants et de connecteurs, il définit la façon dont ils sont reliés entre eux. Cette unité de l'architecture logicielle à base de composants est nécessaire aux bonnes liaisons entre composants. Elle permet de savoir si ces liaisons relatives aux connecteurs permettent une communication correcte. C'est la configuration qui gère tous les composants et tous les connecteurs qui la composent.

Un connecteur explicite est une unité effectuant une liaison entre des composants ou des configurations. Les connections proprement dites s'effectuent grâce à des Rôles (Rôles From et Rôles To) et une "Glue" qui représente la fonction modifiant les informations transférées. Il peut être comparé à un adaptateur entre deux composants.

2.3 Eléments secondaires

L'interface d'un composant, connecteur ou configuration est l'ensemble des éléments fournis pour l'intéraction avec le composant, connecteur ou configuration. Il s'agit plus d'un concept que d'un élément concret. Une interface est généralement constituée de ports et de services.

Un port est un point d'entrée ou de sortie d'un composant ou configuration, il pourrait s'apparenter à un attribut d'une classe si on fait le rapprochement entre le modèle composant et le modèle objet.

Un service est une opération disponible auprès d'un composant, il y a des services fournis par un composant et des services requis pour son fonctionnement. Un service peut être assimilé à une méthode toujours dans le parallèle composant/objet. Un service peut être relié à des ports ce qui signifie que certains ports seraient les entrées du service et d'autres ports seraient la ou les sorties du service.

Les propriétés sont des éléments appartenant aux configurations ou composants, les propriétés peuvent définir une particulatité du comportement ou un élément dont on peut se servir, comme un attribut de classe.

Les contraintes sont des éléments importants, puisque elles définissent ce dont ont besoin les composants ou configurations pour fonctionner, par exemple la présence d'une librairie particulière.

Ce concept est propre aux connecteurs, il y a les roles from et role to qui constituent l'interface des connecteurs, leurs entrées et leurs sorties.

Les liens sont des éléments qui permettent de relier des composant entre eux, ou une configuration à un composant. On distinguera deux types de liens :

- Les attachements qui sont des liens entre composants, lien qui inclut un connecteur,
- Les bindings qui relient les ports de configuration aux port de composant, lien qui n'inclut pas de connecteur.

2.4 Analyse

Pour l'implémentation de notre HADL, nous avons du modéliser le meta modèle (niveau M2) à l'aide de l'uml puisque l'objectif était d'avoir une implémentation objet du HADL. Pour ce faire nous avons commencer à mettre en rapport les concepts liés.

Comme le fait que composant, connecteur ou configuration seraient des

classes puisque représentant des éléments majeur de l'architecture à composant. Et donc pour l'implémentation d'une véritable architecture les composants hériterais de ce meta modèle.

Les configurations sont un assemblage de composant et de connecteur, une composition de ces éléments dans la configuration. De la même manière que ces éléments sont des classes, nous avons considéré que les liens étaient aussi important pour eux aussi être des classes. Par conséquent la classe configuration est aussi composé de lien.

La classe configuration est au centre de toute architecture puisque c'est elle qui contient la façon dont les composants interragissent entre eux. La classe configuration doit intercepter tout ce qui entre et sort des composants pour organiser le routage entre composant des différents messages, c'est une sorte de chef d'orchestre.

Configuration et composant sont des éléments très similaires ils disposent d'éléments commun comme des ports, propriétés et contraintes. Les configurations sont composé de composant et partagent des éléments communs, nous avons reconnu le pattern composite, comme nous le détaillerons au travers des diagrammes.

Le sujet suggèrait que l'on puisse remplacer des composants d'une configuration à tout moment, pour celà il faut que chaque appel d'un port ou service soit capturé par la configuration et que si l'action ne peut être éxécuté l'action soit entreposé et rééxécuté lorsque le composant ou connecteur manquant est de nouveau disponible.

Une partie du travail était aussi de gérer un véritable langage, un peu à la façon d'acme, le plus simple à été de créer une dtd et un parseur xml qui permet d'écrire des configurations en xml.

Dans notre application les ports sont identifiés par des identificateurs numériques, et servent pour l'appel des services des composants. Les services sont en fait des méthodes d'instance, qui sont reliés aux ports.

2.5 Diagrammes

Le diagramme **??**

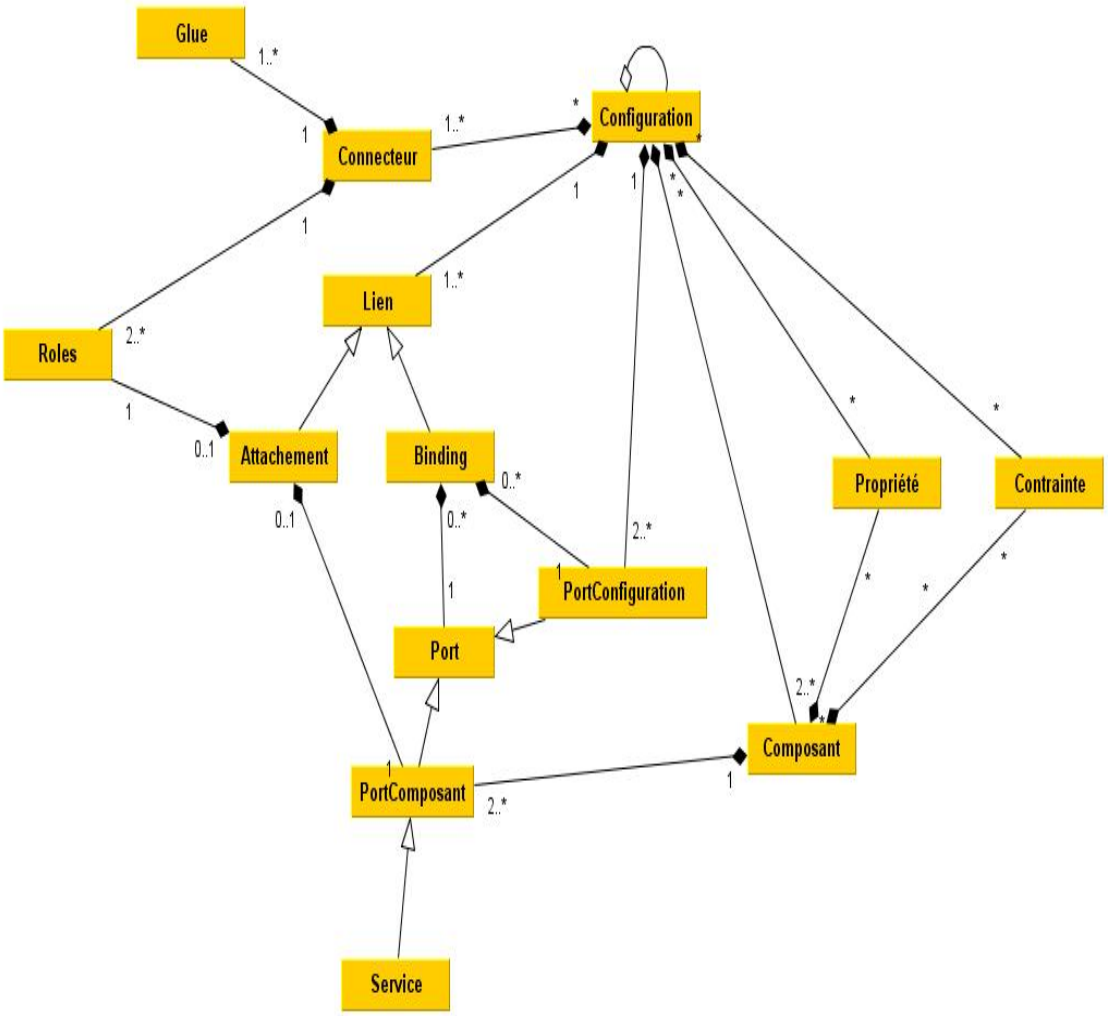
Glue
Connecteur
Configuration
Lien
Roles
Attachement
Binding
Propriété
Contrainte
PortConfiguration
Port
Composant
PortComposant
Service
1..*
1
*
2..*
0..1
0..*
2..*

FIGURE 1 – M2.1

3 Description du système Client/Serveur

3.1 Définitions

Le système Client/Serveur est une bonne représentation au niveau M1 d'une architecture logicielle à composants. En effet, chacune des partie constituant ce système peut être vu comme un composant faisant partie d'une configuration représentant celui-ci. Ces trois composants principaux sont le client, le serveur et le connecteur RPC qui permet la communication entre les deux premières entités.

Le client est donc un simple composant de la configuration "Client/Serveur" permettant l'envoi et la réception de messages. Le serveur, quant à lui, est une configuration comprenant plusieurs autres composants permettant le traitement du message reçu.

3.2 Analyse

Au niveau M1, chacun des composants offre des services. Tout d'abord, le client permet donc d'envoyer et de recevoir des messages. Ces deux services sont liés à deux ports différents.

Les trois composants de la configuration "Serveur" sont ConnectionManager, SecurityDB et Database.

ConnectionManager est le composant qui va recevoir en premier le message envoyé par le client. En effet, via un lien Binding, c'est ce composant qui communique avec la configuration "Serveur". Dans notre implémentation, il propose deux services indiquant l'entrée et la sortie du message dans le composant.

Le composant SecurityDB est celui qui effectue des vérifications liées à la sécurité et aux autorisations concernant le message provenant du client. Nous avons choisi de ne pas implémenter de sécurité pour les messages que nous voulions faire circuler. Le composant affiche uniquement un message dans le terminal signalant le passage du message dans celui-ci.

Enfin, le composant Database représente la base de données interrogée. Pour cela, nous avons mis en place une HashMap avec quelques entrées. Un service de ce composant est de renvoyer la valeur contenu dans la Map correspondant à la clé reçue.

Tous ces composants, qui forment le Serveur, sont reliés entre eux à l'aide de connecteurs. Ces connecteurs sont donc au nombre de trois : ClearenceRequest, SecurityQuery et SQLQuery. La Glue de chacun de ses connecteurs ne fait que transmettre le message, aucune autre action n'est effectuée sur le message.

ClearenceRequest fait le lien entre les composants ConnectionManager et SecurityDB. Quant à lui, le connecteur SecurityQuery relie SecurityDB et Database. Enfin, SQLQuery permet aux composants ConnectionManager et Database de communiquer.

RPC est lui aussi, comme les autres connecteurs décrits précédemment, un connecteur dont la Glue n'effectue uniquement la transmission du message entre le composant Client et la configuration Serveur.

3.3 Diagrammes

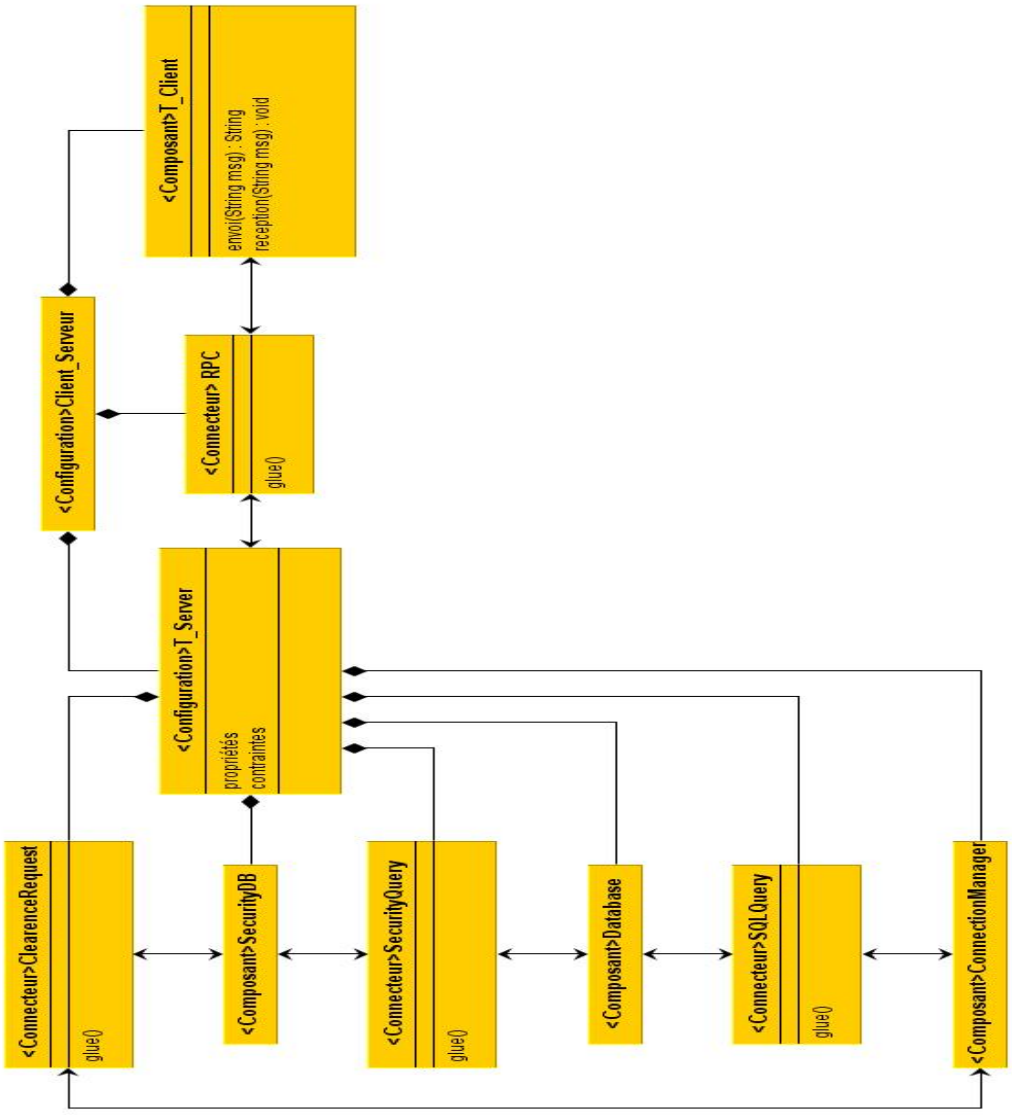

FIGURE 2 – Diagramme du système Client/Serveur

4 Description du méta-méta-modèle HADL

4.1 Définitions

Une méta-entité correspond à une entité comme un composant ou une configuration à un niveau d'abstraction plus grand qu'au niveau méta-modèle.

Une méta-relation correspond, de manière similaire à la méta-entité, à une relation comme un lien d'attachement, un binding ou encore un lien de composition à un niveau d'abstraction plus grand qu'au niveau méta-modèle.

4.2 Analyse

Pour réaliser ce méta-méta-modèle, nous nous sommes aidé de l'exemple du méta-méta-modèle des Langages Orientés Objets étudié en cours.

Nous avons choisi pour ce méta-méta-modèle de ne pas inclure tous ce qui concerne les ports, services et d'autres notions afin de ne pas surcharger le diagramme. Ces notions omises au niveau M2 telles que les ports et services sont des méta-entités et les liens tels que les attachements et les bindings sont des méta-relations.

4.3 Diagrammes

5 Conclusion

En conclusion, le travail effectué sur cet HADL nous a permis de bien observer et de bien comprendre les différents niveaux d'une architecture logicielle à composants. Ceci nous a permis de voir les différents niveaux d'abstraction d'une architecture logicielle et ce qui les relient. De plus, l'implémentation de la méta-modélisation jusqu'à l'instanciation nous a permis de voir concrètement ces niveaux.

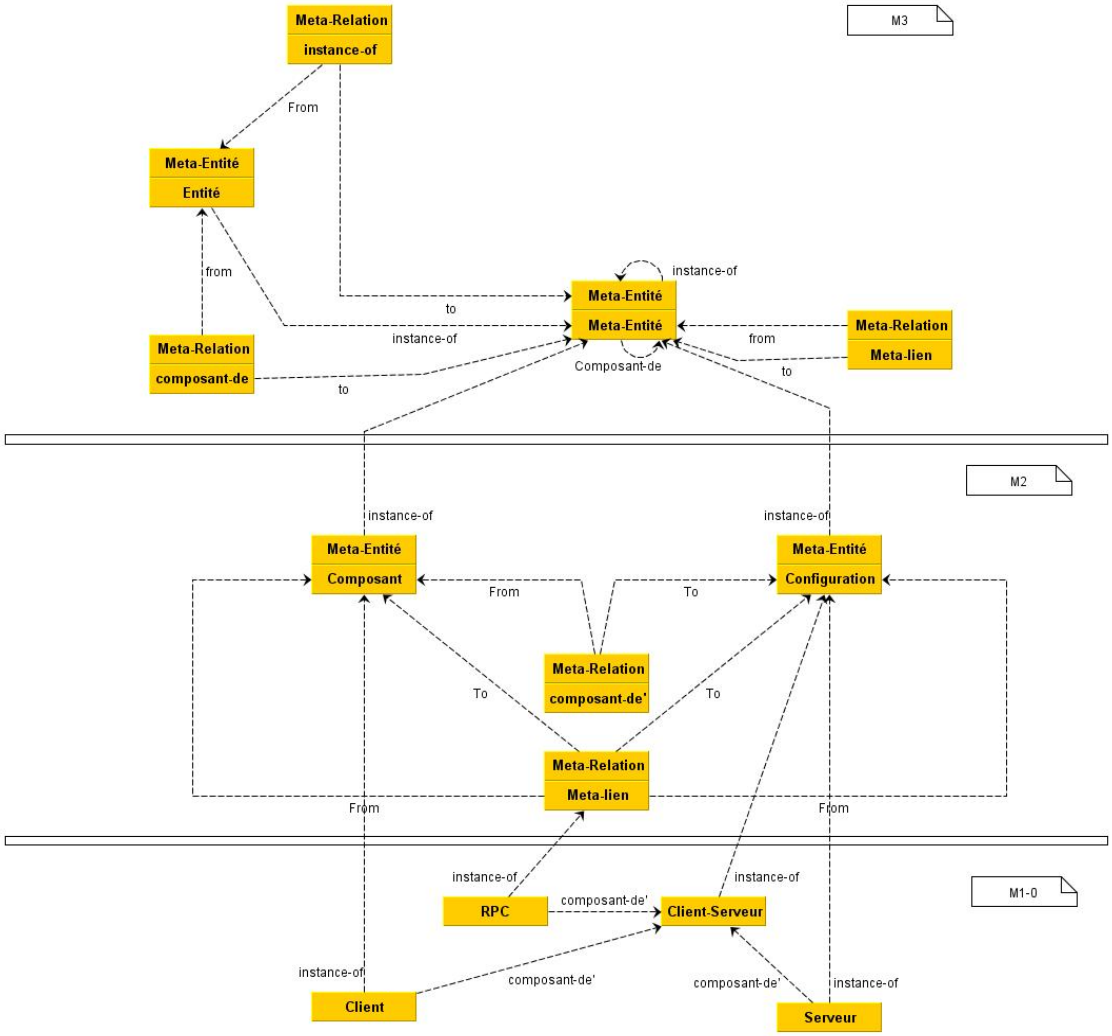

FIGURE 3 – Diagramme du méta-méta-modèle HADL