e Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de
la Recherche Scientifique
Université Ibn Khaldoun – Tiaret –

Faculté des Sciences et de la Technologie et Sciences de la Matière
Département Informatique

MEMOIRE EN VUE DE L'OBTENTION DU DIPLÔME DE MAGISTER

SPECIALITE : Informatique

OPTION : Systèmes d'Information et de Connaissances (SIC)

Présenté par

Mustapha MAASKRI

SUJET DU MEMOIRE :

Simulation à Événements Discrets DEVS : Variantes et Apport

SOUTENU LE ... ... 2012 Devant Le Jury Composé de :

Mr. Amar BALLA	Professeur (ESI) - Alger -	Président
Mr. Omar NOUALI	Directeur de recherche CERIST -Alger-	Examinateur
Mr. Rachid CHALAL	Maître de conférences A (ESI) - Alger -	Examinateur
Mr. Youcef DAHMANI	Maître de conférences A (UIK) -TIARET-	Directeur du mémoire

ANNEE UNIVERSITAIRE 2011/2012

D E D I C A C E

J e d é d i e c e t r a v a i l :

A t o u s c e u x q u i m e s o n t c h e r s …

REMERCIEMENTS

Tout d'abord je tiens à remercier ALLAH LE TOUT PUISSANT pour l'endurance, la patience et la force qu'il m'a donné afin de finir ce mémoire.

Je voudrais exprimer mes forts remerciements à mon encadreur le Dr. DAHMANI Youcef, pour m'avoir donné le goût de la recherche, pour m'avoir poussé et encouragé tout au long de ces dernières années et pour sa patience avec moi.

Ensuite, je tiens à remercier les membres de jury d'avoir accepté d'évaluer ce modeste travail.

Mes remerciements vont à tout le personnel du département d'informatique de l'université de Tiaret et celui de l'Ecole nationale Supérieure de l'Informatique.

Enfin à tous ceux qui ont participés de près ou de loin pour mettre à terme ce travail, je leur transmets toute ma sympathie.

Introduction

Introduction

La modélisation et la simulation ne cessent de s'imposer comme outils incontournables pour analyser le comportement des systèmes dynamiques. Plusieurs méthodes ont été proposées pour améliorer le processus d'analyse de comportement de ces systèmes. Ces propositions tentent d'atteindre des modèles plus réalistes, assez simples et fortement flexibles. Notre intérêt porte sur DEVS (*Discrete Event System Specification*) qui est un formalisme de modélisation des systèmes à événements discrets. Des efforts ont été fournis pour adapter ce formalisme à différents domaines et situations. Dans ce vaste créneau, nous nous penchons sur la problématique de la gestion des imprécisions des paramètres des modèles qui peuvent être numériques ou symbolique, notre objectif étant d'étudier quelque variantes du formalisme DEVS (DEVS, Fuzzy DEVS, Min-Max DEVS, iDEVS etc..) qui traitent cette problématique et de proposer une solution, sans dévier du formalisme DEVS classique.

Ce travail est organisé comme suit. Dans le premier chapitre, nous présentons les concepts à la base de la modélisation et de la simulation. En premier lieu, nous décrivons les différentes approches de modélisation. Nous nous intéressons plus particulièrement à la modélisation systémique. Celle-ci a pour origine la théorie générale des systèmes. Après avoir défini ses concepts de base : notion de système, modélisation, simulation, nous donnons la définition du paradigme de modélisation et de simulation. Cette définition est essentielle, car elle est au centre des mécanismes appliqués dans le formalisme de modélisation et de simulation que nous avons choisi d'utiliser, à savoir DEVS.

Dans le deuxième nous présentons la logique floue, En commençant par énoncer les fondements de la logique floue et de voir comment elle permet d'exprimer selon un formalisme unique des informations très diverses (données incertaines ou imprécises, connaissances exprimées sous forme linguistique,..). Ensuite, on présente les méthodes de raisonnement flou (inférence floue) qui constituent la base de la commande floue.

Dans le troisième Chapitre, nous présentons en deux parties le formalisme DEVS. Chacune de ces parties traite d'un des aspects fondamentaux du formalisme, car celui-ci fait clairement la distinction entre la phase de modélisation et la phase de simulation. Et nous présentons quelques variantes du modèle DEVS à savoir (Fuzzy DEVS, Min-Max DEVS, iDEVS) qui traitent les données imparfaites structurelles ou comportementales du formalisme DEVS.

Le troisième Chapitre décrit notre contribution pour la modélisation de systèmes complexes à paramètres inconnus ou vagues.

Le dernier chapitre présente l'implémentation de l'approche proposée sur la simulation des systèmes photovoltaïques. Cette simulation a été effectuée sur la plate-forme de modélisation et de simulation JDEVS.

Chapitre 1
Modélisation et Simulation

Chapitre 1 : Modélisation et Simulation

I.1 Introduction

L'être humain n'a de cesse de chercher à comprendre le monde qui l'entoure. Les différentes phases d'étude nécessaires à cette compréhension ont toujours évolué de pair avec la technologie: observation, constatation, hypothèse, preuve, explication, acquisition, modélisation, etc. L'avènement de l'informatique et l'apparition de nouveaux formalismes de traitement ont écourté ou fait disparaître certaines de ces phases ; les progrès des méthodes numériques et l'augmentation des performances des ordinateurs permettent aujourd'hui, grâce à des simulations de plus en plus rapides et détaillées, de prédire le comportement de systèmes complexes.

La théorie de la modélisation et de la simulation est basée en grande partie sur la théorie générale des systèmes. La théorie générale des systèmes est un principe selon lequel tout est système, on parle aujourd'hui de *Théorie Systémique [56]*.

Les travaux présentés dans ce chapitre, concernent la modélisation et la simulation informatique de systèmes complexes. Notre réflexion et les développements associés se réfèrent aux notions de système, de modèle, de modélisation, et de simulation.

Ces notions sont dépendantes des techniques et du domaine d'application choisis. Toutes ces entités interviennent dans le processus de modélisation et de simulation. La figure 1 présente ces différents éléments.

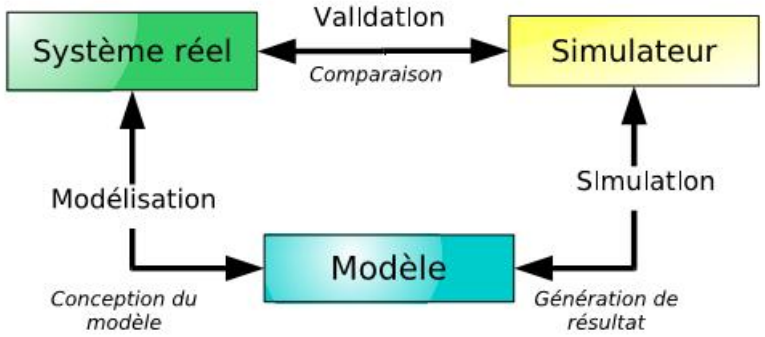

Figure 1 : Etapes du processus de modélisation et de simulation [56]

Le système est l'élément à la base de l'étude, c'est de là que les données nécessaires à l'élaboration du modèle puis à sa validation sont issues. Il est lié au

modèle par une relation appelée la relation de modélisation qui décrit le système dans un environnement choisi.

Le modèle est une représentation du système réel dans un certain environnement. Les modèles sont associés à une structure de contrôle appelée simulateur. Il reproduit le comportement du système sous certaines conditions et génère les résultats issus de la simulation du modèle. Ce processus est appelé simulation.

Le simulateur permet de faire évoluer le modèle dans le temps. Les résultats de ce processus sont comparés aux données du système réel afin de vérifier la conformité du modèle, cette relation entre le simulateur et le système est appelée la relation de validation.

Dans la première partie nous décrivons succinctement différentes approches de modélisation : systémique, empirique mécaniste et approximative.
Dans la seconde partie nous présentons de manière générale l'approche de modélisation systémique. Elle est à la base du formalisme de modélisation utilisé. De plus, nous présentons la théorie systémique, et donnons les définitions de système, de modèle, de modélisation et de simulation.

I.2 Approches de modélisation

D'une façon générale, nous pouvons distinguer plusieurs approches de modélisation : systémique, empirique, mécaniste, approximative [56]. Les approches de modélisation prédictive, que l'on peut qualifier de "non approximatives" font opposition aux méthodes dites approximatives.

L'approche systémique se base sur une description mathématique ou physique de plusieurs processus simultanés et de leurs interactions pour définir des modèles. L'objectif de ce type de modèle est de prendre en compte l'ensemble des variables clés et leurs interactions.

L'approche empirique est déduite des tendances observées à l'intérieur d'un ensemble de données, elle part du postulat que ces tendances ne vont pas changer dans le temps. Les représentations logiques structurent certaines de ces approches empiriques.

L'approche mécaniste est basée sur la connaissance du fonctionnement d'un système. Les différents processus qui composent le système sont modélisés

indépendamment à l'aide de formules qui décrivent une loi ou une règle. Les modèles issus de cette approche comprennent trois parties :

1. Une base de fait, mémoire de travail qui contient les données initiales et les hypothèses émises décrivant le problème à traiter ;
2. Une base de règles constituant la connaissance permanente ;
3. Un moteur d'inférence, mécanisme ou algorithme, qui exploite les règles. Les moteurs d'inférence fonctionnent selon deux mécanismes :
   a) par chaînage avant, ils partent de la base de faits pour aboutir à des conclusions : "si Y alors X" ;
   b) par chaînage arrière ils prennent la procédure dans le sens inverse, des conclusions aux conditions initiales. Ces deux mécanismes peuvent aussi être combinés.

Ce genre d'approche se prête bien à la modélisation de processus complexes qui requièrent l'expertise de spécialistes. Le savoir faire de ces derniers aide à l'élaboration des bases de règles pour la modélisation des processus. Ces modèles sont en revanche peu appropriés pour décrire un système dans sa totalité quand les bases de règles à mettre en œuvre ne peuvent pas prendre en compte l'ensemble des paramètres du phénomène.

L'approche approximative : dans le cadre des modes de raisonnement présentés, un prédicat est soit vrai soit faux. Cependant, le raisonnement et la mise en place de modèles reposent souvent sur des connaissances et des bases de données imparfaites. La mise en place de procédures visant à établir des scénarii prédictifs fiables doit permettre de prendre en compte les données incertaines et / ou imprécises et/ou incomplets des modèles.

Concevoir de telles procédures implique de sortir des approches classiques de la logique. Cela exige aussi de définir une représentation de l'incertitude et de l'imprécision, de choisir des procédures de raisonnement qui prennent en compte ces aspects tout en les propageant au cours des étapes du raisonnement afin de pouvoir les qualifier et les quantifier dans les résultats.

Dans la partie suivante nous revenons en détail sur l'approche systémique. En effet le formalisme de modélisation utilisé est basé sur cette approche.

I.3 Théorie Systémique

Notre étude s'inscrit dans la vision *systémique*. De cette vision découle une pratique particulière de l'activité scientifique. La partie proprement opératoire de l'analyse systémique est née des travaux de *L. von Bertalanffy* [13] ; elle met l'accent sur la notion du système comme faisant partie d'un tout. Jusque là, l'approche scientifique était résolument réductionniste, c'est-à-dire qu'elle procédait par décomposition du réel pour en isoler une partie qui devenait l'objet d'étude. Cette approche a trouvé ses limites dans des perspectives d'explications plus globales où les mécanismes décrits au niveau individuel ne suffisent pas à expliquer le comportement de l'ensemble.

Le mot "*système*" apparaît dès le XVIIIème siècle. Il est à l'origine des concepts fondateurs de la systémique ou de la science des systèmes. Il sera repris en 1950 par le biologiste théoricien *V. Bertalanffy* qui voulait décrire dans le même langage les systèmes artificiels et les systèmes naturels, à l'époque les systèmes fermés ou statiques et les systèmes ouverts ou dynamiques.

Un système permet de représenter un phénomène perçu comme complexe, il peut être déterminé à partir de quatre questions intrinsèquement liées :
Ces questions sont résumées par la figure 2 :

1. *Il fait quoi ?* → Fonction ;
2. *Dans quoi ?* → Environnement ;
3. *Pourquoi ?* → Finalités ;
4. *Devenant quoi ?* → Transformation.

Autrement dit en interrogeant les interrelations qui le constituent, interrelations avec ses sous systèmes et avec son environnement, il est possible de décrire un système.

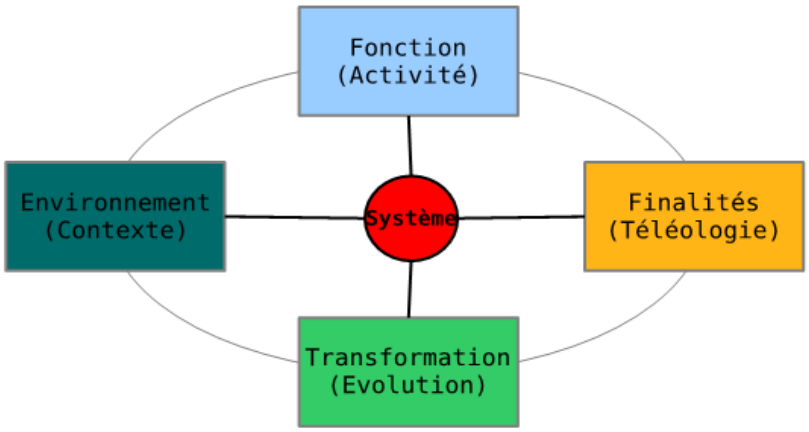

Figure 2 : Interrelations d'un Système

Un système est décrit sous deux aspects : structurel et fonctionnel (comportemental) [92].

Sous son aspect structurel un système comprend quatre composantes :

1. les éléments constitutifs. Nous pouvons en évaluer le nombre et la nature ;
2. une limite, ou une frontière, qui sépare la totalité des éléments de son environnement avec le milieu extérieur. La limite d'un système peut être floue, ou mouvante ;
3. des réseaux de relations (liens). Les éléments sont en effet interreliés ou connectés pour pouvoir communiquer et s'échanger des informations ;
4. des stocks (lieu de stockage) où sont gardés les informations qui doivent être transmises ou réceptionnées.

L'aspect structurel est la représentation de la structure potentielle d'un système.

Sous son aspect fonctionnel un système est aussi décrit par quatre éléments :

1. des flux d'informations, qui empruntent les réseaux de relations et transitent par les stocks. Ils fonctionnent par entrées / sorties avec l'environnement ;
2. des centres de décision qui organisent les réseaux de relations, ils coordonnent les flux et gèrent les stocks ;
3. des boucles de rétroaction, elles servent à informer les centres de décision sur l'état général du système ;
4. des ajustements réalisés par les centres de décision en fonction des boucles de rétroaction.

L'aspect fonctionnel est la représentation du comportement du système.

I.3.1 Systèmes

Un système est un ensemble hiérarchique d'éléments matériels ou immatériels (êtres vivants, machines, méthodes, règles, etc.) en interaction, transformant par un processus des éléments d'entrée en éléments de sortie.

Par exemple, une éolienne transforme le vent en électricité. C'est une organisation hiérarchique de sous systèmes, considérés eux-mêmes comme des systèmes à part entière (le système "éolienne" est composé de sous systèmes "palles", "moteur", etc.).

Nous parlons de systèmes physiques, biologiques ou sociaux, systèmes qui relèvent de la systémique. Nous voyons ici que la systémique se veut transdisciplinaire et / ou pluridisciplinaire en essayant de trouver des lois générales indépendantes des contextes d'application.

La systémique distingue l'aspect structurel et fonctionnel des systèmes. Elle définit aussi les systèmes comme causaux, les sorties étant la conséquence d'une entrée, ou déterministes. A une entrée donnée ne peut correspondre qu'une seule sortie. Cette définition est à la base de nombreux formalismes tels les équations différentielles, les automates à états finis, les réseaux de Pétri, etc.

Plus généralement, un système A peut être défini par l'équation (1) [56] :

$$A = < t, X, W, S, Y, d, \lambda > \qquad (1)$$

Avec :

– t : base de temps ;

– X : ensemble des états d'entrée ;

– W : t → X : états d'entrée courants ;

– S : ensemble des états du modèle ;

– d : W × S → S : fonction de transition, elle fait évoluer l'état du modèle en fonction des états d'entrées (activations) ;

– Y : ensemble des états de sortie ;

– λ : S → Y : fonction de sortie.

Dans l'équation (1), la base de temps t représente la variable de temps ou l'écoulement d'une durée. L'ensemble des états d'entrée X constitue toutes les activations d'entrées possibles pour le système. S représente l'ensemble des états que peut prendre le système.

La dynamique du système est décrite par la fonction de transition « d », elle applique les états d'entrée W à l'état courant *S* pour transiter vers un nouvel état. Le système génère une sortie en appliquant la fonction λ à partir de l'état courant. Les fonctions de transition et de sortie sont activées pour faire évoluer le système dans le temps.

I.3.2 Modélisation

La définition de modèle et de modélisation peut varier notablement. Il y a une relation forte entre contexte et définition. D'après *P.Fishwich* [34] "modéliser c'est décrire une réalité sous la forme d'un système dynamique, à l'aide d'un langage de description, à un certain niveau d'abstraction". Toujours d'après *P. Fishwich* [34] la modélisation est représentée par la symbiose entre le formalisme et les techniques de modélisation qui poursuivent le même objectif : "dégager les meilleures métaphores et analogies permettant de mieux comprendre un phénomène quelconque".

Selon *C. Oussalah* [66] "la modélisation est un épimorphisme entre un système réel et un modèle dont la finalité est de donner une représentation simplifiée et observable de la structure et du comportement du système réel". Ceci nous conduit à définir, dans notre contexte, la modélisation comme un processus d'identification d'un phénomène et l'opération par laquelle on établit son modèle, afin d'en proposer une représentation interprétable, reproductible et simulable. C'est une technique qui consiste à restituer, sous une forme compréhensible par l'ordinateur, un objet ou un phénomène quelconque.

Ce modèle, ainsi constitué, permet d'approcher le fonctionnement ou le comportement du système par l'intermédiaire d'un certain nombre d'hypothèses et de règles qui composent ce que l'on appelle un formalisme de modélisation.
La modélisation et la simulation sont des domaines scientifiques faisant un grand usage de l'informatique, et leur mise en œuvre s'effectue au sein de logiciels appelés environnements de modélisation et de simulation.

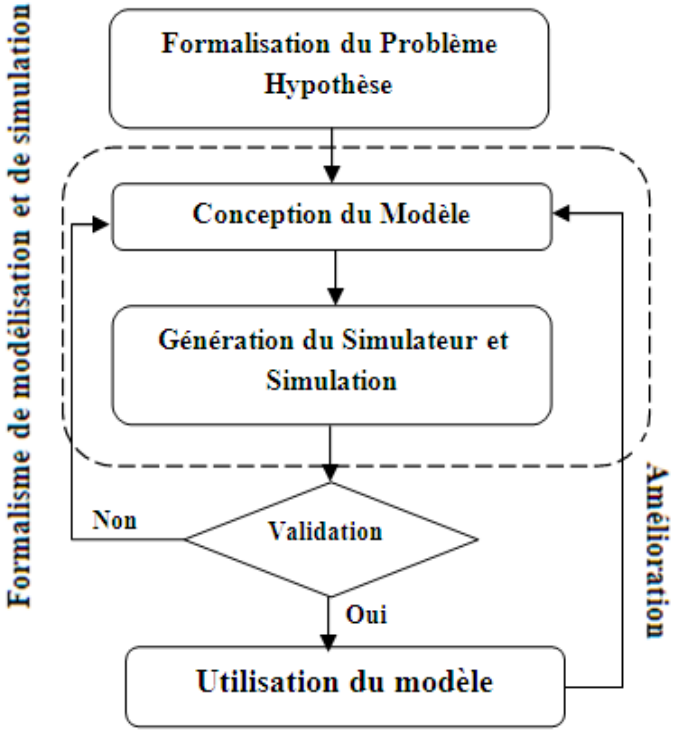

Figure 3 : Cycle de vie d'un modèle [56].

La modélisation systémique caractérise une des grandes méthodes de modélisation contemporaine. Elle veille à expliciter "les points de vue" de l'observateur-concepteur qui la met en œuvre, et à proposer une des formes de compréhension intelligible du système sans prétendre l'expliquer. La modélisation systémique est fondée sur deux hypothèses :

1. rendre compte des fonctions et fonctionnements du système ;
2. expliciter les finalités attribuées au phénomène modélisé en veillant à les différencier explicitement des finalités de l'observateur-concepteur.

En considérant un système de manière globale, par abstraction de certaines contraintes, nous pouvons associer à un système complexe une représentation simplifiée de sa structure et de son fonctionnement.
Cette représentation, plus simple à décrire et utiliser s'appelle un modèle.

L'usage de l'ordinateur et des modèles numériques ouvre, grâce à la modélisation et à la simulation, un champ scientifique nouveau avec sa méthodologie propre de validation expérimentale du modèle (figure 3) et l'usage de celui-ci pour prédire des comportements inaccessibles à la mesure car situés dans le futur lointain ou à des échelles trop grandes ou trop petites pour être instrumentées.

I.3.3 Simulation

La simulation est rapidement devenue incontournable pour la modélisation des systèmes complexes. Elle permet de gérer des modèles afin de produire des données comportementales, c'est-à dire de faire évoluer les états du modèle dans le temps.

Le temps peut être vu de manière continue ou discrète. Selon que les états du système soient spécifiés de manière dénombrable ou non. Dans un modèle on parle de simulation discrète ou continue. Par exemple, les ondes en général, ou la quantité d'oxygène dans l'air, constituent des systèmes à états continus car leurs valeurs changent continuellement. Un système représentant un guichet automatique de banque peut être vu au contraire comme un système à états discrets car l'arrivée d'un client est un évènement subi et dont les quantités sont dénombrables.

La (Figure 4) présente un système continu, qui est un système mettant en jeu des signaux continus. Ils ont les mêmes propriétés qu'une fonction continue.

Dans le cadre de simulation continue, une fonction continue est à la base de l'évolution du modèle dans le temps. Ces simulations nécessitent une description de type mathématique analytique du modèle car il doit être possible de donner l'état du système en tout temps. La complexité de tels modèles grandit toutefois avec le nombre de paramètres et il devient rapidement impossible de modéliser des systèmes complexes de manière purement analytique. Il est alors nécessaire de décrire ces systèmes avec d'autres approches de simulation.

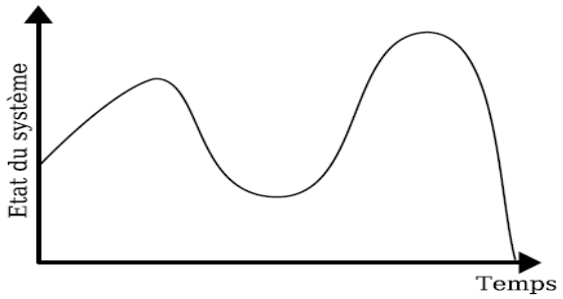

Figure 4: Simulation continue

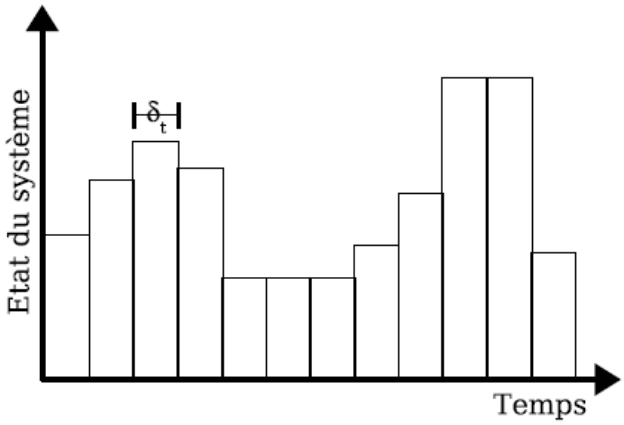

Figure 5 : Simulation discrète dirigée par horloge.

La (Figure 5) présente un système discret est un système qui met en jeu des informations qui ne sont prises en compte qu'à des moments précis. En général ces instants sont espacés d'une durée constante appelée période d'échantillonnage (δ_t). On parle de systèmes discrets par opposition aux systèmes continus. Le terme discret vient des mathématiques. Dans une approche de simulation discrète, l'état futur du modèle dépend de son état actuel. La simulation de ce type de modèles implique que les changements d'états s'effectuent de manière discrète dans le temps. Il existe deux façons de gérer le temps en simulation discrète. La simulation est :

- Dirigée par une horloge, lorsque l'état du modèle est réévalué à intervalles réguliers. Dans la Figure 5, un intervalle δ_t sépare deux transitions d'états.
- Dirigée par les évènements, lorsque l'état du modèle est réévalué en fonction de l'arrivée d'évènement. Dans ce cas la simulation est dite à évènements discrets (Figure 6).

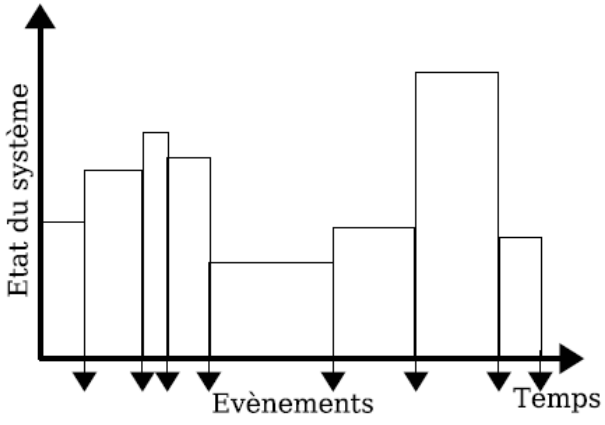

Figure 6 : Simulation discrète dirigée par évènements.

Les méthodologies de simulation à évènements discrets présentent de nombreux avantages sur la simulation dirigée par horloge. En effet, il est possible de simuler un modèle en temps discret grâce aux évènements discrets en programmant des évènements d'activation à intervalles réguliers.

L'intérêt de l'utilisation de la simulation à évènements discrets apparaît lorsque le phénomène simulé utilise des échelles de temps très différentes, de l'ordre de la seconde pour une partie du modèle et de l'année pour une autre ; dans ce cas, si la simulation est dirigée par une horloge, la règle veut que le pas de temps utilisé soit celui du modèle utilisant la plus petite échelle de temps, même si le sous modèle n'est actif que pendant une petite partie du temps complet de la simulation. Les simulations dirigées par évènements discrets permettent de ne pas réévaluer l'état du modèle lorsque ce n'est pas jugé nécessaire.

Dans notre contexte, nous pouvons définir la simulation comme un procédé informatique visant à faire évoluer un système afin de prédire son comportement. Ceci dit, il ne faut pas oublier que les résultats obtenus par simulation dépendent des hypothèses retenues pour construire les modèles (Figure 3) ; et qu'il ne faut évidemment pas confondre résultat de simulation et résultat réel. La simulation ne doit pas être utilisée sans prise de recul ; la vérification de la validité des modèles (Figure 3), et si nécessaire leurs améliorations sont des phases importantes afin de ne pas prendre de risque inconsidéré dans les décisions.

Comme nous l'avons vu, la modélisation peut être représentée par la symbiose entre le formalisme et les techniques de modélisation ; en fonction du système à étudier ou du domaine d'application, il peut être nécessaire d'utiliser différentes techniques de modélisation. Celles-ci peuvent être regroupées au sein d'un paradigme de modélisation et de simulation.

I.4 Paradigme de Modélisation et de Simulation

Un paradigme est un modèle exemplaire d'une chose ou d'une réalité. En sciences le paradigme désigne une vision du monde ou un mythe fondateur d'une communauté scientifique particulière [56].

Nous pouvons alors parler d'un paradigme de modélisation et de simulation comme étant l'ensemble des définitions et formalismes, des méthodes, des outils et techniques qui caractérisent une activité de modélisation. Par exemple, nous parlons du paradigme objet, caractérisé par les notions d'encapsulation, d'héritage et de

polymorphisme. Il existe beaucoup de paradigmes de modélisation. Parmi les plus connus, nous pouvons citer les modèles stochastiques, les algorithmes évolutionnaires ou les techniques d'apprentissage comme les réseaux de neurones.

L'augmentation de la puissance de calcul des ordinateurs nous autorise aujourd'hui à être plus pointus dans nos représentations des systèmes. La question n'est pas de savoir si leurs états sont dénombrables ou indénombrables, mais plutôt de faire cohabiter plusieurs représentations au sein d'un même modèle. Il semble assez évident que la complexité et la diversité des systèmes soient mieux appréhendées par une diversité des modèles. L'hétérogénéité des modèles nous amène donc à la construction de multi-modèles. Un multi-modèle rassemble plusieurs paradigmes et / ou formalismes dans sa réalisation - nous parlons alors de multi-modélisation. Ce terme a été introduit par *T.I. Oren* en 1989 [65] et s'est fait connaître par les travaux de *P. Fishwick* et *B.P. Zeigler* [36]. Il existe aussi le terme de modélisation multi-paradigmes ; H. Vangheluwe [82] le définit comme s'adressant à trois axes de recherches orthogonaux :

1. à la modélisation liée au multi-formalisme (donc à la multi-modélisation), c'est-à-dire au couplage de modèles spécifiés dans différents formalismes ;
2. au problème de changement de niveau d'abstraction dans les modèles ;
3. à la méta-modélisation, c'est-à-dire la construction de modèles de modèles.

Pour intégrer différents paradigmes, nous considérons les travaux effectués par *B.P. Zeigler* [93]. Ces travaux se basent sur les mathématiques discrètes et plus particulièrement sur la théorie générale des systèmes. Dans ce contexte, des travaux formels ont été menés pour développer les fondements théoriques de la modélisation et de la simulation des systèmes dynamiques [96]. Ces travaux ont notamment donné naissance au formalisme *DEVS* (Discrete EVent system Spécification) pour la spécification des systèmes à évènements discrets.

Nous avons choisi comme paradigme de modélisation et de simulation le multi formalisme DEVS [93]. Il offre la possibilité d'encapsuler, pour un même système, des modèles décrits à partir de différents formalismes tels que les équations différentielles [96] ou les réseaux de Pétri [48], etc. De plus, il permet d'appréhender la complexité des systèmes étudiés, et il s'abstrait totalement de la mise en œuvre des simulateurs associés aux modèles. Il existe également plusieurs extensions du formalisme DEVS adaptées à des domaines précis, mais nous y reviendrons par la suite.

I.5 Conclusion

Dans ce chapitre nous avons vu que DEVS est un formalisme de modélisation et de simulation hiérarchique et modulaire, basé sur la théorie des systèmes à évènements discrets. Il est de plus "structuré sous composition" et permet la séparation de la phase de modélisation de celle de simulation. La simulation est automatisée en fonction du modèle, le simulateur se générant automatiquement (simulateur abstrait). Néanmoins il doit être adapté et étendu lorsqu'il est mis dans le contexte spécifique d'un domaine d'application. Le formalisme étant souvent trop abstrait, il est nécessaire d'enrichir sa syntaxe pour permettre de simplifier la spécification de types de modèles particuliers.

Chapitre 2
Systèmes Flous

Chapitre 2 : Systèmes Flous

L'acquisition de connaissances d'un domaine d'expertise particulier est une étape essentielle pour effectuer un raisonnement efficace. De telles connaissances sont de deux formes : simples (par exemple, des faits, des observations, des mesures...), ou complexes (par exemple, des règles, des lois, des relations...). Une méthode classique d'acquisition de connaissances consiste à les obtenir d'experts (par exemple un expert d'un domaine précis dans lequel on souhaite réaliser notre étude).

Cependant, il apparaît que, dans certains cas, il devient difficile pour quelqu'un de formaliser et d'énoncer les connaissances pertinentes utilisées dans la réalisation de la tâche à effectuer. Des paramètres inconnus, vagues, ou non mesurables, associés à certains systèmes complexes, sont difficiles à modéliser.

La théorie des ensembles flous offre une palette d'outils pour formaliser les processus de raisonnement. C'est un moyen efficace de prendre en compte l'imprécision dans des connaissances. A la différence de l'incertitude inhérente à une connaissance qui donne le degré avec lequel on peut être certain qu'un évènement se produira, l'imprécision d'une connaissance est liée à la mesure qui en est faite. Ainsi, la théorie des ensembles flous ne préjuge pas de l'occurrence d'évènements mais plutôt de la manière dont des éléments sont perçus ou connus. Manipuler les imprécisions est une capacité humaine par excellence. Le mode de raisonnement humain est capable d'appréhender des données imprécises et d'arriver à en déduire des conclusions adéquates.

La théorie des ensembles flous formalise et prend en compte cette capacité. Elle permet alors de modéliser pour les systèmes informatiques des modes de raisonnement proches des modèles humains [16].

Dans la suivante nous présentons les notions de base de la théorie des ensembles flous et la logique floue, leurs modes de représentation de paramètres (les fonctions floues).

II.1 Données imparfaites

Une proposition comme "demain il y aura beaucoup de vent" est à la fois imprécise et incertaine voire incomplète :

II.1.1 Imprécise, car comment quantifier "beaucoup de vent" ? Une imprécision est une difficulté à exprimer clairement un fait, c'est ce qui est relatif à son contenu "environ 20 Km/h", on ne donne pas de valeur précise mais un intervalle. Elle se manifeste généralement lorsque la donnée est exprimée sous forme linguistique "beaucoup" ;

II.1.2 Incertaine, car comment être sûr que "demain il y aura vraiment beaucoup de vent", une incertitude représente un doute sur la validité d'un fait. Elle fait référence à la véracité de l'information, c'est un coefficient apporté au fait qu'une proposition soit vraie ou fausse ;

II.1.3 Incomplète, car à partir de cette proposition nous n'avons pas exactement la vitesse du vent à une heure fixée. Les incomplétudes sont des absences de connaissances ou des connaissances partielles sur certaines caractéristiques du système. Elles peuvent être dues à l'impossibilité d'obtenir certains renseignements ou à un problème au moment de l'acquisition des connaissances. L'incomplétude peut être considérée comme un cas particulier de l'imprécision.

Ces trois types d'imperfections ne sont pas indépendants les uns des autres, bien que naturellement manipulés par l'être humain, les transcrire de manière informatique et numérique sur ordinateur peut s'avérer compliqué, les ordinateurs n'ayant pas nos capacités d'abstraction et de compréhension.

II.2 Notions basiques de logique floue et ensembles flous

La logique classique a un rôle important dans de nombreux domaines. Or, sa structure qui ne peut exprimer des faits qu'avec "vrai" ou "faux" (0 ou 1) limite son champ d'action dans des techniques et des applications qui veulent imiter le raisonnement et l'esprit humain (l'Intelligence Artificielle, l'Aide à la décision...), c'est à dire des techniques qui s'appuient sur l'incertitude pour leur bon fonctionnement. La logique floue est justement conçue pour répondre à ce problème, pour permettre la caractérisation des éléments de façon "graduelle". Elle a été introduite par L. Zadeh

[84] comme extension de la logique booléenne. Cette logique ne consiste pas à être précis dans les affirmations, mais au contraire à répondre à des propositions vagues, nécessitant une incertitude (un flou).

Par exemple, en logique classique, à la question : "Est-ce que cette personne est grande?", on ne peut répondre que par vrai, si c'est le cas, ou faux dans le cas contraire. Avec la logique floue, on peut représenter les cas où la personne est très petite, moyennement petite, normale, pas très grande, grande, etc.

La logique floue fournit un cadre formel pour construire des systèmes qui offrent à la fois une bonne performance numérique (précision), et une représentation linguistique (interopérabilité).

D'un point de vue numérique, les systèmes flous sont des systèmes non linéaires capables de traiter une information imprécise et incomplète. Linguistiquement, ils représentent les connaissances sous forme de règles, ce qui est une façon naturelle d'expliquer des processus décisionnels. La logique floue est un paradigme informatique qui fournit des outils mathématiques pour représenter et utiliser des informations d'une manière proche du processus de communication et de raisonnement humains.

Les ensembles flous sont des généralisations des ensembles réels. Soit un ensemble A, un élément x Є A si $\mu_A(x) = 1$, μ_A est la fonction d'appartenance (Figure7) dans cet exemple x=[5 10].

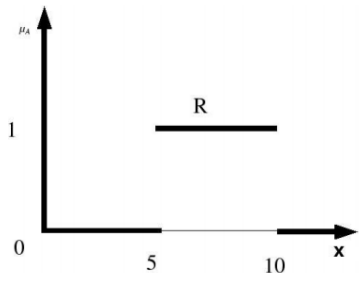

Figure 7 : Fonction d'appartenance μ_A à un ensemble réel

Pour les ensembles flous il en va de même, l'élément x Є A si $\mu_A(x) = 1$. Mais en plus il y a une certaine possibilité pour que **x** appartienne à un ensemble A appelé degré d'appartenance déférente de 1. Par exemple dans la Figure 8 (pour x = 4,5, $\mu_A(x) = 0,33$).

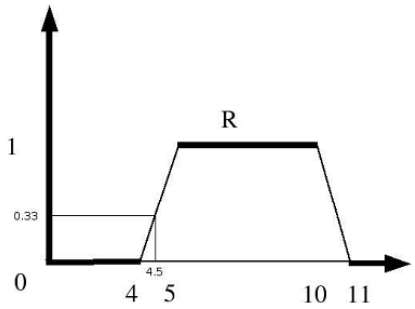

Figure 8 : Fonction d'appartenance à un ensemble flou

L'opération qui consiste à transformer un ensemble réel en ensemble flou est appelée "fuzzyfication" (le passage de la figure 7 vers la Figure 8), et l'opération inverse est appelée "defuzzyfication".

II.3 Fonctions floues

En logique booléenne un évènement a lieu à une date donnée t1 ; en logique floue comme le montre la figure 9, un évènement peut avoir lieu à n'importe quelle date entre t1-dt et t1 + dt avec, en fonction, des possibilités différentes. Ceci n'est qu'un exemple car le flou peut intervenir à d'autres niveaux que le temps, notamment sur les données. D'après D. Dubois [26], la logique floue introduit l'échelle de nuances.

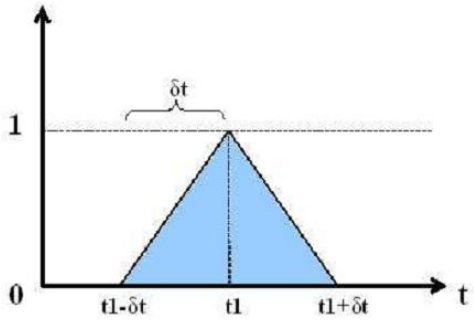

Figure 9 : Exemple de fonction flou

La figure 9 est un exemple de représentation de fonction floue, on en distingue plusieurs types pouvant être utilisés pour décrire les valeurs d'un ensemble flou, notamment les fonctions triangulaires (figure 10.a), les fonctions trapézoïdale (figure 10.b) et les fonctions paraboliques (figure 10.c). Les plus simples à mettre en œuvre sont les fonctions triangulaires.

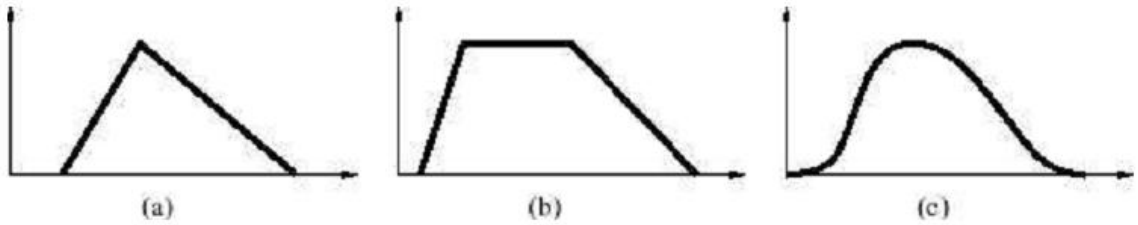

Figure 10 : Différents types de fonctions floues

Il est possible d'effectuer plusieurs types d'opérations sur ces fonctions, comme l'intersection (opération « ET» Figure 11) et l'union (opération « OU» Figure12) d'intervalle. Ces deux types d'opérations seront utilisés par la suite pour : soit fixer un intervalle où se chevauchent deux fonctions (intersection), soit définir une fonction globale à partir de plusieurs fonctions (unions).

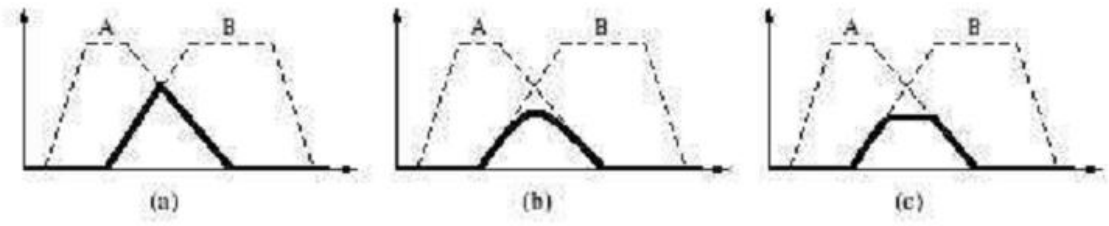

Figure 11 : Intersection de fonctions floues

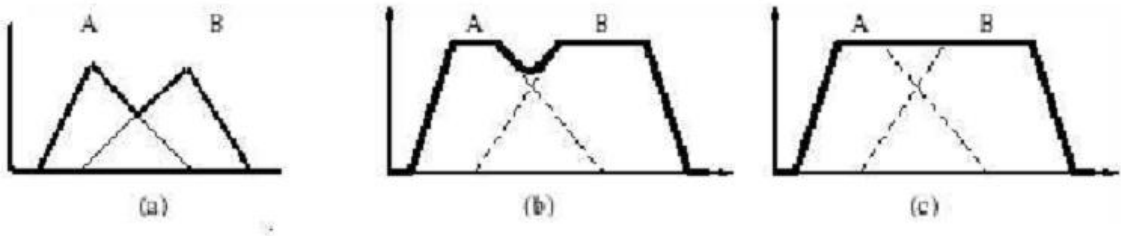

Figure 12 : Union de fonctions floues

II.4 Intervalle flou

Les opérations floues que nous venons de définir peuvent être relativement difficiles à appliquer, pour cela, nous nous intéressons à des types de quantités floues de formes particulières conduisant à la simplification des calculs. Les fonctions d'appartenance de forme trapézoïdale et triangulaire en font partie, de plus elles permettent respectivement de décrire un intervalle et un nombre flou. Ceux-ci peuvent être représentés à partir d'un quadruplet [a, b, α, β], avec a < b, [α, β] comme valeurs modales et (a − α), (b + β) respectivement les limites inférieure et supérieure de l'intervalle flou (voir Fgure 13).

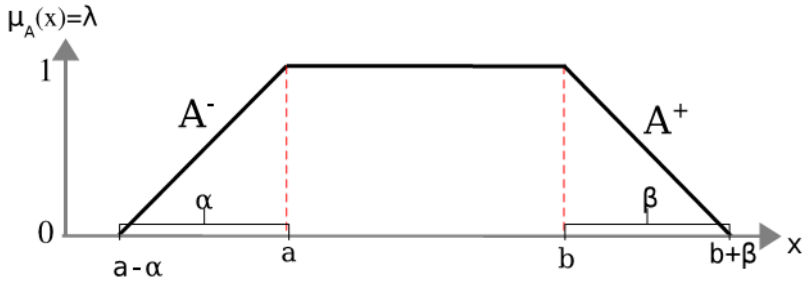

Figure 13 : Intervalle Flou

Cette notation permet de représenter uniformément un nombre réel, un intervalle classique, un nombre flou et un intervalle flou.

– Un nombre réel par [a, a, 0, 0] ;

– Un intervalle classique [a, b] par [a, b, 0, 0] ;

– Un nombre flou X par [a, a, α, β] ;

– Un intervalle flou X par [a, b, α, β].

De plus, on peut généraliser les opérations suivantes aux intervalles flous [70].

Soit M et N deux intervalles flous définis par un quadruplet [a, b, α, β] :

– M + N = [aM + aN , bM + bN , α M + α N , β M + β N]

– λ ×M = [λ ×aM , λ ×bM , λ × α M , λ × β M]

– M −N = [aM −bN , bM −aN , α M + β N , β M + α N]

Ces formules sont plus rapides et plus faciles à utiliser, elles s'appliquent uniquement sur les bornes et le noyau de l'intervalle, c'est-à-dire sur le quadruplet [a, b, a, b]

II.5 Système d'Inférence Floue (SIF)

Nous avons vu que les imprécisions et les incertitudes pouvaient être prises en compte avec les théories des sous ensembles flous et des possibilités, ces deux imperfections sur les connaissances sont souvent liées, dans ce cas, il est nécessaire d'utiliser la théorie des possibilités sur un sous ensemble flou.

Pour les systèmes dont le comportement n'est pas précisément connu (incomplet) des outils comme le raisonnement approximatif et les systèmes d'inférence floue permettent à la fois de décrire son comportement mais aussi d'en exploiter les résultats. Les systèmes d'inférence floue (SIF) sont composés d'une collection de règles floues qui ont la forme générale : "Si p alors q".

La conception des SIF s'appuie en général sur des connaissances expertes pour la définition des termes linguistiques correspondant à chaque variable (ensemble de fonctions d'appartenance) d'une part, et sur des algorithmes d'apprentissage pour la génération des règles d'autre part. Par exemple, les termes linguistiques associés à la vitesse du vent (faible, modérée, rapide, violente) correspondent à des vitesses définies dans l'échelle de Beaufort.

Les SIF sont à utiliser quand :

1. Il existe une expertise humaine que l'on veut exploiter et introduire dans un système automatique.
2. On veut extraire des connaissances à partir de données numériques, en les exprimant dans un langage proche du langage naturel.
3. On veut réaliser une interface homme-machine, donner des explications ou établir des diagnostics immédiatement interprétables.

La réalisation d'un système d'inférence passe par plusieurs étapes (figure 14 [39] [68]), la fuzzyfication et la défuzzyfication des variables d'entrées et de sorties, et la réalisation d'un moteur d'inférence.

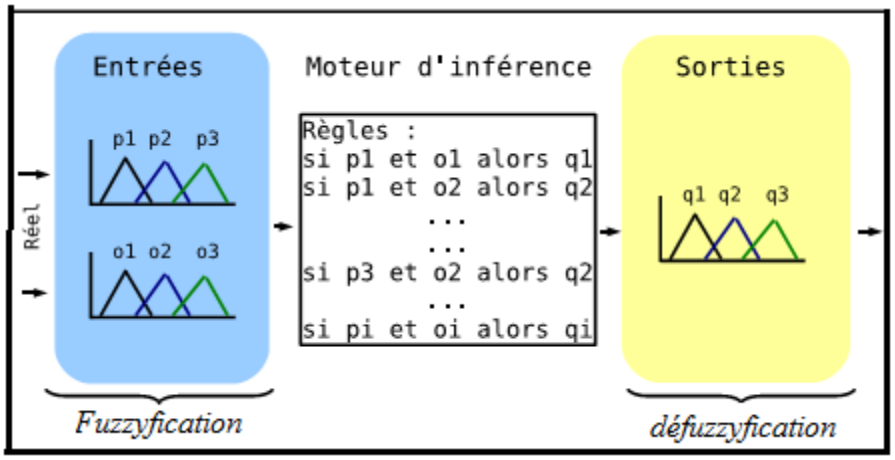

Figure 14: Système d'inférence floue

II.6 Fuzzyfication et défuzzyfication

L'opération qui consiste à transformer un ensemble réel en un ensemble flou est appelée "fuzzyfication", et l'opération inverse est appelée "défuzzyfication". La fuzzyfication est l'étape qui consiste à quantifier à l'aide de termes linguistiques ou de valeurs floues les valeurs réelles d'une variable. car il faut connaître toutes les variations possibles de la variable, sa ou ses fonctions d'appartenance.

La défuzzyfication est une phase décisionnelle, qui permet de transformer une valeur floue d'une variable en une valeur réelle.

Il existe de nombreuses méthodes de défuzzyfication :

- Méthode des hauteurs ;
- Méthode du centre de gravité (dite de Mamdani [57] [58], [59]), elle consiste à prendre pour valeur finale, en sortie, l'abscisse du centre de gravité de l'ensemble flou agrégeant les conclusions ;
- Méthode des moyennes des maxima ;
- Méthode des aires (dite de Sugeno [75] [76]), elle consiste à prendre la médiane qui fait le partage de l'aire en deux.

II.7 Conclusion

Dans ce chapitre, nous avons présenté plusieurs théories permettant la description et la manipulation d'informations imparfaitement définies. La théorie des sous ensembles flous et la théorie des possibilités constituent aujourd'hui ce que l'on appelle la Logique Floue. La Logique Floue permet la formalisation des imperfections

dues aux connaissances inexactes d'un système et à la description du comportement du système par des mots. Elle permet donc la description d'un système et le traitement de données aussi bien numériques que linguistiques.

Nous avons axé notre présentation sur la théorie des sous ensembles flous, car elle 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.

Chapitre 3
Le Formalisme DEVS

Chapitre 3 : Le Formalisme DEVS

III.1 Introduction

Depuis les années 1970, des travaux formels ont été menés pour développer les fondements théoriques de la modélisation et de la simulation des systèmes dynamiques à évènements discrets. Le formalisme *DEVS*, de l'anglais *Discrete EVent system Spécification*, a été introduit par le professeur *B.P. Zeigler* [93] comme un formalisme abstrait pour la modélisation à évènements discrets.

Le formalisme DEVS est une approche de modélisation basée sur la théorie générale des systèmes. Plus précisément, c'est un formalisme modulaire et hiérarchique pour la modélisation, centré sur la notion d'état. Un système est représenté, pour sa forme structurelle, par deux types de modèles (atomiques et couplés).

La modélisation consiste à interconnecter ces différents types de modèles afin de former un nouveau modèle décrivant le comportement du système étudié, c'est l'aspect fonctionnel. Les modèles atomiques sont les composants de base du formalisme, ils décrivent le comportement du système. Leur fonctionnement est proche de celui des "states machine" (machines d'états).

Pour décrire un système plus complexe nous interconnectons plusieurs modèles atomiques pour former un modèle couplé. Ce nouveau modèle peut être utilisé comme modèle de base dans une description de plus haut niveau, c'est l'aspect hiérarchique du formalisme.

Au niveau de la structure du système, cette approche peut sembler statique ; le formalisme DEVS dans sa forme basique ne tient pas compte de l'évolution potentielle de la structure du système, seul les états peuvent évoluer. Toutefois le formalisme a été étendu pour permettre ces changements de structure, et il en est, ou peut en être de même pour d'autres aspects.

Le formalisme DEVS peut être vu comme un environnement de multi-modélisation regroupant de manière cohérente d'autres formalismes de modélisation basés eux aussi sur la théorie générale des systèmes et centrés sur les états. Sa capacité d'ouverture, au sens informatique, en fait un formalisme adapté à un grand nombre de domaines d'application [4][80][64][79].

Au niveau de la simulation, il permet l'analyse de systèmes complexes à évènements discrets décrits par des fonctions de transitions d'états, et des systèmes continus décrits par des équations différentielles [96]. A ce niveau, le principal avantage de l'approche tient au fait que, pour un modèle décrit suivant les spécifications du formalisme, les algorithmes de simulations sont générés automatiquement. Cela permet de s'abstraire totalement de l'implémentation des simulateurs lors de la phase de modélisation, ce qui conduit à une séparation explicite entre la modélisation et la simulation.

Les deux types de modèles (atomique et couplé) du formalisme sont décrits dans la première section, dédiée à la phase de modélisation, et dans la seconde section, nous présentons les aspects de simulation.

III.2 Modélisation DEVS

Le formalisme DEVS repose sur la définition de deux types de modèles : les *modèles atomiques* et les *modèles couplés*. Les modèles atomiques permettent de représenter le comportement de base du système. Les modèles couplés, quant à eux, sont définis par un ensemble de sous modèles atomiques et / ou couplés. Ils permettent de représenter la structure interne du système grâce à la définition de couplages entre modèles.

III.2.1 Modèle Atomique

Le modèle atomique peut être vu comme une machine d'états basé sur le temps. Il permet de décire l'aspect fonctionnel ou comportemental du système. Le modèle atomique fournit une description autonome du comportement du système, défini par des états et des fonctions d'entrées / sorties et de transitions internes du modèle. L'évolution du modèle se fait par changement d'état en fonction de stimuli externes (via une entrée) ou internes (via une fonction de transition). Ces changements d'états ont pour but de déterminer la réponse comportementale du système à ces stimuli.

Dans le formalisme DEVS (classique) la notion de couple (*numéro de port, valeur*) est introduite pour chaque *entrée, ou sortie* d'un modèle atomique. Une entrée correspond à la réception d'une ou plusieurs valeur(s) sur un numéro de port. Une sortie correspond à l'émission d'une ou plusieurs valeur(s) sur un numéro de port.

Le modèle atomique, figure 15, est caractérisé par la spécification suivante (équation2):

$$MA = < X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta > \qquad (2)$$

Avec

– $X = \{(p_{IN}, v) \mid p_{IN} \in Ports\ d'entrée, v \in X_{p_{IN}}\}$: la liste des entrées du modèle, chaque entrée étant caractérisée par un couple (numéro du port / valeur) ;

– $Y = \{(p_{OUT}, v) \mid p_{OUT} \in Ports\ de\ sortie, v \in Y_{p_{OUT}}\}$: la liste des sorties du modèle, chaque sortie étant caractérisée par un couple (numéro du port / valeur) ;

– S : l'ensemble des états ou des variables d'états du système ;

– $\delta_{ext} : Q \times X \rightarrow S$: la fonction de transition externe, où :

  – $Q = \{(S_i, e) \mid S_i \in S, 0 \leq e \leq ta(S_i)\}$: l'ensemble des états $S_{\{1,2,\ldots,n\}}$;
  – e : est le temps écoulé depuis la dernière transition ; La fonction de transition externe spécifie comment le modèle atomique change d'état (passage de l'état S_1 à l'état S_2 (figure 16)) quand une entrée survient (évènement externe) avant que $ta(S_1)$ ne soit écoulé ;

– $\delta_{int} : S \rightarrow S$: 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 n'arrive durant le temps de vie de l'état $ta(S_1)$ (Figure 16) ;

– $\lambda : S \rightarrow Y$: la fonction de sortie.

– $ta : S \rightarrow R^+$: la fonction d'avancement du temps, ou le temps de vie de l'état S ;

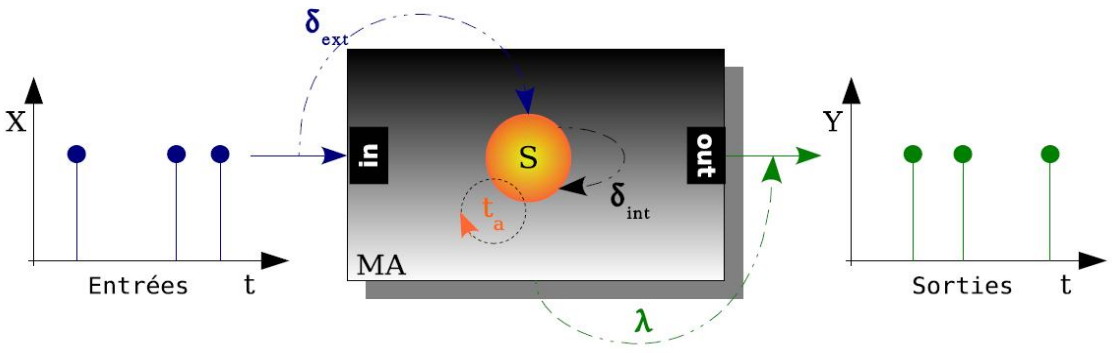

Figure 15 : Description d'un modèle atomique DEVS

Les modèles atomiques réagissent à deux types d'évènements (stimuli) externes ou internes. Un évènement externe provient d'un autre modèle, il déclenche la fonction de transition externe (δ_{ext}) et met à jour le temps de vie de l'état ($ta(Si)$). Un évènement interne entraîne un changement d'état du modèle. Il déclenche les fonctions de transition interne (δ_{int}) et de sortie (λ). Le modèle calcule ensuite avec la fonction d'avancement du temps (ta) la date du prochain évènement interne. Ces enchaînements d'actions et la description du comportement du modèle sont présentés figure 16.

La figure 16 c'est un exemple présente l'évolution des états d'un modèle. $X_{i=\{1,2\}}$ représente les entrées, $S_{i=\{1,2,3\}}$ les états du modèle, e l'écoulement du temps, il est remis à zéro à chaque changement d'état. Ta représente la durée de vie d'un état, elle est mise à jour après chaque changement d'état, si elle est égale à zéro la fonction de transition interne est déclenchée. Y représente les sorties du modèle.
A chaque instant le modèle est dans un état ($S_{i=\{1,2,3\}}$). Si un évènement externe $X_{i=\{1,2\}} \in X$ est détecté avant que $e = ta(Si)$, le système change d'état grâce à la fonction de transition externe ($\delta_{ext}(Si,e,Xi)$).

Dans la figure 16, nous passons de l'état $S1$ à l'état $S2$ lorsque l'entrée $X1$ est détectée, puis de l'état $S1$ à l'état $S3$ lorsque $X2$ est détectée à son tour. Si aucun évènement externe ($X_{i=\{1,2\}}$) n'est détecté, le modèle reste dans le même état pendant un temps donné par la fonction $ta(Si)$. Lorsque le temps du vie de l'état est écoulé, c'est-à-dire lorsque $e=ta(Si)$ le système active sa fonction de sortie ($\lambda(Si)$) (envoie de Y_0 sur la figure 16). De plus, l'état du système est aussi mis à jour grâce à l'exécution de la fonction de transition interne ($\delta_{int}(Si)$). Dans les deux cas, le système est dans un nouvel état (figure 16 : $S1$ avec δ_{int} et $S2$ et $S3$ avec δ_{ext}), avec un nouveau temps de vie et ainsi de suite.

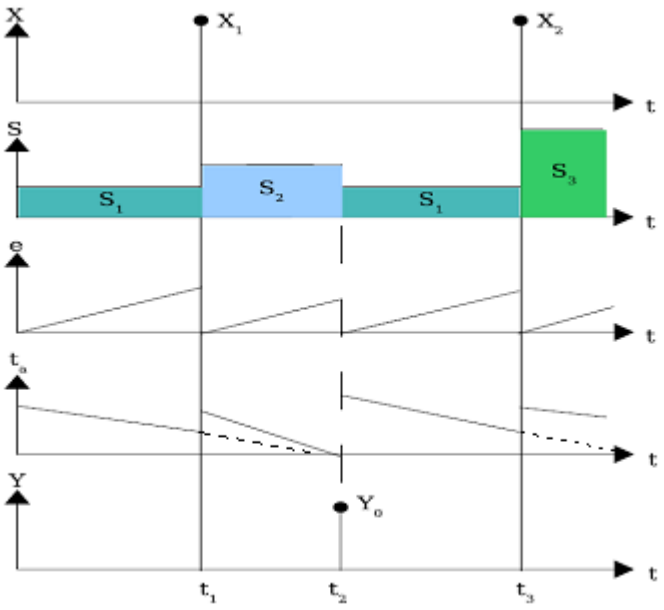

Figure 16 : Description de l'évolution des éléments d'un modèle atomique

En fonction de son temps de vie, l'état d'un modèle peut être défini comme *transitoire* ou *passif*. Si $t_a(S) = 0$ alors la durée de vie de l'état est tellement courte qu'aucun évènement externe ne peut intervenir avant l'arrivée du prochain changement d'état, le système est dans un état transitoire. Si $t_a(S) = \infty$ le système restera dans le même état tant qu'aucun évènement externe n'est détecté, il est dans un état passif.

A partir de modèles atomiques nous pouvons représenter un grand nombre de systèmes en les interconnectant au sein d'un modèle couplé de plus haut niveau.

III.2.2 Modèle Couplé

Le formalisme DEVS utilise la notion de hiérarchie de description qui permet la construction de modèles dits "couplés" à partir d'un ensemble de modèles atomiques et / ou couplés, et de trois relations de couplage :

- une relation de couplage interne (*IC : Internal Coupling*) pour le couplage des ports des sous-modèles qui composent le modèle couplé (figure 17) ;
- une relation de couplage des entrées externes (*EIC : External Input Coupling*) pour le couplage des ports d'entrée du modèle couplé avec les ports d'entrées de ses sous-modèles (figure 17) ;
- une relation de couplage des sorties externes (*EOC : EIC : External Output Coupling*) pour le couplage des ports de sortie du modèle couplé avec les ports de sortie de ses sous-modèles (figure 17).

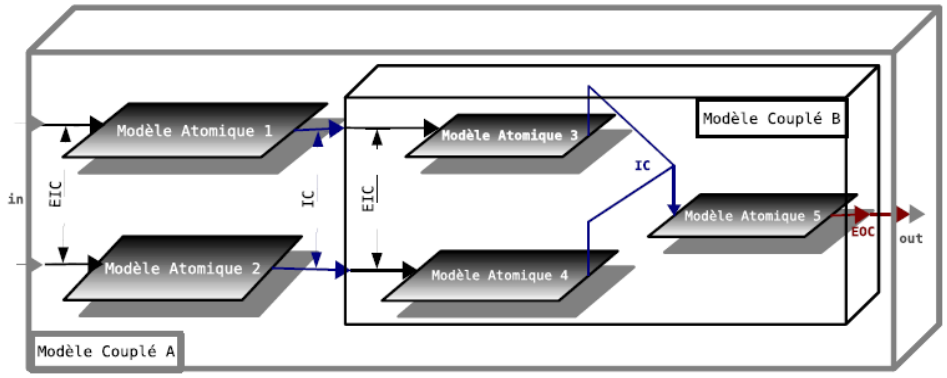

Figure 17 : Description d'un modèle couplé DEVS

La Figure 17 décrit un exemple d'un modèle couplé (équation 3) est une composition de modèles atomiques et / ou de modèles couplés. Il présente un exemple de hiérarchie entre modèles, il est composé de deux modèles couplés (A et B) et cinq modèles atomiques (1, 2, 3, 4 et 5).

Le modèle de plus haut niveau, qui contient tous les autres modèles est le modèle couplé *A*. Le second modèle couplé *B* est composé des modèles atomiques (3, 4, 5). La figure 17 présente le modèle couplé *A* avec 2 entrées "*IN*" et une sortie "*OUT*". Il contient 2 modèles atomiques MA1 et MA2 et un modèle couplé *B*.

Un modèle couplé DEVS est modulaire et présente une structure hiérarchique, ce qui permet la création de modèles complexes à partir de modèles atomiques et / ou couplés. Il est décrit par la formule :

$$MC = < X, Y, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\}, Select > \qquad (3)$$

Avec

– $X = \{(p_{IN}, v) \mid p_{IN} \in Ports\ d'entrée, v \in X_{p_{IN}}\}$: la liste des entrées du modèle, chaque entrée étant caractérisée par un couple (numéro du port / valeur) ;

– $Y = \{(p_{OUT}, v) \mid p_{OUT} \in Ports\ de\ sortie, v \in Y_{p_{OUT}}\}$: la liste des sorties du modèle, chaque sortie étant caractérisée par un couple (numéro du port / valeur) ;

– D : la liste des modèles composant le modèle couplé MC ;

– $M_i = < X_i, Y_i, S_i, \delta ext_i, \delta int_i, \lambda_i, ta_i >$: un modèle atomique ;

– Pour chaque modèle $i \in D\{MC\}$, I_i est l'ensemble des modèles qui influence i ;

– $Z_{i,j}$ est la fonction de transition des sorties du modèle i vers le modèle j, telle que :

  – $Z_{MC,j}$: $X_{MC} \rightarrow X_j$ est la fonction de couplage des entrées externes (EIC) ;

  – $Z_{i,MC}$: $Y_i \rightarrow X_{MC}$ est la fonction de couplage des sorties externes (EOC) ;

  – $Z_{i,j}$: $Y_i \rightarrow X_j$ est la fonction de couplage interne (IC) ;

– *Select* : la liste des priorités entre modèles.

La structure d'un modèle couplé doit répondre à des contraintes telles que, $\forall i \in D$:

1. M_i doit être un modèle atomique ;
2. Une seule fonction $Z_{i,j}$ peut contenir l'ensemble des informations sur le couplage du modèle couplé ;
3. I_i est un sous ensemble de $D \cup MC$ et $i \notin I_i$.

La cohérence et la conservation des propriétés du système entre ces niveaux de hiérarchie sont résumées par la propriété de "fermeture sous composition[1]". En effet dans le formalisme DEVS, chaque modèle est indépendant et peut être considéré comme une entité à part entière ou comme le modèle d'un système plus grand. Il a été montré dans [94] que le formalisme DEVS est fermé sous composition, c'est à dire que pour chaque modèle couplé DEVS, représenté par le couplage d'un ensemble de sous modèles, il est possible de construire un modèle atomique DEVS équivalent.

III.2.3 Exemple

Supposons une lampe (Lumière) commandée par une personne (ActeurLumière). A chaque fois que la personne s'aperçoit que la lampe est éteinte, il l'allume. Le temps de sa réaction est estimé à 2 unités de temps (u.t). La lumière est maintenue pendant 5 u.t, ensuite la lampe s'éteint.

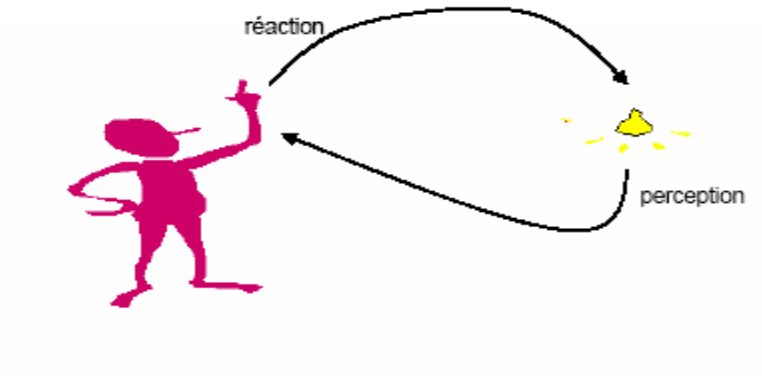

Figure 18 : Système Lumière (Allumer/ Eteindre)

Les modèles atomiques DEVS décrivant le comportement de la personne (ActeurLumière) et celui de la lampe (Lumière), et le modèle DEVS-Couplé (LumièreCouplé) groupant les deux sous-modèles sont définis comme suit :

ActeurLumière = (X, Y, S, δ_{int}, δ_{ext}, λ, t_a)

X={(In, OFF)}

Y={(Out, ON)}

S={Eteint, Allume}

δ_{int}(Eteint)=Allume

δ_{ext}(Allume, e, In ?OFF)=Eteint

λ(Eteint)=(Out !ON)

t_a(Eteint)=2

t_a(Allume)=infini

Lumière =(X, Y, S, δ_{int}, δ_{ext}, λ, t_a)

X={(In, ON)}

Y={(Out, OFF)}

S={Eteint, Allume}

δ_{int}(Allume)=Eteint

δ_{ext}(Eteint, e, In ?ON)=Allume

λ(Allume)=(Out !OFF)

t_a(Eteint)=infini

t_a(Allume)=5

LumièreCouplée=(XI, YI, D, {M_d | dϵD}, EIC, EOC, IC, select)

XI=YI={}

D={ActeurLumière, Lumière}

EIC=EOC={}

IC={((ActeurLumière,Out),(Lumière,In)),((Lumière,Out),(ActeurLumière,In))}

La fonction « **select** » n'est pas définie, car il n'existe que deux composants et il est inutile de définir des règles de priorité entre eux.

III.3 Simulation DEVS

La simulation est un procédé informatique visant à faire évoluer un système afin de prédire son comportement. Etablir une simulation exige donc la définition précise du comportement ainsi que la description des interactions qui existent entre les modèles.

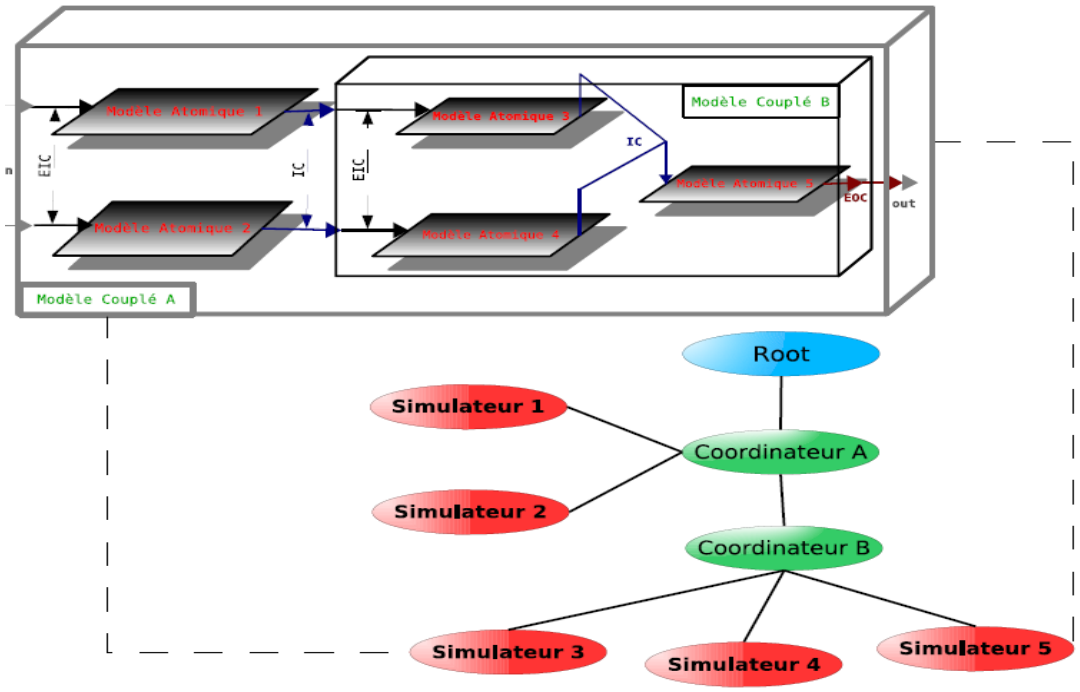

Figure 19 : Arbre de classe du simulateur DEVS

L'une des propriétés importantes du formalisme DEVS est qu'il fournit automatiquement un processeur pour chacun des modèles. Le formalisme DEVS établit une distinction entre la modélisation et la simulation (figure 19) d'un système tel que n'importe quel modèle DEVS puisse être simulé sans qu'il soit nécessaire d'implémenter un processeur spécifique.

La figure 19 montre comment sont organisés les processeurs (*Root, coordinateur, simulateur*) dans le cas l'exemple décrit à la figure 17

Chaque modèle atomique est associé à un *Simulateur* (figure 19) 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 de ces modèles est géré par un processeur spécifique appelé *Root* [94] [81].

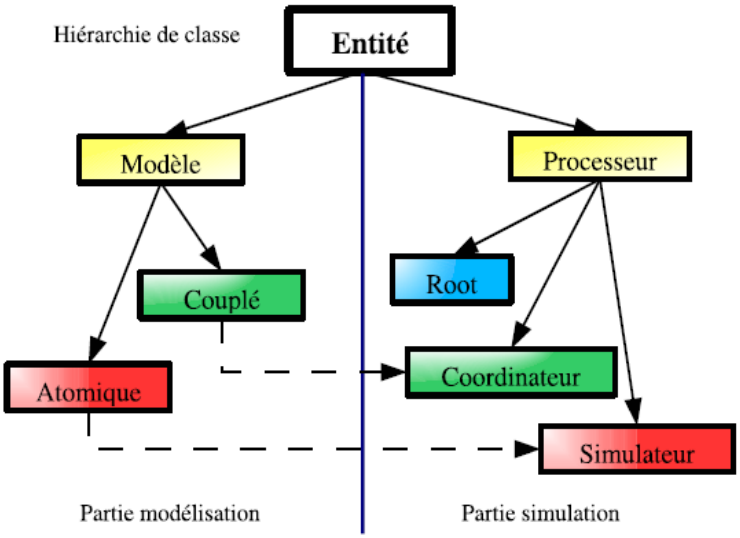

Figure 20 : Hiérarchie de Classe d'un environnement DEVS

Comme nous l'avons vu, le formalisme DEVS est basé, pour la modélisation, sur deux types de modèles : les modèles couplés et atomiques. Ces modèles possèdent des ports d'entrée, des ports de sortie et des variables d'état. Chaque modèle communique grâce à l'envoi et à la réception de plusieurs types de messages. Le principe est décrit dans [94] [81]. Chaque message génère des évènements qui sont stockés dans un échéancier, qui est une structure de données composée d'évènements 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 d'évènements. Nous distinguons deux types d'évènements : les évènements externes et les évènements internes :

– Un évènement externe, prévu à l'instant *t* sur un port d'entrée donné du modèle représente une modification d'un des états du modèle ;
– Un évènement interne, prévu à l'instant *t*, correspond à une modification d'un des états du modèle et provoque l'émission d'une valeur de sortie via la fonction (λ) sur un des ports de sortie du modèle.

Un évènement interne ou externe DEVS peut être caractérisé de la façon suivante (équation 4) :

$$E = (temps, port, valeur) \qquad (4)$$

Le premier champ représente la date d'occurrence de l'évènement, le second désigne le port sur lequel l'évènement intervient, il n'est pas renseigné pour les évènements internes, et le troisième symbolise la valeur de l'évènement. Il a été défini

dans le formalisme *DEVS* [98], dans le cas où plusieurs évènements sont programmés au même instant t, que le choix de l'évènement à exécuter en premier se fait en fonction de la liste de priorité.

Zeigler [93] a proposé un simulateur conceptuel pour simuler les modèles atomiques DEVS. Ce simulateur utilise les informations suivantes :

- l'état courant du modèle s,
- la date d'occurrence du dernier événement tl,
- la date du prochain événement interne tn = tl + ta(s)
- le temps écoulé depuis le dernier événement e = t - tl
- le temps restant avant le prochain événement interne σ = tn - t = ta(s) - e

Le simulateur reçoit trois types de messages, des messages de type (x, t) précisant la date d'occurrence t de l'événement de valeur x, des messages de type (*,t) dits messages de synchronisation des messages internes et un message d'initialisation (i,t). Et il émet deux types de messages, des messages de type (y, t) pour émettre les événements de sortie (fonction de sortie λ) et des messages de type (done, tn) pour prévoir la prochaine transition interne dans le modèle à la date tn.

```
when receive an input (x, t):
   done :=false
        if tl ≤ t ≤ tn then
        e:= t-tl
        s:= δext(s,e,x)
        tl := t
        tn := tl + ta(s)
        send (done, t)
when receive an input (*, t)
   done=true
   if (t = tn) then
        y := λ(s)
        send y-message (y, t)
        s := δint(s)
        tl := t
        tn :=t + ta(s)
        send (done, t)
when receive an (i, t)
        tl := t-e
        tn := tn + ta(s)
```

Figure 21 : Le simulateur conceptuel DEVS [93][95]

La mécanique de simulation est très simple. Au début de chaque simulation, un message d'initialisation est envoyé au simulateur, il permet de réinitialiser la date du dernier événement tl et de calculer la date du prochain événement tn. Lorsque le simulateur reçoit un message (x, t), il exécute la fonction de transition externe

$\delta_{ext}(s,e,x)$ et affecte t à tl et recalcule la date d'occurrence du prochain événement. La réception du message (*, t) permet d'envoyer le message de sortie y-message et d'exécuter la transition interne correspondante.

III.4 Les différentes variantes du DEVS

III.4.1 Introduction

On peut intégrer dans le multi-formalisme DEVS de nombreux autres formalismes ou méthodes de modélisation. Dans cette partie nous nous intéressons aux extensions de DEVS qui permettent de prendre en compte imprécision et incertitude sur les événements ou sur les états. L'incertitude intervient au niveau des changements d'états, alors que l'imprécision intervient sur le temps ou les valeurs des événements. Dans les deux cas, ces changements modifient la structure des modèles et les algorithmes de simulation.

Le formalisme Fuzzy-DEVS [54] permet de traiter des données incertaines, cette incertitude intervient uniquement au niveau des changements d'états du système. Une méthode de défuzzyfication permet de transformer les incertitudes sur le temps des évènements en valeurs réelles, et effectue l'opération inverse pour fournir un temps de sortie floue. Le formalisme Min-Max DEVS [38] [43] permet de prendre en compte uniquement une imprécision sur le temps d'un événement pour des systèmes électroniques, et le modèle iDEVS manipule des variables précises ou imprécises. Son rôle est de décrire l'aspect comportemental d'une partie d'un système à paramètres imprécis. Si tous les paramètres du modèle iDEVS sont précis, il a le même comportement qu'un modèle DEVS classique. Ces trois formalismes sont détaillés dans ce chapitre.

IV.4.2 Fuzzy-DEVS

Le formalisme Fuzzy-DEVS introduit par *Y. Kwon* dans [54], dérive du formalisme DEVS tout en conservant sa sémantique, une partie de ses concepts et sa modularité. Il est basé sur la logique floue, une règle "max-min" définie dans [54] et les méthodes de fuzzyfication et de défuzzyfication pour prendre en compte les incertitudes sur le temps des évènements. Pour permettre la simulation, les incertitudes sur le temps doivent être transformées en valeurs réelles (défuzzyfication); pour pouvoir être exploitées, les temps de sortie sont de nouveau transformés en données incertaines (fuzzyfication).

La structure du modèle atomique flou décrite dans [54] est la suivante :

$$\tilde{AMF} < X, Y, S, \tilde{\delta}_{\text{int}}, \tilde{\delta}_{ext}, \tilde{\lambda}, \tilde{ta} > \qquad (5)$$

avec :

– X, Y, l'ensemble des ports d'entrée et de sortie non flous ;

– S, l'ensemble des états séquentiels non flous ;

– $\tilde{\delta}_{\text{int}} : S \times S \rightarrow [0,1]$, fonction floue de transition interne ;

– $\tilde{\delta}_{ext} : Q \times X \times S \rightarrow [0,1]$, fonction floue de transition externe, avec :

  – $Q = \{(s,e) \mid s \in S, 0 \leq e \leq ta(s)\}$, ou

  – $ta(s)$ est la valeur de défuzzyfication de $\tilde{ta}(S)$;

– $\tilde{\lambda} : S \times Y \rightarrow [0,1]$, fonction floue de sortie ;

– $\tilde{ta} : S \times \tilde{N} \rightarrow [0,1]$, fonction floue d'avancement du temps, où $\tilde{N}$ est l'ensemble des nombres flous appartenant à $R_{[0,\infty[}$.

Tel qu'il est décrit, le modèle atomique Fuzzy-DEVS ne permet de prendre en compte que les différentes possibilités de transition $(\tilde{\delta}_{\text{int}}, \tilde{\delta}_{ext})$ entre état. Les entrées et les sorties du modèle ne sont d'ailleurs pas représentées comme imprécises. De plus, le modèle atomique flou de Fuzzy-DEVS, contrairement au modèle atomique DEVS, est non déterministe, c'est-à-dire qu'il ne répond pas aux deux conditions suivantes :

1. la fonction de transition interne est exécutée ($\delta_{\text{int}}(s_i) = s_{i+1}$) quand la durée de vie de l'état est écoulée ($ta = 0$ ou $ta = e$) et la fonction de transition externe $(\delta_{\text{Ext}}(S_i, e_i, X_i) = S_{i+1})$ est exécutée lorsqu'un évènement externe arrive ;
2. la fonction de sortie ($\lambda(s_i) = Y_i$) est exécutée quand la durée de vie d'un état est finie ($ta = 0$).

Dans le formalisme Fuzzy-DEVS, l'état suivant du système S_{i+1} n'est pas déterminé avec δ_{int} et δ_{ext} mais avec la règle "max-min" ; elle détermine, suivant un algorithme, l'état qui a le plus de chances d'être mis à jour. Les différentes possibilités de changement d'état sont représentées par des matrices, et l'évolution du modèle par des arbres de possibilités [54] [1].

Fuzzy-DEVS ne traite pas les valeurs imprécises d'un modèle, mais propose une méthode qui fournit un arbre de possibilités représentant l'évolution des états du système, les feuilles sont les états et les branches la possibilité associée à un état.

La figure 21 présente l'évolution d'un système en fonction des possibilités μ_s de transition entre états. Le système est défini par deux entrées $X = \{x_1, x_2\}$, deux sorties $Y = \{y_1, y_2\}$ et trois états $S = \{s_1, s_2, s_3\}$.

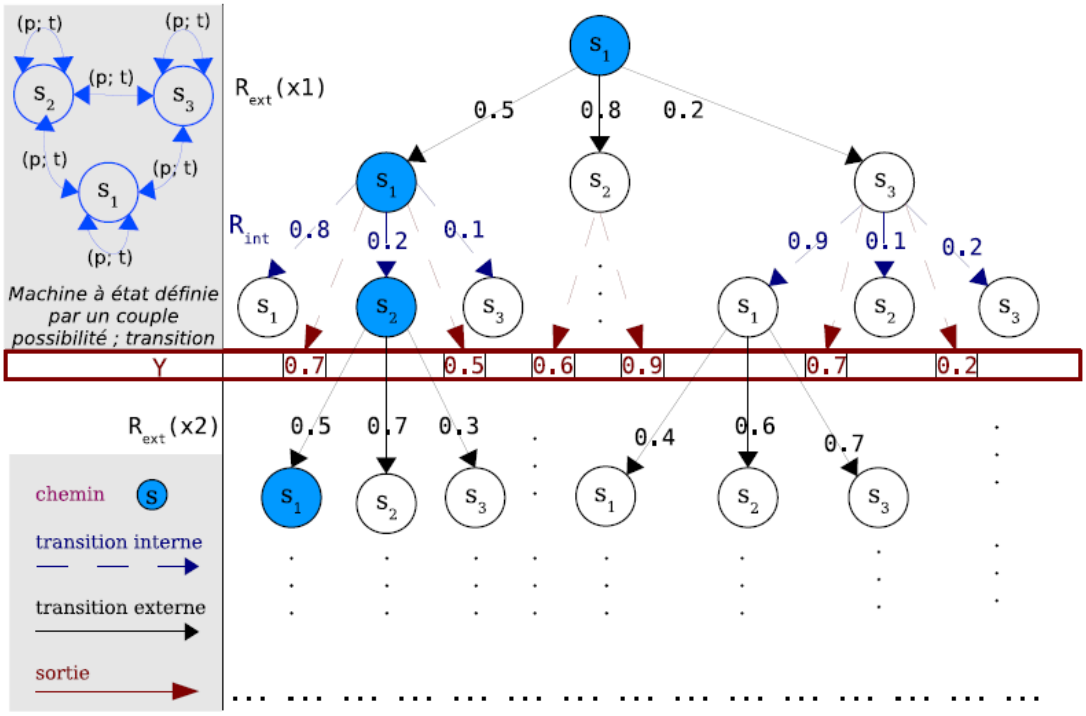

Figure 22 : Arbre de simulation Fuzzy-DEVS [56]

L'évolution du système se fait non pas en fonction de δ_{int} et δ_{ext} comme dans le formalisme DEVS classique, mais en fonction de matrices représentant les relations (R) entre fonctions de transition et états.

Relation interne

$$R_{int} = \begin{array}{c} \\ S1 \\ S2 \\ S3 \end{array} \begin{array}{c} \begin{array}{ccc} S1 & S2 & S3 \end{array} \\ \begin{pmatrix} 0,8 & 0,2 & 0,1 \\ 0,6 & 0,1 & 0,3 \\ 0,9 & 0,1 & 0,2 \end{pmatrix} \end{array}$$

Relation de sortie

$$R_{out} = \begin{array}{c} \\ S1 \\ S2 \\ S3 \end{array} \begin{array}{c} \begin{array}{cc} Y1 & Y2 \end{array} \\ \begin{bmatrix} 0,7 & O,5 \\ 0,6 & 0,9 \\ 0,7 & 0,2 \end{bmatrix} \end{array}$$

Relations externes $R(x_1)$:

$$\begin{array}{c} \\ S1 \\ S2 \\ S3 \end{array}\begin{array}{c} \begin{array}{ccc} S1 & S2 & S3 \end{array} \\ \begin{pmatrix} 0,5 & 0,8 & 0,2 \\ 0,7 & 0,4 & 0,2 \\ 0,2 & 0,7 & 0,6 \end{pmatrix} \end{array} \quad \text{, et} \quad \begin{array}{c} \\ S1 \\ S2 \\ S3 \end{array}\begin{array}{c} \begin{array}{ccc} S1 & S2 & S3 \end{array} \\ \begin{pmatrix} 0,4 & 0,6 & 0,7 \\ 0,5 & 0,7 & 0,3 \\ 0,1 & 0,4 & 0,8 \end{pmatrix} \end{array}$$

L'état suivant du système s_{i+1} est choisi en fonction de la règle "max-min" définie, pour les transitions :

– interne par : $\mu_{\tilde{S}_{i+1}}(S_{i+1}) = \max(\min(\mu_{\tilde{S}_i}(S_i), \tilde{\delta}_{int}(S_i, S_{i+1})))$;

– externe par : $\mu_{\tilde{S}_{i+1}}(S_{i+1}) = \max(\min(\mu_{\tilde{S}_i}(S_i), \tilde{\delta}_{ext}(S_i, X_i, S_{i+1})))$

L'état suivant du système (figure 21), après détection 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 $s_1 \rightarrow s_1$.

Après une transition interne $max\ (min\ (0.8, 0.2), min\ (0.8, 0.1)) = 0.2$, nous passons dans l'état $s_1 \rightarrow s_2$.

Nous pouvons remarquer que comme dans le formalisme DEVS classique, avant une transition interne, la fonction de sortie est déclenchée.

Fuzzy-DEVS est un formalisme qui traite l'imprécision [54] [1]. L'idée de base est très intéressante, mais est limitée à l'étude de systèmes à états finis. De plus, ce formalisme ne paraît pas totalement cohérent avec le formalisme DEVS, mais il fournit les idées pour l'amélioration de ce domaine, comme la possibilité de définir le temps de vie d'un modèle t_a à l'aide d'un label linguistique [54]. Pour autant il ne permet pas de répondre aux problématiques de prise en compte des paramètres imprécis et pas seulement au niveau des transitions entre états.

III.4.2 Min-Max DEVS

Le formalisme Min-Max DEVS [38] [43] introduit par *N. Giambiasi* fait suite aux travaux de M. Smaili [73] sur la modélisation et la simulation de circuits logiques à retard flou. Le but de Min-Max DEVS est de permettre de modéliser et de simuler des systèmes réels pour lesquels les valeurs des retards ne sont pas connues avec exactitude. Ce formalisme a été proposé pour modéliser les systèmes dans lesquels la durée de vie des états transitoires est représentée par des intervalles de temps et non par les valeurs moyennes comme dans le formalisme DEVS classique. Les modèles

atomiques Min-Max DEVS et DEVS classique sont quasiment identiques, le seul changement est la représentation de la fonction d'avancement du temps ta. Dans Min-Max DEVS elle est définie comme suit :

– $ta(si) : S \rightarrow R^+ \times R^+$;

– $ta(si) = (d_{min}, d_{max})$; où

- d_{min} est le temps minimum pendant lequel le modèle reste dans le même état si, d_{min} représente l'évolution la plus rapide du système ;
- d_{max} est le temps maximum pendant lequel le modèle reste dans le même état si, d_{max} représente l'évolution la plus lente du système.

Cette modification entraîne deux changements importants par rapport au formalisme DEVS classique. Le premier intervient au niveau de l'évolution du modèle; dans Min-Max DEVS les évènements externes sont choisis en fonction du temps minimum ; s'il n'y a pas d'évènement externe détecté, un évènement interne est déclenché au temps maximum. Le second changement touche les algorithmes de simulation, la fonction ta ayant été modifiée, il faut en tenir compte au niveau de la simulation. Chaque évènement est déclenché en fonction du temps, et tous les temps des évènements internes sont fixés par la fonction ta. Pour cela deux nouvelles variables de temps ont été définies, ts représentant le temps maximum avant le prochain évènement et tf représentant le temps minimum avant le prochain évènement. Ces évolutions ont engendré une modification des algorithmes de simulation, ils sont présentés dans [38].

Ces travaux permettent la représentation et la prise en compte des imprécisions au niveau du temps de vie de l'état ta, mais sont essentiellement applicables dans les domaines de la détection de fautes ou de la prise en compte de retards flous dans la définition de circuits digitaux.

IV.4.3 Modèles iDEVS

Les deux modèles présentés précédemment sont considérés comme un cheminement scientifique qui a conduit [56] à proposer une nouvelle méthode de modélisation et de simulation de système à paramètre imprécis, méthode appelée iDEVS. Pour prendre en compte les imprécisions, les modèles iDEVS doivent dériver ou instancier la classe ***IntervalleFlou*** *(une classe qui représente une librairie pour la définition et la manipulation des données de type Flou)*. Le but est de fournir au concepteur les moyens de définir des modèles 'imprécis' indépendamment d'une

plateforme ou d'une application données, juste en important la classe dans l'environnement de modélisation. Le concepteur peut ainsi modéliser son système imprécis à partir de la combinaison de modèles dérivés de la classe *IntervalleFlou* et de modèles atomiques DEVS classiques. Il est à noter que le modèle couplé résultant d'une combinaison de modèles 'imprécis' et de modèles classiques est automatiquement un modèle qui retourne des résultats imprécis. Tous les modèles iDEVS manipulent des données imprécises (données de type *IntervalleFlou*), ils ont donc la possibilité d'utiliser les fonctions définies dans la classe *IntervalleFlou*. Dans la suite, nous décrivons les évolutions apportées aux modèles DEVS pour permettre la prise en compte d'imprécisions.

III.4.3.1 Modèle atomique iDEVS

Un modèle atomique iDEVS, est semblable au modèle atomique DEVS. Sa particularité est qu'il peut manipuler des variables précises ou imprécises. Son rôle est de décrire l'aspect comportemental d'une partie d'un système à paramètres imprécis. Si tous les paramètres du modèle iDEVS sont précis, il a le même comportement qu'un modèle DEVS classique.

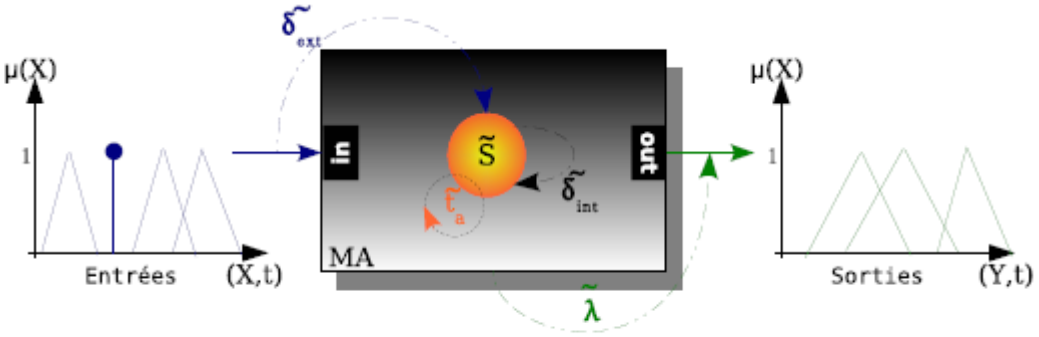

Figure 23 : Modèle atomique iDEVS

Au niveau des modèles atomiques DEVS et iDEVS, le comportement des fonctions de transitions interne et externe (δ_{int} ,δ_{ext}) est strictement identique. Par contre, pour la manipulation de variables imprécises, nous utilisons les fonctions surchargées dans la classe *IntervalleFlou* (+,−, /,*cos*,*etc*.). Pour décrire δ_{int} et δ_{ext} , l'utilisateur n'a qu'à indiquer quelles sont les variables imprécises, l'emploi des fonctions appropriées se fait automatiquement.

Pour les fonctions d'avancement du temps et de sortie, des modifications structurelles ont été introduites, mais elles restent imperceptibles pour l'utilisateur

final. La fonction d'avancement du temps, doit retourner une valeur précise afin que l'évènement correspondant puisse être inséré dans l'échéancier.

Pour cela ils ont ajouté une fonction de défuzzyfication qui est présentée dans l'algorithme ci-aprés.Cette fonction est automatiquement déclenchée et l'utilisateur final n'a pas à se soucier de son fonctionnement. Un utilisateur averti peut modifier l'implémentation de cette fonction à partir de la classe *IntervalleFlou*.

Les données manipulées par un modèle atomique flou sont représentées par un quadruplet [*a*, *b*, ψ, ω], défini dans la classe *IntervalleFlou* (Figure23). Si *a* = *b* et α = β = 0 le modèle iDEVS devient un modèle DEVS classique (non flou) manipulant des données précises $AM_{\tilde{i}DEVS} \supset AM_{DEVS}$.

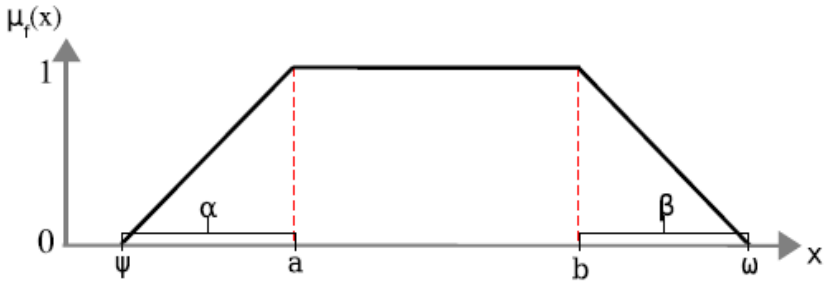

Figure 24 : Intervalle flou

L'équation 6 présente en détail le modèle atomique iDEVS général. Le "tilde" (˜) sur un paramètre signifie qu'il est imprécis, ou qu'il manipule des variables imprécises. Les valeurs d'entrée peuvent être imprécises $\tilde{X}$; à la réception d'une valeur en entrée, la fonction floue de transition externe $\tilde{\delta}_{ext}$ est déclenchée ; elle met à jour l'état du système $\tilde{S}$ et sa durée de vie **σ** en fonction des spécifications définies par le concepteur. Si aucune entrée n'est détectée avant l'expiration de la durée de vie, les fonctions floues de transition interne $\tilde{\delta}_{int}$ et de sortie $\tilde{\lambda}$ sont déclenchées. $\tilde{\delta}_{ext}$ met à jour l'état du système en fonction des spécifications définies par le concepteur et $\tilde{\lambda}$ génère les sorties sur $\tilde{Y}$.

$$\tilde{MA} < \tilde{X}, \tilde{Y}, \tilde{S}, \tilde{ta}, \tilde{\delta}_{int}, \tilde{\delta}_{ext}, \tilde{L} > \quad (6)$$

avec :

– $\tilde{X} = \{(p, \tilde{v}) \mid p \in Ports\ d'entrée, \tilde{v} \in \tilde{X}_p\}$: la liste des ports d'entrée, chaque port étant caractérisé par un couple (numéro du port / valeur), où la valeur peut être définie comme précise ou imprécise ;

– $\tilde{Y} = \{(p, \tilde{v}) \mid p \in Ports\ de\ sortie, \tilde{v} \in \tilde{Y}_p\}$: la liste des ports de sortie, chaque port étant caractérisé par un couple (numéro du port / valeur), où la valeur est précise ou imprécise en fonction du comportement du modèle ;

– $\tilde{S}$: l'ensemble des états ou des variables d'états précises S ou imprécises $\tilde{S}$ du système $S \in \tilde{S}$;

– $\tilde{ta}(\tilde{S}) \rightarrow \Re^+$: la fonction d'avancement du temps, l'algorithme 1 décrit la fonction $\tilde{ta}$;

Algorithme : Fonction d'avancement du temps imprécis $\tilde{ta}$

```
// déclaration des variables de classe
IntervalleFlou τ= [0,0,0,0]    // l'intervalle représentant le temps de fin de simulation
réel Λ                         // la somme des degrés d'appartenance l défuzzifié
réel nbrDefuz ← 1              // variable qui compte le nombre de défuzzifications
réel moyΛ= Λ / nbrDefuz        // variable qui conserve la moyenne des λ, elle est renvoyée à la fin de la
                               simulation par chaque modèle
// Fonction d'avancement du temps imprécise
Fonction réel ta (état S) {
 Soit σ le temps de vie de l'état S
si σ est précis // la nature de σ est testé, si σ est précis la fonction ta a un comportement classique
 ta ← σ
 τ ← τ + σ // l'intervalle τ est incrémenté de σ, en fin de simulation il va fournir un intervalle de
 temps
sinon
 ta ← σ.coefEEM() // σ est une instance de la classe IntervalleFlou.
 Λ ← Λ + μ(σ.coefEEM()) // Λ est la somme des λ défuzzifiés, la fonction μ(x) renvoie la valeur
 de λ pour x
 nbrDefuz ← nbrDefuz+1
 τ ← τ + σ // nous ajoutons à l'intervalle t l'intervalle σ
retourner ta )
```

- $\tilde{\delta}_{ext} : \tilde{Q} \times \tilde{X} \rightarrow \tilde{S}$: la fonction de transition externe, où :
  - $\tilde{Q} = \left\{ (\tilde{S}_i, e) \mid \tilde{S}_i \in S, 0 \leq e \leq ta(\tilde{S}_i) \right\}$: l'ensemble des états précis ou imprécis $\tilde{S}_{\{1,2,\ldots,n\}}$;
  - e : est le temps écoulé depuis la dernière transition ; la fonction de transition externe spécifie comment le modèle atomique change d'état (passage de l'état $\tilde{S}_1$ à l'état $\tilde{S}_2$ quand une entrée précise ou imprécise survient (évènement externe) avant que $ta(\tilde{S}_1)$ ne soit écoulé ;
- $\tilde{\delta}_{int} : \tilde{S} \rightarrow \tilde{S}$: la fonction de transition interne. Elle permet de passer d'un état $\tilde{S}_1$ à l'instant t_1, à un état $\tilde{S}_2$ à l'instant t_2 lorsqu'aucun évènement externe n'arrive durant le temps de vie de l'état $ta(\tilde{S}_1)$;
- $\tilde{\lambda} : \tilde{S} \rightarrow \tilde{Y}$: la fonction de sortie, elle renvoie les sorties du modèle et aussi les variables de classe τ et $moy\Lambda$ (Algorithme 1).

IV.4.3.2 Modèle couplé iDEVS

Le modèle couplé iDEVS a la même forme que le modèle couplé DEVS, ils sont tous deux décrits de la même façon. La seule différence est que le modèle couplé iDEVS peut être composé de modèles DEVS classiques ou de modèles iDEVS. Par conséquent les variables d'entrée et de sortie sont définies comme imprécises. Une donnée imprécise inclut une donnée précise.

Dans l'équation 7 nous décrivons en détail les différents aspects du modèle couplé iDEVS.

$$CM = < \tilde{X}, \tilde{Y}, D, \{M_{\tilde{i}}\}, \{I_{\tilde{i}}\}, \{Z_{\tilde{i},\tilde{j}}\}, Ic > \quad (7)$$

avec :

- $\tilde{X} = \left\{ (p, \tilde{v}) \mid p \in Ports \text{ d'entrée}, \tilde{v} \in \tilde{X}_p \right\}$: la liste des ports d'entrée, chaque port étant caractérisé par un couple (numéro du port / valeur) ;

– $\tilde{Y} = \left\{ (p, \tilde{v}) | p \in Ports \text{ de sortie}, \tilde{v} \in \tilde{Y}_p \right\}$: la liste des ports de sortie, chaque port étant caractérisé par un couple (numéro du port / valeur) ;

– D : la liste des modèles composant le modèle couplé CM ;

– $M_{\tilde{i}} < X_{\tilde{i}}, Y_{\tilde{i}}, S_{\tilde{i}}, \delta_{ext_{\tilde{i}}}, \delta_{\text{int}_{\tilde{i}}}, \lambda_{\tilde{i}}, ta_{\tilde{i}} >$: un modèle atomique imprécis $M_{\tilde{i}} \subset M_{\tilde{i}}$;

– Pour chaque modèle $\tilde{i} \in D \cup \{CM\}$, $I_{\tilde{i}}$ est l'ensemble des modèles qui influencent $\tilde{i}$;

– $Z_{\tilde{i},\tilde{j}}$ est la fonction de transition des sorties du modèle $\tilde{i}$ vers le modèle $\tilde{j}$, telle que :

  – $Z_{CM,\tilde{j}} : X_{CM} \to X_{\tilde{j}}$ est la fonction de couplage des entrées externes (EIC) ;

  – $Z_{\tilde{j},CM} : Y_{\tilde{i}} \to X_{CM}$ est la fonction de couplage des sorties externes (EOC) ;

  – $Z_{\tilde{i},\tilde{j}} : Y_{\tilde{i}} \to X_{\tilde{j}}$ est la fonction de couplage interne (IC) ;

  – L : la liste des priorités entre modèles.

L'intérêt de la méthode iDEVS est la prise en compte des imprécisions en définissant les paramètres des modèles sous forme d'intervalles. Les intervalles peuvent être décrits à partir de valeurs numériques ou linguistiques. Ces modèles peuvent être utilisés pour la description de systèmes spécifiques, c'est-à-dire que le concepteur doit lui-même coder son modèle pour représenter le système à simuler.

Cependant il faut noter que l'algorithme de simulation a largement été modifie (fonction d'avancement du temps) et de ce fait une modification du concept de DEVS classique.

III.5 Conclusion

Nous avons présenté dans ce chapitre le formalisme DEVS développé par B.P Zeigler.ce formalisme peut être défini comme une méthodologie universelle et générale qui fournit des outils pour modéliser et simuler des systèmes dont le comportement repose sur la notion d'évènements, et permet la spécification de systèmes complexes à évènements discrets sous forme modulaire et hiérarchique.

Dans la deuxième partie du chapitre nous avons présenté trois variantes basées sur le formalisme DEVS permettant de prendre en compte des imperfections sur les paramètres des modèles. Fuzzy-DEVS ne traite que les incertitudes au niveau des transitions entre états, et Min-Max-DEVS est trop spécifique à un domaine

d'application, et ne traite que les retards sur le temps de déclenchement des évènements et le formalisme iDEVS qui traite le problème de paramètres imprécis par la définition d'une classe qui traite les opérations floues, mais en modifiant le profond de fonctionnement du formalisme DEVS classique décrit par Zeigler.

Ces variantes nous ont servi pour définir une nouvelle méthode de prise en compte des imprécisions au niveau des paramètres des modèles DEVS. Dans le chapitre suivant nous présentons cette nouvelle méthodologie de modélisation basée sur la théorie des sous ensembles flous et sur le formalisme DEVS.

Chapitre 4

Approche de modélisation imprécise

Chapitre 4 : Approche de modélisation imprécise

IV.1 Introduction

L'étude de systèmes complexes nécessite la prise en compte de nombreux paramètres, souvent liés ou dépendants de phénomènes mal connus, ou difficilement étudiables. Un formalisme de modélisation bien adapté et notamment qui prend en compte l'incertitude de tous ces paramètres peut être d'une grande aide. La modélisation étant une étape essentielle de notre processus, nous nous sommes donc basés sur DEVS, qui est déjà en soi un outil bien abouti et qui ne demande qu'à être étendu. La théorie des ensembles flous, qui propose une palette intéressante pour la représentation de connaissances et de données incertaines, nous aide à définir un formalisme de modélisation de systèmes complexes à paramètres flous ou vagues. Notre formalisme doit bien entendu respecter les exigences de compatibilité définies par DEVS afin d'être totalement associable avec celui-ci.

En vue de l'utilisation de DEVS, quelques réajustements ont été nécessaires, car originellement, DEVS ne prend en compte que des données réelles. Ce chapitre présente les améliorations apportées à la partie modélisation de DEVS nécessaires à l'intégration de la logique floue. La première partie donne la définition d'un évènement DEVS basique, la seconde présente l'évènement imprécis et la troisième partie propose une approche de traitement. Enfin, une quatrième partie donne un exemple d'application sur le système de photovoltaïque.

IV.2 Évènements DEVS

Le formalisme DEVS est basé pour la modélisation sur deux types d'éléments : les modèles couplés et atomiques. Ces éléments 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 modèle 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.

Comme nous l'avons mentionné dans le chapitre (3) dans la section (2.1), un évènement DEVS peut être caractérisé par :

E = (temps; port; valeur)

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 dans l'échéancier de DEVS suivant une liste de priorité ; on éclate l'évènement E qui a lieu au temps T en plusieurs évènements E1, E2, E3 pris en compte toujours au temps T mais suivant la liste de priorité définie par l'utilisateur.

Comme nous l'avons déjà vu, le formalisme DEVS ne permet pas la prise en compte de données floues. Celle-ci passe par une évolution dans la définition des évènements DEVS, qui sont à la base de la simulation. La partie suivante présente une extension des évènements originaux DEVS afin de permettre le traitement de telles données.

IV.2.1 Évènements Imprécis

Nous avons vu qu'un évènement dépendait du temps, d'un port et d'une valeur qu'il soit interne ou externe. Le port est fixé et ne peut être incertain mais au niveau des valeurs ou du temps un flou peut être envisagé.

Par exemple pour les incendies :

– Sur sa valeur, à **t** donné la puissance du vent sera comprise entre 60 et 90 Km/h. On ne connaît pas la valeur réelle de l'évènement ;

– Sur son temps, la valeur est connue le feu va parcourir 2Km mais le temps n'est pas fixé en 40 ou 60 minutes ;

– Sur la valeur et le temps.

Cette partie présente un évènement DEVS imprécis basé sur les évènements DEVS mais qui permet de prendre en compte une incertitude sur les valeurs et sur le temps. La première section présente un évènement qui prend en compte un flou sur les valeurs. L'incertitude des données pouvant intervenir sur les valeurs ou sur les dates, la seconde section décrit un évènement imprécis qui prend en compte un flou sur le temps. La troisième section propose la prise en compte du flou sur les valeurs et les dates d'occurrence.

IV.2.2 Évènements à valeurs incertaines

La figure 24 décrit les différents composants d'un évènement à valeurs floues, et permet de définir, à un temps donné, les possibilités qui seront calculées grâce à l'interprétation de fonctions floues et à un ensemble de règles définies par des spécialistes du domaine.

Pour décrire une variable floue, et afin de fournir des outils de représentation adaptés aux spécialistes et à leurs domaines d'étude, nous utilisons la méthode de Bisgambiglia [56] qui permet de décrire rapidement et simplement une imprécision. Elle est basée sur une description de l'intervalle à partir de points références, c'est-à-dire quatre couples : valeur-degré d'appartenance, de type [(a,1); (b,1); (ψ,0); (ω,0)]. Ils sont présentés sur la figure 24 :

a et b sont les sommets de la fonction d'appartenance ;

$\psi = a - \alpha$ est la limite inférieure de l'intervalle ;

$\omega = b + \beta$ est la limite supérieure de l'intervalle ;

IV.2.3 Évènements à dates incertaines

Exemple d'évènement à date floue : un feu parcourt 2 kilomètres dans un intervalle de 40 à 60 minutes (figure 25).

Le simulateur abstrait de DEVS insère les évènements 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.

Comment prévoir quand aura lieu le prochain évènement si l'on ne connaît pas la date exacte de l'évènement actuel ? De plus celui-ci influence peut être tous les évènements suivants ! Comment faire pour continuer la simulation ?

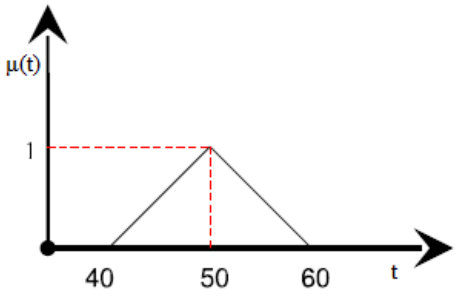

Figure 25 : Fonction Floue : temps mis par un feu pour parcourir 2 Km

Un des moyens est de transformer le flou au niveau des dates (du temps) en flou au niveau des valeurs. Pour revenir sur notre exemple : on fixe une date représentant la plus grande possibilité que l'évènement ait lieu (ici : que le feu ait parcouru 2 Km, soit 50 minutes). A partir de là, il faut définir une fonction floue représentant la distance que le feu peut parcourir en 50 minutes.

L'utilisation de fonctions de transition de domaine permet d'effectuer cette étape. Ces fonctions doivent être écrites par les spécialistes du domaine et permettent de passer d'un flou sur une date à un flou sur une valeur.

Dans notre exemple :

– Au minimum le feu parcours 2 Km en 60 minutes, donc en 50 il en aurait parcouru 1,6 Km ;

– Au maximun le feu parcours 2 Km en 40 minutes, donc en 50 il en aurait parcouru 2,5 Km.

A partir de ces nouvelles données on peut créer une nouvelle fonction représentant la distance parcourue par le feu en 50 minutes (figure 25).

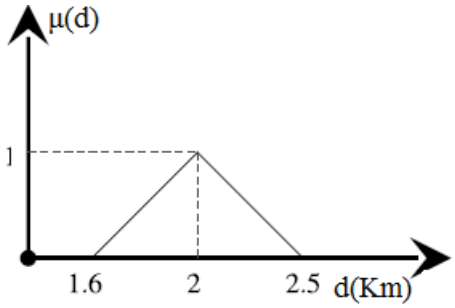

Figure 26 : Fonction Floue : Distance parcourue par un feu en 50 mn.

Ce mécanisme revient à "défuzzifier" le temps, c'est-à-dire à supprimer l'incertitude pour la transformer en valeur réelle.

La logique floue 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 le temps pourra être flou (on parle des valeurs du temps ou des dates), des fonctions de "Fuzzyficication" et "Défuzzycation" (figure 26) permettront de gérer ce flou au niveau des modèles (atomiques et couplés) et de la simulation. Ces fonctions contiendront les fonctions de transition de domaine.

Ceci permet de ne jamais interrompre le dialogue avec les spécialistes de domaine. Leurs données ou connaissances pourront être utilisées en entrée (elles seront adaptées ou traduites pour l'application par nos fonctions), de même pour les sorties, qu'ils pourront ainsi comprendre, interpréter et exploiter.

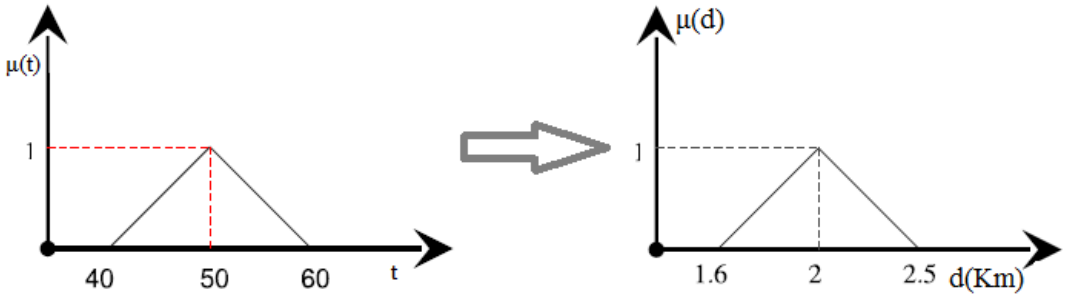

Figure 27 : Fonction Défuzzyfication de temps

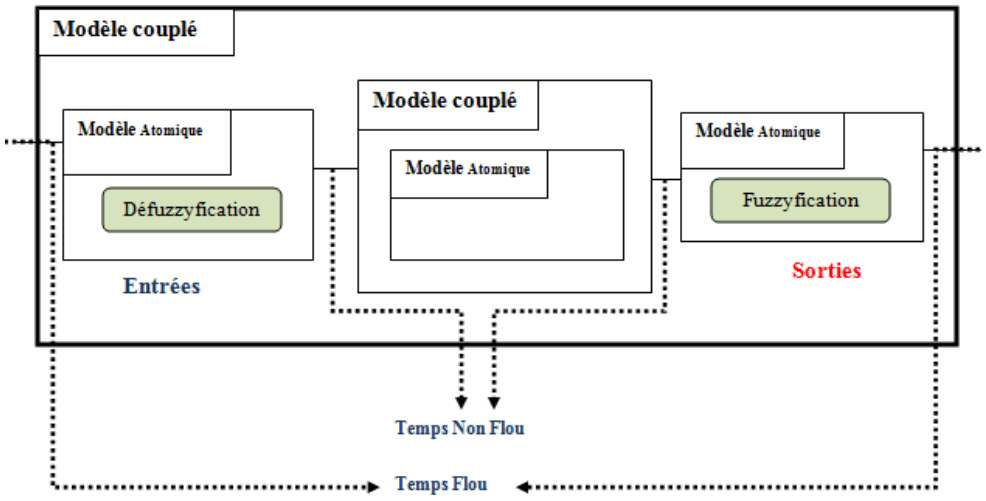

Figure 28 : Architecture et description de l'approche proposée.

Les modules "Defuzzyfication" et "Fuzzyfication" permettent en modifiant le flou sur le temps de résoudre notre problème.

– La fonction Defuzzyfication : transforme en entrée un évènement à temps flou en évènement à valeur floue et à temps fixé. Par exemple, pour un feu qui parcourt 2 Km en 40 ou 60 minutes, au temps t= 50 minutes il aura parcouru entre 1,6 et 2,5 kilomètres ;

– La fonction Fuzzyfication : transforme en sortie un évènement à temps fixé en évènement à temps flou, c'est la fonction inverse.

Vraisemblablement, ces fonctions seront définies comme modèle atomique afin de ne pas s'éloigner de DEVS qui impose un modèle couplé de haut niveau pour respecter l'ordre de la simulation.

Notre modèle (Figure 28) traite le flou au niveau du temps et des valeurs des variables d'états, mais pour le temps, l'utilisation de fonctions de changement de domaine sera nécessaire en vue de la bonne démarche de la simulation. Il sera donc utilisé pour traiter les évènements à dates et valeurs floues.

IV.2.4 Spécification des Modèles proposés

- Un Defuzzyficateur peut être défini comme un modèle atomique tel que les ensembles flous sont des entrées et la valeur defuzzyfiée comme sortie.

$$\mathbf{Defuzzyfication} = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a)$$

$X = \{In\}$ telque x est un nombre flou représenté par quadruplé $[a, b, \alpha, \beta]$

$Y = \{Out\}$

$S = \Re^{+}$

$\delta_{int}(S) = infini$

$\delta_{ext}(q, s, x) =$ Fonction de Defuzzyfication

$\lambda(S) = S.$

$t_a = infini$

Plusieurs fonctions de defuzzyfication permettent de transformer la valeur defuzzifiée de l'ensemble flou x en valeur précise, dans notre cas nous avons utilisé la méthode EEM [2], elle permet d'ajouter un support d'aide à la décision ; en définissant un coefficient de confiance, notre choix est basé sur l'étude effectuée par BISGAM BIGLIA [9] [7].

- Un fuzzificateur peut être défini comme un modèle atomique tel que des nombres précis sont des entrées et des nombre flou comme sortie.

$$\mathbf{fuzzyfication} = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a)$$

$X = \{In\}$

$Y = \{Out\}$

$S = \Re^{+}$

$\delta_{int}(S) = infini$

$\delta_{ext}(q, s, x) =$ Fonction de fuzzyfication

$\lambda(S) = S.$

$t_a = infini$

IV.3 Conclusion

Ce chapitre a présenté notre approche de modélisation imprécise basé sur DEVS et sur la logique floue. Nous nous sommes attachés à définir un formalisme conforme aux exigences de compatibilité définies par DEVS, afin de pouvoir s'appuyer sur les outils qu'il fournit.

L'approche proposée pour contourner le problème d'imprécision des paramètres, elle s'appuie sur des fonctions utilisant les principes de la théorie des ensembles flous et sur la définition de fonctions de changement de domaines. Ces fonctions permettent de représenter des paramètres imprécis, tout en restant cohérent avec DEVS et proche du mode de raisonnement des données des experts du domaine.

Chapitre 5
Implémentation

Chapitre 5 : Implémentation

V.1 Introduction

JDEVS est un outil de modélisation et de simulation développé par une équipe du laboratoire systèmes physiques pour l'environnement de l'université de Corse [95]. Nous avons choisi l'environnement JDEVS comme un outil de simulation pour notre proposition. Parce qu'il offre les avantages suivants :

- ✓ JDEVS est une implémentation du Formalisme DEVS en Java.
- ✓ Permet au modélisateur de spécifier directement les modèles atomiques.
- ✓ Cet outil permet également la définition de modèles DEVS couplés.
- ✓ JDEVS fournit des classes et des objets JAVA.
- ✓ Fournit des outils graphiques qui facilitent la conception et la simulation des modèles DEVS complexes.
- ✓ Un outil robuste et évolutif de simulation.
- ✓ Un outil gratuit.
- ✓ Permet l'intégration et le couplage de méthodologies diverses d'analyses de systèmes par la simulation.

Cet outil a été amélioré afin de répondre aux besoins du problème traité.

V.2 Le logiciel JDEVS

JDEVS est composé de quatre modules indépendants présentés en figure 29 : un moteur de simulation, une interface graphique de modélisation, un module de stockage dans des bibliothèques et les cadres expérimentaux de visualisation et de simulation.

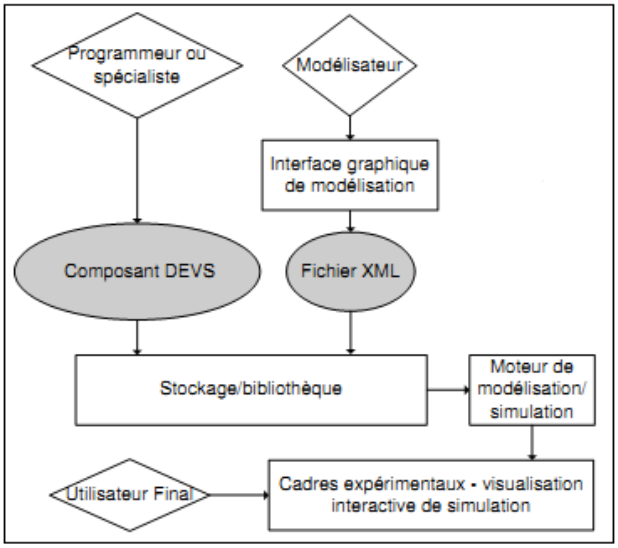

Figure 29 : Cadre de la plateforme JDEVS

Chacun de ces modules reprend l'architecture de classe définie dans le cadriciel ce qui permet à ces modules de fonctionner ensemble et de pouvoir être éventuellement remis à jour indépendamment. Sans la prise en compte de techniques de modélisation spécifique, JDEVS prend pour base la technique de modélisation classique DEVS [95]. La première section est consacrée au moteur de modélisation et de simulation implémentant cette technique.

V.3 Moteur de modélisation et de simulation

Le moteur de modélisation et de simulation est une implémentation de la méthodologie classic-DEVS avec ports. Les modèles DEVS se traduit en instructions JAVA.

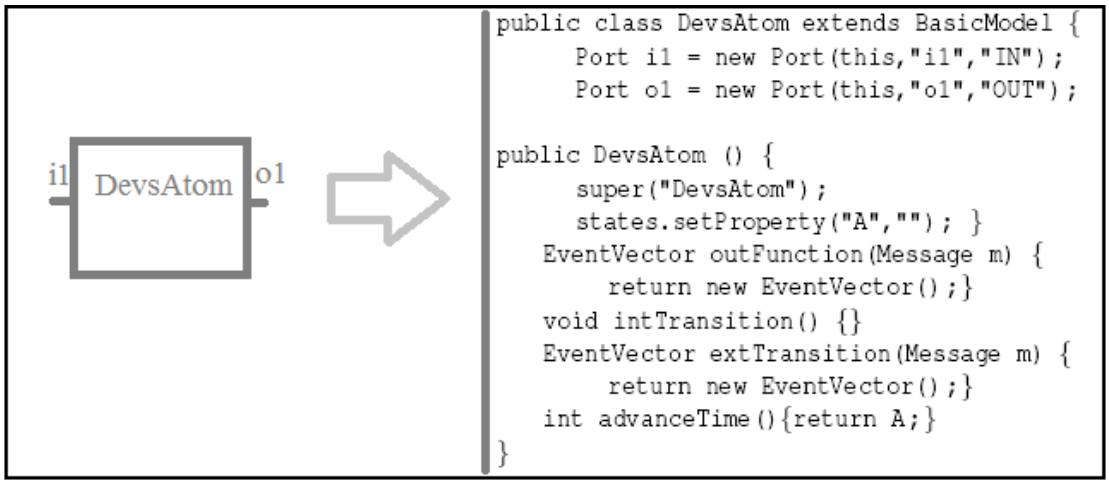

Figure 30 : Code source JAVA pour un composant atomique DEVS standard

Le formalisme DEVS propose des interfaces bien définies pour la description de systèmes. Ces interfaces nous permettent de pouvoir utiliser des modèles programmés dans de nombreux langages orientés objets et d'y accéder ensuite grâce à des appels de méthodes à distance (comme Java RMI). La modélisation de modèles atomiques peut toutefois se faire directement dans l'interface et en utilisant Java. Aussi, il est possible d'ajouter un composant atomique vide, de définir ses interfaces d'entrée/ sortie et de générer le squelette du code du modèle. Ce modèle vide est ensuite enregistré dans la bibliothèque et compilé. La figure 30 présente le code généré automatiquement

Pour $DevsAtom = < X(i1); S(A); Y(o1); \delta int; \delta ext; \lambda; ta >$

Les fonctions de sortie et de transition externe renvoient des vecteurs d'événements qui sont ajoutés à la liste générale des événements. La seule tâche de programmation qui incombe au spécialiste du domaine, est de spécifier la dynamique des modèles atomiques à travers ses quatre méthodes. Une fois le modèle atomique créé, il est stocké dans la bibliothèque pour être utilisé plus tard dans une composition de modèle à l'intérieur d'un modèle couplé. Les compositions de modèles sont réalisées à travers l'interface graphique de modélisation présentée dans la section suivante.

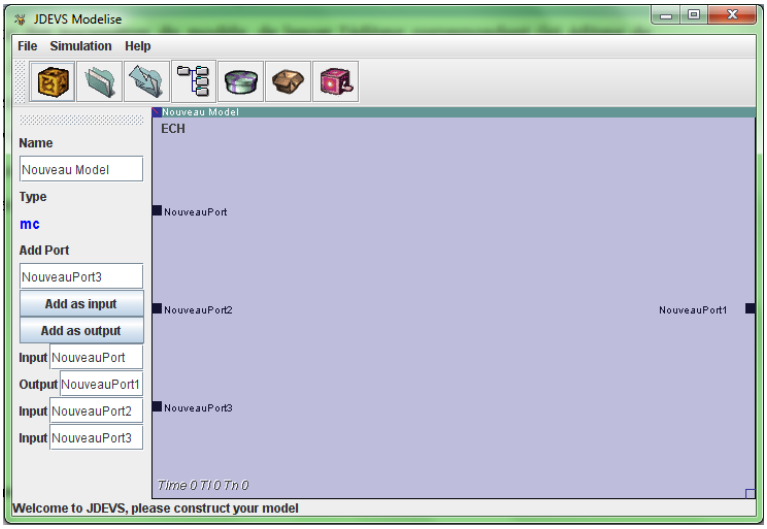

Figure 31 : Modèle couplé et son panneau de propriétés

V.4 Interface graphique de modélisation

L'interface de modélisation en diagrammes permet aux utilisateurs de construire graphiquement leurs modèles. La figure 31 représente une vue de l'outil constitué d'un modèle couplé avec deux ports d'entrée et un port de sortie.

L'implémentation de cette interface est conforme au cadriciel, ainsi chaque modèle est présenté dans un espace de travail (partie centrale de la figure 31) et est associé à un panneau de propriétés (situé à droite dans la figure 31). De plus, l'interface dispose d'une barre de menu simple permettant d'ajouter des modèles vides (atomique, couplés ou d'autres techniques de modélisation), d'ouvrir une bibliothèque. Les modèles atomiques et couplés ont des panneaux de propriétés différents, la figure 31 présente celui d'un modèle couplé permettant d'ajouter des ports et de changer son nom.

Le panneau de modèles atomiques figure 32 permet d'inspecter et de modifier la valeur des paramètres du modèle, de lancer l'éditeur correspondant (ici éditeur de code Java) et de compiler le modèle afin de le recharger dans l'interface graphique. Les composants peuvent être déplacés et redimensionnés dans ce canevas.

Un port est représenté par une poignée noire dans un composant : les couplages s'effectuent en cliquant dans les ports source puis destination.

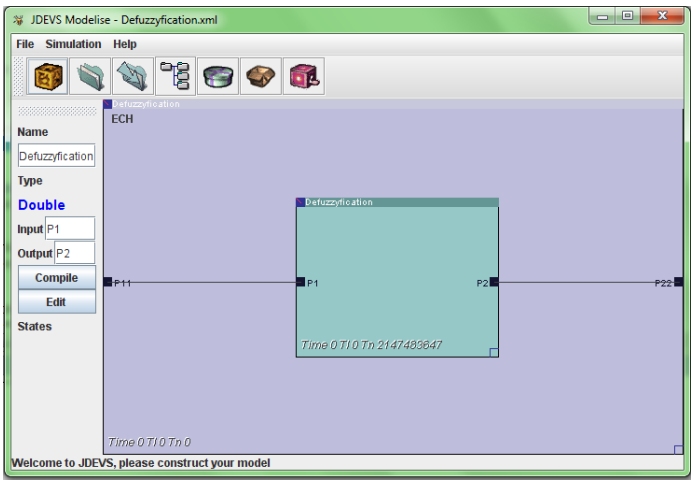

Figure 32 : Modèle atomique et son panneau de propriétés

Pour ajouter des modèles existant il suffit de les glisser depuis le composant bibliothèque vers l'espace de travail. La bibliothèque est l'élément essentiel permettant le stockage de modèles ; il est brièvement présenté dans la section suivante.

V.5 Cadres expérimentaux

Les cadres expérimentaux sont les interfaces utilisateur de simulation des modèles stockés dans la bibliothèque. Les expérimentations sur des modèles en diagramme peuvent être effectuées directement depuis l'interface graphique de modélisation. Il est ainsi possible de vérifier le comportement d'un modèle en cours

de création et donc d'ajuster ses paramètres ou corriger les erreurs de conception sans explicitement utiliser le module cadre expérimental.

L'application permettant d'exécuter les expérimentations se présente avec la même interface que l'interface de modélisation, chaque modèle possède un panneau de propriétés. Le panneau de propriété affiché est le panneau du modèle actif (i.e. le composant sélectionné).

La figure 33 présente un modèle en cours de simulation dans l'interface. Le lancement de la simulation depuis l'interface graphique permet de charger une liste d'événements d'entrée et de spécifier les paramètres de l'expérimentation ; ces actions sont effectuées depuis le panneau de simulation (figure 33, en haut à gauche). L'option "track simulation" permet de réaliser la simulation en insérant un temps d'attente entre l'envoi de deux événements d'entrée pour vérifier le comportement du modèle.

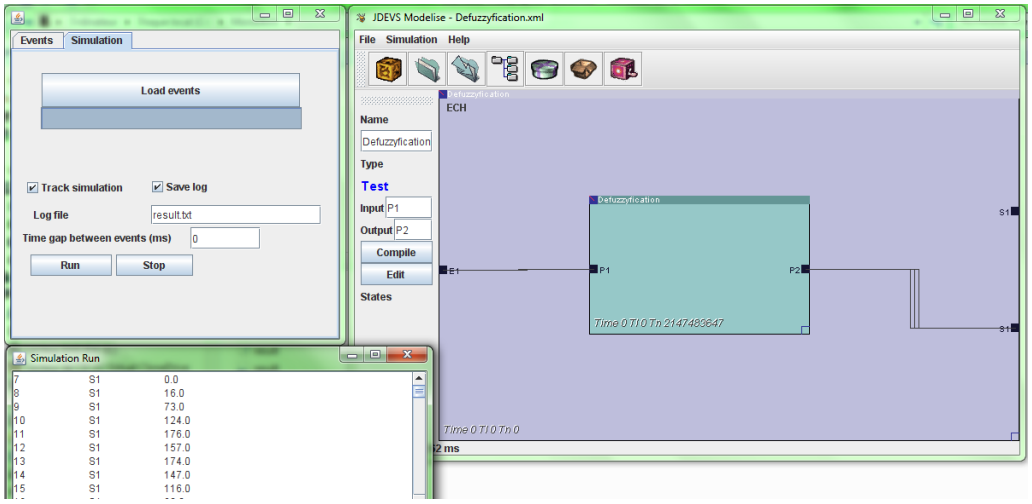

Figure 33 : Interface de modélisation de l'environnement JDEVS

La liste des événements à traiter est affichée à l'intérieur de chaque modèle couplé et évolue au cours de la simulation. Il est possible de changer les paramètres de chaque modèle atomique durant la simulation en l'activant puis en modifiant les valeurs à l'intérieur de son panneau de propriétés (figure 32, à droite). Le résultat de simulation est une liste d'événements de sortie qui peut être soit affichée (figure 33, en bas à gauche), soit sauvegardé dans un fichier.

V.7 Modélisation d'un système photovoltaïque

Les systèmes à énergie solaire PhotoVoltaïque, présentés en figure 34, sont aussi connus comme systèmes "PV". Ils permettent de convertir la radiation solaire incidente en puissance électrique dont on distingue deux types d'installations [13]:

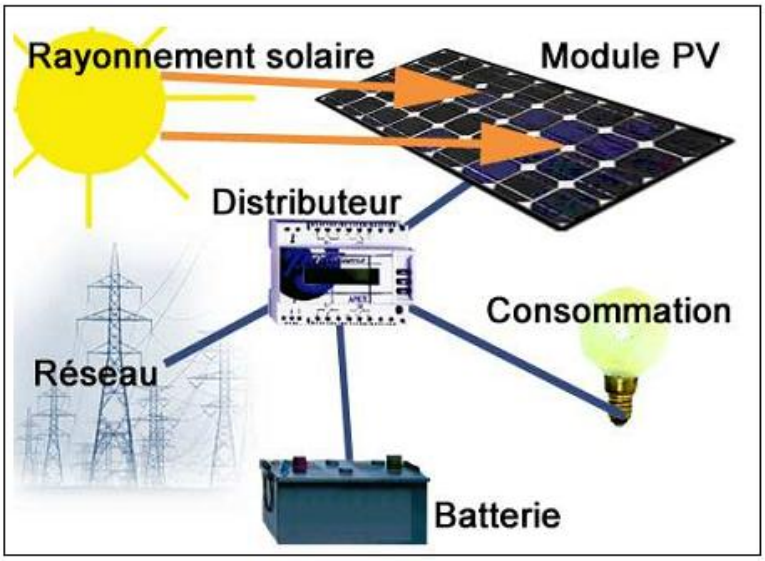

Figure 34 : Système photovoltaïque

1- Les systèmes "PV" autonomes sont composés d'une batterie optionnelle de panneaux de production d'électricité photovoltaïque et d'un répartiteur [47]. Leur mode de fonctionnement est simple : le rayonnement solaire est converti en électricité par le panneau photovoltaïque, l'énergie produite est ensuite partagée par le distributeur entre la recharge de la batterie et les postes de consommation. Si le soleil ne brille pas suffisamment, la batterie est utilisée pour fournir la puissance manquante. Enfin si la batterie est complètement chargée, l'énergie non consommée est perdue.

2- Les systèmes "PV" connectés utilisent le réseau électrique comme batterie, ils sont uniquement composés d'un répartiteur et de panneaux photovoltaïques. Dans un système connecté, l'énergie produite peut être renvoyée sur le réseau tandis que pendant les périodes de sous production, le réseau complète l'apport énergétique, assurant du même coup un fonctionnement sans interruption de service.

Le but final de ce modèle est d'effectuer des expérimentations virtuelles sur ces systèmes et aider à leur dimensionnement en fonction des sites d'implantation. Il a été développé par une équipe énergétique de laboratoire qui traite cette problématique [63]. Cette équipe, en tant que spécialiste de ce domaine, a défini le protocole permettant de préciser le cadre expérimental et donc les paramètres et variables d'entrée et de sortie du modèle. La figure 34 présente le modèle implémenté dans JDEVS.

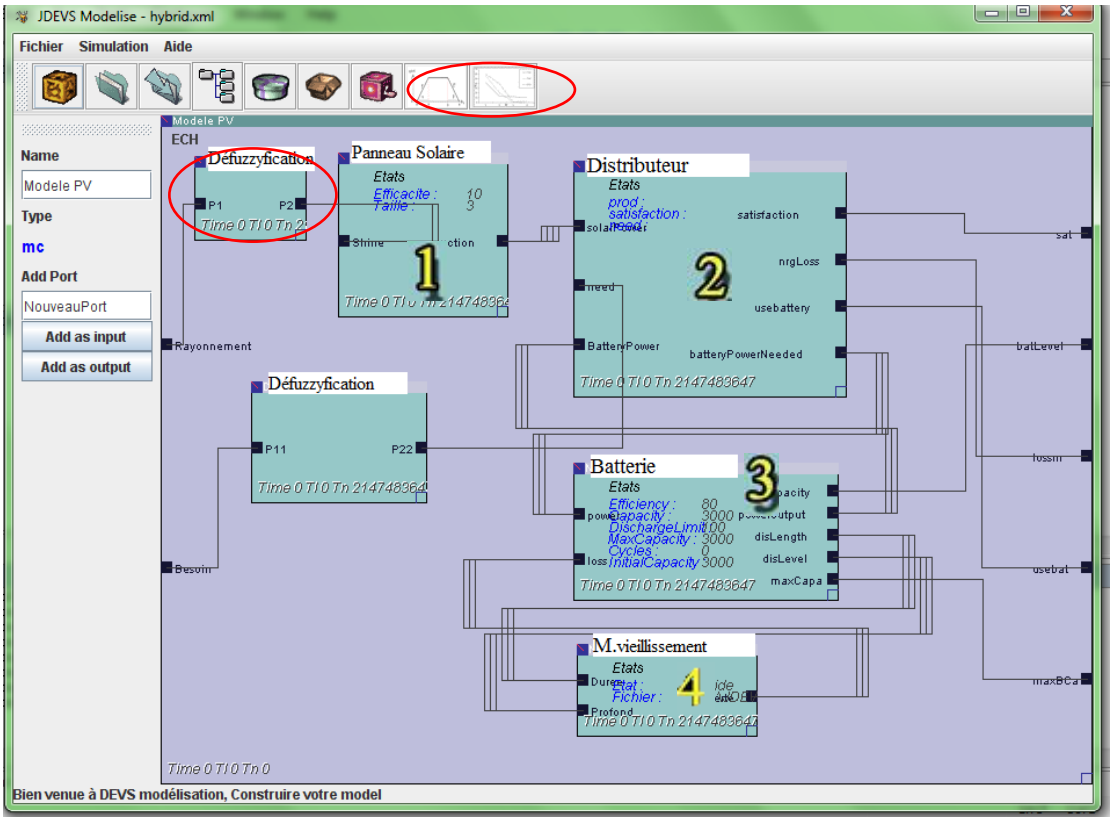

Figure 35 : Modèle du système PV dans JDEVS

Le modèle général a deux *entrées* :

– **ensol** : l'ensoleillement en Watt/Heure/mètre carré.

– **besoin** : le besoin en consommation en Watt/heure

Il est possible de suivre plusieurs variables de *sorties* du modèle suivant le cadre expérimental dans lequel l'utilisateur se place. Cependant la majorité des études dans le domaine énergétique (par exemple [41] ou [63]) s'intéresse à un nombre restreint de ces variables. Nous avons pris les mêmes ports de sortie que celle définis par [67] pour permettre de suivre l'évolution du modèle à l'occurrence :

– **Satisfaction** : le pourcentage de besoin satisfait par le système par rapport au besoin réel de consommation.

– **Energ_Perdu** : la quantité d'énergie solaire perdue en Watt/heure.

– **Batt_Util** : le pourcentage d'utilisation d'énergie provenant de la batterie par rapport à l'énergie du panneau photovoltaïque.

– **Niveau_Batt** : le niveau de la batterie en Watt (pour une tension constante de 12 volts).

– **MaxBattCapa** : la capacité maximum de la batterie en Watt (pour une tension constante de 12 volts).

Une des contraintes posées par les spécialistes du domaine et les utilisateurs finaux est la modalité de mesure des données d'entrée et de sortie. En effet, la grande majorité des stations de mesures, comme les modèles existants, fonctionnent en temps discret.

Donc le modèle doit être capable de recevoir en entrée des événements arrivant à intervalles de temps réguliers (ici l'heure). Les événements de sortie sur les ports Energ_Perdu et Niveau_Batt doivent aussi être générés à intervalles de temps régulier. Les autres ports de sortie sont plus conformes à un mode de simulation à événements discrets et ne sont actifs que pour informer d'un changement de valeur. Les sous-modèles peuvent ainsi fonctionner sur ce mode mixte temps/événements discrets. Ils sont présentés en détail dans la suite de cette section.

V .7.1 Le panneau photovoltaïque

Le modèle atomique du panneau photovoltaïque (Figure 35(1)) représente le système de production d'énergie, il possède deux ports :

– Un port d'entrée, **Ensol** : l'ensoleillement en Watt/Heure/mètre carré.

– Un port de sortie, **Production** : la production électrique en watt/heure.

Il y à deux paramètres dans ce modèle pour permettre à l'utilisateur final d'effectuer plusieurs expérimentations :

– **Taille** : la taille en mètres carrés.

– **Rendement** : le rendement en % de conversion énergétique (solaire vers électrique).

Le comportement des cellules photovoltaïques composant un panneau est bien connu, il peut donc être modélisé comme un modèle atomique DEVS classique, en utilisant les données du fabricant. L'énergie de sortie calculée par le modèle est envoyée au distributeur.

$$Panneau = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a)$$

$X = \{ Ensol \}$

$Y = \{ Production \}$

$S = \{ (Taille, Efficacité) \}$

$\delta_{int}(S) = infini$

$\delta_{ext}(q, s, x) = (Ensol * Taille * Efficacité)/100$

$\lambda(S) = S$

$t_a = infini.$

V.7.2 Le distributeur

Le modèle du distributeur (Figure 35(2)) est aussi un modèle atomique DEVS classique, ce modèle est chargé d'assurer la répartition des charges entre la batterie, le panneau solaire et les postes de consommation. Ce sous-modèle est dans une position centrale car il est connecté en entrée, à la batterie, au panneau photovoltaïque et aux postes de consommation par les ports :

– **Prod** : la production en Watt/heure du système source d'énergie,

– **Besoin** : la demande électrique en watt/heure.

– **EnergBatt** : l'énergie disponible dans la batterie, en Watt.

En sortie, ce modèle est connecté à la batterie pour communiquer le besoin en énergie. Les autres ports de sortie sont utiles à la génération de données pour l'analyse des résultats. Le sous-modèle est globalement composé de quatre ports de sortie :

– **Satisfaction** : le pourcentage de besoin satisfait par le système par rapport au besoin réel de consommation,

– **Energ_Perdu** : la quantité d'énergie solaire perdue en Watt/heure,

– **Batt_Util** : le pourcentage d'utilisation d'énergie provenant de la batterie par rapport à l'énergie du panneau photovoltaïque,

– **Dech_batt** : le besoin en énergie (décharge) ou bien la quantité d'énergie disponible pour la recharge de la batterie (si la batterie n'est pas pleine), en Watt.

Dés que le niveau change dans le modèle de la batterie, un message est envoyé au distributeur pour l'informer de la nouvelle valeur ; ainsi le distributeur connaît constamment le niveau d'énergie dans la batterie. Lorsqu'il y a un besoin en énergie et que l'énergie disponible est suffisante, un message est envoyé par le port « Dech_batt », donnant la quantité d'énergie à enlever à la batterie. Lorsqu'il y a un surplus d'énergie et que le niveau de la batterie n'est pas au maximum, un message est envoyé pour la recharge de la batterie.

Distributeur = **(X, Y, S, δ_{int}, δ_{ext}, λ)**

X={ Prod ,Besoin, EnergBatt }

Y={ (Satisfaction , Energ_Perdu, Batt_Util , Dech_batt) }

S= {Prod, Satisfaction, Besoin}

δ_{int}(S)=infini

δ_{ext}(q,s,x) =

```
//si le besoin est supérieur à la demande on pompe la batterie sinon on la recharge.
Dech_batt = prod - besoin
  si (besoin > 0) et (prod > 0)
// on utilise la batterie
 si (energBatt > 0)   alors   satisfaction =((prod + energBatt)/besoin)*100
                              Batt_util= energBatt /(prod + energBatt)*100
                              Energ_perdu=0
//   on recharge
Sinon si ((energBatt == 0.0) et (prod > besoin))
                               satisfaction=100
                               Batt_util =0.0
                               Energ_perdu =0.0
// completement dechargée
 Sinon si ((energBatt == 0.0) et (prod < Need)) {
                                 satisfaction=((prod + energBatt)/Need)*100))
                                 Batt_util =0.0
                                 Energ_perdu =0.0
// utilisation soleil, perte d'energie
Sinon
                 Batt_util =0
                 Satisfaction=100
                 Energ_perdu = energBatt *-1
```

λ(S) = S

Le comportement de la batterie évolue durant la simulation à cause de son vieillissement. Ce vieillissement est souvent propre à un type ou même une marque de batterie. Il est alors difficile de le modéliser avec des techniques conventionnelles.

V.7.3 La batterie et le modèle de vieillissement

Le comportement physique d'une batterie est très complexe et fait intervenir de nombreuses réactions chimiques qui nécessitent beaucoup de données pour pouvoir être simulées de manière conventionnelle. Nous avons donc adapté un modèle existant, décrit dans [47], afin d'obtenir des résultats satisfaisants. Dans ce modèle, la batterie est considérée comme un réservoir d'électricité avec une capacité et une efficacité de charge en paramètres et peut donc être modélisée comme un modèle DEVS classique. Chaque batterie est attachée un modèle de vieillissement qui modifie sa capacité de charge en cours de simulation. Le modèle de vieillissement est ici un modèle DEVS qui implémente un réseau de neurones [28] utilisant la technique de rétro-propagation du gradient [42], [18].

Selon [44] "Un réseau de neurones est un processeur massivement distribué en parallèle qui a une propension naturelle pour stocker de la connaissance empirique (experiential knowledge selon l'auteur) et la rendre disponible à l'usage. Il ressemble au cerveau sur deux aspects : La connaissance est acquise par le réseau au travers d'un processus d'apprentissage. Les connexions entre les neurones, connues sous le nom de poids synaptiques servent à stocker la connaissance."

Un réseau de neurones peut être considéré comme un modèle mathématique de traitement réparti, composé de plusieurs éléments de calcul non linéaire (neurones), opérant en parallèle et connectés entre eux par des poids [44]. Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Le lecteur intéressé trouvera dans [42] ou [32] une description détaillée de cette technique.

Ce modèle de vieillissement permet de calculer la perte de capacité à partir de la profondeur et de la durée de la décharge maximale. La batterie est considérée comme étant en décharge lorsque la puissance disponible est inférieure de moitié à la capacité maximale.

Le modèle de batterie (Figure 35(3)) est ici considéré comme un réservoir de puissance mesurée en Watt. Il possède deux ports d'entrée :

– **puissance** : la puissance qui doit être ajoutée ou retranchée à la quantité présente dans la batterie,

– **Perte** : la perte de capacité de stockage due au vieillissement.

De nombreuses variables de sortie sont nécessaires au calcul de vieillissement et à l'analyse du comportement de la batterie ; le modèle est composé de cinq ports de sortie :

– **Capacité** : la puissance disponible dans la batterie, en Watt, remise à jour dès que la capacité change. Ce port est destiné à être connecté à une sortie du modèle général pour permettre l'analyse de résultats.

– **Port_capa** : Identique au port précédent mais destiné au couplage avec les autres sous-modèles,

– **Duré_Decharge** : durée de la décharge en cours, exprimée en heures. Ce port émet toutes les heures dès que la puissance disponible est inférieure à la moitié de la capacité maximum,

– **Niveau_Decharge** : profondeur de la décharge en cours exprimé en pourcentage de la capacité initiale,

– **Max_Capacite**: capacité maximum de la batterie.

Le modèle de batterie dispose aussi de six paramètres dépendant des expérimentations et qui peuvent être ajustés avant ou au cours de la simulation :

– **Rendement** : l'efficacité de charge et de décharge, exprimée en pourcentage.

– **Capacité** : la puissance courante disponible, exprimée en Watts.

– **DischargeLimit** : la limite de décharge ou seuil au delà duquel la batterie ne peut plus de se décharger, exprimée en Watt.

– **Max_Capacité** : capacité maximum de la batterie.

– **Cycles** : nombre de cycles charge/décharge.

– **Capacit_Initial** : capacité initiale maximum de la batterie.

Batterie **= (X, Y, S, δ_{int}, δ_{ext}, λ)**

X={ puissance ,Perte }

Y={ capacité , port_capa, duré_décharge

,Niveau_decharge,Max_capacité, Rendement,décharge_limit,

cycles, capacité_initial }

S= { (capacité , port_capa, duré_décharge

,Niveau_decharge,Max_capacité) }

δ_{int}(S)=infini

δ_{ext}(q,s,x)

```
λ(S)= Si (capacité + puissance >  Max_Capacité
          capacité = Max_Capacité  - (capacité + puissance)
// charger
 Si ((capacité + puissance) < decher_limt  ){
          capacité = capacité - decher_limt }

Si (puissance > 0)  Capacité=capacité *(rendement/100);
 // décharger
Capacité = Capacité -( capacité *(rendement/100)
```

λ(S)=S.

Le modèle utilisé pour calculer le vieillissement de la batterie est une implémentation du modèle décrit dans [47]. Il utilise la technique des réseaux de neurones à rétro-propagation du gradient pour estimer la perte de capacité d'une batterie en fonction de la profondeur et de la durée d'une décharge.

vieillissement = (X, Y, S, δ_{int}, δ_{ext}, λ)

X={ Durée ,profondeur }

Y={ perte }

S= {puissance}

δ_{int}(S)=infini

δ_{ext}(q,s,x)= Foction de calculi de perte (utilisant la technique des réseaux de neurones)

λ(S)=S

Un des avantages majeurs de l'utilisation du modèle décrit dans [47] est que l'on dispose des données pour entraîner et valider le réseau de neurones. Ces données sont collectées au Florida Solar Energy Center [35] qui dispose d'un grand centre d'étude sur les batteries. Cependant le modèle pose un problème lors de son implémentation dans un environnement de simulation à événements discrets car il est dirigé par une horloge. En effet, à cause des données disponibles nous ne pouvons

calculer la perte de capacité que toutes les heures, en fonction de la durée de la décharge en cours. Le modèle de vieillissement n'est toutefois activé que pendant les périodes de décharges.

```
PVCouplée=(XI, YI, D, {Md | dєD}, EIC, EOC, IC)
XI={Ensol,Besoin}
YI={satisfaction,capacite, Ener_Perdu,Batt_util, Max_capacite}
D={Defuzzyfication, panneau, Distributeur }
EIC={( PVCouplée.Ensol, Defuzzyfication.Ensol),
      ( PVCouplée.besoin, Defuzzyfication.besoin)
     }
EOC={(distributeur.satisfaction, PVCouplée.satisfaction ),
      (Batterie.capacité, PVCouplée. capacité),
      (distributeur.Ener_perdu, PVCouplée. Ener_perdu ),
      (Batterie.capacité, PVCouplée.capacité),
      (Batterie.Batt_util, PVCouplée.Batt_util),
      (Batterie.Max_capacité, PVCouplée.Max_capacité),
      (Batterie.Max_capacité, PVCouplée.Max_capacité),

     }
 IC={(panneau.production, Distributeur.production),
      (Distributeur.Besoin, Batterie.puissance),
      (Batterie.Port_capacité, Distributeur.Energ_Batt),
      (Batterie.Duré_decharge, vieillissement.durée),
      (Batterie.Niveau_decharge, vieillissement.profondeur),
      (vieillissement.Perte, Batterie.Perte)
     }
```

L'interface graphique de JDEVS permet de créer simplement et graphiquement la structure de tous ces modèles mais le comportement doit être spécifié en code Java. Les expérimentations, réalisées à partir d'une liste d'événements d'entrée et les différentes configurations, sont obtenues en modifiant les paramètres des modèles grâce à leurs panneaux de propriétés. Quelques résultats de simulations sont présentés dans la section suivante.

V.7.4 Résultats de simulation

Dans le cadre d'une simulation standard de ce modèle, les données d'entrée sont fournies par l'utilisateur, dans notre cas on a essayé d'adapter les même entrées utilisées par [67] et les événements de sorties sont directement stockés dans des fichiers. Toutefois, grâce à la méthodologie utilisée, il est très simple d'intégrer ce

modèle à l'intérieur d'un autre plus complexe comprenant, par exemple, un modèle de consommation et un modèle de calcul de pollution en fonction de l'énergie économisée.

Nous avons effectué douze tests sur une période de trois ans et une combinaison de quatre tailles de panneaux différents et de trois capacités de batterie. Les profils de consommation ont été obtenus auprès du FSRC [35], et correspondent à la consommation d'une maison avec :

- Cinq ampoules de type basse consommation utilisées 8h/jour.
- Une télévision utilisée 5h/jour.
- Un petit réfrigérateur (consommation totale journalière sous 12 Volts : 1600 Watts).

La figure 36 présente une expérimentation d'une simulation avec une taille de panneau solaire de 6 mètres carrés et une puissance de 6 KW de batterie. L'effet du modèle de vieillissement est ici clairement visible ; la capacité maximum de la batterie est fortement dégradée chaque hiver.

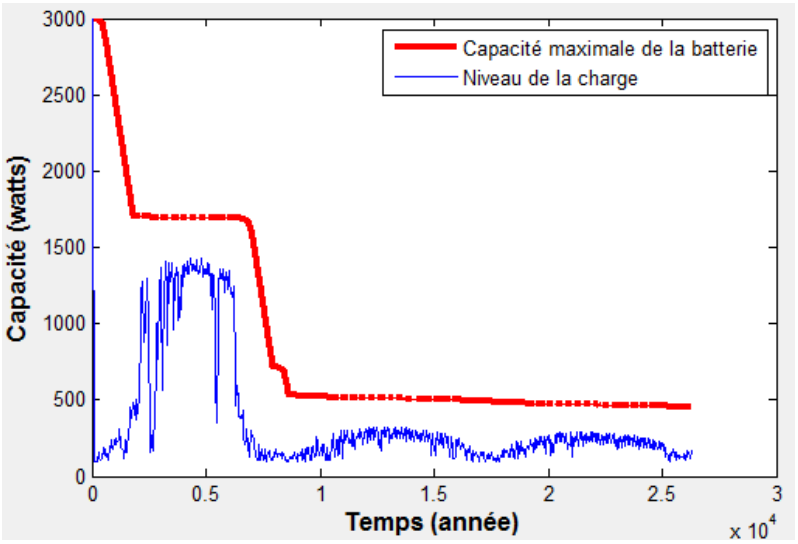

Figure 36 : Simulation du vieillissement

La figure 37 montre que la perte de capacité de batterie après trois ans est dépendante de la taille du panneau photovoltaïque. Le calcul de coût d'un système photovoltaïque déconnecté est fortement dépendant de l'usure de la batterie car on considère qu'une batterie qui a perdu 40% de sa capacité doit être changée.

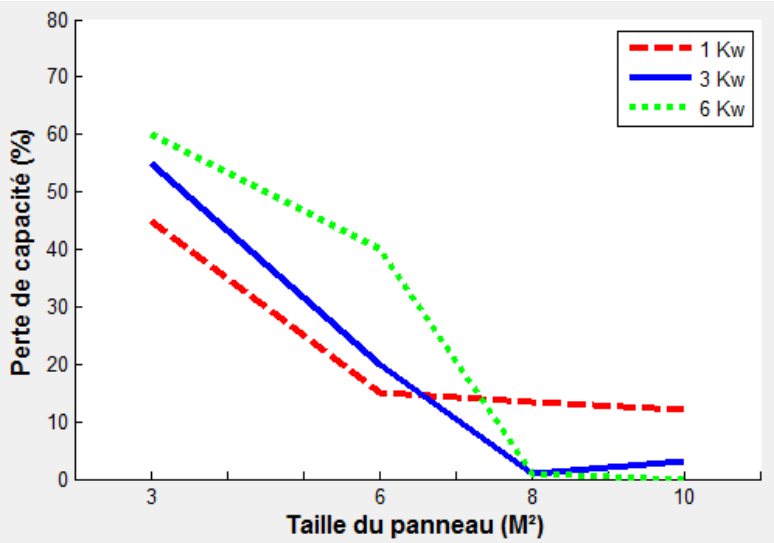

Figure 37 : Perte de capacité de batterie après trois ans.

La figure 38 montre le prix cumulé d'un système connecté au réseau après 15 ans. Le coût d'un système déconnecté est calculé en ajoutant le coût des batteries de rechange au coût initial de l'installation. Le réseau électrique peut être concurrencé si le système PV est bien dimensionné, mais cela se fait au détriment de la qualité de service (faible satisfaction).

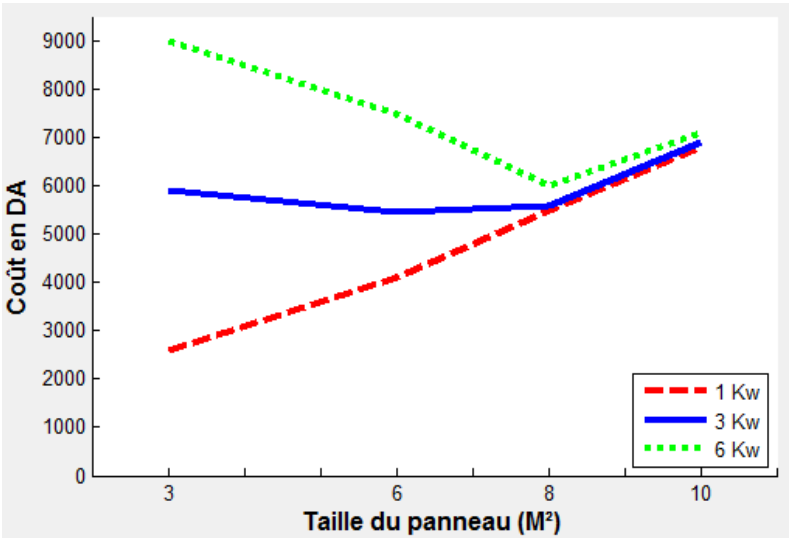

Figure 38 : Prix d'un système PV

La figure 38 présente la satisfaction moyenne. La satisfaction est un ratio entre la puissance demandée et la puissance fournie qui atteint 100% lorsque l'utilisateur ne manque jamais d'électricité. Un système PV doit être bien dimensionné pour offrir une satisfaction suffisante à de moindres coûts.

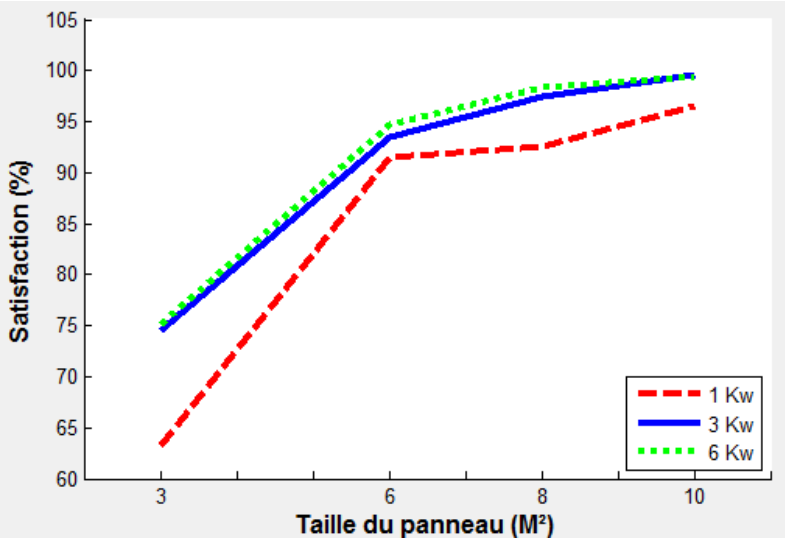

Figure 39 : Satisfaction moyenne sur 3 ans

La figure 40 présente le coût d'une installation par % de satisfaction. Le réseau électrique est ici bien meilleur que les systèmes PV.

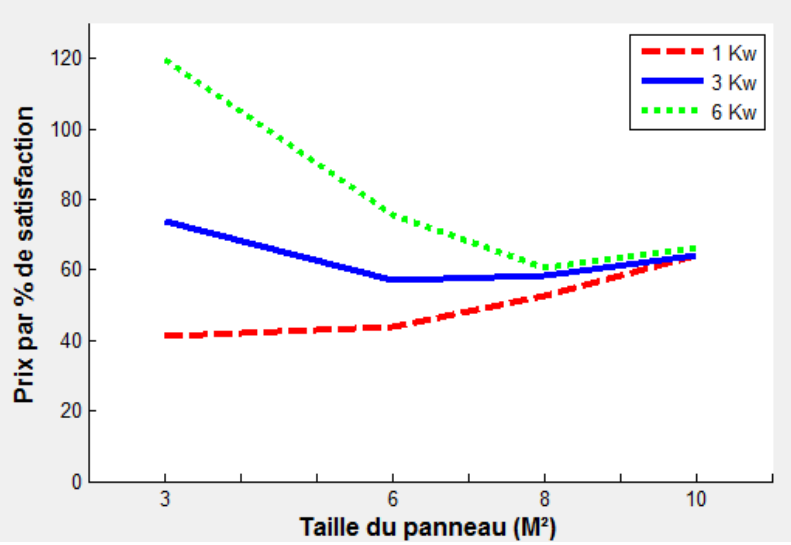

Figure 40 : Prix par % de satisfaction pour un système PV

Bien que ce modèle produise des résultats exploitables, plus de données sont nécessaires pour entraîner le réseau de neurones de la batterie de manière plus satisfaisante. Il serait aussi nécessaire de simuler de manière plus adéquate les cellules photovoltaïques en prenant en compte la chaleur qui est un paramètre important dans le rendement et qui n'est pas pris en compte dans ce modèle.

V.8 Conclusion

Dans ce chapitre nous avons illustré l'implémentation de notre approche à l'aide de la plateforme JDEVS, qui est un environnement de modélisation et de simulation développé par une équipe du laboratoire systèmes physiques pour l'environnement de l'université de Corse.

L'exemple traité dans notre étude est la modélisation et la simulation des systèmes à énergie solaire photovoltaïque. Ils permettent de convertir la radiation solaire incidente en puissance électrique. Le but de ce modèle est d'effectuer des expérimentations virtuelles sur ces systèmes et aider à leur dimensionnement (les panneaux PV). Nous avons présenté quelques résultats obtenus par cette approche.

Conclusion et Perspectives

Conclusion et perspectives

Ce mémoire a présenté notre formalisme de modélisation imprécise basé sur DEVS et sur la logique floue. Nous nous sommes attachés à définir un formalisme conforme aux exigences de compatibilité définies par DEVS, afin de pouvoir s'appuyer sur les outils qu'il fournit. Nous avons implémenté les fonctions "Defuzzyfication" et "Fuzzyfication" et les intégrer dans le simulateur JDEVS.

L'approche proposée pour objectif de contourner le problème d'imprécision des paramètres, nous paraît tout de même intéressant. Elle s'appuie sur des fonctions utilisant les principes de la théorie des ensembles flous et la logique floue et sur la définition de fonctions de changement de domaines. Ces fonctions permettent de représenter des paramètres flous, tout en restant cohérent avec DEVS et proche du mode de raisonnement des données des experts du domaine.

A terme, cette méthodologie va être employée pour permettre la modélisation et la simulation de systèmes complexes à paramètres (valeur et temps) flous ou vagues. En ce qui concerne les discussions avec différents spécialistes de domaines (base de l'étude), la méthode courante de représentation de paramètres incertains, via des fonctions floues (triangles ou trapèzes), et l'utilisation de descriptions par variables linguistiques seront conservées car elles nous paraissent plus proches et plus adéquates pour modéliser leurs connaissances et expériences.

Ceci dit, le travail est loin d'être fini, de nombreuses améliorations peuvent être considérées sans pour autant prétendre l'exhaustivité, l'ajout du facteur imparfait des paramètres structurels (incertain, incomplet ou imprécis) et comportemental, cette dynamique peut être vu comme modèle générique du formalisme DEVS classique.

Sommaire

Liste des Figures

Bibliographie

[1] A. Anglani, P. Caricato, A. Grieco, F. Nucci, A. Matta, G. Semeraro, and T. Tolio. Evaluation of capacity expansion by means of fuzzy-devs. citeseer.ist.psu.edu/499458.html, May 2000. 14th European Simulation MultiConfer-ence. Ghent, Belgium.

[2] A. Anglani, A. Grieco, F. Nucci, G. Semeraro, and T. Tolio. A new algorithm torank temporal fuzzy sets in fuzzy discrete event simulation. Fuzzy Systems, 2000.FUZZ IEEE 2000. The Ninth IEEE International Conference on, 2 :923–928, 2000.

[3] A. Aiello. Environnement Orienté Objet de Modélisation et de Simulation à Evènements Discrets de Systèmes Complexes. PhD thesis, Université de Corse, 1997.

[4] F. Barros. Dynamic structure discrete event system specification : a new formalism for dynamic structure modelling and simulation. In Proceedings of Winter Simulation Conference 1995, 1995.

[5] P.-A. Bisgambiglia, L. Capocchi, E. de Gentili, and P.A. Bisgambiglia. Manipulation of incomplete or fuzzy data for DEVS-based systems. In SCS, editor, Pro-ceedings of the International Modeling and Simulation Multiconference (IMSM) -Conceptual Modeling Simulation (CMS), pages 87–92, 2 2007.

[6] P.-A. Bisgambiglia, E. de Gentili, P.A. Bisgambiglia, and J.F. Santucci. Modélisa-tion floue basée sur le formalisme DEVS et sur la méthode des fronts. Cépaduèseditions edition, 11 2007.

[7] P.-A. Bisgambiglia, E. de Gentili, P.A. Bisgambiglia, and J.F. Santucci. Discreteevents system simulation-based defuzzyfication method. In Proceedings of The 14thIEEE Mediterranean Electrotechnical Conference (MELECON), pages 132–138,05 2008.

[8] P.-A. Bisgambiglia, E. de Gentili, P.A. Bisgambiglia, and J.F. Santucci. Fuzzy modeling for discrete events systems. In Proceedings of The 14th IEEE Mediterranean Electrotechnical Conference (MELECON), pages 151–157, 05 2008.

[9] P.-A. Bisgambiglia, E. de Gentili, P.A. Bisgambiglia, and J.F. Santucci. Fuzzy modlation for discrete events systems. In Proceedings of the 2008 IEEE World Congresson Computational Intelligence (WCCI 2008) - IEEE International Conference onFuzzy Systems (FUZZ-IEEE), pages 688–694, 06 2008.

[10] P.-A. Bisgambiglia, E. de Gentili, J.B. Filippi, and P.A. Bisgambiglia. DEVS-Flou :a discrete events and fuzzy logic-based new method of modelling. SIMULATIONSERIES, VOL 38, PART 4, pages 83–90, 7 2006.

[11] P. Bosc, D. Dubois, and H. Prade. An introduction to fuzzy sets and possibilitytheory based approaches to the treatment of uncertainty and imprecision in database management systems. 2nd Workshop on Uncertainty Management in Information Systems : From needs to solutions, 1993.

[12] F. Bernardi. Conception de bibliothèques hiérarchisées de modèles réutilisables selon une approche orientée objet. PhD thesis, Université de Corse, 2002.

[13] L.V. Bertalanffy. Théories générale des Systèmes. 1968.

[14] P.-A. Bisgambiglia, J.B. Filippi, and E. de Gentili. A fuzzy approach of modeling evolutionary interfaces systems. In IEEE, editor, Proceedings of the ISEIM 2006, Corte (France), pages 98–103, 6 2006.

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

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

[17] L. Capocchi. Simulation concurrente de fautes comportementales pour des systèmes à évènements discrets : Application aux circuits digitaux. PhD thesis, Université de Corse, 2005.

[18] Carling, A. (1992). Introducing neural networks. Sigma Press, New York.

[19] M. Delhom. Modélisation et Simulation Orientées Objet, Contribution à l'Etude du Comportement Hydrologique d'un bassin Versant. PhD thesis, Université de Corse,1997.

[20] A.P. Dempster. Upper and lower probabilities induced by a multivalued mapping.Annals of Mathematical Statistics, (38) :325–339, 1967.

[21] DEVS-Group (2003). Groupe de standardisation de DEVS. http ://www.sce.carleton.ca/faculty/wainer/standard/.

[22] D. Dubois, H. Fargier, and J. Fortin. A generalized vertex method for computing with fuzzy intervals. In IEEE, editor, Proc. of the International Conference on Fuzzy Systems, pages 541–546. IEEE, July 2004. Budapest, Hungary.

[23] E. de Gentili. Contribution au développement d'une méthode de validation d'architecture logicielle de télécommunication à l'aide de la modélisation et de la simulation à évènements discrets. PhD thesis, Université de Corse, 2002.

[24] D. Dubois and H. Prade. Fuzzy sets and systems. 1980.

[25] D. Dubois and H. Prade. Representation and combination of uncertainty with belief functions and possibility measures. Comput. Intell, (4) :244–264, 1988.

[26] Dubois, D. et Prade, H. (1993). Fuzzy set in approximate reasoning.

[27] Dukelow, J. (1994). Verication and validation of neural networks. Dans Actes de la conférence Neural Network Workshop for the Hanford Community, pages 76–80. Ri-chland, Washington, USA.

[28] Filippi, J., Bisgambiglia, P., et Delhom, M. (2001). Neuro-DEVS, an hybrid methodology to describe complex systems. Dans Actes de SCS ESS 2001 conference on simulation in industry, volume 1, pages 647–652. Marseille, France

[29] J.-B. Filippi, F. Bernardi, and M. Delhom. The JDEVS environmental modeling and simulation environment. IEMSS, Integrated Assessment and Decision Support,Lugano Suisse, pages 283–288, 2002.

[30] Jérôme Fortin, Didier Dubois, and Hélène Fargier. Le calcul des intervalles flous par la méthode des fronts. Fuzzy interval analysis using the method of profiles.Cépaduès editions edition, 2004.

[31] J.B. Filippi. Une architecture logicielle pour la multi-modélisation et la simulation à évènement discrets de systèmes naturels complexes. PhD thesis, Université de Corse, 2003.

[32] Filippi, J.-B. (2000). Analyse, conception et programmation d'un logiciel de simulation de bassins versants. Rapport technique, University of Corsica. TCUDC5092000.

[34] P. Fishwick. Simulation Model Design and Execution : Building Digital Worlds,1995.

[35] FSERC (2003). Florida solar energy center. http ://www.fsec.ucf.edu/.

[36] P. Fishwich and B.P. Zeigler. A multi-model methodology for qualitative model engineering. ACM transaction on Modeling and Simulation, vol. 2, (1) :52–81,1992.

[37] S. Garredu, P.-A. Bisgambiglia, E. Vittori, and J.F. Santucci. Toward the definition of an intuitive specification language. In SCS, editor, Proceedings of the International Modeling and Simulation Multiconference (IMSM) - Conceptula Modeling Simulation (CMS), pages 187–192, 2 2007.

[38] N. Giambiasi and S. Ghosh. Min-Max-DEVS : A new formalism for the specification of discrete event models with min-max delays. pages 616–621. 13th European Simulation Symposium, 2001.

[39] P.Y. Glorennec. Algorithmes d'apprentissage pour systèmes d'inférence floue. Hermès Science, 1999.

[40] A.M. Grishin. Mathematical modelling of forest fires and new methods of fighting them. House of the Tomsk State University, 1997.

[41] Halupka, C., Bisgambiglia, P., et Santucci, J. (2000). Devs-based modelling of a pure thermal system. Dans Actes de la conférence WMC 2000, San Diego, California

[42] Hecht-Nielsen, R. (1989). Neural network primer : part i. AI Expert.

[43] A. Hamri, N. Giambiasi, and C. Frydman. Min-Max DEVS modeling and simulation. Simulation Modelling Practice and Theory (SIMPAT), 14(7) :909–929,October 2006. Ed. Elsevier, ISSN 1569–190X.

[43] D. R. Hild. Discrete Event System Specification Distributed Object Computing Modeling and Simulation. PhD thesis, 2000.

[44] Haykin, S. (1994). Neural Networks : A Comprehensive Foundation. Macmillan.

[45] L.S. Iliadis. A decision support system applying an integrated fuzzy model for longterm forest fire risk estimation. ELSEVIER, Environmental Modelling and Software 20 (2005), pages 613–621, 2005.

[46] L.S. Iliadis, A.K. Papastavrou, and D. Lefakis. A computer-system that classifies the prefectures of Greece in forest fire risk zones using fuzzy sets. ELSEVIER, Forest Policy and Economics 4 (2002), pages 43–54, 2001.

[47] Jungst, R., Urbina, A., et Paez, T. (2000). Stochastic modeling of rechargeable battery life in a photovoltaic power system. Rapport Technique 1541C, Sandia national laboratories

[48] C. Jacques and G.A. Wainer. Using the cd++ DEVS tookit to develop petrinets. In SCS, editor, Proceedings of the SCS Conference, 2002.

[49] A. Kaufmann. Introduction à la théorie des sous ensemble flous. Number 1. Masson edition, 1973.

[50] E. Kofman, N. Giambiasi, and S. Junco. Fdevs : A general DEVS based formalism for fault modeling and simulation. European Simulation Symposium, volume 1.Hamburg (2000), pages 77–82, 2000.

[51] E. Kofman, M. Lapadula, and E. Pagliero. PowerDEVS : A DEVS-based environment for hybrid system modeling and simulation. Technical report, 2003. LSD0306, Rosario National University.

[52] F.N. Kogan. Global drought watch from space. Bull. Amer. Meteor. Society,(4) :621–636, 1997.

[53] F.N. Kogan. Global monitoring fire potential from operational satellites. 1999.

[54] Y. Kwon, H. Park, S. Jung, and T. Kim. Fuzzy-DEVS Formalisme : Concepts, Realization

and Application. Proceedings AIS 1996, pages 227–234, 1996.

[55] Seungsoo Lee, Kwang H. Lee, and Doheon Lee. Ranking the sequences of fuzzy values. Inf. Sci. Inf. Comput. Sci., 160(1-4) :41–52, 2004.

[56] M. Paul-Antoine BISGAMBIGLIA.thèse: Approche de modélisation approximative pour des systèmes à événements discrets : Application à l'étude de propagation de feux de forêt (Décembre 2008)

[57] E.H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. International Journal of Man-Machine Studies, (7) :1–13, 1975.

[58] E.H. Mamdani. Advances in the linguistic synthesis of fuzzy controllers. International Journal of Man-Machine Studies, (8) :669–678, 1976.

[59] E.H. Mamdani. Applications of fuzzy logic to approximate reasoning using linguistic synthesis. IEEE Transactions on Computers, (26) :1182–1191, 1977.

[60] A.G. McArthur. Weather and grassland fire behaviour. the Forest and Timber Bureau, 1966.

[61] R. Moore. Interval analysis. 1977.

[62] A. Muzy. Elaboration de modèles déterministes pour la simulation de systèmes spaciaux complexes : appliquion à la propagation des feux de forêt. PhD thesis, Université de Corse, 2004.

[63] Notton, G., Muselli, M., Cristofari, C., Poggi, P., et Louche, A. (2002). Intégration de systèmes hybrides à sources renouvelables d'énergie pour la production d'écentralisée d'électricité en sites isolés. Actes du Congrès International Environnement et Identité en Méditerranée, 1(1) :243–251

[64] Lewis Ntaimo and Bernard P. Zeigler. Expressing a forest cell model in parallel DEVS and timed cell-DEVS formalisms. Proceedings of the 2004 Summer Computer Simulation Conference, 2002.

[65] T.I. Oren. Knowledge-Based Simulation : Methodology and Application. chap.Dynamics Templates and Semantic Rules for Simulation Advisters and Certifiers,1989.

[66] C. Oussalah. Modèles hiérachisés multi-vues pour le support de raisonnement dans les domaines techniques. Technical report, 1988.

[67] Philipi 2003 Thèse (Une architecture logicielle pour la multi-modélisation et la simulation à évènements discrets de systèmes naturels complexes)

[68] Pena-Reyes. Fuzzy modeling. PhD thesis, 2002.

[69] F.P. Ramsey. Truth and probability. The Foundations of Mathematics and other Logical Essays, 1931.

[70] D. Rocacher and P. Bosc. Sur la définition des nombres rationnels flous. Cépaduès editions edition, 11 2003.

[71] R. C. Rothermel. A mathematical model for predicting fire spread in wildland fuels. Research Paper INT- 115, Ogden, UT : U.S. Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station. :40p, 1972.

[72] G. Shafer. A Mathematical Theory of Evidence. 1976.

[73] Mohamed Smaili. Modélisation à retards flous de circuits logiques en vue de leur simulation. PhD thesis, 1994.

[74] Stimpfi-Abele, G. (1995). Validation of input data for trained neuralnet Computer Physics Communications, 85(2) :176–188s.

[75] M. Sugeno. Fuzzy measures and fuzzy integrals : a survey. Fuzzy Automata and Decision Processes, pages 89–102, 1977.

[76] M. Sugeno. Industrial applications of fuzzy control. Elsevier Science Pub. Co,1985.

[77] Hessam S. Sarjoughian and Bernard P. Zeigler. The role of collaborative devs modeler in federation development. In Proceedings of the 99 System Interoperability Workshop, 1999.

[78] Liem Tran and Lucien Duckstein. Comparison of fuzzy numbers using a fuzzy distance measure. Fuzzy Sets Syst., 130(3) :331–341, 2002.

[79] Alejandro Troccoli and Gabriel Wainer. Implementing parallel cell-DEVS. In IEEE, editor, Proceedings of the 36th Annual Simulation Symposium, 2003.

[80] A. Uhrmarcher. Dynamic Structures in Modeling and Simulation : A Reflective Approach. ACM Transactions on Modeling and Computer Simulation vol. 11 2001, pages 206–232, 2001.

[81] H. Vangheluwe. The Discrete EVent System specification DEVS Formalism. Technical report, 2001. http ://moncs.cs.mcgill.ca/.

[82] H. Vangheluwe, J de Lara, and P.J. Mosterman. An introduction to multi paradigm modelling and simulation. In SCS Editions, editor, Inproceedins of AIS 2002, 2002.

[83] R.O. Weber. Modeling fire spread through fuel beds. Prog. Energy Combust. Sci,vol. 11, pages 67–82, 1991.

[84] L.A. Zadeh. Fuzzy sets. Information Control, 8 :338–353, 1965.

[85] L.A. Zadeh. A fuzzy-set theoretic interpretation of linguistic hedges, 1972.

[86] L.A. Zadeh. Outline of a new approach to the analysis of complex systems and decision process. IEEE Trans. Systems, Man and Cybernetics, (3) :28–44, 1973.

[87] L.A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning, 1975. parts 1 and 2.

[88] L.A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning. Information Sciences, (9) :43–80, 1976.

[89] L.A. Zadeh. Fuzzy sets as a basics for a theroy of possibility. Fuzzy Sets and Systems, pages 3–28, 1978.

[90] L.A. Zadeh. The role of fuzzy logic in the management of uncertainty in expert systems. Fuzzy Sets and Systems, pages 199–227, 1983.

[91] Lofti A. Zadeh. Fuzzy logic. Computer, 21(4) :83–93, 1988.

[92] Bernard P. Zeigler. A theory-based conceptual terminology for M and S, 2000. H. et Kim, T. G. « Theory of modeling and simulation ».Deuxième édition Academic Press, 2000-1976

[93] Bernard P. Zeigler. Theory of Modeling and Simulation. Academic Press,1976.

[94] B.P. Zeigler. Multifaceted modelling and discrete event simulation. Academic Press, 1984.

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

[96] Bernard P. Zeigler, Tag Gon Kim, and Herbert Praehofer. Theory of Modeling and Simulation. Academic Press, Inc., Orlando, FL, USA, 2000.

[97] Hong Zhang, Tam C. M., and Heng Li. Modeling uncertain activity duration by fuzzy number and discrete-event simulation. European journal of operational research, 164(3) :573–695, 2005.

[98] Bernard P. Zeigler and S. Vahie. DEVS formalism and methodology - unity of conception diversity of application. In SCS Editions, editor, Proceedings of the 1993 Winter Simulation Conference, pages 573–579, 1993.