Fuzzy-PDEVS: towards handling δcon with conceptual method

Conference Paper · November 2015

4 authors:

Paul-Antoine Bisgambiglia
Université de Corse Pascal Paoli
47 PUBLICATIONS **77** CITATIONS
SEE PROFILE

Romain Franceschini
Université de Corse Pascal Paoli
8 PUBLICATIONS **7** CITATIONS
SEE PROFILE

Damien Foures
Université de Corse Pascal Paoli
10 PUBLICATIONS **18** CITATIONS
SEE PROFILE

Eric Innocenti
University of Corsica, Corsica Institute of Te…
34 PUBLICATIONS **197** CITATIONS
SEE PROFILE

Fuzzy-PDEVS : vers une méthode conceptuelle pour la gestion de conflits

Fuzzy-PDEVS: towards handling δ_{con} with conceptual method

P.-A. Bisgambiglia[1] R. Franceschini[1] D. Foures[1,2] E. Innocenti[1]
[1] UMR SPE 6134 CNRS – Université de Corse Pasquale Paoli

[1]Université de Corse – CNRS UMR SPE – équipe TIC
[2]Université de Toulouse – LAAS CNRS

Campus Grimaldi, Bat. Conrad - Boîte aux lettres 81 - 20250 Corte (France), bisgambiglia@univ-corse.fr

Résumé :
Dans cet article, nous démontrons qu'une ambiguïté de conception lors de la réalisation d'un modèle PDEVS peut avoir un impact important sur les résultats de simulation. Nous proposons une solution, nommée Fuzzy-PDEVS, basée sur un coefficient de confiance afin d'évaluer le comportement du modèle et d'aider le modélisateur dans sa tâche.

Mots-clés :
PDEVS, DEVS, modèle, conflit, croyance, crédibilité

Abstract:
In this work, we demonstrate that a design ambiguity when coding a model can have a significant impact on simulation results. We propose a method based on a belief rate to evaluate the model and help the modeller in its design task.

Keywords:
PDEVS, DEVS, model, confluent, belief, credibility

Introduction

Cet article a pour objectif de poser les bases conceptuelles d'une approche décisionnelle pour la simulation de systèmes à évènements discrets fondée sur le formalisme PDEVS (Parallèle DEVS [1]). PDEVS est une évolution du formalisme DEVS (Discrete EVent system Specification [2]) pour les systèmes parallèles et distribués.
Dans ce travail exploratoire, nous souhaitons proposer une évolution de PDEVS visant à évaluer la confiance potentielle accordée à un modèle ou à son modélisateur. L'objectif est de faciliter l'expression des fonctions comportementales du formalisme PDEVS, et notamment la fonction de conflit (δ_{con}). En effet, dans sa forme originale, le formalisme PDEVS possède une fonction de conflit (ou de confluence) qui peut engendrer des ambiguïtés lors du processus de modélisation ou d'utilisation du modèle. Celle-ci permet de gérer et de prioriser l'exécution des fonctions comportementales d'un modèle PDEVS (δ_{int}, δ_{ext}) lorsque surviennent deux événements simultanés. Ce mécanisme de gestion est trop générique pour pouvoir exprimer sans ambiguïté un comportement concurrent. En effet, il est à la charge du modélisateur ou de l'utilisateur, d'exprimer comme il l'entend l'algorithme comportemental concurrent. Un algorithme séquentiel est proposé par défaut dans PDEVS (δ_{int} puis δ_{ext} ou δ_{ext} puis δ_{int}). Cependant, dans de nombreux cas, le modélisateur éprouve trop souvent un problème de conception quant à la logique à mettre en œuvre. La source de cette ambiguïté peut être très variée, mauvaise connaissance du système, utilisation dans un contexte incertain, mauvais modélisateur, etc.
À partir d'un cas d'étude didactique, nous allons montrer que l'ordre d'exécution des fonctions comportementales δ_{int} et δ_{ext} peut avoir une influence non négligeable sur les résultats d'une simulation. L'approche que nous proposons repose sur l'intégration dans les modèles PDEVS d'un coefficient nommé B_r. Nous souhaiterions définir ici B_r comme le niveau de

confiance accordé au modèle vis-à-vis des objectifs définis en rapport avec un cadre expérimental explicite, ou encore la confiance accordée au modèle par l'utilisateur. Nous proposons une évolution du formalisme PDEVS, qui va intégrer et utiliser le coefficient B_r.

Dans une première partie, nous rappelons les principes du formalisme DEVS et son évolution parallèle et distribuée PDEVS. Nous soulignons ses avantages et ses limites. Ensuite, nous proposons un exemple de simulation à partir d'un modèle théorique. L'objectif de notre exemple, est de dresser un constat sur les ambiguïtés de modélisation introduites dans le formalisme PDEVS. Puis, nous ferons part de notre réflexion pour pallier à ces problèmes, et aider le modélisateur dans ses choix. Enfin, après une conclusion, nous énoncerons nos perspectives pour finaliser l'approche proposée.

1 Background

1.1 Le formalisme DEVS

Le formalisme DEVS [2] propose une approche formelle facilitant la modélisation et la simulation de systèmes complexes à évènements discrets. Ce formalisme est basé sur la théorie des systèmes. Il est courant que le formalisme DEVS, dans sa forme originale, 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 systèmes flous [3], [4], ou d'autres types de systèmes [5], [6].

Le formalisme DEVS repose sur la définition de deux types de composants de modélisation: les modèles atomiques (M cf formule 1) et les modèles couplés. Les modèles atomiques permettent de décrire le comportement du système à étudier à l'aide de fonctions. M évoluent en fonction d'occurrences d'événements qui engendrent des transitions d'états internes ou externes.

Un modèle atomique est défini par le tuple : M $< X, Y, S, t_a, \delta_{int}, \delta_{ext}, \lambda >$ (1)

Avec :

- X : l'ensemble des ports d'entrée ;
- Y : l'ensemble des ports de sortie ;
- S : l'ensemble des états du système;
- $t_a : S \rightarrow R^+$ la fonction d'avancement du temps (ou de durée de vie d'un état) ;
- $\delta_{int} : S \rightarrow S$ 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 lorsqu'aucun évènement externe ne survient durant le temps de vie de l'état $t_a(S_1)$;
- $\delta_{ext} : Q \times X \rightarrow S$ la fonction de transition externe. Elle spécifie comment le modèle change d'état (passage de l'état s_1 à l'état s_2) quand une entrée survient (x) avant que $ta(s_1)$ ne soit écoulé ; Q est l'ensemble des états tels que $\{(e, s)|s \text{ dans } S, 0 \leq e \leq t_a(s)\}$; e est le temps passé dans l'état.
- $\lambda : S \rightarrow Y$ la fonction de sortie ;

Les modèles couplés décrivent la structure, et fixent une priorité entre composants du modèle grâce à une fonction adaptée nommée « select ». Plusieurs composants de modélisation peuvent donc être interconnectés entre eux dans un modèle couplé, et des évènements simultanés peuvent se produire. Cependant, 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 [7].

Le formalisme PDEVS a été proposé afin de les dépasser.

1.2 Le formalisme PDEVS

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 évènements 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 évènements d'entrées simultanés. Ce nouvel ensemble permet de collecter des événements émis au même instant. Ainsi, le formalisme PDEVS, permet de spécifier plusieurs événements externes à une 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 conflits à un instant donnée 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 évènements 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, δ_{con}, δ_{int}, δ_{ext}, λ > (2)
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,
- $t_a : S \rightarrow R^+$ est la fonction d'avancement du temps,
- $\delta_{int} : S \rightarrow S$ est la fonction de transition interne,
- $\delta_{ext} : Q \times X^b \rightarrow S$ est la fonction de transition externe où, X^b est l'ensemble des sacs d'entrées appartenant à X, Q est l'ensemble des états totaux, Q = {(s, e) | s dans S, $0 \leq e \leq t_a(s)$}, e est le temps écoulé depuis la dernière transition
- $\delta_{con} : S \times X^b \rightarrow S$ la fonction de conflit,
- $\lambda : S \rightarrow Y$ 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 évènement 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 modèle(s) PDEVS. Si un événement survient sur X à e = t_a(s), le simulateur appelle la fonction δ_{con}(s, e, x^b). L'algorithme comportemental de la fonction de conflit δ_{con} doit être implémenté par le modélisateur. Par défaut $\delta_{con} = \delta_{ext}(\delta_{int}$ (s, e), 0, x^b), donnant ainsi la priorité à la transition interne lors d'un conflit dans un modèle PDEVS.
L'objectif de ce travail est d'exprimer ce choix en le soumettant à une fonction d'évaluation F_{eval}.

1.3 DEVS et le flou

Le formalisme DEVS a déjà été combiné avec les *théories de l'incertain* [3], [8], [9]. Nous pouvons citer notamment les travaux concernant le formalisme Fuzzy-DEVS [3]. Celui-ci est basé sur la logique floue, intègre une règle "max-min", des méthodes de fuzzification et de défuzzification afin de gérer les incertitudes sur le temps des évènements. Tel qu'il est spécifié, un modèle atomique Fuzzy-DEVS ne permet de prendre en compte

que les différentes possibilités de transitions entre états. Dans Fuzzy-DEVS, les entrées et les sorties du modèle ne sont pas représentées comme floues. Le modèle atomique flou Fuzzy-DEVS, contrairement à son pendant DEVS, est non déterministe, c'est-à-dire qu'il ne répond pas aux deux conditions suivantes :

1. la fonction de transition interne δ_{int} est exécutée quand la durée de vie de l'état est écoulée (e = 0) et que la fonction de transition externe δ_{ext} est exécutée lorsqu'un évènement externe survient ;
2. la fonction de sortie λ est exécutée quand la durée de vie d'un état est finie ($e = t_a$).

Dans le formalisme Fuzzy-DEVS, l'état suivant du système n'est pas déterminé par les fonctions δ_{int} et δ_{ext} mais avec une règle "max-min" ; Celle-ci exprime sous la forme d'un algorithme, l'état le plus probable nécessitant une mise à jour. Les différentes possibilités de changement d'état sont habituellement représentées par des matrices, et l'évolution d'un modèle FuzzyDEVS par des arbres de possibilités. Les feuilles représentent les états et les branches expriment la possibilité associée.

La figure 1 présente l'évolution d'un tel modèle en fonction des possibilités associées aux fonctions de transition δ_{int} et δ_{ext}. Le modèle est défini selon deux entrées $X = \{x_1, x_2\}$, deux sorties $Y = \{y_1, y_2\}$ et trois états discrets possibles $S = \{s_1, s_2, s_3\}$.

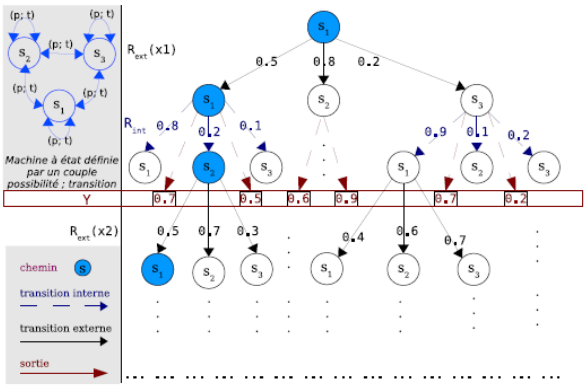

Figure 1 – Une modélisation floue de type Fuzzy-DEVS

L'état suivant du modèle de la figure 1, après survenue d'une entrée (x_1) et déclenchement de la fonction de transition externe, sera choisi en fonction du degré de possibilité défini par :

- *max(min(0.5,0.8), min(0.5,0.2)) = 0.5*

Nous restons donc dans l'état s_1.

Après une transition interne :

- *max(min(0.8,0.2), min(0.8,0.1)) = 0.2*

Nous passons dans l'état s_2.

Remarquons que comme dans le formalisme DEVS original, avant une transition interne δ_{int}, la fonction de sortie λ doit être exécutée.

D'autres travaux comme [8] proposent de prendre en compte les imprécisions des données d'entrées ; Dans [9]–[12] les auteurs proposent d'intégrer des modèles de contrôle ou d'optimisation basés sur les systèmes d'inférence. Dans tous les cas, nous notons qu'il n'y a pas de travaux dans le domaine qui proposent d'aider à l'implémentation de la fonction de conflit ou qui associent PDEVS et la logique floue.

2 Cas d'étude théorique

L'objectif est ici de démontrer, au travers d'un exemple théorique, que l'ordre d'exécution des fonctions de transition (δ_{int} et δ_{ext}) choisi par le modélisateur engendre des résultats de simulation qui peuvent être différents. Nous avons choisi d'utiliser un modèle théorique de croissance de stock, qui repose sur l'équation : $\varphi_{t+n} = \varphi_t + r \times \varphi_t \times (1 - (\varphi_t / K)$ avec r un coefficient d'accroissement de la population, et K la quantité que peut absorber le milieu. Cette équation est issue de [13].

Le modèle PDEVS considéré est défini tel que :

- X : les entrées $\{x_1 ; x_2\} \in X$ qui expriment un prélèvement sur une quantité φ stocké dans le modèle PDEVS,
- Y : modélise une valeur de sortie qui permet de lire la quantité φ
- $\varphi \in S$: une variable d'état qui définit une quantité (de combustible, de poissons, etc.), initialement $\varphi = 80$ unités.
- δ_{int} exprime la croissance φ (évolution de population) conformément à : $\varphi_{t+n} = \varphi_t + r \times \varphi_t \times (1 - (\varphi_t / K)$ avec $r = 0,5$ et $K = 100$. Nous proposons dans le tableau 1 un

exemple d'évolution à partir d'une valeur initiale de $\varphi = 80$.

- δ_{ext} engendre une diminution de φ elle représente un prélèvement fixé aléatoirement à *10%* pour x_1, *2%* pour x_2.
- δ_{con} est utilisée avec son comportement par défaut. En cas de conflit entre un événement externe et interne nous privilégierons δ_{int} puis δ_{ext}
- $t_a = 50$

Tableau 1: Evolution de la quantité en fonction de l'équation proposée

temps	0	50	100	150	200	250	300
φ	80	88	93,3	96,4	98,1	99,1	99,5

2.1 Fonction de conflit

Nous proposons une simulation à temps discret (voir tableau 2) avec un grand nombre de conflits. Tous les 50 pas de temps, le modèle doit exécuter sa fonction de transition interne et il reçoit au même moment une demande de prélèvement en X. Les résultats sont présentés dans le tableau 2 ci-dessous :

Tableau 2 : Exemple d'évolution du modèle PDEVS avec la fonction de conflit qui exécute la transition interne puis externe

temps		0	50	100	150	200	250	300
φ	*après* δ_{int}	80	88,0	87,4	87,1	86,8	86,7	86,6
φ	*après* δ_{ext}	80	79,2	78,7	78,4	78,2	78,0	77,9

Il est assez simple de tester le scénario inverse, dont les résultats sont indiqués dans le tableau 3:

Tableau 3 : Exemple d'évolution du modèle PDEVS avec la fonction de conflit qui exécute la transition externe puis interne

temps		0	50	100	150	200	250	300
φ	*après* δ_{ext}	80	72,0	73,9	75,2	76,1	76,6	77,0
φ	*après* δ_{int}	80	82,1	83,5	84,5	85,2	85,6	85,9

Nous constatons que le comportement du modèle entre la simulation 1 (S1) et la simulation 2 (S2), présente un différentiel d'environ 10% (cf. tableau 4).

Tableau 4 : Erreur entre les deux simulations

$\lvert\varphi_{s1} - \varphi_{s2}\rvert$	0,0	2,9	4,8	6,1	7,0	7,6	7,9
$\lvert\varphi_{s1} - \varphi_{s2}\rvert / \varphi_{s1}$	0	0,04	0,06	0,08	0,09	0,10	0,10

Le cas d'étude présenté est adapté à l'exécution de la fonction de conflit, d'où l'utilisation du temps discret, et le partage entre les deux fonctions de transition d'une ressource unique φ. Dans d'autres cas d'étude, notamment si les fonctions de transition n'impactent pas la même ressource, les résultats seraient très différents. Il découle de ce constat une forte contrainte qui doit permettre, en fonction du modèle, d'activer ou non notre approche.

2.2 Entrées simultanées

Dans le cas d'entrées simultanées, la problématique est relativement similaire à la précédente, puisqu'il s'agit de récupérer au même instant, plusieurs valeurs d'entrées. Outre la fonction de gestion de conflits, PDEVS inclus pour cela une structure de données de type *sous-ensemble*, appelée *sac* qui permet la collecte des valeurs d'entrée x. Le formalisme PDEVS laisse le choix au modélisateur d'implémenter une méthode spécifique de prise en compte (sélection). Il peut alors opter pour une des stratégies suivantes:

1. ne rien faire ;
2. vider séquentiellement le *sac* ;
3. élire un des éléments du *sac* (définir des priorités) ;
4. établir une stratégie pour chaque sac recevable ;

Nous nous plaçons ici dans le cadre d'une gestion séquentielle des entrées, où deux évènements simultanés impactent la même ressource.

Tableau 5 : Exemple d'évolution avec deux entrées simultanées et prise en compte de l'entrée 1 puis l'entrée 2

temps		0	50	100	150	200	250	300
φ	*après* δ_{int}	80,0	88,0	86,3	85,2	84,5	84,0	83,7
φ	*après* x_1	80,0	79,2	77,7	76,7	76,0	75,6	75,3
φ	*après* x_2	80,0	77,6	76,1	75,2	74,5	74,1	73,8

Sur la base du même modèle, nous avons ajouté une autre entrée x_2 qui va elle aussi appliquer un prélèvement de 2% au même moment que x_1. Nous pouvons constater sur les tableaux 5, 6 et 8 que les résultats du modèle de production ne sont pas impactés mais que les quantités prélevées par x_1 et x_2 (voir tableau 8) présentent des différences.

Tableau 6 : Exemple d'évolution avec deux entrées simultanées et prise en compte de l'entrée 2 puis l'entrée 1

temps		0	50	100	150	200	250	300
φ	*après* δ_{int}	80,0	88,0	86,3	85,2	84,5	84,0	83,7
φ	*après* x_2	80,0	86,2	84,6	83,5	82,8	82,3	82,0
φ	*après* x_1	80,0	77,6	76,1	75,2	74,5	74,1	73,8

Par contre, nous pouvons voir dans le tableau 7 qu'en utilisant l'union des deux entrées (addition des événements du sac), il y a un impact sur l'évolution de la quantité φ.

Tableau 7 : Exemple d'évolution avec deux entrées simultanées et prise en compte de l'union de l'entrée 2 et l'entrée 1

temps		0	50	100	150	200	250	300
φ	*après* δ_{int}	80,0	88,0	80,8	86,2	81,4	85,0	81,8
φ	$x_1 \cup x_2$	80,0	70,4	77,4	71,1	75,8	71,6	74,8

Le tableau 8 détaille les différences entre ces trois exemples de simulation, nous avons noté :

- s1 la première simulation qui donne une priorité à x_1 sur x_2
- s2 la seconde simulation qui donne une priorité à x_2 sur x_1, et
- s3 la troisième simulation avec l'union de x_1 et x_2 soit un prélèvement de 12%.

Tableau 8 : Différentiel entre les trois simulations (s1, s2, s3)

$\lvert x_{1\,s1} - x_{1\,s2} \rvert$	0,0	1,6	1,6	1,5	1,5	1,5	1,5
$\lvert x_{2\,s1} - x_{2\,s2} \rvert$	0,0	8,62	8,46	8,35	8,28	8,23	8,20
$\lvert \varphi_{s1} - \varphi_{s3} \rvert$	0,0	0,00	5,48	0,97	3,10	0,98	1,91
$\lvert \varphi_{s2} - \varphi_{s3} \rvert$	0,0	0,00	5,48	0,97	3,10	0,98	1,91

Cette problématique peut être, et est d'ailleurs actuellement totalement gérée par le modélisateur. L'intérêt de notre proposition est d'essayer d'offrir une méthodologie d'aide à la conception pour adapter le comportement du modèle et la simulation à un contexte particulier.

3 Une proposition et des perspectives

Le cas d'étude précédent, nous permet d'identifier trois problématiques majeures quant à l'utilisation du formalisme PDEVS : (1) La principale : comment automatiser, ou aider le modélisateur à prendre en compte des conflits entre fonctions de transitions ou entre entrées simultanées ? Une solution est de définir un coefficient B_r pour chaque modèle et d'utiliser ce coefficient pour définir les actions à mener. Cette proposition entraine deux sous problématiques : (2) comment définir le coefficient ? (3) comment le faire évoluer au cours de la simulation (influenceurs et influencés) ?
Nous ne traitons dans ce travail que la première problématique (problématique principale).

Nous définissons un modèle Fuzzy-PDEVS tel que : $< X, Y, S, B_r, t_a, \delta_{con}, \delta_{int}, \delta_{ext}, \lambda >$
En fonction du niveau de connaissances sur le système, B_r sera définit par le modélisateur comme un coefficient. Il est possible de faire évoluer B_r au cours de la simulation en fonction des influences des autres composants du modèle. Si B_r vaut 1 le modèle Fuzzy-PDEVS doit être considéré comme "crédible".
Afin d'illustrer cette solution, nous l'avons appliqué au cas d'étude précédent. Dans les cas les plus simples, cette solution peut s'apparenter à une gestion de priorités, nous exécutons la fonction de transition du modèle ayant le coefficient le plus élevé. Un changement d'état entraîne la mise à jour de B_r en appliquant une fonction définie par le modélisateur.

3.1 Fonction de conflit

Soit M_1 le modèle de production, δ_{int} et δ_{ext} ses fonctions de transition et $M_1(B_r)$ son

coefficient. La fonction de conflit engendre les cas suivants :

- δ_{int} puis δ_{ext} , cas par défaut si $M_1(B_r) = 1$; ou $M_1(B_r) = M_2(B_r)$;
- δ_{ext} puis δ_{int} si $M_1(B_r) < M_2(B_r)$;
- aucune action $\emptyset$;
- une action spécifique.

En ce qui concerne l'expression de la fonction d'évolution du coefficient B_r, nous laissons le choix au modélisateur d'utiliser soit une fonction d'agrégation (Min, Max, Moyenne, etc.), soit une règle de combinaison.

3.2 Entrées simultanées

Soit M un modèle Fuzzy-PDEVS et X l'ensemble de ses entrées. Les entrées simultanées engendrent les différentes possibilités suivantes:

- de vider X séquentiellement en fonction de B_r;
- de n'exécuter que le sous-ensemble de X ayant le B_r le plus grand;
- de "combiner" les sous-ensembles de X.

3.3 Exemple

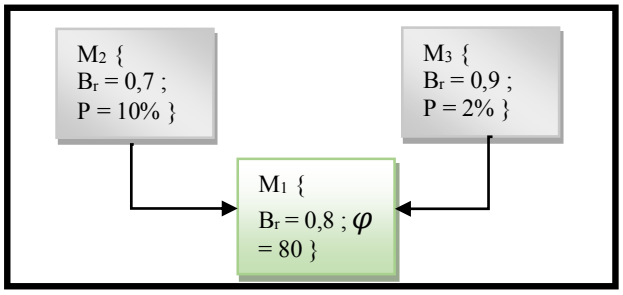

Figure 2 – Modèles $\{M_1 ; M_2 ; M_3\}$

La figure 2 est un exemple de modèle Fuzzy-PDEVS possédant différents coefficients. L'utilisation de deux générateurs en entrée (M_2, M_3) influence beaucoup les résultats car ceux-ci sont fixes et n'évoluent pas pendant la simulation. Le coefficient B_r du modèle de production (M_1) va donc en fonction de la "règle de combinaison" fixée converger vers 0.7 ou 0.9. Toutefois, nous présentons également un cas d'étude dans lequel nous faisons varier le coefficient B_r de l'un des deux générateurs. Cela nous permet d'expliciter plus avant notre démarche.

Dans le tableau 9 nous présentons les résultats obtenus avec une fonction d'agrégation *max* afin de mettre à jour $M_1(B_r)$.

A t = 50 le simulateur reçoit deux entrées X {0.7 ; 0.9} et doit aussi exécuter δ_{int} {0.8}. Il doit faire un choix. δ_{con} est alors exécutée : max ({0.8} ; {0.7 ; 0,9}). C'est donc δ_{ext} qui va être exécutée puis δ_{int}. Ces deux fonctions, vont venir mettre à jour φ en y appliquant un prélèvement (Pt) puis une production (Pr), $\varphi_{t+50} = \varphi_t - Pt + Pr$.

Enfin, le coefficient B_r de M_1 va être mis à jour, B_r = max ({0.8} ; {0.7 ; 0,9})

A t = 100 comme $M_1(B_r)$ = Max ($M_2(B_r)$; $M_3(B_r)$), le comportement par défaut est appliqué dans δ_{con}, et φ est mise à jour $\varphi_{t+50} = \varphi_t + Pr - Pt$.

$M_1(B_r)$ n'évoluant plus jusqu'à t = 300, le même comportement est reproduit.

A t = 300, le B_r du modèle M_1 est modifié par l'utilisateur pour prendre une valeur supérieure. Dans ce cas, δ_{ext} est à nouveau privilégiée par rapport à δ_{int}.

Tableau 9 : Exemple de simulation

temps		0,0	50,0	100,0	150,0	200,0	250,0	300,0
φ		80,0	φ -Pt+Pr	φ +Pr-Pt	φ +Pr-Pt	φ +Pr-Pt	φ +Pr-Pt	φ -Pt+Pr
	B_r	0,8	0,9	0,9	0,9	0,9	0,9	0,95
	δ_{con}	δ_{ext} puis δ_{int}	δ_{int} puis δ_{ext}	δ_{int} puis δ_{ext}	δ_{int} puis δ_{ext}	δ_{int} puis δ_{ext}	δ_{int} puis δ_{ext}	δ_{ext} puis δ_{int}
	δ_{int}	{0,8}	{0,9}	{0,9}	{0,9}	{0,9}	{0,9}	{0,9}
x^b	δ_{ext}	{0,7 ; 0,9}	{0,7 ; 0,9}	{0,8 ; 0,8}	{0,8 ; 0,8}	{0,9 ; 0,7}	{0,9 ; 0,7}	{0,95 ; 0,6}

4 Conclusions et perspectives

Dans ce travail exploratoire, nous proposons une approche conceptuelle et théorique pour aider le modélisateur à lever les ambiguïtés de conception de modèles PDEVS. Nous posons les bases d'une approche décisionnelle pour la simulation de systèmes à évènements discrets basée sur le formalisme PDEVS. PDEVS propose deux mécanismes de gestion des fonctions comportementales. Ce 'flou' peut entraîner une influence non négligeable sur les résultats d'une simulation. Nous proposons une solution, consistant à ajouter à chacun des composants PDEVS un coefficient B_r. Un cas

d'étude est présenté. Au cours de la simulation, B_r est évalué et permet de définir automatiquement les actions à entreprendre en cas de conflit et donc d'incertitude.

Nous souhaitons maintenant orienter notre travail afin de transformer le coefficient B_r en masse de croyance. À partir de la définition suivante : « *Pour modéliser et quantifier la crédibilité attribuée à des faits, dont on ne connaît pas la probabilité d'occurrence, on utilise des fonctions de croyance, qui indiquent un ordre dans la confiance attribuée à ces faits.* » [14].

Nous pouvons raisonnablement penser que la théorie des fonctions de croyance est un cadre formel et/ou théorique pouvant accompagner notre démarche. Dans ce cas B_r deviendrait une masse de croyance fixant ainsi la confiance que le modélisateur attribut à son modèle.

Cette perspective soulève d'autres problématiques : (1) comment estimer le niveau de croyance des modèles ? (2) et comment définir l'influence d'un composant du modèle Fuzzy-PDEVS sur le suivant ? Dans le premier cas, nous pensons utiliser des travaux permettant de définir le niveau de validité des modèles [15], [16]. Il serait aussi possible d'évaluer le modèle en fonction de nos connaissances sur le système et sur le cadre expérimental. Il est également possible de combiner ces premiers éléments avec un degré de confiance lié au modélisateur lui-même, et même dans le cas de modèle à états finis de définir un B_r par état.

Dans le second cas, nous pensons étudier l'impact de l'utilisation de fonctions d'agrégation ou de la règle de combinaison de Dempster [17], [18].

Références

[1] A. C. Chow et B. P. Zeigler, « Revised DEVS : A Parallel Hierarchical Modular Modeling Formalism », 2003.

[2] B. P. Zeigler, H. Praehofer, et T. G. Kim, *Theory of Modeling and Simulation, Second Edition*. 2000.

[3] Y. Kwon, H. Park, S. Jung, et T. Kim, « Fuzzy-DEVS Formalisme : Concepts, Realization and Application », *Proc. AIS 1996*, p. 227–234, 1996.

[4] P.-A. Bisgambiglia, B. Poggi, et C. Nicolai, « Models-Based Optimization Methods for the Specification of Fuzzy Inference Systems in Discrete EVent Simulation », in *Proceedings in the 7th conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-2011) and (LFA-2011)*, 2011, vol. 1-1, p. 957-964.

[5] G. Quesnel, R. Duboz, É. Ramat, et M. K. Traoré, « VLE: A Multimodeling and Simulation Environment », in *Proceedings of the 2007 Summer Computer Simulation Conference*, San Diego, CA, USA, 2007, p. 367–374.

[6] K. Al-Zoubi et G. Wainer, « Distributed simulation of DEVS and Cell-DEVS models using the RISE middleware », *Simul. Model. Pract. Theory*, vol. 55, p. 27-45, juin 2015.

[7] M. Akplogan, *Approche modulaire pour la planification continue : application à la conduite des systèmes de culture*. Toulouse 3, 2013.

[8] P.-A. Bisgambiglia, E. de Gentili, P. A. Bisgambiglia, et J.-F. Santucci, « Fuzz-iDEVS: Towards a fuzzy toolbox for discrete event systems », in *Proceedings of the SIMUTools'09, Rome (Italie)*, 2009.

[9] B. P. Zeigler, Y. Moon, V. L. Lopes, et J. Kim, « DEVS approximation of infiltration using genetic algorithm optimization of a fuzzy system », *Math. Comput. Model.*, vol. 23, nº 11-12, p. 215-228, juin 1996.

[10] P.-A. Bisgambiglia, L. Capocchi, P. Bisgambiglia, et S. Garredu, « Fuzzy inference models for Discrete EVent systems », in *FUZZ-IEEE*, 2010, p. 1-8.

[11] J. F. Santucci et L. Capocchi, « Fuzzy Discrete-Event Systems Modeling and Simulation with Fuzzy Control Language and DEVS Formalism », in *Sixth International Conference on Advances in System Simulation (SIMUL2014)*, Nice, France, 2014, p. 250-255.

[12] G. Kim et P. Fishwick, « Validation method using fuzzy simulation in an object-oriented physical modeling framework », *Enabling Technol. Simul. Sci. II*, vol. 3369, p. 143-153, 1998.

[13] M. B. Schaefer, « Some aspects of the dynamics of populations important to the management of the commercial marine fisheries », *Bull. Math. Biol.*, vol. 53, nº 1-2, p. 253-279, 1991.

[14] B. Bouchon-Meunier, *Logique floue et ses applications*. Addison-Wesley, 1995.

[15] D. Foures, V. Albert, et A. Nketsa, « Simulation validation using the compatibility between Simulation Model and Experimental Frame », in *Proceedings of the 2013 Summer Computer Simulation Conference*, 2013, p. 55.

[16] M. Olsen et M. Raunak, « A Method for Quantified Confidence of DEVS Validation », in *Proceedings of SpringSim TMS/DEVS. 2015*, USA, 2015.

[17] G. Shafer, *A mathematical theory of evidence*. 1976.

[18] P. Smets et R. Kennes, « The transferable belief model », *Artif. Intell.*, vol. 66, nº 2, p. 191-234, avr. 1994.