Architecture à composants de co-simulation appliquée au couplage de la thermique du bâtiment au comportement de l'usager

Sana GAALOUL[1], Xuan Hoa Binh LE[2], Benoit DELINCHANT[1], Frédéric WURTZ[1], Stéphane Ploix[2]

[1]G2Elab–LABoratoire de Génie Electrique de Grenoble
ENSE3 : 961, Rue de la Houille Blanche, 38402 Saint Martin d'Hères

sana.gaaloul@g2elab.grenoble-inp.fr

[1]GSCOP– Laboratoire des Sciences pour la Conception, l'Optimisation et la Production de Grenoble
46, avenue Félix Viallet - 38031 Grenoble Cedex 1 - France

RÉSUMÉ. *Cet article développe une architecture de co-simulation basée sur un standard de composants logiciels. Il propose une norme de composant permettant d'unifier les interfaces de co-simulation de plusieurs outils de simulation énergétique du bâtiment. Il permettra ainsi de faciliter et de généraliser la mise en place d'une co-simulation. L'intérêt de cette stratégie sera mis en évidence à travers le couplage entre le logiciel COMFIE de simulation de l'enveloppe thermique et l'outil Brahms de simulation du comportement des occupants dans l'environnement Matlab/SIMULINK. La mise en relation de ces deux outils était indispensable pour profiter de leurs compétences spécifiques dans chaque domaine et assurer la complémentarité entre eux.*

ABSTRACT. *This paper develops a co-simulation architecture based on software components standard. It proposes a component to unify the co-simulation interfaces of several building energy simulation tools.It will thus facilitate and generalize the establishment of a co-simulation. The advantage of this strategy will be highlighted through the coupling between the COMFIE software: thermal envelope simulator and the Brahms software: occupant behavior simulator, in the Matlab / Simulink, software for the orchestration. The linking of these two tools was essential to take advantage of their specific skills in each area and ensure complementarity between them.*

MOTS-CLÉS : *co-simulation, composant logiciel, enveloppe thermique, comportement de l'usager.*

KEY WORDS: *co-simulation, software component, thermal envelope, inhabitant behaviour.*

1. Introduction

Le bâtiment est un secteur hétérogène faisant cohabiter plusieurs composants issus de différents domaines d'ingénierie et qui interagissent ensemble de manière plus ou moins forte. Dans le contexte des bâtiments basses consommations, l'isolation thermique est de plus en plus performante grâce à des matériaux et des techniques avancées, rendant de ce fait les apports internes (usagers, appareils électriques…) plus influents sur la gestion énergétique du bâtiment. Ils ne peuvent plus être négligés car ils permettent de :

- réduire la facture énergétique en participant au chauffage d'hiver,
- augmenter la facture énergétique en impactant le confort d'été.

Donc pour réduire les consommations énergétiques tout en assurant le confort, il est indispensable de tenir compte de différents acteurs hétérogènes et de leurs interactions pour une simulation complète du bâtiment.

Pour répondre à ces besoins de nombreux outils de simulation ont été développés ces dernières années. Le département de l'énergie des Etats Unis[1] a recensé 404 outils de simulation associés à l'évaluation de l'efficacité énergétique, les énergies renouvelables et le développement durable… Cependant, chaque outil est spécialisé dans un domaine donné et prend moins en considération les autres [Cra 05]. Il n'existe pas d'outil complet permettant de répondre inconditionnellement aux besoins des utilisateurs et ainsi simuler les différents domaines et composants

[1] http://apps1.eere.energy.gov/buildings/tools_directory/

dans plusieurs phases de vie du bâtiment. En effet, les besoins et les objectifs des bureaux d'études évoluent d'un projet à un autre mais également au fil du temps selon le contexte énergétique. D'où la nécessité de pouvoir répondre dynamiquement et facilement à leurs exigences à travers la co-simulation. La co-simulation permettra ainsi de simuler n'importe quel composant dans le logiciel approprié sans avoir à se soucier de son couplage avec d'autres composants dans d'autres logiciels bien différents.

2. La cosimulation à base de composants logiciels: une solution pertinente d'interopérabilité dans le bâtiment

2.1. Les solutions existantes d'interopérabilité

Selon l'IEE, l'interopérabilité se définie comme *la capacité de deux ou plusieurs systèmes ou composants d'échanger l'information et d'utiliser cette information qui a été échangée*. Une interopérabilité peut être spécifique, réalisée pour un cas précis entre des acteurs définis, ou être plus générique et indépendante de l'outil utilisé.

Les approches d'interopérabilité peuvent être classées en trois principales catégories en se basant sur les études de Hensen [Hen 04]:

- Interopérabilité des données des modèles : il s'agit d'assurer le transfert des données entre les programmes au niveau des modèles. Elle peut être réalisée à travers l'établissement d'une large base de données, tel était l'objectif du projet COMBINE (Computer Models for the Building Industry in Europe) [Aug 94]. Un format standard de données peut être également mis en place pour structurer ces données afin de faciliter l'échange, tel est le cas du standard des IFC (*Industry Foundation Classes*) [Baz 99].
- L'interopérabilité des processus des modèles : cette stratégie peut être également assurée à travers deux moyens. Le premier consiste à la projection des modèles dans le même langage qui doit être neutre ou multi physiques comme le langage Modelica[2]. Plusieurs bibliothèques ont été développées dans ce langage comme la bibliothèque *ATplus* [Fel 02] pour la simulation du comportement thermique du bâtiment, *HumanComfort* [Mic 09] des modèles de prédiction du confort de l'occupant. La deuxième consiste à l'échange et la réutilisation des composants modèles. A ce niveau, une standardisation des différents modèles du bâtiment a été proposée à travers l'établissement d'une norme de composants logiciels [Gaa 11 b].
- Coopération des données et des processus des modèles : cette approche se focalise sur les couplages des programmes en vue d'un échange mutuel d'information d'une manière prédéfini pendant la simulation. Cette stratégie est appelé également « co-simulation » et permet plus de flexibilité de modélisation et simulation que les autres approches car les modèles sont indépendamment créés et séparément simulés dans leur propre outil. Dans certains cas, elle s'avérer la seule solution possible d'interopérabilité, quand il est délicat de séparer les modèles et leurs données de leur solveur. En outre, les autres solutions sont préférables lorsqu'elles sont possibles car la co-simulation offre de faibles performances et d'éventuels problèmes d'instabilité numérique. A part plusieurs couplages directs réalisés entre quelques logiciels de simulation du bâtiment, un effort de généralisation et d'automatisation pour faciliter la mise en œuvre d'une co-simulation a été réalisé. Il s'agit du BCVTB[3], une plateforme assurant le couplage de plusieurs outils et avec laquelle plusieurs connexions ont été développées. Il s'agit ici d'une standardisation du moyen de communication : l'échange d'information via des *sockets*[4]. D'un autre côté aucun travail de standardisation des acteurs de la co-simulation n'a été répertorié dans le domaine de la simulation énergétique du bâtiment malgré l'utilité de ce concept démontré dans d'autres domaines [Fmi 10].

2.2. Le standard de composant logiciel pour la co-simulation

Une co-simulation met en œuvre deux types de composants principaux :

- un orchestrateur: qui gère les appels successifs aux différents modèles à coupler

[2] www.modelica.org

[3] simulationresearch.lbl.gov/bcvtb

[4] socket : interface logicielle avec les services du système d'exploitation, grâce à laquelle un développeur exploitera facilement et de manière uniforme les services d'un protocole réseau tel que TCP/IP

- les modèles: qui fournissent des résultats de calcul en réponse aux requêtes de l'orchestrateur.

Un modèle peut être intégré à une co-simulation s'il peut offrir des interfaces de communication avec son solveur afin de le démarrer et de l'arrêter depuis l'extérieur. La solution se basant sur la co-simulation ne nécessite pas des modifications importantes des logiciels utilisés et permet une communication simple entre les outils.

Cependant, les interfaces offertes par les outils de modélisation sont très variées au niveau de la syntaxe, de la nature des données, de la stratégie adoptée... Ceci qui nécessite un traitement et un développement au cas par cas.

Par exemple, la mise en œuvre d'une co-simulation dans un orchestrateur X1 (maître) de deux modèles X2 et X3 (esclaves) se fait généralement de manière séparée (Figure 1 gauche). En vue de généraliser la procédure de co-simulation, nous avons opté pour une technique se basant sur un standard de composant logiciel. Cette technique consiste à encapsuler le modèle et ses capacités de calcul dans un composant logiciel. Ce dernier, étant une entité indépendante de déploiement, va garantir la portabilité et l'exploitabilité de l'outil via ses interfaces qui seront exposées alors que tout le contenu du composant est caché, tel est le principe des composants logiciels [Szy 96].

Un standard de composant pour la co-simulation signifie essentiellement une normalisation des interfaces et de cette manière le cas de la co-simulation traité précédemment peut être développé de la façon suivante : X2 et X3 sont encapsulés sous la forme d'un composant logiciel, donc ils ont les mêmes interfaces. Il suffit alors de développer un unique connecteur du composant vers X1 qui permettra de charger et communiquer avec X2 et X3 (Figure 1 droite).

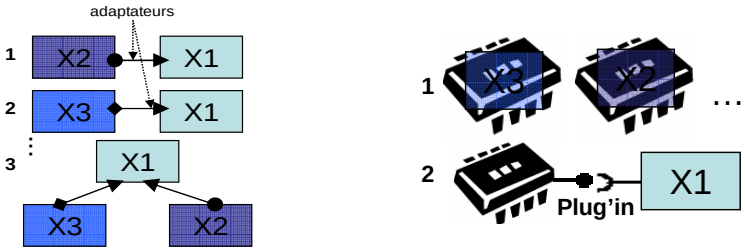


Figure 1. *La co-simulation sans et avec l'utilisation d'un standard de composant logiciel.*

3. La norme de composants logiciels ICAr pour la cosimulation

3.1. La norme ICAr

ICAr (Interface for Component ARchitecture) [Del 02] est une norme de standardisation des composants logiciels définie initialement pour la conception des systèmes d'ingénierie électrique et élargie par la suite pour couvrir d'autres métiers et d'autres domaines grâce au concept « muti-services » (Figure 2). Cette architecture évolutionnaire permet d'accéder à plusieurs services (optimisation, simulation…) à travers les différentes facettes accessibles via l'interface « Component » (Figure 3). Cette interface représente un moyen pour instancier un composant, accéder à ses ports et interagir avec ses différents services ou « facettes ». Chaque facette du composant correspond à une tâche particulière capable de fournir les sorties désirées à parties des entrées fournies tel que un calcul du jacobien ou des dérivées du vecteur d'état.

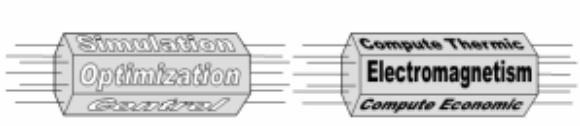


Figure 2. *Le principe du Muti-services*



Figure 3. *Le composant ICAr*

Un composant ICAr est indépendant de la plateforme d'utilisation vu qu'il est sous forme d'un code java exécutable et donc autonome. Il est déployable de façon normalisé grâce à ses interface standardisées, et ainsi, utilisable dans différents environnements.

3.2. La facette de co-simulation

La définition de la facette de co-simulation « CosimulationSlave » du composant ICAr a nécessité l'étude de plusieurs interfaces de communication (API[5]) offertes par plusieurs solveurs, ainsi que les différentes stratégies de couplages possibles : synchrone, asynchrone, avec un pas de synchronisation fixe ou variable. Ceci est dans le but d'avoir une norme générique pouvant couvrir la majorité des cas tests de la simulation du bâtiment.

La norme ICAr définit le composant principalement en terme de ports et d'interfaces. Le composant est définit par ces ports d'entrées : – *U*, le vecteur des sources du modèle et – *P* : les paramètres du modèle

A la sortie de l'ICAr on retrouve : – *Y*, le vecteur des sorties.

Les principales interfaces de co-simulation sont (Figure 4):

- *Initialize ()*: permet d'initialiser le modèle (esclave) et préparer les éléments dont il a besoin pendant la simulation (librairies, fichiers, machine virtuelle, configuration…)
- *start (starTime, simStep, synchTime, InitialValues)* : Elle permet de lancer ou de relancer (en cas de retour arrière) une simulation à partir d'une date starTime, avec un pas de simulation de simStep jusqu'à l'instant de synchronisation synchTime en lui affectant les valeurs initiales dont il a besoin.
- *Resume(simStep, synchTime)*: reprend la suite de la simulation du modèle de là où elle s'est arrêtée à la dernière étape, avec un pas de simulation de *simStep* jusqu'à l'instant de synchronisation synchTime.
- *Pause()* : Permet d'interrompre, quand c'est possible, le processus de simulation avant sa fin.
- *Stop ()*: permet de terminer la simulation en libérant les mémoires qui ont pu être alloué ou supprimer des fichiers qui ne sont plus nécessaires…

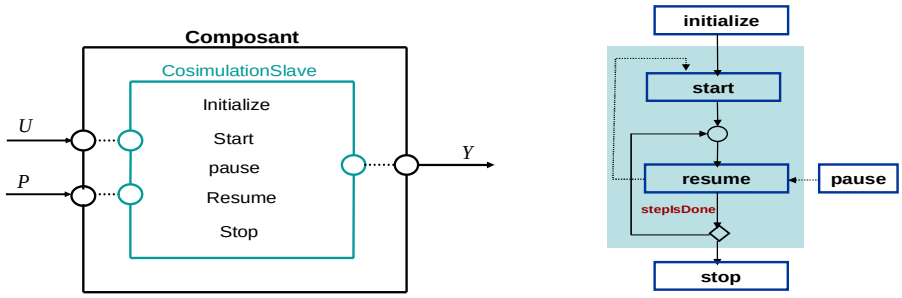


Figure 4. *Les interfaces de co-simulation.*

Le composant contient également des méta données pour renseigner l'utilisateur quand il ne connaît pas son contenu pour mieux gérer sa connexion et orchestration avec des stratégies avancées comme : – les pas de temps quand il est fixé par le modèle, – la capacité de varier le pas de temps, et si oui – les pas de temps supportés ou son intervalle de variation, – la possibilité d'interrompre une simulation, – la capacité de se réinitialiser…

4. Cas d'application: co-simulation entre d'une simulation thermique (COMFIE) et d'une simulation d'occupants (BRAHMS)

4.1. La nécessité et l'intérêt de la co-simulation

[5] Application Programming Interface

Un cas d'étude a été sélectionné pour mettre en évidence l'importance et l'apport de cette nouvelle approche. Il s'agit d'une simulation détaillée d'un système de bâtiment composé de deux zones thermiques et habité par une personne qui peut intervenir dynamiquement sur la consigne de chauffage quand elle dépasse son seuil de confort.

Pour assurer la simulation thermique de l'enveloppe, un large choix d'outils est possible. Nous avons choisi COMFIE [Sal 05] vu qu'il assure une modélisation assez fine du bâtiment s'appuyant sur l'approche des différences finies et en même temps assez rapide grâce à une réduction par approche modale. Il permet de tenir compte de plusieurs paramètres : la géométrie du bâtiment, les caractéristiques thermiques et le climat...

COMFIE, comme plusieurs autres logiciels de simulation énergétique, ne néglige pas l'usager car il prend en compte la puissance thermique qu'il dégage à travers des scénarios prédéfinis. Par contre, cette méthode ne prend pas en considération la variation et la complexité du comportement de l'usager, ne représente pas son interaction avec son environnement et ne peut pas vérifier le respect de son confort. La modification de ces outils nécessite des efforts conséquents au niveau méthodologique ainsi qu'en programmation alors qu'il existe d'autres outils qui pourraient les compléter. A ce niveau, les travaux de Kashif [Kas 11] ont démontrés la capacité de l'outil BRAHMS[6] à assurer une simulation performante du comportement de l'usager dans les bâtiments pour une meilleure gestion de l'énergie.

BRAHMS est une plateforme multi agents permettant de simuler le comportement réactif et délibératif d'un agent au sein de son groupe. Le modèle de l'occupant dans BRAHMS est réactif, il est conscient de son environnement et peut donc réagir sur des objets et modifier leurs états en fonction des informations perçues. Il est également délibératif car l'occupant peut prendre une décision en concertation avec les autres membres de son groupe.

Ainsi, un agent de Brahms est en relation avec les autres agents ou objets... La simulation de chaque agent nécessite un traitement particulier de la machine virtuel de Brahms (solveur) pour tenir compte de son environnement et pour bien définir l'ordre de priorité des différentes activités et évènements qui peuvent avoir lieu. Il est donc impossible de séparer le modèle de son solveur pour une résolution externe et ainsi de procéder à une interopérabilité par processus des modèles. L'interopérabilité des données est insuffisante dans ce cas pour bénéficier des compétences des deux outils. La co-simulation s'avère donc l'unique solution permettant d'avoir à la fois une simulation détaillée de l'enveloppe du bâtiment par COMFIE et du comportement des usagers via BRAHMS.

4.2. La modélisation de l'usager: Brahms

4.2.1. Le principe de modélisation dans Brahms

Le comportement humain est modélisé dans Brahms selon le principe des agents et il est basé sur une structure BDI (belief, desire, intention). Un agent (occupant par exemple) dans Brahms est définit, dans le cadre d'un groupe d'agent, par ses activités (exemple : déplacement) qui peuvent être en lien avec d'autres objets, son emplacement géographique et des contraintes temporelles qui conditionnent son comportement.

Pour répondre aux besoins du cas test décrit précédemment, l'occupant a été modélisé par un agent régit par un ensemble de conditions temporelles de déplacement entre les deux zones du bâtiment et l'extérieur (Figure 5). Il est capable également de modifier l'état de l'objet chauffage s'il a la sensation de chaleur ou de froid (besoin de chauffer ou de refroidir) selon la température perçue.

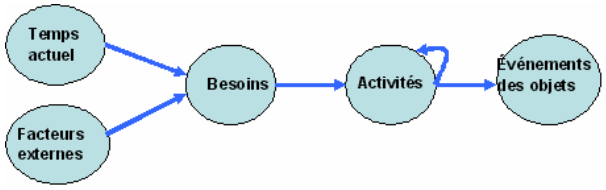


Figure 5. *L'évolution du comportement d'un occupant dans Brahms.*

4.2.2. Le plug'out de BRAHMS

[6] http://www.agentisolutions.com/

Brahms a l'avantage d'avoir des interfaces variées et développées pour interagir avec sa machine virtuelle. Il s'agit des interfaces de programmation en Java (*Japi*[7]) ce qui rend leur utilisation plus facile pour la génération du plug'out étant donné que la norme ICAr est développé en Java. Il offre des interfaces pour communiquer avec les modèles en affectation et en récupération (exemple : setAgentAttributeDouble, getAgentLocation, getObjectAttributeString…). On trouve également des interfaces pour le contrôle de la machine virtuelle de Brahms (JAPI.getVMController().start(), JAPI.getVMController().resume(), JAPI.getVMController(). pause()) ainsi que des informations sur la fin d'une étape de calcul ou de la détection d'un événement…

La construction du composant logiciel à partir de BRAHMS consiste à compléter les interfaces de l'ICAr de co-simulation avec les bonnes commandes de BRAHMS en se basant sur le kit de développement.

4.3. La modélisation de l'enveloppe thermique: COMFIE

4.3.1. Le principe de la modélisation dans COMFIE

Comme décrit dans [Gaa 11 a], la modélisation dans COMFIE se base sur l'établissement d'un système d'état global des différentes zones à partir des équations d'échange de chaleurs établies pour chaque maille de la zone. On aboutit après plusieurs traitements mathématiques au système global final suivant :

.

$$\begin{cases} X^{n+1} = \exp\left(F \cdot \Delta t\right) \cdot X^{n} + W^{\alpha} \cdot \left(U^{n+1} - U^{n}\right) + W^{\beta} \cdot \left(Y^{n+1} - Y^{n}\right) \\ Y^{n+1} = MGIF \cdot X^{n} + MGIE \cdot U^{n+1} + MGID \cdot U^{n} - MGIC \cdot Y^{n} \end{cases}$$

Le calcul itératif permet d'obtenir les valeurs de X et Y au temps (n+1).Δt, notées Xn+1 et Yn+1, en fonction des valeurs Xn et Yn au temps n.Δt (méthode récursive). On note de même Un+1 et Un les vecteurs des sollicitations aux temps (n+1).Δt et n.Δt.

COMFIE permet ainsi d'étudier l'évolution des températures internes en fonctions des différentes sollicitations externes (flux solaire, température extérieure…) ou internes (apport des usagers, des équipements…).

4.3.2. Le plug'out COMFIE

Contrairement à BRAHMS, COMFIE est plus fermé et ne propose pas d'API pour la commande externe de son algorithme de résolution. Le seul moyen d'interopérabilité offert par COMFIE est l'externalisation des matrices d'états et des sollicitations après traitement par le cœur de calcul de COMFIE.

Donc, pour construire le composant de co-simulation de COMFIE on procède à l'extraction des matrices générées par COMFIE et la reconstruction en Java du système d'état [Gaa 11 b]. Il s'agit de compléter, en s'appuyant sur le kit de développement, les interfaces nécessaires du composant avec les informations convenables à partir de ce système d'état. Il s'agit essentiellement de calculer la sortie *Y* et mettre à jour le système d'état à chaque appel de l'interface *resume*(). La sauvegarde des états précédents est assurée par le composant et le pas de simulation est imposé à l'intérieur également.

Le bâtiment étudié est composé de deux zones thermiques. On aura donc un composant logiciel ayant en entrée les puissances internes injectées dans chaque zone et en sortie les températures des deux zones.

4.4. Mise en place de la co-simulation dans Matlab SIMULINK

4.4.1. Le plug'in Matlab SIMULINK

SIMULINK, permet l'introduction des nouveaux modèles via les S-Function. Un block S-fucntion est définit par un ensemble d'instructions, chaque une est exécutée lorsque le solveur l'appelle en envoyant le *flag* qui lui correspond. Dans le plug'in, chaque instruction Matlab, appelle une interface équivalente de l'ICAr qui assure la même fonctionnalité qu'elle. Ainsi, à chaque étape de la simulation le solveur va appeler une interface de l'ICAr en lui affectant ces entrées et récupérant ces sorties. Il faut préalablement avoir chargé l'ICAr et ses librairies dans l'espace de travail de Matlab.

[7] Java Application Programming interfaces: les interfaces java de commande de la machine virtuelle de Brahms

Une fois réalisé, le plug'in permet d'importer, introspecter et utiliser plusieurs ICar de la même manière (Figure 6). Ainsi, les composants ICAr de COMFIE et de Brahms peuvent être exécutés dans Simulink qui jouera le rôle de l'orchestrateur.

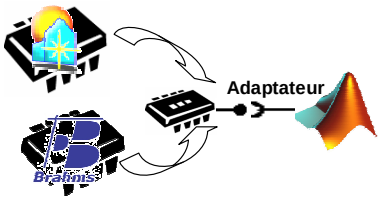


Figure 6. *Le plug'in de l'ICAr dans SIMULINK.*

4.4.2. L'orchestration de la co-simulation dans SIMULINK

La première zone du bâtiment est équipée d'un chauffage donc ses puissances internes sont constituées de la puissance thermique du chauffage et de la puissance thermique dégagée par l'usager, contrairement à la deuxième zone où la puissance thermique apportée ne provient que d'un éventuel occupant. Ce dernier étant conscient de son environnement et de la température ambiante, agit sur la consigne de chauffage pour l'éteindre ou l'allume en fonction de seuils de température. Le schéma global de la co-simulation est présenté dans la figure suivante :

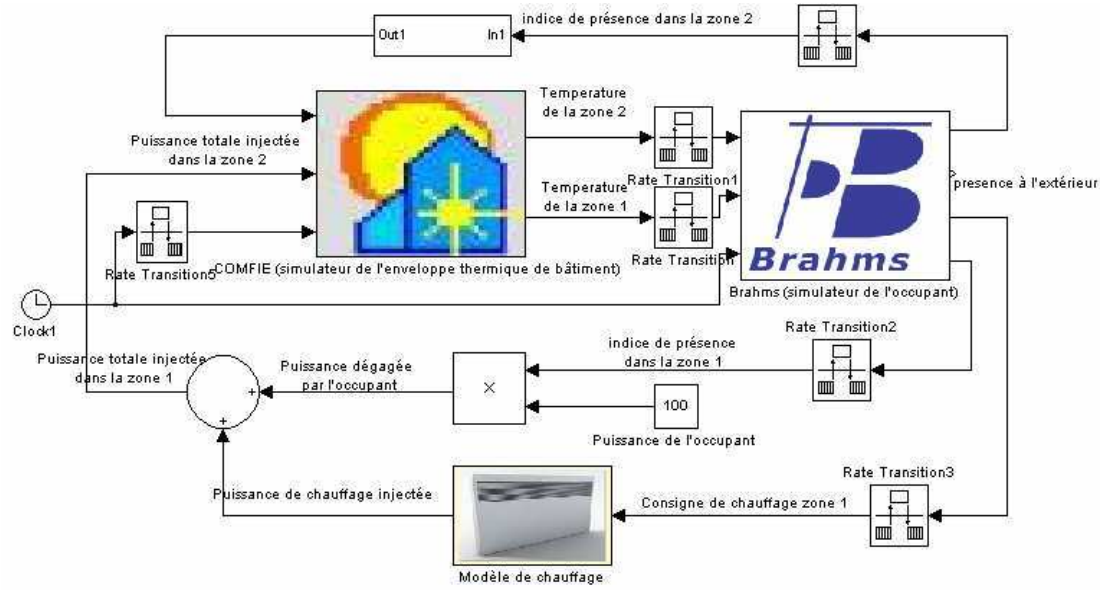


Figure 7. *Le schéma de la co-simulation entre COMFIE et BRAHMS orchestrée dans SIMULINK.*

4.4.3. Analyse des résultats de la co-simulation

Le pas de simulation de Brahms a été fixé à 60 secondes pour qu'il puisse avoir un échange avec Matlab régulièrement afin de l'informer des variations du comportement de l'occupant. Toutes les 60 secondes, Brahms l'informe des différents évènements qui ont eu lieu entre deux pas consécutifs. Le pas de simulation de COMFIE est fixé par le modèle à 1 heure. La synchronisation des deux acteurs et l'échange d'informations se fait au plus grand pas de temps qui est celui de COMFIE : 1 heure (Figure 8). Cette différence au niveau des pas de simulation est gérée par Simulink via les blocks « rate Transition » qui permettent l'adaptation des données entre les deux acteurs.

La simulation de 20 heures du système complet a duré 1174 secondes, environ 20 minutes et donc la simulation d'une année complète nécessite des journées entières. Cette lenteur est due principalement à la communication entre

Simulink et Brahms (essentiellement l'arrêt et la reprise de la machine virtuelle de Brahms), qui a multiplié par 10 les temps de simulation. La simulation du modèle Brahms depuis son composeur dédié est déjà assez lente : 2 minutes pour une journée contrairement à une simulation COMFIE. Le temps simulation du modèle de COMFIE dans Simulink reste assez faible par rapport à une simulation dans son propre environnement: quelques 1/dixièmes de secondes pour simuler une journée.

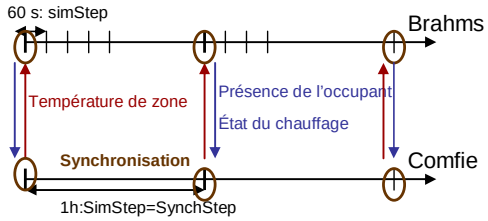


Figure 8. *La synchronisation entre COMFIE et BRAHMS.*

On retrouve les scénarios des déplacements prédéfinis dans BRAHMS et on peut ainsi observer la présence de l'occupant dans la zone principale (Figure 9). On constate également la variation de la consigne de chauffage donnée par l'occupant selon les limites des températures internes atteintes : Extinction si T> Tmax =20.5°C et mise en marche si T< Tmin=19.5°C. Par contre, si l'occupant n'est pas dans la zone chauffée, il ne peut plus agir sur la consigne même si la température dépasse le seuil de confort et elle continue donc d'augmenter pour atteindre des niveaux plus hauts, comme ce qui s'est passé à temps t=14h. On peut ainsi envisager de comparer l'impact énergétique de deux comportements différents de l'occupant celui qui est représenté ici et un autre plus conscient énergétiquement qui éteint le chauffage avant de quitter le local pour une longue durée.

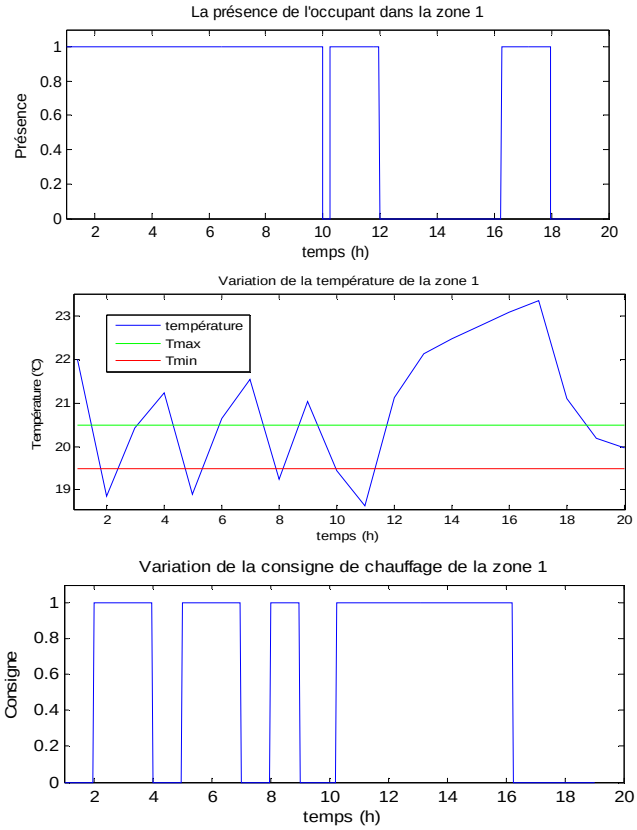


Figure 9. *Les résultats de la co-simulation.*

L'utilisation d'un pas de simulation faible de Brahms, n'avait pas de grande influence sur l'amélioration des résultats de simulation et de sa précision avec cette stratégie de co-simulation vue que la synchronisation entre Brahms et COMFIE ne se fait dans tout les cas que toutes les heures même si on détecte les événements de Brahms avant. L'adoption d'un faible pas de temps de Brahms peut s'avérer efficace avec d'autres méthodes de synchronisation.

- On peut prévoir par exemple d'interrompre, quand c'est possible, une simulation COMFIE dès la détection d'un événement de Brahms (variation de l'état de présence) pour qu'il puisse se synchroniser et récupérer les nouveaux états et reprendre la simulation à partir de l'événement (Figure 10, gauche). Cela pose le problème de reprendre à un instant différent des instants initialement prévus par le modèle. Deux solutions sont envisagées, celle de disposer d'un modèle avec un pas de temps plus faible (6 minutes) ou simplement par une méthode d'interpolation des données en repartant sur un pas d'une heure mais décalé dans le temps.
- On peut également procéder à un retour en arrière lors de la détection d'un évènement tel qu'un franchissement de seuil au niveau des températures. De la même manière, plusieurs solutions sont envisageables en fonction des possibilités du modèle. Une première serait que COMFIE reparte au pas de synchronisation précédant en adoptant un pas de simulation et de synchronisation plus faible pour mieux détecter l'événement (Figure 10, droite). Une deuxième, consiste simplement à une interpolation permettant de détecter le franchissement de seuil et de relancer les modèles à ce pas de temps.

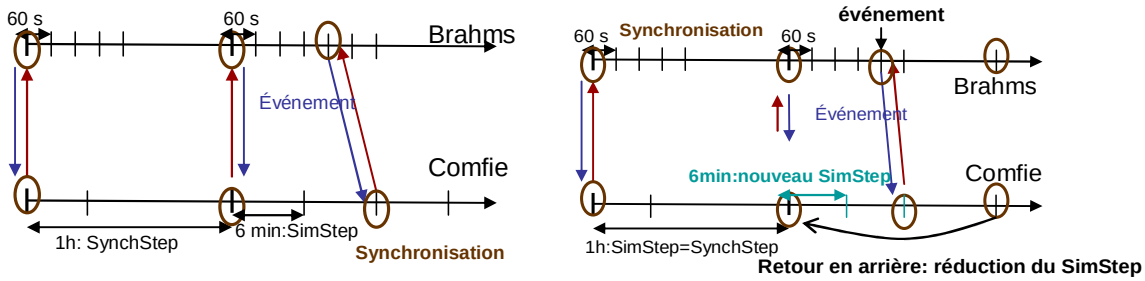


Figure 10. *Stratégies d'orchestration avancées.*

Ces méthodes ont été étudiées pour définir les interfaces de composants afin qu'elles soient compatibles avec des plus larges cas tests. La faisabilité de ces méthodes a été vérifiée, cependant leurs mises en œuvre, ne sont pas actuellement réalisées vu que Simulink ne le permettaient pas facilement. Ce qui nécessite donc le développement d'autres algorithmes de synchronisation plus adaptés.

5. Conclusion

Ces travaux ont été réalisés dans le cadre du projet SIMINTHEC visant à trouver des solutions efficaces d'interopérabilité entre plusieurs domaines du bâtiment (thermique, électrique, usager…). Cet article présente une nouvelle approche de co-simulation basée sur un standard de composant logiciel. Il a démontré la capacité de cette méthode à améliorer la flexibilité des outils et leur coopération pour pouvoir dépasser leurs limites. Il s'est focalisé sur les limites des outils existants comme COMFIE au niveau de la modélisation de l'occupant et l'intérêt d'utiliser des outils plus spécialisés tel que BRAHMS. Une co-simulation entre ces deux outils a été mise en place pour montrer l'intérêt de l'architecture à base de composant. Ce travail s'est focalisé essentiellement sur la standardisation des acteurs de co-simulation sans rentrer dans les détails des orchestrateurs et leurs algorithmes qui peuvent être des perspectives de ce travail. Plusieurs stratégies de co-simulation peuvent être implémentées et testées pour étudier leurs stabilités et comparer leurs performances.

6. Bibliographie

[Aug 94] AUGENBROE G.L.M., « An overview of the COMBINE project », *Acte de la première conférence ECPPM*, Dresden, Germany, 1994, pp.547-554.

[Baz 99] BAZJANAC V., CRAWLEY D.B., « Industry foundation classes and interoperable commercial software in support of design of energy-efficiant buildings An overview of the COMBINE project », *Acte de la première conférence ECPPM*, Dresden, Germany, 1994, pp.547-554.

[Cra 05] CRAWLEY D.B., HAND J.W., KUMMERT M., GRIFFITH B.T., « Contrasting the capabilities of building energy performance simulation programs», la conference *building simulation*, 1999.

[Del 02] DELINCHANT B., WURTZ F., FANDINO J., « Mixing of FEM and Analytical Modeling for the Preliminary Design of a Transformer », *International workshop on OIPE*, Lodz, Poland, 2002.

[Fel 02] FELGNER F., AGUSTINA S., CLADERA B. MERTZ R., LITZ L., « Simulation of thermal building behaviour in Modelica », la deuxième conference *Modelica*, Oberpfaffenhofen, Allemagne, 2002.

[Fmi 10] la spécification des fmi de co-simulation version 1.0, projet Modelisar, 12 Octobre 2010 : http://www.modelisar.com/specifications/FMI_for_CoSimulation_v1.0.pdf.

[Gaa 11 a] GAALOUL S., DELINCHANT B., WURTZ F., THIERS S., Peurportier B., « Strong coupling of a building thermal model with a controlled electrical heater for an hybrid energy simulation », *Electrimacs*, Paris, France, 2009.

[Gaa 11 b] GAALOUL S., DELINCHANT B., WURTZ F., VERDIERE F., « Software components for dynamic building simulation », B*uilding simulation* conference, Sidney, Australie, 2011.

[Hen 04] HENSEN J., Djunaedy E., Radosevic M. Yahiaoui A., « Building performance simulation for better design : some issues and solutions », *Acte de la 21 ème édition de la conference PLEA*, Eindhoven, 9-21 September, pp. 1185-1190.

[Kas 11] KASHIF A., LE X. H. B.., DUGDALE J., PLOIX S., « Agent based framework to simulate inhabitants' behaviour in domestic settings for energy management », *ICAART 2011*, Rome, Italie.

[Mich 09] MICHAELSEN J., EIDEN J., « HumanComfort Modelica-library thermal comfort in buildings and mobile applications » *Modelica conference*, Como, Italie, 2009.

[Szy 96] SZYPERSKI C., *Beyond object-oriented programming*, Boston, Addison-Wesley Professional, 2ème edition 2002.

[Sal 05] SALOMON T., MIKOLASEK R., PEUPORTIER.B., «Outil de simulation thermique du bâtiment, Comfie » *Journée thématique SFT-IBPSA, 2005.*