Merging FMI and HLA for Systems Engineering

Couplage de FMI et HLA pour l'Ingénierie de Systèmes

Simon Gorecki[1] Jalal Possik[1] Gregory Zacharewicz[2] Yves Ducq[1]
[1] Univ. Bordeaux, CNRS, IMS, UMR 5218, 33405 Talence, France
[2] IMT, Mines Ales

simon.gorecki@u-bordeaux.fr
jalal.possik@u-bordeaux.fr
gregory.zacharewicz@mines-ales.fr
yves.ducq@u-bordeaux.fr

Abstract:
Modeling and Simulation (M&S) are important steps in design process. Increasing the complexity of engineered system tend to raise Distributed Simulation (DS). Papyrus, an open source UML/SysML modeler of the Eclipse foundation provides a tool to model and simulate these two languages thanks to the fUML standard. However, Papyrus is not yet able to deal with DS. In this paper, we propose a DS composed of several Papyrus instances. Each made of an UML Profile, a Moka extension, and software architecture. The main objective is to synchronize the execution of these Papyrus instances. We are using Functional Mockup Interface (FMI) to enable communication between them. Nevertheless, the FMI standard does not natively include time management. Indeed, we propose to use the High-Level Architecture (HLA) standard to manage time between Papyrus simulations with FMI communication.

Keywords:
M&S, HLA, FMI, System Engineering.

Résumé :
La modélisation et la simulation (M&S) sont des étapes importantes dans le processus de conception. La complexité grandissante des systèmes tend à accroitre le besoin de segmenter les modèles, leurs exécutions, se dirigeant vers des approches distribuées. Papyrus, un modeleur open source permettant la M&S UML/SysML via le standard fUML. Cependant, Papyrus n'est pas en mesure de gérer la simulation distribuée (SD). Dans cet article, nous proposons une SD composée de plusieurs instances de Papyrus. L'objectif principal est de synchroniser l'exécution de ces instances, en utilisant le standard Functional Mock-up Interface (FMI) pour la communication. Néanmoins, la norme FMI n'incluant pas nativement la gestion du temps, nous proposons d'utiliser le standard High-Level Architecture (HLA) pour gérer la temporalité entre les instances de simulation Papyrus, via la communication FMI.

Mots-clés :
M&S, HLA, FMI, Ingénierie Système.

1 Introduction

De nos jours, avec l'évolution des technologies, les systèmes sont de plus en plus difficiles à modéliser et simuler (M&S). Plusieurs recherches à propos de ces complexités démontrent l'importance de prendre en compte ces problématiques dès la phase de modélisation d'un système [1], et que cette tâche peut drastiquement augmenter la difficulté de M&S. Dans ce contexte à haut risque, il peut être nécessaire de diviser la complexité d'un système en utilisant la simulation distribuée (SD) [2]. Le contexte industriel qui guide cette recherche impose l'utilisation de diagrammes fUML via Papyrus, non compatible avec les principes de la SD.
Dans le but de résoudre cette problématique, nous allons utiliser une extension du moteur d'exécution de Papyrus développée pour externaliser la gestion des risques et des aléas du modèle principal via la norme FMI [3]. Dans la continuité de ces travaux, nous proposons dans ce papier de gérer la complexité d'un projet Papyrus en divisant différentes instances Papyrus en différentes simulations distribuées pour que les utilisateurs et les développeurs puissent fragmenter la simulation de projets complexes en sous-simulations de projets moins-complexes.
Nous verrons premièrement l'état de l'art des différents projets connexes à la proposition. Les parties 3 et 4 présentent la contribution plus en détail, pour enfin conclure en partie 5

2 Contributions connexes

La contribution de ce papier est basée sur deux précédents travaux.
Le premier porte sur le contrôle du temps au sein du moteur d'exécution de Papyrus : Moka, via un composant FMU (Fonctional Mock-up Unit) [3]. Ce composant (voir Figure 1), a pour fonction de générer et injecter des aléas dans le processus de simulation via le moteur d'exécution. Ainsi, il nous est possible de ralentir, stopper, ou accélérer une tâche via ce composant externe.

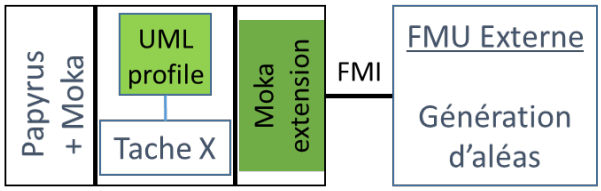


Figure 1 - Extension du moteur Moka

L'extension Moka possède un accès au profil UML associé à une tâche désignée. Ce profil UML contient des paramètres et constantes définis par l'utilisateur. Via cette extension, nous pouvons :

- spécifier un comportement qui agira en fonction des paramètres contenu dans le profil UML
- implémenter une communication avec un composant de co-simulation (FMU).

Dans notre cas (voir Figure 2) le profil UML contient des informations précisant la nature de la tâche X, qui seront récupérées par l'extension Moka, puis envoyées au FMU. Le composant de génération d'aléas récupère les informations, détermine un scénario, et retourne son résultat à l'extension Moka qui impacte la durée de la tâche X durant l'exécution de la simulation.

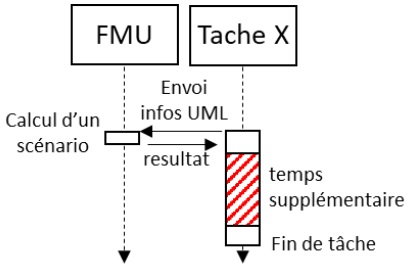


Figure 2 - Diagramme séquence Moka –FMI

Le second papier concerne l'articulation de la technologie HLA et de la norme FMI. Il présente une solution permettant une communication entre les deux standards [4].
La Figure 3 est un exemple illustrant la cohabitation des standards HLA et FMI. On observe en haut à droite de la figure un composant FMU simulant le mouvement d'une balle rebondissante. Ce FMU est instancié depuis un fédéré membre d'une fédération HLA. Ainsi, au cours du temps de la simulation, les informations de position et mouvement de la balle sont transmises à un second fédéré, qui trace la trajectoire de la balle (en haut à gauche de la figure).

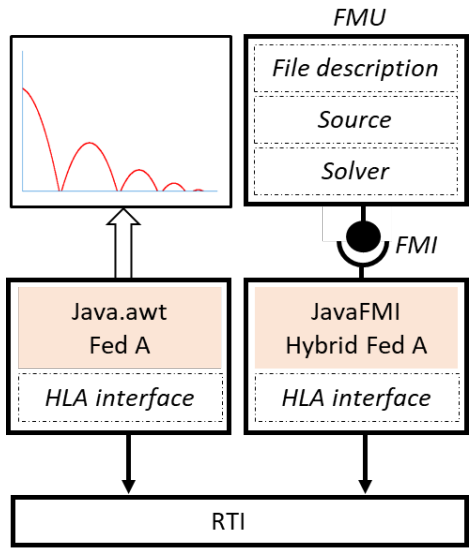


Figure 3 - Exemple de composant hybride HLA - FMI

Cet exemple est une preuve de concept illustrant les possibilités de connexion entre les deux standards qui vont être utiles dans le contenu de ce papier.

3 Composant hybride HLA - FMI pour articuler des instances de Papyrus

L'objectif de ce papier est d'exécuter différentes instances de Papyrus, chacune d'entre elles ayant son propre composant hybride HLA – FMU connecté au moteur d'exécution (comme on peut le constater sur la

Figure 4). Le standard FMI sert de point d'entrée dans une simulation : il permet d'échanger des informations avec le moteur d'exécution de Papyrus, « Moka ». Le standard HLA fait le lien entre les différents FMU. Il est utilisé pour ses fonctionnalités de management du temps et de communication. L'objectif principal de cette implémentation est de gérer des points de synchronisation entre des instances de Papyrus. En utilisant ce composant hybride, on permet à Papyrus, ou tout outil de simulation compatible avec HLA ou FMI, d'être interopérable avec d'autre outils de simulation distribuée.

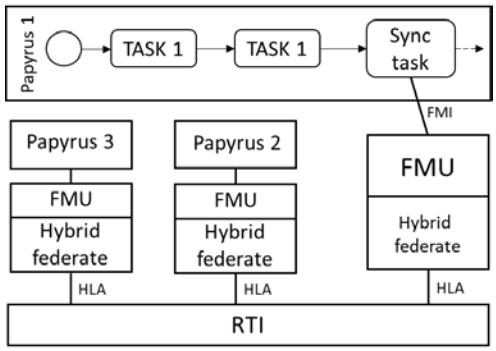


Figure 4 - Architecture distribuée Papyrus

La Figure 4 décrit l'architecture technique d'un exemple illustrant notre proposition. On peut observer trois instances de Papyrus travaillant ensemble. Chacune d'entre elles ayant son propre composant hybride permettant la communication inter-simulations à travers les mécanismes de communication HLA fourni par le Run Time Infrastructure (RTI).
Lorsqu'une instance de Papyrus doit attendre une autre instance, des messages sont échangés entre les deux par le biais du mécanisme de communication HLA. Au début de la fédération, chaque fédéré souscrit à une classe d'interaction « Message » (décrite dans la Figure 5) afin d'être informé de chaque modification d'état au cours de la simulation distribuée. Les mécanismes HLA ne permettent pas à un fédéré d'adresser un message directement à un destinataire. Les fédérés peuvent seulement publier ou s'abonner à des « Objets » ou « Interactions ».

A l'initialisation de la fédération, chaque fédéré souscrit à la même classe d'interaction (Figure 5). Ainsi, quand un message est envoyé, tous les composants de la simulation distribuée en sont informés.

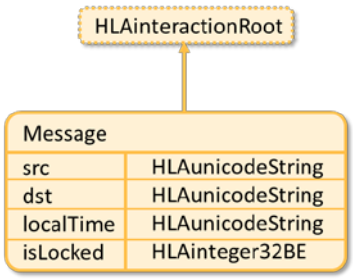


Figure 5 - Fichier "FOM"

Durant la simulation, tous les fédérés reçoivent les mêmes interactions. Ils utilisent alors les attributs « src » et « dst » afin d'identifier si ils sont concernés par l'interaction. Si non, ils ignorent le message, si oui, ils publient une réponse (destinée à l'expéditeur). Quand une instance de Papyrus entre dans une étape synchrone, le fédéré associé publie une interaction dans le but d'informer les autres membres de la simulation distribuée.
Les attributs « localTime » et « isLocked » servent respectivement à envoyer la valeur du temps local du fédéré correspondant ainsi que d'informer si un message est un message annonçant un blocage, ou annonçant un déblocage.

4 Gestion du temps entre des instances de Papyrus

La gestion du temps entre les instances Papyrus est un point important. Un point de synchronisation (où tâche synchrone) représente un élément bloquant. L'utilisateur définit une tâche comme bloquante en lui associant un profil UML prédéfini (Figure 6). A l'exécution de cette tâche, l'instance sera mise en attente, avant d'être débloquée par une autre instance de la SD. Quand une instance doit en attendre une autre, cette dernière est bloquée par le moteur d'exécution. Le FMU se met en attente d'un message de déblocage provenant d'un autre membre de la fédération. Le temps

d'attente αt doit être simulé par le moteur d'exécution de Papyrus.

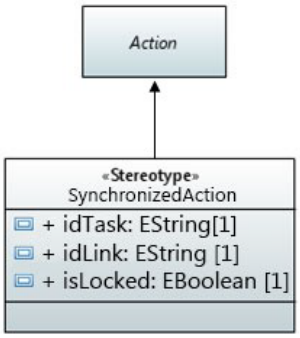


Figure 6 - Profil UML de tâche bloquante

La Figure 6 décrit le profil UML que l'on applique à une tâche afin de la définir comme étant « synchrone ». L'utilisateur définit un identifiant unique à la tâche (idTask), ainsi que l'identifiant de la tâche à laquelle elle est liée (idLink). Le champ « isLocked » permet de définir si la tâche est :

- Bloquante : elle sera débloquée par la tâche désignée par idLink ;
- Débloquante : elle libèrera la tâche désignée par idLink

Ainsi, avec l'implémentation d'un simple FMU au sein du moteur d'exécution de Papyrus, grâce au composant hybride HLA/FMI, il est très simple d'implémenter des mécanismes de co-simulation dans un outil qui n'est à la base pas prévu pour, et intégrer des communications avec des outils externes.

5 Conclusion

La communauté de Papyrus tend, dans les prochaines versions, à ouvrir l'outil aux principes de co-modèles avec l'implémentation du standard FMI. Cependant, FMI ne prenant pas en charge les mécanismes de gestion du temps, nous avons présenté dans ce papier, une extension du moteur Moka compatible FMI-HLA pour franchir cette limitation. Un des principaux avantages de cette solution hybride se distingue par la simplicité de mise en œuvre. En effet, les mécanismes d'implémentation, de communication, et de gestion du temps imposés par HLA deviennent transparent avec l'utilisant d'un simple composant FMU. La solution proposée est actuellement à un stade initial de conception. Jusqu'à présent, elle ne peut partager que de simples jetons comme ressources. Un autre travail futur pourrait être la mise en œuvre d'un partage de ressources plus complexe tel que des variables de simulations, ressources, etc.

6 Références

[1] M. Better, F. Glover, G. Kochenberger, et H. Wang, « Simulation optimization Applications in risk management », *International Journal of Information Technology & Decision Making, vol. 7, No. 4*, p. 571-587, 2008.

[2] J. Possik, A. D'Ambrogio, G. Zacharewicz, A. Amrani, et B. Vallespir, « A BPMN/HLA-Based Methodology for Collaborative Distributed DES », in *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, Capri, Italy, 2019, p. 118-123.

[3] S. Gorecki, Y. Bouanan, J. Ribault, G. Zacharewicz, et N. Perry, « Including Co-Simulation in modeling and Simulation tool for supporting risk management in industrial context », *International multidisciplinary modeling & simulation multiconference (I3M)*, Budapest, Hungary, sept-2018.

[4] Y. Bouanan, S. Gorecki, J. Ribault, G. Zacharewicz, et N. Perry, « Including in HLA Federation Functional Mockup Units for Supporting Interoperability and reusability in Distributed Simulation », *50th Computer Simulation Conference (SummerSim)*, Bordeaux, France, p. 1-12, juill-2018.