INTRODUCTION

La théorie de la modélisation et de la simulation (TMS) a été développée dans les années 1970 par le professeur B.P. Zeigler aux États-Unis à l'université d'Arizona (Zeigler, 1976). Associant approche formelle et théorie générale des systèmes, elle a connu un large écho dans les années 1990 auprès de la communauté francophone. On compte aujourd'hui une petite dizaine d'équipes ou de laboratoires français qui font vivre la communauté et développent des méthodes, outils et applications basés sur la formalisation de la TMS à savoir le formalisme DEVS et ses extensions ou déclinaisons. Les domaines d'applications sont très variés ; cette diversité d'utilisation est une des forces de la TMS, qui a su évoluer et étendre ses champs d'utilisation.

Le formalisme DEVS (Zeigler *et al*., 2000) propose une approche formelle facilitant la modélisation et la simulation de systèmes complexes à événements discrets. Ce formalisme est basé sur la théorie générale des systèmes (Bertalanffy, 1968). Il est courant que le formalisme DEVS, dans sa forme originale (Zeigler, 1976), soit adapté et étendu afin d'être replacé dans des contextes plus spécifiques d'un domaine d'application. C'est par exemple le cas quand il s'agit de modéliser des équations différentielles (Kofman, 2000), des systèmes flous (Kwon *et al*., 1996), des systèmes proches des automates cellulaires (Wainer, 2015), etc.

Le formalisme DEVS apporte une solution pour répondre aux enjeux de modélisation multiformalismes, voire de co-simulation. En effet, modéliser un système complexe hétérogène peut nécessiter l'utilisation de plusieurs formalistes (Vangheluwe, 2000), équations différentielles ou algébriques, automate ou réseau de pétri, évolution dans le temps continue ou discrète, etc. Un cadre formel rigoureux est alors nécessaire pour intégrer différents formalismes ou méthode de modélisation et aussi définir les algorithmes des simulateurs. Le formalisme DEVS et ses déclinaisons (DESS, DEV&DESS, DSDE, PDEVS, etc.) permettent de décrire le système à partir d'une sémantique de modélisation (modèle) et de simulation (simulateur) univoque.

Il repose sur la définition de deux types de composants de modélisation : les modèles atomiques M (*cf.* eq. (1)) et les modèles couplés N (*cf.* eq. (2)).

Les modèles atomiques permettent de décrire le comportement du système à étudier à l'aide de fonctions comportementales. M évoluent en fonction d'occurrences d'événements qui engendrent des transitions d'états internes ou externes. C'est une sorte de machine à états.

DOI: 10.3166/tsi.2018.00013

Le modèle atomique M est défini par le *tuple* :

$$X,\ Y,\ S,\ t_a,\ \delta_{int},\ \delta_{\text{ext}},\ \lambda, \tag{1}$$

avec :

– X : l'ensemble des ports d'entrée ;

– Y : l'ensemble des ports de sortie ;

– S : l'ensemble des états du système ;

– t_a : la fonction d'avancement du temps (ou de durée de vie d'un état) ;

– δ_{int} : la fonction de transition interne. Elle permet de passer d'un état s_1 à l'instant t_1, à un état s_2 à l'instant t_2 tant qu'aucun événement externe ne survient durant le temps de vie de l'état $ta(S_1)$;

– δ_{ext} : la fonction de transition externe. Elle spécifie le changement d'état (passage de l'état s_1 à l'état s_2) quand une entrée survient (x) avant que $t_a(s_1)$ ne soit écoulé ;

– Q est l'ensemble des états tels que $\{(e,\ s)|\text{s dans } S,\ 0 \leq e \leq t_a(s)\}$;

– e est le temps passé dans l'état ;

– λ : la fonction de sortie.

Les modèles couplés (N) décrivent la structure et fixent une priorité entre composants du modèle grâce à une fonction adaptée nommée *select*. Ils définissent comment sont interconnectés des sous-modèles (atomiques ou couplés) afin de former un nouveau modèle (couplé). Une hiérarchie de composition est donc possible. Un modèle couplé englobe les informations suivantes :

– l'ensemble des modèles qui le compose D ;

– l'ensemble des ports d'entrée qui recevront les événements externes X ;

– l'ensemble des ports de sortie qui émettront les événements Y ;

– les couplages en ports d'entrée et de sortie des modèles composant le modèle couplé.

N possède la structure suivante :

$$X,\ Y,\ D,\ \{M_d/d \text{ dans } D\},\ EIC,\ EOC,\ IC,\ Select. \tag{2}$$

Les définitions de X et Y sont identiques à celles du modèle atomique. D est l'ensemble des noms des composants (modèles) du modèle couplé. M_d est un modèle DEVS atomique ou couplé. Les variables représentant les entrées et les sorties du modèle seront indexées par l'identifiant du modèle. Les entrées et les sorties du modèle couplé sont connectées aux entrées et sorties des modèles composant le modèle couplé. *EIC* représente l'ensemble des ports d'entrée ip_{N} du modèle couplé connectés aux ports d'entrée ip_{d} des sous-modèles les composant. On a la même situation pour les ports de sortie *EOC*. À l'intérieur du modèle couplé, les sorties d'un modèle peuvent être couplés

aux entrées des autres modèles *IC*. Une sortie d'un modèle ne peut pas être couplée à l'une de ses entrées.

Plusieurs composants de modélisation peuvent être interconnectés entre eux dans un modèle couplé et des événements simultanés peuvent se produire. Or, dans sa formulation originale, le formalisme DEVS impose de briser le lien de causalité entre composants dans le cas d'événements simultanés. Si plusieurs composants interconnectés s'influencent mutuellement, les événements de sortie des influenceurs ne seront pas reçus par les influencés aux mêmes instants. Cela résulte de la définition de la fonction *select* empêchant toute possibilité de simultanéité dans le modèle DEVS original. De plus, un événement externe peut survenir sur les ports d'entrées d'un composant du modèle au même instant que celui déclenchant la transition interne. Il y a donc conflit et la prise en compte de ce conflit doit être décrite dans un algorithme adapté à la charge du modélisateur. Les limites conceptuelles qu'implique cette concurrence sont détaillées dans Zeigler *et al*. (2000). Le formalisme PDEVS (Chow et Zeigler, 1994) a été proposé afin de les dépasser.

Le formalisme PDEVS (pour *Parallel Discrete EVent system Specification*) est une extension du formalisme DEVS. Il est complété d'une fonction (δ_{con}) dont l'objectif est de permettre au modélisateur de gérer les conflits survenant entre événements internes et externes (fonctions δ_{int} et δ_{ext}). PDEVS inclut aussi un mécanisme (structure de données de type sac, X^b sous-ensemble de X) pour gérer les événements d'entrées simultanés. Ce nouvel ensemble permet de collecter des événements émis au même instant. Ainsi, le formalisme PDEVS permet d'exprimer plusieurs événements externes à la même date. Ces événements sont collectés et stockés dans des ensembles d'événements survenus au même instant. Les sorties réalisées par les « modèles imminents », c'est-à-dire les modèles en conflit à un instant donné pour lesquels une transition interne est prévue au même instant, sont stockées dans un sous-ensemble d'entrées noté X^b. Chacun des événements de l'ensemble X^b est identifié par son temps d'occurrence. Aucune relation d'ordre n'est préconisée pour les événements appartenant à un même ensemble. On peut ainsi autoriser et dénombrer les événements simultanés sur chaque port d'entrée X. La prise en compte des événements de l'ensemble n'est autorisée qu'après les transitions internes de tous les modèles imminents.

De ce fait, les transitions externes sont réalisées par des ensembles représentant ainsi la réponse agrégée des événements simultanés.

Le modèle atomique PDEVS M est décrit par un *tuple* :

$$X,\ Y,\ S,\ t_a,\ \delta_{con},\ \delta_{int},\ \delta_{ext},\ \lambda, \tag{3}$$

où :

– X est l'ensemble des ports *ip* et des valeurs d'entrées ;

– Y est l'ensemble des ports *op* et des valeurs de sorties ;

– S est l'ensemble des états partiels du système ;

– t_a est la fonction d'avancement du temps ;

– δ_{int} est la fonction de transition interne ;

– δ_{ext} est la fonction de transition externe où, X est l'ensemble des sacs d'entrées appartenant à X, Q est l'ensemble des états totaux, $Q = \{(s, e) \mid s \text{ dans } S, 0 \leq e \leq t_a(s)\}$, e est le temps écoulé depuis la dernière transition ;

– δ_{con} est la fonction de conflit ;

– λ est la fonction de sortie.

De la même manière que pour la version classique de DEVS, en l'absence d'événements sur les ports d'entrées, le modèle conserve un état passif jusqu'au prochain événement déclenchant la transition interne (δ_{int}). Une sortie est alors générée par la fonction de sortie (λ), suivie de l'exécution de la fonction de transition interne (δ_{int}).

Si un événement externe survient sur l'un des ports d'entrée avant l'instant prévu pour la transition interne, l'état du système passe à δ_{ext} (s, e, x^{b}). À la différence d'une approche DEVS classique, la transition externe recalcule l'état s depuis les ensembles d'événements (X^{b}) provenant d'un ou de plusieurs modèles PDEVS. Si un événement survient sur X à $e = t_a(s)$, le simulateur appelle la fonction $\delta_{\text{con}}(s, e, x^{\text{b}})$. L'algorithme comportemental de la fonction de conflit δ_{con} doit être implémenté par le modélisateur. Par défaut $\delta_{\text{con}} = \delta_{\text{ext}}(\delta_{\text{int}}\ (s,\ e),\ 0,\ x^{\text{b}}))$, cette définition donne ainsi la priorité à la transition interne lors d'un conflit dans un modèle PDEVS.

Au niveau de la simulation, l'une des propriétés importantes du formalisme DEVS est qu'il fournit automatiquement un simulateur pour chacun des modèles. Le formalisme DEVS établit une distinction explicite entre la partie modélisation et les algorithmes de simulation telle que n'importe quel modèle DEVS puisse être simulé sans qu'il ne soit nécessaire d'implémenter un simulateur spécifique. C'est la notion de *simulateur abstrait* définit dans Zeigler *et al.* (2000).

Chaque modèle atomique est associé à un simulateur chargé de gérer le comportement du modèle, et chaque modèle couplé est associé à un coordinateur chargé de la synchronisation temporelle des modèles sous-jacents. L'ensemble des composants de modélisation est géré par un coordinateur spécifique nommé *root* qui centralise et organise l'échéancier de la simulation. L'échéancier est une structure de données composée d'événements classés suivant un ordre chronologique, la tête de l'échéancier représentant le futur immédiat, et la queue le futur plus lointain. La simulation consiste à faire évoluer les états des modèles dans le temps en fonction des événements.

Finalement, un autre avantage du formalisme est sa forte compatibilité avec les propriétés des langages orientés objets. En résulte un nombre important d'implémentations, dans différents langages, souvent adaptées à un contexte d'application. Nous pouvons lister :

– aDEVS (Nutaro, 1999) ;

– CD++ (Wainer, 2002) ;

– fwkDEVS (Bisgambiglia *et al*., 2016) ;

– MS4ME (Zeigler, 2013) ;

– VLE (Quesnel *et al*., 2007) ;

– PyDEVS et PyPDEVS (Bolduc et Vangheluwe, 2002 ; Van Tendeloo et Vangheluwe, 2016a)

– DEVSimPY (Capocchi *et al*., 2011) ;

– PowerDEVS (Bergero et Kofman, 2011) ;

– ProDEVS (Bisgambiglia *et al*., 2016) ;

– DEVS-Ruby (Franceschini *et al*., 2014a) ;

– SimStudio (Traoré, 2008).

Ces outils de modélisation et simulation sont comparés dans Franceschini *et al*. (2014b) et Van Tendeloo et Vangheluwe (2016b).

Ce numéro spécial sur les avancées en TMS fait suite à la première conférence (Journées DEVS francophone) organisée par le réseau DEVS (RED) (Bisgambiglia *et al*., 2016) ; elle a rassemblé en 2016 à l'institut des études scientifiques de Cargèse (IESC) des doctorants, chercheurs et enseignants-chercheurs sur ces thématiques.

Le RED a pour objectif de promouvoir la TMS et les outils associés. Les thématiques du réseau sont :

– l'ingénierie dirigée par modèle ;

– la validation et la certification des simulateurs ;

– la performance des simulateurs ;

– la validation et la vérification des modèles.

Une sélection des meilleurs articles présentés en version longue compose ce numéro :

– le premier article, « Vers une distribution optimale des modèles DEVS dans un contexte pessimiste. De l'apprentissage à la distribution », traite de l'optimisation des simulations DEVS distribuées dans un contexte pessimiste. Les auteurs proposent de mettre en place une approche d'optimisation des simulations distribuées par restructuration de la hiérarchie des modèles à l'aide d'outils de partitionnement de graphes ;

– le deuxième article, « Wrapping DEVS de modèles IP dans MECSYCO pour la co-simulation de systèmes cyber-physiques », aborde les notions de co-simulation, il présente l'intérêt de la plateforme MECSYCO, basée sur le wrapping DEVS, en termes d'intégration de l'hétérogénéité des modèles et des simulateurs : formalismes, représentation du temps et des données, logiciels, etc. ;

– le troisième article, « Intégration de la hiérarchie d'abstraction et de la granularité temporelle au sein de la modélisation et la simulation DEVS », propose une extension du formalisme SES (*System Entity Structure*) afin d'intégrer dans DEVS les concepts de hiérarchie d'abstraction et de granularité temporelle ;

– enfin, le dernier article élargi les thématiques abordées dans ce numéro pour traiter d'un problème d'importance pour tout usagé de la simulation, intitulé « Répétabilité et reproductibilité numérique. Constat, conseils et bonnes pratiques pour le cas des simulations stochastiques parallèles et distribuées », il propose des recommandations et un état des lieux sur les causes de non-reproductibilité numérique et de la non-répétabilité des expériences computationnelles.

Paul-Antoine Bisgambiglia
SPE CNRS – Université de Corse Pasquale Paoli,
Campus Grimaldi, Bâtiment PPDB, 20250 Corte
Gauthier Quesnel
MIAT – INRA Toulouse,
24 Chemin de Borde Rouge – Auzeville CS 52627,
31326 Castanet Tolosan cedex

Bibliographie

Bergero F., Kofman E. (2011). PowerDEVS: a tool for hybrid system modeling and real-time. *Simulation*, vol. 87, p. 113-132. doi: 10.1177/0037549710368029.

Bertalanffy L.V. (1968). *Théorie générale des systèmes*.

Bisgambiglia P.-A., Quesnel G., Duboz R. (2016). Actes des Journées DEVS francophones (JDF2016) : théorie et applications. In: *Workshop RED, Cépaduès*. Cépaduès édtions, Cargèse (Corse).

Bolduc J.S., Vangheluwe H. (2002). *A modeling and simulation package for classic hierarchical DEVS*.

Capocchi L., Santucci J.F., Poggi B., Nicolai C. (2011). DEVSimPy: a collaborative Python software for modeling and simulation of DEVS systems. In: *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2011 20th IEEE International Workshops on*. Presented at the WETICE, IEEE, Paris, p. 170-175. doi: 10.1109/WETICE.2011.31.

Chow A.C.H., Zeigler B.P. (1994). Parallel DEVS: a parallel, hierarchical, modular modeling formalism. In: *Simulation Conference Proceedings, 1994*. Winter, p. 716-722. doi: 10.1109/WSC.1994.717419.

Franceschini R., Bisgambiglia P.-A., Bisgambiglia P.A., Hill D.R.C. (2014a). DEVS-Ruby: a domain specific language for DEVS modeling and simulation (WIP). In: *CD Proceedings of the Symposium on Theory of Modeling & Simulation – DEVS Integrative M&S Symposium, DEVS 14*. Presented at the TMS14, SCS, Tampa, FL, USA.

Franceschini R., Bisgambiglia P.-A., Touraille L., Bisgambiglia P., Hill D. (2014b). A survey of modelling and simulation software frameworks using discrete event system specification. In: Neykova R., Ng N., eds. *2014 Imperial College Computing Student Workshop, Open Access Series in Informatics (OASIcs)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, p. 40-49. doi: http://dx.doi.org/10.4230/OASIcs.ICCSW.2014.40.

Kofman E. (2000). *Discrete event based simulation and control of continuous systems*.

Kwon Y., Park H., Jung S., Kim T. (1996). Fuzzy-DEVS formalisme: concepts, realization and application. *Proc. AIS*, vol. 1996, p. 227-234.

Nutaro J. (1999). aDEVS (A Discrete EVent System simulator). *Ariz. Cent. Integr. Model. Simul. ACIMS Univ. Ariz. Tucson*. Available https://web.ornl.gov/~nutarojj/adevs/.

Quesnel G., Duboz R., Ramat É., Traoré M.K. (2007). VLE: a multimodeling and simulation environment. In: *Proceedings of the 2007 Summer Computer Simulation Conference, SCSC '07*. Society for Computer Simulation International, San Diego, CA, USA, p. 367-374.

Traoré M.K. (2008). SimStudio: a next generation modeling and simulation framework. In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, Simutools '08*. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (ICST), Brussels, Belgium, p. 67:1-67:6.

Van Tendeloo Y., Vangheluwe H.2016a. An overview of PythonPDEVS. In: *Collectif Workshop RED*, Editor JDF, p. 59-66.

Van Tendeloo Y., Vangheluwe H. (2016b). An evaluation of DEVS simulation tools. *Simulation*, 0037549716678330.

Vangheluwe H.L. (2000). DEVS as a common denominator for multi-formalism hybrid systems modelling. In: *Proceedings of ISCACS 2000*.

Wainer G. (2002). CD++: a toolkit to develop DEVS models. *Softw. Pract. Exp.*, vol. 32, p. 1261-1306.

Wainer G.A. (2015). The cell-DEVS formalism as a method for activity tracking in spatial modelling and simulation. *Int. J. Simul. Process Model.*, vol. 10, p. 19-38. doi: 10.1504/IJSPM.2015.068517.

Zeigler B.P. (2013). *Guide to modeling and simulation of systems of systems – User's reference, Springer briefs in computer science*. Springer.

Zeigler B.P. (1976). *Theory of modeling and simulation*. Academic Press, USA.

Zeigler B.P., Praehofer H., Kim T.G. (2000). *Theory of modeling and simulation*, 2nd ed.