Stage recherche - Wrapper Java pour un code C++

Arouna GANOU
INRIA Nancy
Email: arouna.ganou@gmail.com

***Résumé*—Nous voulons expliquer les résultats de notre stage recherche, les limites de la solution trouvée et les ouvertures possibles pour les travaux futurs.**

I. Introduction

Le but du stage est de voir les possibilités d'utilisation d'un code C++ dans un code Java. Nous avons choisi utiliser JNI(Java Native Interface) parmi tant d'autres possibilités telle que JNA(Java Native Access).

II. Problème

Mecsyco étant un logiciel de co-simulation de systèmes hétérogènes, il serait intéressant d'intégrer les simulateurs écrits dans des langages différents. Cette intégration exige un travail d'interfaçage entre les différents langages utilisés. Dans ce stage, nous avons exploré la possibilité d'utiliser des simulateurs en C++ dans la version Java de Mecsyco en implémentant une bibliothèque contenant l'implémentation native.

III. Méthode

La solution apportée se base sur le formalisme DEVS. Les échanges que nous avons avec chaque simulateur écrit en C++ utilise un modèle artifact. Donc notre travail est focalisé uniquement sur le modèle artifact qui est en C++. Nous avons développé une classe en Java qui est une représentation du modèle artifact C++. Mais les méthodes de cette classe Java sont implémentées dans une bibliothèque dynamique C++ qui est chargée de gérer les appels que nous effectuerons depuis Java. Cette bibliothèque est principalement codée avec le framework JNI. Ce cadre a l'avantage de donner un plan de travail global de la classe Java initial jusqu'à l'utilisation de la bibliothèque par un code Java quelconque en passant par les adaptations nécessaires au code C++ existante. Les outils de compilation que nous avons utiliser sont **cmake** et **make** sous **ubuntu** et sous **windows**(en utilisant **cygwin**).

IV. Résultats et limites

Nous avons pu implémenter une bibliothèque dynamique fonctionnelle que nous avons testée avec un modèle de Lorenz. Dans cette simulation le comportement de la variable Z est implémentée en C++ à travers la bibliothèque native. Notre est réutilisable sans modifications majeures lorsque les données échangées sont des tuples de réels; les modifications qui devront être faites seront sur les étapes de sérialisation-deserialisation en C++ comme en Java.
Parmi les limites de ce travail, il est utile de citer la perte de portabilité due au fait que chaque système a son type de bibliothèque. Donc il faut compiler pour chaque système une bibliothèque. En plus, le comportement lorsque deux bibliothèques natives sont chargées par la même classe est incohérente. Dans ce cas, les résultats de la simulation ne sont pas ceux attendus. Les configurations nécessaires expliquées dans le rapport sont vraiment limitées à **Eclipse** mais elles sont facilement adaptables à d'autres IDEs.

V. Conclusion

Le développement de ce wrapper avec JNI a utilisé surtout de la sérialisation-déserialisation pour les types non primitifs : Json a été massivement utilisé. Il serait intéressé de voir comment ne pas l'utiliser ou d'utiliser un transport de données plus efficace comme XML.