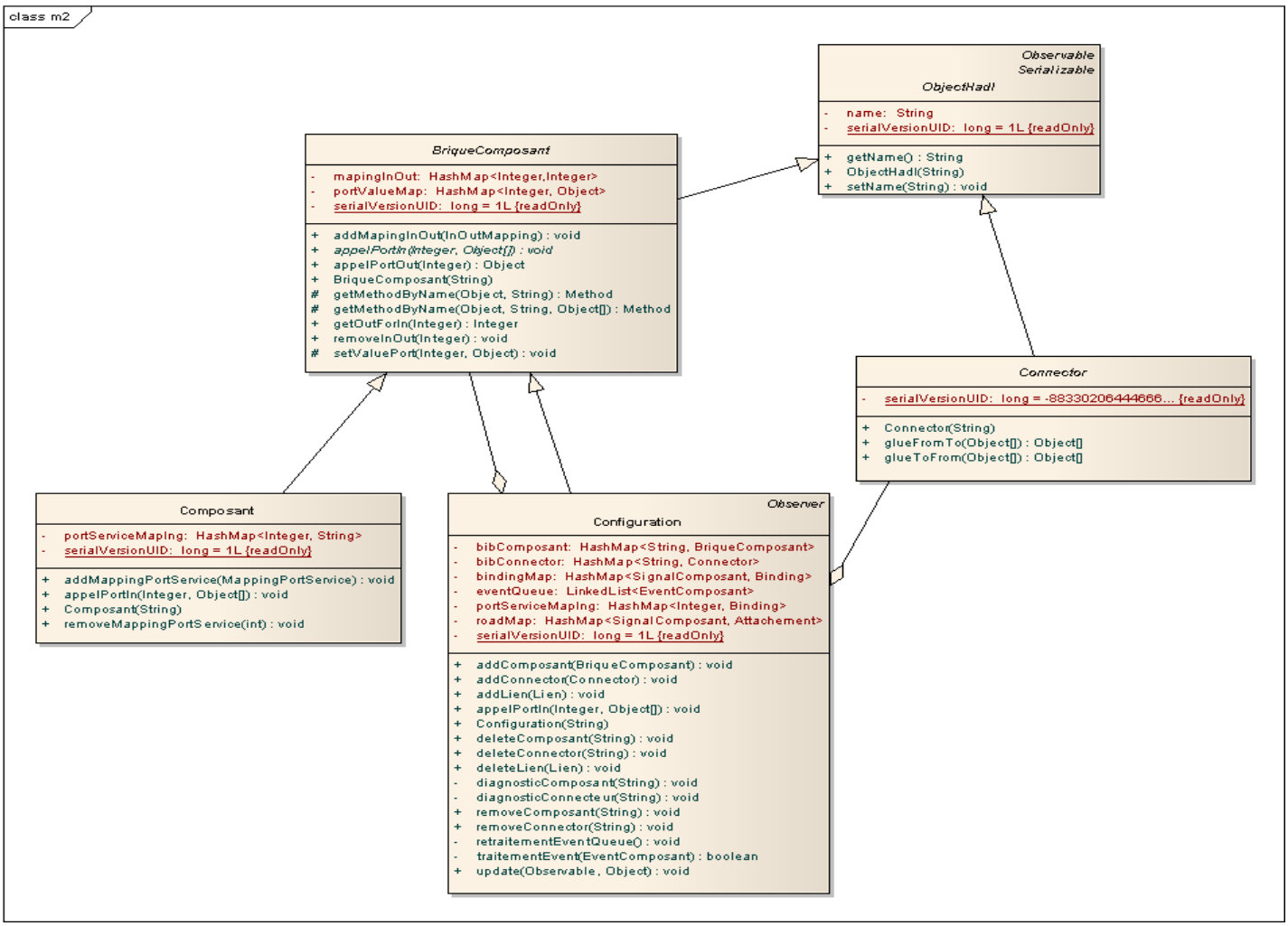
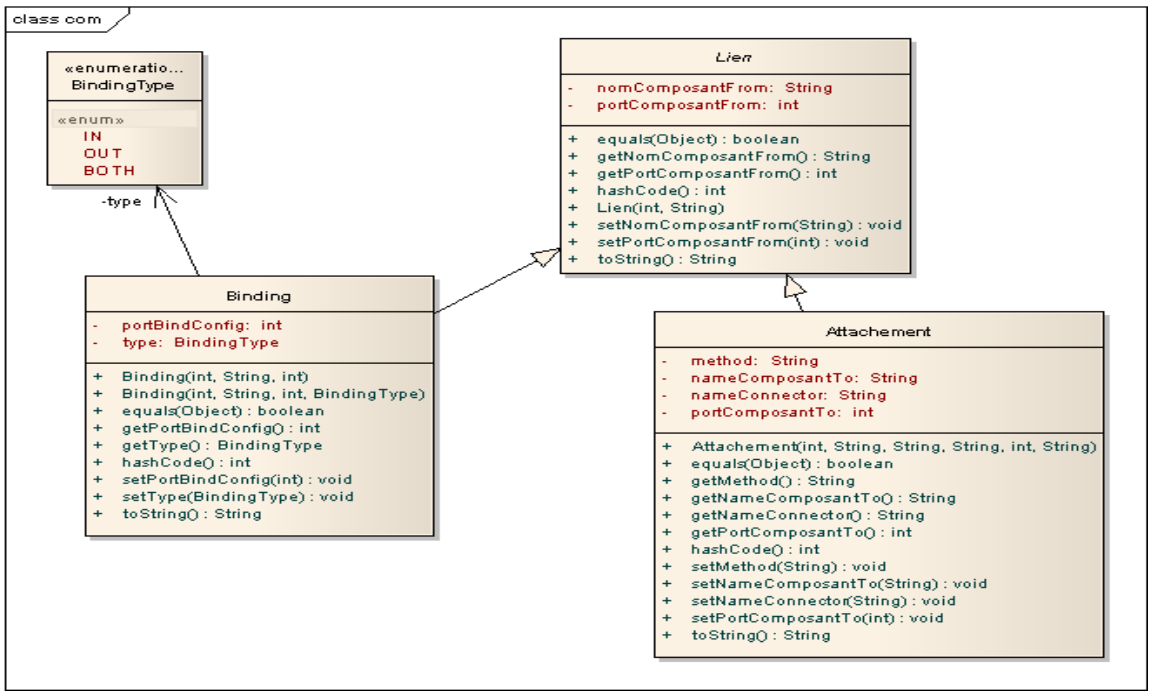
Fortun Manoël
Caillaud Anthony

Synthèse composant

1) Diagramme de classe





Voici les diagrammes du niveau m2 représentant l'ensemble des concepts liés aux développements par composant.

Le premier diagramme représente vraiment les objets de composant et leurs différentes interactions. Le second montre les différents types de lien unissant les composants et configurations entre eux. Ces liens sont contenus dans les configurations pour la communication entre ses composants et elle-même. On distingue à ce propose que le lien binding peut être de plusieurs types : entrée, sortie ou les deux pour différents types de communications et configurations.

Notre architecture est composé des packages suivant:

hadl.m2 : contient les briques basiques (cf diagramme 1)
hadl.m2.com : contient les liens (cf diagramme 2)
hadl.m2.com.event : contient des classes pour la reprise après reconfiguration d'une configuration
hadl.m2.com.param : contient des classes de paramétrage de composants et configurations
hadl.m2.parseur : contient le langage et parseur hadl
hadl.m2.testm2 : contient un test basique du m2
hadl.m2.testxml : contient un test basique du m2 par langage xml

hadl.m1.client : composant exemple client
hadl.m1.rpc : connecteur exemple rpc
hadl.m1.serveur : configuration et composant exemple du serveur
hadl.m1.ClientServeur : configuration client serveur

hadl.m0 : contient les instanciations selon les deux possibilités de l'application

2)Nos choix particuliers

Nous avons choisi de prendre le biais d'avoir des ports identifiés de façon numérique, cela a facilité le développement pour le rendre un peu plus rapide. Le soucis de ce choix c'est que des chiffres ne sont pas forcément très parlant rétrospectivement.

Nous avons choisi pour la configuration et les composants, le parttern observer/observable. Tout objet hadl est donc observables, seul les configurations sont observers. Et logiquement, elles observent les éléments dont elles sont composées.

Nous avons mis en place le pattern composite pour les composants et configurations, en extrayant un objet brique composant rassemblant les caractéristiques communes des deux types d'objets.

Nous avons aussi défini dans tous les objets hadl un élément commun, un attribut de type texte, un nom qui permet son identification dans l'architecture.

Nous avons fait en sorte que pour la création d'un m1, l'utilisateur n'est qu'à étendre nos classes composant, configuration et connecteur. Il n'a plus qu'à simplement écrire les services internes sous forme de méthodes, et bindé port et service via une classe de paramétrage.

3) Fonctionnalités particulières

Nous avons implémenté un enlèvement et rajout « à chaud » de composants(ou configurations) et connecteurs permettant la sauvegarde des traitements en cours. Par exemple en enlevant un composant nécessaire à l'exécution d'une routine, si on appelle la routine ça ne fonctionne pas, mais en remettant le composant la routine est exécuté sans que l'on ait à la relancer.

Nous avons aussi écrit un langage xml décrit par une dtd pour l'instanciation de configuration facilité.

Nous avons aussi en prévision du langage de définition écrit des classes java représentant tous ce qui peut servir pour paramétré des composants ou des configurations, le langage s'appuie sur ces classes pour sa définition et l'instanciation liée,