DEVS-Flou: a Discrete Events and Fuzzy Sets Theory-Based Modeling Environment

Paul-Antoine Bisgambiglia, E. de Gentili, J.F. Santucci, P. Bisgambiglia

DEVS-FLOU : ENVIRONNEMENT DE MODELISATION BASEE SUR LE PARADIGME DES EVENEMENTS DISCRETS ET SUR LA THEORIE DES ENSEMBLES FLOUS

P. A. BISGAMBIGLIA, E. DE GENTILI, J.F. SANTUCCI, P.A. BISGAMBIGLIA

UMR SPE CNRS 6134
Faculté des sciences,
Quartier Grossetti 20250 Corte, Corse
{bisgambiglia/gentili/santucci/bisgambi}@univ-corse.fr

Abstract : *L'objectif de cet article est de présenter une approche générique de multi-modélisation et de simulation de systèmes complexes. Afin de prendre en compte des paramètres flous ou mal connus, nous nous proposons de définir une approche permettant l'ajout d'une technique basée sur la théorie des ensembles flous, dans un environnement de multi-modélisation JDEVS basé sur le formalisme unificateur DEVS.*

Keywords : *Environnement de multi-modélisation, DEVS, Théorie des ensembles flous, systèmes complexes*

1. INTRODUCTION

L'objectif de cet article est de proposer une approche générique de multi-modélisation et de simulation de systèmes complexes. La représentation des systèmes est réalisée à partir d'un ensemble de modèles. Un modèle décrit en utilisant un formalisme spécifique, est en général mieux adapté pour décrire et simuler une partie d'un système. La composition de différents types de modèles, ou de différentes techniques et leurs abstractions à différents niveaux est donc indispensable pour pouvoir étudier un système dans son ensemble. Une telle association de modèles est appelée multi-modèles.

L'étude de systèmes complexes, nécessite donc l'intégration de plusieurs techniques au sein d'un modèle. Pour prendre en compte des paramètres flous ou mal connu, nous nous proposons de définir une approche permettant l'ajout d'une technique basée sur la théorie des ensembles flous, dans l'environnement de multi-modélisation JDEVS développé dans notre laboratoire. Ceux-ci afin de traiter l'incertitude de tels paramètres. L'environnement JDEVS s'appuie sur le formalisme unificateur DEVS (Discrete EVent System specification) introduit par le Professeur Zeigler.

La première partie de cet article introduit les concepts de base de nos travaux : DEVS, JDEVS, la théorie des ensembles flous, et le formalisme fuzzy-DEVS. La seconde partie propose une vision générale de notre environnement de modélisation. La troisième partie présente notre méthodologie de modélisation floue. Enfin, une conclusion reprend les points essentiels de notre travail et fournit des perspectives de travaux restant à effectuer.

2. ETAT DE L'ART

Dans cette partie nous introduisons les différentes approches et travaux à la base de notre étude. Nous décrivons dans un premier temps le formalisme DEVS, puis l'environnement générique JDEVS. La troisième partie est consacrée à la théorie des ensembles flous et dans la dernière nous présentons une approche fuzzy-DEVS mettant en œuvre la théorie du flou dans un environnement DEVS.

2.1. Le formalisme DEVS

Les concepts de base de l'approche DEVS sont destinés à contrôler la difficulté du problème étudié, en réduisant l'analyse du système, à l'étude d'une somme de sous-systèmes plus simples. Notre environnement basé sur la notion de hiérarchie, permet ainsi d'appréhender la complexité inhérente à un problème de manière tout à fait graduelle.

La représentation des systèmes complexes est généralement difficile à mettre en œuvre. En effet, celle-ci implique la prise en compte d'une multitude d'éléments, reliés les uns aux autres par de nombreuses connections. Pour palier ce problème, nous utilisons l'approche de modélisation hiérarchique introduite par B.P. Zeigler, qui a pour objectif l'introduction graduelle des composants du système par masquage successifs des sous composants.
Cette approche utilise la notion de Modèle Atomique et de Modèle Couplé. Un modèle atomique permet de rendre compte des comportements d'entrées/sorties et des changements d'états du système étudié. Les modèles couplés décrivent comment connecter de manière hiérarchique plusieurs composants (modèles atomique et/ou modèles couplés) pour obtenir un modèle couplé de plus haut niveau.
Les composants communiquent entre eux par l'envoi et la réception de messages. Ces messages peuvent être

(1) soit des messages internes, message d'un modèle à destination de lui-même, pour prendre en compte un changement d'état, (2) soit des messages externes émis par un modèle à destination d'un autre modèle.

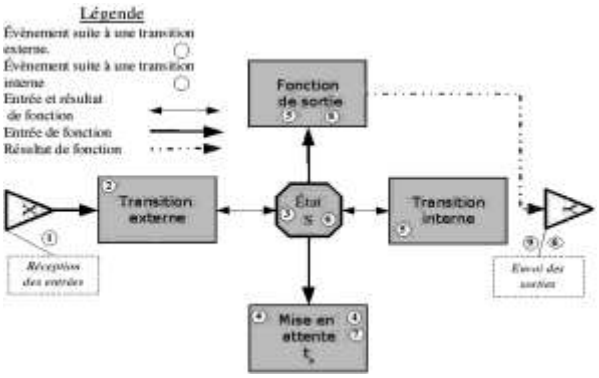

Figure 1. Comportement d'un modèle atomique DEVS

Le modèle atomique est défini par la structure suivante : $AM = < X, Y, S, t_a, \delta_{int}, \delta_{ext}, \lambda >$ (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 : la fonction d'avancement du temps (ou de durée de vie d'un état) ;
- δ_{int} : la fonction de transition interne déclanché par l'arrivée d'un message interne ;
- δ_{ext} : la fonction de transition externe déclenché par l'arrivée d'un message externe ;
- λ : la fonction de sortie ;

Le modèle couplé est défini par la structure suivante : $CM = < X, Y, C, EIC, EOC, IC, L >$ (2)
Avec :

- X_M : l'ensemble des ports d'entrée ;
- Y_M : l'ensemble des ports de sorties ;
- C_M : la liste des modèles composant le modèle couplé CM ;
- EIC : l'ensemble des liens d'entrée connectant le modèle couplé à ses composants ;
- EOC : l'ensemble des liens de sortie connectant les composants au modèle couplé ;
- IC : l'ensemble des liens internes connectant les composants entre eux ;
- L : liste des priorités entre composants ;

La simulation est effectuée en associant à chaque composant un simulateur permettant de mettre en œuvre les algorithmes de simulation correspondants. DEVS permet donc au modélisateur de s'abstraire totalement de l'implémentation des simulateurs. Une fois le modèle construit et quelle que soit sa forme, B.P. Zeigler a défini un simulateur capable de prendre en compte n'importe quel modèle décrit suivant le formalisme DEVS, et a développé le concept de simulateur abstrait (Zeigler, 1984). L'architecture d'un tel simulateur représente une description algorithmique permettant de mettre en œuvre les instructions implicites des modèles issus du formalisme DEVS, afin de générer leur comportement. L'intérêt majeur d'un tel simulateur réside dans le fait que sa construction est indépendante du modèle.

Lorsque le formalisme DEVS est replacé dans le contexte spécifique d'un domaine d'application, il se montre trop souvent abstrait, il est alors nécessaire d'enrichir sa syntaxe pour permettre de simplifier la spécification de types de modèles particuliers.

2.2 Le formalisme JDEVS

Le formalisme JDEVS [Filippi, 2003] fournit un outil générique et évolutif de modélisation et de simulation de systèmes complexes.
Il est basé sur une architecture logicielle, organisée sous la forme d'un cadriciel (framework), qui permet l'ajout ultérieur de techniques (telles que la logique floue, les réseaux de neurones …), par la voie de packages spécifiques. L'utilisation de packages permet de respecter la cohérence de l'architecture globale du cadriciel tout en assurant son évolutivité. De plus, grâce à l'utilisation d'un formalisme unificateur, DEVS [Zeigler, 2000], sur lequel est basé chaque technique ajoutée au cadriciel, il est possible d'assurer la compatibilité entre les différents modèles créés.
Cet outil a pour but de servir toute la communauté scientifique étudiant les systèmes complexes, il utilise des méthodes originales de modélisation et de simulation accessibles à tous : (1) le modèle est facile à créer et réutilisable (stocké dans des bibliothèques [Bernardi et Santucci, 2002]), grâce à une interface graphique ; (2) l'approche est modulaire ; (3) la phase de modélisation est indépendante de la simulation…
De plus des techniques d'étude de systèmes permettant au modélisateur de conceptualiser les modèles selon différents paradigmes ont enrichi le cadriciel : (1) Feedback-DEVS, s'inspire du formalisme F-DEVS [Kofman et al., 2000], il permet la spécification de modèles adaptatifs, auto-apprenants, autorisant la construction de modèles empiriques ; (2) les automates cellulaires, cette méthode est fréquemment utilisée pour l'étude de systèmes ayant des dimensions spatiales et est associé à un mode de représentation de l'espace (SIG) ; (3) Vector-DEVS, cette approche originale est adaptée des méthodes mathématiques de "front tracking" [Glimm et al., 1996] permettant l'étude de la propagation d'un phénomène par la simulation de la dynamique de son interface physique.
L'intérêt de ce cadriciel est d'être ouvert et de fournir une base théorique forte pour permettre la réutilisation, le stockage et le couplage simplifié de modèles hétérogènes. Toutefois, sans extensions, ce cadriciel ne permet d'étudier qu'un nombre limité de systèmes, en particulier nous proposons l'ajout d'un package spécifique basé sur la théorie de ensemble flou permettant l'étude de systèmes a paramètres vagues.

2.3 La théorie des ensembles flous

La théorie des ensembles flous introduite par Zadeh (Zadeh, 1965) offre une palette d'outils pour formaliser

les processus de raisonnement humain. C'est un moyen efficace de prendre en compte l'imprécision dans des connaissances. Elle permet alors de modéliser pour les systèmes informatiques des modes de raisonnement proches des modèles humains (Bouchon Meunier, 1985).

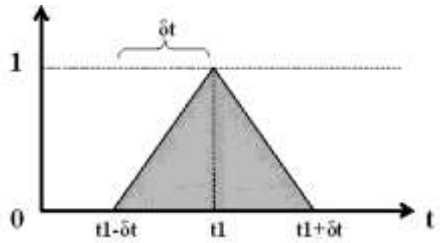

Figure 2. Exemple de fonction floue

En logique booléenne un évènement a lieu à une date donnée $t1$; en logique floue comme le montre la figure 2, un évènement peut avoir lieu à n'importe quelle date entre $t1-\delta t$ et $t1+\delta t$ avec, en fonction de la date, des possibilités différentes.

2.4 Le formalisme fuzzy-DEVS

Le formalisme fuzzy-DEVS introduit par Y. Kwon dans (Kwon et al., 1996) dérive du formalisme DEVS tout en conservant sa sémantique, ses concepts et sa modularité, il étend les fonctions caractéristiques de DEVS aux ensembles flous. Il permet la modélisation et l'analyse de systèmes complexes à évènement discret avec une structure et un comportement partiellement inconnus, bien que lors de la simulation les paramètres flous soient transformés en données réelles à l'aide de fonctions de défuzzyfication (Anglani et al).

De plus le modèle atomique fuzzy-DEVS, contrairement au modèle atomique DEVS est non déterministe (Kwon et al., 1996) et ne permet pas de détermine l'état suivant du système. Enfin la propriété de fermeture sous composition du formalisme DEVS (Zeigler, 1984) (Vangheluwe, 2001) n'est pas respectée. Ces contraintes ne nous permettent pas d'intégrer les modèles fuzzy-DEVS dans un environnement de multi-modélisation.

3. ENVIRONNEMENT DE MODELISATION

Le formalisme DEVS a montré sa pertinence comme base unificatrice permettant la coopération de différents types de modèles. Le développement formel d'un environnement basé sur ce formalisme est le fruit de travaux développés depuis huit ans dans notre laboratoire [Aiello, 1997], [Delhom, 1997], [Filippi, 2003].
Ces travaux constituent les bases de notre environnement logiciel.

3.1 Description de l'approche

Nos travaux ont pour but de définir, non seulement un nouveau module permettant la modélisation et la simulation de systèmes complexes à paramètres vagues, mais également une nouvelle approche d'utilisation qui permet au spécialiste du domaine de représenter les systèmes à étudier indépendamment de l'outil utilisé. Ce module sera intégré au cœur du formalisme déjà développé à l'université de Corse : JDEVS.

3.2 Rôles et acteurs

Afin de décrire notre approche, nous avons à l'aide du langage UML [Booch et al., 1998] axé notre réflexion sur la définition des rôles, et les exigences des utilisateurs envers l'environnement. Ainsi nous avons pu identifier les différents acteurs, décrire leurs rôles, et définir les interactions existantes entre chaque niveau.

4.2.1 Constructeur d'infrastructure CI

Il développe le noyau du système (figure 3), celui-ci doit être assez stable et ouvrable pour permettre d'effectuer des mises à jour et d'y intégrer de nouveaux formalismes. Les packages (liés à des formalismes spécifique) doivent permettre de modéliser et de simuler les problèmes posés par le spécialiste du domaine et traduits par le constructeur d'infrastructure. Dans notre cas, son rôle est d'intégrer le formalisme (DEVS-Flou) au cœur de l'application, c'est-à-dire à JDEVS.
Il est indispensable que le constructeur d'infrastructure possède des compétences en programmation et connaisse DEVS. L'intérêt de cette démarche est double : premièrement valider notre approche et enrichir son environnement générique tout en complétant l'outil JDEVS et deuxièmement en développant une approche modulaire fondée sur DEVS permettre d'agréger un package sans avoir à réaliser de modification sur le modèle de base.

Figure 3. Diagramme de cas d'utilisations

4.2.2 Constructeur d'application CA

Ils vont récupérer les renseignements en provenance des spécialistes du domaine et les transposer de manière formelle (figure 3). Ils font l'interface entre les experts de domaines (spécialistes) et le constructeur d'infrastructure (CI), pour ce faire ils doivent réaliser les fonctions de transfert de niveaux et posséder des compétences en informatique et dans le domaine étudié. Leur tâche est d'interpréter le langage des experts pour le rendre accessible au constructeur d'infrastructure, puis de choisir ou créer le modèle le plus adapté aux problèmes posés, de transcrire les besoins collectés en fonction du temps en évaluant les caractéristiques des paramètres à prendre en compte.

4.2.3 Spécialiste du domaine

Ils vont nous donner des informations en provenance du terrain ainsi qu'une vue générique du problème tout

en ayant à la fois une expérience et des besoins précis en matière de résultats (figure 3).
Ils n'ont pas forcément de connaissances en informatique, et sont spécialistes du domaine à la base de l'expérience, par exemple pour l'astronomie : physiciens, astrophysicien, ingénieur en électronique spatiale, spécialiste des télescopes... Leurs connaissances exprimées de manière linguistique aident à la réalisation du modèle de base ainsi qu'à regrouper les données d'entrées et exploiter les données de sorties.

3.2.4 Interactions entre niveaux

La figure 4 représente et décrit les interactions entre niveaux nécessaires à l'intégration du flou dans notre infrastructure logicielle.
L'expert décrit en langage naturel les spécifications du système étudié, c'est-à-dire qu'il traduit des données extraites du terrain ou issus de son expérience de manière compréhensible et exploitable par le constructeur d'application (CA). L'interface Expert-CA est la représentation de données à l'aide de fonctions floues, cette interface permet de faire travailler l'expert et le CA en collaboration et d'exploiter les outils que fournit la théorie des ensembles flous pour la représentation de systèmes à paramètres flous.

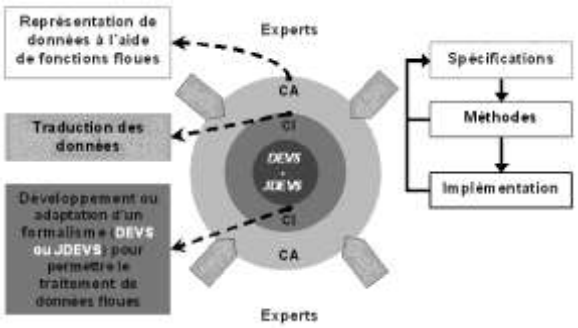

Figure 4. Rôle des intervenants dans l'environnement

Le constructeur d'application traduit les données et modélise le système. Il regroupe les données d'entrées et définit les méthodes permettant de traiter le flou. L'organisation de l'application de façon modulaire en vue de son intégration par le constructeur d'infrastructure (CI) se fait au niveau de l'interface CA-CI. Les spécifications et les méthodes permettant le traitement de systèmes flous sont adaptées afin de permettre leur intégration au cœur du logiciel.
Le constructeur d'infrastructure définit les caractéristiques du formalisme DEVS-Flou, et implémente les méthodes définies ci-dessus. Au niveau de l'interface CI-Cœur, il développe et/ou adapte le formalisme pour permettre le traitement de données floues. Enfin il implémente DEVS-Flou et l'intègre à JDEVS.
Nous pouvons constater que chaque intervenant peut être lié, ou est responsable d'une des étapes nécessaires à la réalisation du logiciel : spécification, définition des méthodes de traitement, et implémentation. Ceci nous permet d'utiliser au mieux les compétences de chacun et d'avoir une vision élargie du problème.

3.3 Place dans l'environnement

Nous avons vu que le formalisme JDEVS ne demandait qu'a être étendu. Tout comme les packages déjà développés (Feed back-DEVS, Vector-DEVS …) et intégrés au cœur, le formalisme DEVS-Flou doit s'insérer dans cet environnement (figure 5).

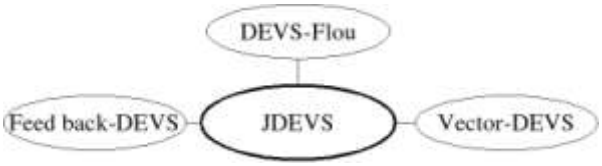

Figure 5. Environnement logiciel

Tout au long de son développement du package DEVS-Flou les caractéristiques de compatibilité avec DEVS et JDEVS seront au centre de nos préoccupations. La partie suivante présente notre méthodologie de modélisation ainsi que les aménagements effectués, nécessaire à la compatibilité du formalisme avec l'approche DEVS.

4. METHODOLOGIE DE MODELISATION FLOUE

Notre méthodologie de modélisation doit être compatible avec les standards DEVS. DEVS est basé sur le paradigme des évènements discrets, afin de prendre en compte des données incertaines, il est nécessaire d'apporter quelques modifications à la structure de ces évènements.

4.2 Evènement DEVS

Le formalisme DEVS est basé pour la modélisation sur deux types composants : les modèles couplés et atomiques. Ces composants possèdent des ports d'entrée, des ports de sortie et des variables.
L'échange des données s'établit à travers les ports des différents éléments d'un modèle, grâce à deux types d'évènements fondamentaux : les évènements externes et les évènements internes.
Un évènement externe prévu à l'instant *t* représente une modification (*à l'instant t*) de la valeur d'un ou plusieurs ports d'entrée appartenant à un élément donné *M*. Ceci a pour conséquence une modification des variables de *M*, à l'instant *t*.
Un évènement interne prévu à l'instant *t* représente une modification des variables de *M* (*à l'instant t*), sans qu'aucun évènement externe n'intervienne. De plus, l'arrivée d'un évènement interne provoque, à l'instant *t*, un changement de valeur sur un ou plusieurs ports de sortie du modèle *M*.
Un évènement DEVS peut être caractérisé par **:**
E = (temps; port; valeur) (3)
Le premier champ représente le temps d'occurrence de l'évènement, le second désigne le port sur lequel l'évènement intervient, et le troisième symbolise la valeur de l'évènement.
Dans DEVS un évènement a lieu à un temps donné, il modifie l'état (la valeur) d'une seule variable. Dans le

cas où l'état de plusieurs variables doit être modifié, on génère plusieurs évènements à la même date (au même moment), qui sont traités par les algorithmes de simulation suivant une liste de priorité. Par exemple si trois variables doivent être modifiées par un événement E qui a lieu au temps T on l'éclate en trois évènements E1, E2, E3 pris en compte toujours au temps T mais suivant une liste de priorité définie par l'utilisateur.

Comme nous venons de la préciser, dans le formalisme DEVS un événement doit être traité à une date T bien précise. Comme la notion d'événements est à la base du processus de simulation et que notre approche est uniquement basée sur la modification du processus de modélisation, il n'est donc pas possible d'intégrer la notion de logique floue pour la variable de temps. Ceci passe par une évolution dans la définition des évènements DEVS.

4.2 Evènement DEVS-Flou

A partir de la définition d'un évènement DEVS, on constate que les paramètres incertains peuvent intervenir à plusieurs niveaux : sur le temps, sur les valeurs, et sur le temps et les valeurs. On distingue donc trois types d'évènements DEVS-Flou.

4.2.1 Evènement à valeurs floues

Un évènement à valeurs floues est décrit par la formule (4) qui permet de définir, à un temps donné, le degré de possibilité qu'une certaine variable d'état (*var*) prenne une nouvelle valeur (*val*).
Ces possibilités seront calculées grâce à l'interprétation de fonctions floues et à un ensemble de règles défini par les spécialistes du domaine.

Évènement : $< t; port; \text{var}; \tilde{val} >$ avec *var* la variable d'état que doit modifier l'évènement et

$$\tilde{val} =< val, \pi^{val}_{(\text{var},t)}, \zeta^{val}_{\text{var}} > \quad (4)$$

Où :

- $\tilde{val}$: valeur floue ;
- $\pi^{val}_{(\text{var},t)}$: degré de possibilité que *var* prenne la valeur *val* au temps *t* ;
- ζ^{val}_{var} : pente qui oriente la courbe qui décrit la possibilité qu'a var d'être à l'état val à partir de t.

A partir de la formule (4) il est possible de calculer la valeur de la variable à n'importe quel temps t, en connaissant les valeurs de π et de ζ au temps courant t_{cour} , on utilise pour cela la formule suivante :

$$\pi^{val}_{(\text{var},t_{cour})} = \zeta^{val}_{\text{var}} \times (t_{cour} - t) + \pi^{val}_{(\text{var},t)} \quad (5)$$

4.2.2 Evènement à temps flou

Le simulateur abstrait de DEVS traite les évènements en les insérant dans un échéancier [] en fonction de la différence entre leur temps et le temps courant. Traiter une date incertaine au niveau du simulateur est très délicat car la simulation a lieu à une date fixée à l'avance, et de plus un évènement influence souvent les évènements suivants.
L'un des moyens pour ne pas avoir à gérer les événements avec un temps T qui soit incertain est de transformer le flou au niveau des dates (du temps) en flou au niveau des valeurs.
Ce mécanisme revient à "defuzzifier" le temps, c'est-à-dire à supprimer l'incertitude pour la transformer en valeur réelle.
La théorie des ensembles flous nous permet de rester proche du mode de représentation de données des experts, qui utilisent des fonctions floues pour représenter des incertitudes au niveau du temps et des valeurs des variables. En entrée et en sortie des composants le temps pourra être flou, par contre le flou sur le temps devra être transformé en flou sur les variables au niveau des composants.

Formule d'un évènement à temps flou :

$$< \tilde{t}, port, \text{var}, val > \text{avec } \tilde{t} =< t_\alpha, \pi^{t_{occ}}_{(t,\text{var})}, \zeta^{t_{occ}}_{t} > \quad (6)$$

Où :

- $\pi^{t_{occ}}_{(t,\text{var})}$: degré de possibilité que t soit à $t_\alpha \in [t_{\min}, t_{\max}]$, t date d'occurrence d'un évènement sur var (var prend val).
- $\zeta^{t_{occ}}_{t}$: pente qui oriente la courbe décrivant la possibilité que t soit à t_α .

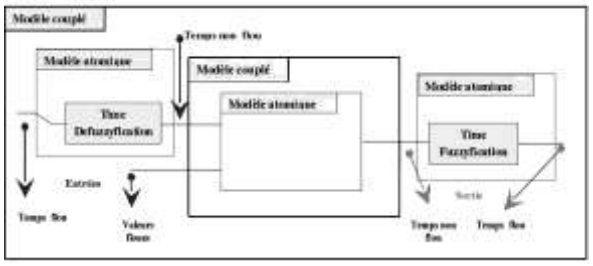

Figure 7. Composant DEVS-Flou, modèle à temps et valeurs flous.

La figure 7, présente un composant DEVS-Flou pouvant être assimilé à un modèle couplé DEVS. Ce modèle peut prendre en entrée des évènements à paramètres flous sur les valeurs ou les dates. Comme nous ne pouvons pas traiter avec la même méthodologie ces deux paramètres, les modules "Time-Defuzzyfication" et "Time-Fuzzyfication" permettent en modifiant le flou sur le temps de résoudre notre problème.

- La fonction *Time-Defuzzyfication* : transforme en entrée un évènement à temps flou en évènement à valeur floue et à temps fixé.
- La fonction *Time-Fuzzyfication* : transforme en sortie un évènement à temps fixé en évènement à temps flou, c'est la fonction inverse.

Ces fonctions seront définies comme des modèles atomiques afin de respecter l'approche DEVS.

Pour les évènements à temps et valeurs flous, les deux approches seront utilisées.

4.3 Évènements externes ou internes flous

Les évènements DEVS permettent l'échange des données à travers les ports des différents éléments d'un modèle (atomique et couplé). Ces messages sont envoyés grâce à deux types d'évènements fondamentaux : les évènements externes et les évènements internes. Dans notre cas le flou intervient de deux manières différentes en fonction du type d'évènement.
Pour les évènements externes, le flou est traité à l'aide de la méthodologie énoncée ci-dessus, c'est-à-dire les valeurs floues sont traitées dès l'entrée et le temps flou est transformé en date fixe.
Pour les évènements internes le flou n'intervient qu'au niveau des valeurs des variables d'état car toute incertitude sur le temps est transformée grâce aux fonctions "Time-Defuzzyfication".

4.4 Exemple d'utilisation

Cette partie propose un exemple détaillé d'utilisation du formalisme DEVS-Flou. Celle-ci peut être décrite en quatre étapes.

1. La première étape est la récupération de données, elles peuvent être extraites de connaissances d'experts ou de n'importe quelle autre source fiable. Dans notre cas : "la sonde européenne Rosetta qui doit se placer en orbite autour de la comète Chyurumov-Gerasimenko en 2014 parcourt 135000Km en 50 à 70 minutes".

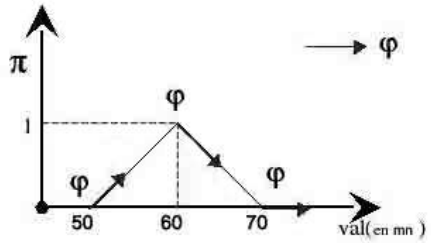

Figure 8. Fonction floue représentant le temps mit par Rosetta pour parcourir 135000KM

2. La seconde étape consiste à modéliser à l'aide de fonctions floues ces données. La figure 8 représente cette modélisation.
3. Nous avons pu voir que pour traiter un flou sur des données temporelles, il est nécessaire de transformer ce flou en flou sur les valeurs des variables. La troisième étape a pour but de transformer avec des *fonctions de transition de domaine* un flou sur le temps en flou sur les valeurs. Au minimum Rosetta parcours 135000Km en 70 minutes, donc en 60 elle en parcourt 115000. Au maximum elle parcourt 135000Km en 50 minutes donc en 60, elle en parcourt 160000. La figure 9 représente la distance en Km parcourue par Rosetta en 60 minutes et est le fruit de ce changement de domaine.

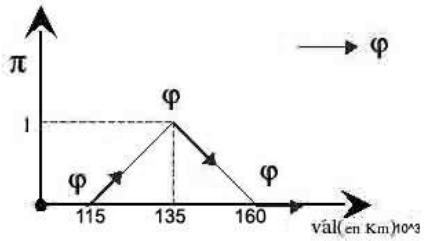

Figure 9. Fonction floue représentant la distance parcourue par Rosetta en 60 minutes après avoir appliqué la fonction Time-Defuzzyfication

4. Le flou intervenant maintenant au niveau des valeurs des variables, la dernière étape consiste à appliquer la formule (4).

Où :

- *val* est la valeur choisie en abscisse ;
- π la valeur correspondant à *val* en ordonnée ;
- ζ le sens du vecteur correspondant au coefficient directeur de la courbe aux points [*val*;π].

$$\begin{array}{lll} val & \pi & \zeta \\ 115000 & 0 & \vec{\zeta} \\ 135000 & 1 & -\vec{\zeta} \\ 160000 & 0 & \vec{0} \end{array} \qquad (7)$$

La matrice (7) donne les valeurs possibles de π et ζ pour les valeurs de val [115000; 135000; 160000]. Les valeurs sont choisies aux extremums et minimums (règle Max-Min), mais dans notre cas, n'importe quel point peut être calculé grâce au vecteur $\vec{\zeta}$.

5. CONCLUSION

Cet article a présenté un environnement de multi-modélisation basée sur le formalisme unificateur DEVS. Nous avons montré conceptuellement, comment il est possible de traiter les paramètres flous d'un système à l'aide de la théorie des ensembles flous en vue de l'intégrer dans l'environnement JDEVS.
Un des objectifs à court terme est d'implémenter le package décrivant cette technique.

A plus long terme nous souhaitons définir un ensemble de fonctions de changement de domaines, et identifier les différentes méthodes à employer pour passer d'un domaine à un autre. On entend par changement de domaine la définition de fonction liant deux domaines différents et permettant de passer plus ou moins facilement de l'un à l'autre tout en conservant la validité de tous les paramètres.
Il sera ensuite nécessaire de valider la généricité de l'approche sur plusieurs domaines d'application.

REFERENCES

Aiello, A. (1997). *Environnement Orienté Objet de Modélisation et de Simulation à Evénements Dis-*

crets de Systèmes Complexes. Thèse de Doctorat, Université de Corse.

Anglani, P., Caricato, A., Grieco, A.., Nicci, F., Matta, A., Semeraro, Q. et Tolio, T. *Evaluation of capacity expansion by means of Fuzzy-DEVS*, MURST Project

Bernardi, F. *Conception de bibliothèques hiérarchisées de modèles réutilisables selon une approche orientée objet*, Thèse de doctorat, (2002)

Bernardi, F. et Santucci, J. (2002), *Model design using hierarchical web-based libraries*, Dans Actes de la conférence 39th conference on Design automation, volume 1, pages 14.17. New Orleans, USA.

Booch, G., Rumbaugh, J., et Jacobson, I. (1998), *The Unified Modeling LanguageReference Manual*. Addison-Wesley.

Bouchon Meunier, B. La logique floue et ses applications, (1985)

Delhom, M. (1997). *Modélisation et Simulation Orientées Objet, Contribution àl'Etude du Comportement Hydrologique d'un Bassin Versant*. Thèse de Doctorat, Université de Corse.

Filippi, J.-B., Bernardi, F., et Delhom, M. (2002), *The JDEVS environmental modeling and simulation environment*, Actes de la conférence IEMSS 2002 conference on Integrated Assessment and Decision Support, 3 :283.288. Lugano, Suisse.

Filippi, J. *Une architecture logicielle pour la multi-modélisation et la simulation à évènement discrets de systèmes naturels complexes*. Thèse de Doctorat, Université de Corse, (2003)

Fishwick, P. *Simulation Model Design and Execution: Building Digital Worlds*, Prentice Hall, nj, englewood cliffs edition, (1995)

Glimm, J., Simanca, S., Smith, T., et Tangerman, F. (1996), *Computational physics meet computational geometry, Rapport Technique SUNYSB-96-19*, State University of New York at Stony Brook.

Kofman, E., Giambiasi, N., et Junco, S. (2000). *Fdevs : A general DEVSbased formalism for fault modeling and simulation*. Dans Actes de la conférence 2000 European Simulation Symposium, volume 1, pages 77.82. Hamburg, Germany.

Kwon, Y., Park, H., Jung, S., et Kim, T. (1996), *Fuzzy-DEVS Formalisme :Concepts, Realization and Application*, Proceedings AIS 1996, pages 227.234.

Lee, K. et Fishwick, P. *Dynamic Model Abstraction*, Proceedings of the Winter Simulation Conference, (1996)

Vangheluwe, H.L.M. *The discrete event system specification (DEVS) Formalism*, CS522 Fall Term, (2001)

Zadeh, L. *Fuzzy Sets*, Inform Control, Vol. 8 pp. 338-353, (1965)

Zeigler, B. *Theory of Modeling and Simulation*, Academic Press, (1976)

Zeigler, B. *Multifaceted modelling and discrete event simulation*, Academic Press, (1984)

Zeigler, B. (2000), *Theory of Modeling and Simulation*. Academic Press, 2nd Edition, ISBN : 0127784551.