REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

Université 20 Août 1955 Skikda

Faculté des sciences.

Département Informatique

Mémoire de fin d'études :

Présentée en vue de l'obtention du diplôme en Master 2

Option: Réseaux et Systèmes Distribués (RSD)

Thème :

Un modèle basé DEVS

Pour la simulation et la vérification d'un procédé industriels

Présenté par :

- **Ramdane Moncef Abdelkrim**
- **Hafsi Houssem**

Encadré par:

Dr. Seddari Noureddine

Année universitaire : 2021-2022

REMERCIEMENTS

Je remercie tout d'abord ALLAH pour m'avoir donné la force et la volonté tout au long de mon travail de fin d'étude car sans lui rien n'aurait pu être.

Un grand merci à mon encadreur de mémoire, Mr Seddari Nour-Eddine pour ses précieux Conseils et son Orientation. Ainsi que son support, sa patience et ces efforts durant ce travail que dieu le bénisse.

Je tiens à remercie également les membres du jury d'avoir accepté d'évaluer mon travail.

Je remercie tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

Enfin, Un grand merci à tous ceux qui m'ont accompagné au cours de toutes ces années, famille et amis.

Merci.

Dédicace

A mes deux parents, eux qui m'ont offert l'un des plus beaux cadeaux de la vie : le savoir. Je leur dis merci pour tout ce qu'ils ont fait et continuent à faire pour moi.

A mon encadreur Mr . Seddari, à toute ma famille, à mes amis, eux qui m'ont toujours soutenu dans les moments difficiles tout au long de mes études.

A tous ceux qui me sont chers

Résumé

La modélisation et la simulation des systèmes revêtent une grande importance dans l'étape de laconception et du contrôle des systèmes industriels. En effet, ces techniques permettent de fournir un prototype formel ou matériel conforme aux spécifications du système réel ou à une ou plusieurs de ses parties et dont le comportement, quand il est soumis à un ensemble d'entrées provoque un ensemble de sorties, qui permettent lorsqu'elles sont projetées sur le système réel d'en déduire un ensemble de conclusions.

Plusieurs travaux de recherche et de nombreuses implémentations ont été conduits sur le thème de la modélisation et de la simulation (M&S) des systèmes industriels. Le travail présenté dans cette mémoire est une contribution dans le domaine de la M&S des systèmes industriels.

Après avoir mené une étude approfondie dans le cadre de la modélisation la simulation des systèmes industriel, Nous avons proposé une méthode qui est basée principalement sur le formalisme DEVS (Discrète Event system Spécification).

Ainsi une version de ce travail a été implémentée sur la plate-forme JAVADEVS et en utilisant l'outil DEVS - Suite et a abouti à l'implémentation d'un simulateur de système industriel (Procédé de compression de la réfrigération de propane pour le nouveau projet de train Skikda LNG, Algérie).

Mots-clés : Simulation, Modélisation, DEVS (Discrète Event system Spécification), DEVS- suite, DEVS-JAVA, Système industriel.

Abstract

System modeling and simulation are of great importance in the design and control stage of industrial systems. Indeed, these techniques make it possible to provide a formal prototype or material conforming to the specifications of the real system or to one or more of its parts and whose behavior, when it is subjected to a set of inputs causes a set of outputs, which allow when they are projected onto the real system to deduce a set of conclusions.

Several research works and numerous implementations have been carried out on the subject of modeling and simulation (M&S) of industrial systems. The work presented in this thesis is a contribution in the field of M&S of industrial systems.

After having carried out an in-depth study within the framework of modelling and simulationof industrial systems, we proposed a method based mainly on the simulation of an industrial process using the DEVS (Discrete Event system Specification) formalism.

Thus a version of this work was implemented on the JAVADEVS platform and using the DEVS - Suite tool and resulted in the implementation of an industrial system simulator (Propane refrigeration compression process for the new Skikda LNG train project, Algeria).

Keywords: Simulation, Modeling, DEVS (Discrete Event system Specification), DEVS-suite, DEVS-JAVA, Industrial system.

الملخص:

نمذجة النظام والمحاكاة لهما أهمية كبيرة في مرحلة التصميم والتحكم في الأنظمة الصناعية في الواقع، تجعل هذه التقنيات من الممكن تقديم نموذج أولي رسمي أو مادي يتوافق مع مواصفات النظام الحقيقي أو لجزء أو أكثر من أجزائه وسلوكه، عندما يخضع لمجموعة من المدخلات، يتسبب في مجموعة من المخرجات، والتي تسمح عندما يتم إسقاطها على النظام الحقيقي لاستنتاج مجموعة من الاستنتاجات.

تم إجراء العديد من الأعمال البحثية والعديد من التطبيقات حول موضوع النمذجة والمحاكاة (M&S) للأنظمة الصناعية. العمل المقدم في هذه الأطروحة هو مساهمة في مجال M&S للأنظمة الصناعية. بعد إجراء دراسة متعمقة في سياق نمذجة ومحاكاة الأنظمة الصناعية، اقترحنا طريقة تعتمد بشكل أساسي على شكليات DEVS (مواصفات نظام الأحداث المنفصلة).

وهكذا تم تنفيذ نسخة من هذا العمل على منصة JAVADEVS وباستخدام أداة DEVS – Suite وأسفرت عن تنفيذ محاكي النظام الصناعي (عملية ضغط تبريد البروبان لمشروع قطار سكيكدة للغاز الطبيعي المسال الجديد، الجزائر).

الكلمات الرئيسية:

المحاكاة، النمذجة، DEVS (مواصفات نظام الأحداث المنفصلة)، مجموعة DEVS، DEVS– JAVA، النظام الصناعي.

TABLE DES MATIERES

Liste de figure

Liste des tableaux

Introduction Générale

Depuis la nuit des temps, les êtres humains ont cherché à comprendre le monde qui les entoure, à transfert des connaissances et la prévision des phénomènes complexe, le domaine de la modélisation et simulation (M&S) apporte un cadre formel pour la spécification de systèmes, continus ou discrets, naturels ou artificiels. Il est défini par la spécification d'un système à l'aide d'un ou plusieurs modèles, selon un formalisme de modélisation.

La simulation des systèmes complexes constitue important majeur dans de nombreux domaines de la société. Elle permet de manipuler, d'observer et d'améliorer la compréhension des phénomènes complexes. Où la simulation imite le fonctionnement d'un processus ou d'un système du monde réel au fil du temps (tester des hypothèses sur l'existant) de les exposer et d'en formuler de nouvelles a posteriori, ce qui fait de la simulation un outil d'investigation unique quel que soit le domaine considéré.

Les premiers simulateurs ont été inventés pour la formation des pilotes d'avion, à moindre risque et à moindre coût. Après cela, il comprend de nombreux secteurs comme : l'industrie pétrolière [Nikiforov, 1999], la modélisation de « crash » sur les voitures, ou encore la simulation CAO/Robotique pour calculer la trajectoire des robots [Riat, 1992]. Ils aident à détecter les erreurs d'exigences et de conception au tout début des cycles de développement de produits, cette capacité peut réduire considérablement le coût associé à l'élimination des erreurs dans la mise en œuvre du système et les étapes de développement des tests.

Le processus de simulation se divise en trois phases itératives qui sont l'élaboration du modèle dans un langage de modélisation, l'exécution de ce modèle dans une plate-forme de simulation et l'analyse des résultats de la simulation [Fishwick, 1994].

Notre travail rentre dans ce cadre ; nous avons opté pour une méthode à événement discret basée sur le formalisme DEVS Discrete Event Systems Specification [Zeigler, 1976 ; Zeigler,1984].

Dans le monde de la M&S, le formalisme DEVS, est dédié à la spécification de systèmes évoluant en fonction d'évènements discrets (i.e. non continus dans le temps). Sa flexibilité et son adaptabilité le rendent adapté à la simulation de modèles dans une variété de

domaines. Il a été implémenté sur plusieurs plateformes.

DEVS détermine les structures et les comportements des entités constituant le système, ce qui permet de créer des interconnections de modèles et d'inclure des modèles dans d'autres modèles (hiérarchie). il doit, pour être productif, être implémenté le modèle dans un langage de programmation. Les algorithmes de simulation de DEVS sont réputés robustes, et plusieurs extensions sont venues compléter ce formalisme.

Une application de cette approche, a été menée pour la simulation de procédés de industriels. Selon son importance, un système industriel peut contenir des milliers d'éléments et même plus. C'est le cas, par exemple des systèmes pétroliers.

L'implémentation du système est faite grâce à la plateforme JAVADEVS et en utilisant l'outil DEVS - Suite ; cette plateforme a été mise au point par l'équipe du B.P. Zeigler au Arizona Center for Integrative Modeling and Simulation (ACIMS) [zeiglar, et sarjoughian, 2000] et repose sur plusieurs outils, par exemple une interface graphique de modélisation et simulation.Ceci, dans le but de profiter de la puissance de cet outil évolué.

Notre travail se compose de quatre chapitres organisés de la façon suivante :

- **Modélisation et Simulation des Systèmes**: ce chapitre constitue une introduction à notre travail par la présentation des notions sur la modélisation, la simulation et la vérification d'une façon générale.
- **Le formalisme DEVS** : ce chapitre constitue une présentation du formalisme DEVS, la modélisation de ce formalisme. Ainsi que quelques une de ses extensions. Nous étudions quelques plateformes et environnements basés sur ce formalisme, a la fin de ce chapitre nous présentons les motivations de choix de ce formalisme.
- **Procédé de compression la réfrigération de propane:** ce chapitre constitue des informations pour les procédés industriels, les régulateurs et les vannes, ou été proposé un système industriel (boucles de réfrigération au propane pour le nouveau projet de train Skikda GNL, Algérie) et proposé aussi un modèle DEVS est conforme au systèmeproposé.
- **L'implémentation** : Ce dernier chapitre présente la réalisation et l'implémentation du notre système en utilisant la plateforme JAVADEVS et l'outil DEVS – Suite.

Problématique et Motivation

Les systèmes industriels sont suffisamment complexes pour être analysés, compris ou même maîtrisés par de simples équations ou d'applications de théorie sans avoir recours à des outils spécifiques.

La complexité d'un système industriel ne réside pas dans le nombre de variables qui le caractérisent, mais plutôt, dans les interconnexions qui existent entre elles Mais, si en plus de cette complexité spatiale vient s'ajouter le temps, on devra alors tenir compte de dynamique du système. Toutefois, si nous avons l'intention de bien analyser le système et de ne rien laisser au hasard, bref de nous rapprocher le plus possible de la réalité afin de pouvoir le maîtriser, nous devons faire attention au comportement temporel incertain (systèmes chaotiques, bifurcatoires).Afin d'analyser ce genre de systèmes, l'analyse mathématique est déconseillée par ce que elle est trop difficile, donc nous pouvons nous référer à la dynamique des systèmes qui nous fournit une démarche s'appuyant sur la science des systèmes asservis etla simulation sur ordinateurs. Ces outils nous permettent prévoir les comportements dynamiques des systèmes en question et de suggérer des modifications possibles sans pour autant être obligé de faire des expérimentations souvent difficiles à réaliser ou nécessitant un temps d'exécution assez long [Karsky, 2002].

La dynamique des systèmes industriels est la science du changement et de l'évolution, ceci implique le fait que l'avancement du système dans le temps s'effectue suivant de petits pas : nous désirons analyser et comprendre l'état du système à chaque instant, c'est pourquoi nous adoptons une approche discrète

Suivre l'approche discrète permet d'analyser l'état du système à tout moment et de le traiter de manière discontinue et discrète. A chaque instant, on maintient une liste linéaire d'évènements prévus avec leur date d'occurrence correspondante. Ce calendrier d'évènements est appelé « échéancier ». Elle est soumise à une relation d'ordre qui est l'occurrence chronologique. La tête de la liste est constituée par l'événement en cours de déroulement, et la fin de la liste par l'événement leplus éloigné dans le temps. Chaque phase de résolution consiste à chercher dans l'échéancier, l'événement qui a la plus petite date prévisionnelle d'occurrence. On sait qu'il n'y aura pas de changements d'état entre la date courante et cette date. Il s'agit, donc, de rechercher cette date, et d'avancer le temps de la simulation jusqu'à celle-ci. Il faut remarquer qu'à part l'événement courant, tous les évènements qui lui succèdent sont potentiels, car leur apparition et leur rang, si jamais n'ils

vont être exécutés, dépendent des traitements des évènements antérieurs.

Les simulations discrètes nous permettent de visualiser le système sous forme d'unités discrètesà mesure que l'état du modèle change au fil du temps. Chaque changement d'état représente un événement qui consiste en la modification des variables d'entrée et de sortie nécessaires pour décrire le système au fil du temps. Ces variables sont discrètes et souvent aléatoires [Leroudier, 1980].

A chaque fois qu'on avance dans le temps, la simulation met en évidence les différents évènements. Nous distinguons, alors deux sortes d'évènements : les évènements externes imposés par l'extérieur, et les évènements internes (conséquences des changements d'état du système). Entre deux évènements successifs, l'état du système reste constant. L'instant auquel l'événement se produit, est appelé date d'occurrence de l'événement. d'un autre côté, lesactions qui durent dans le temps, et qui sont délimitées par des évènements, sont dites activités [Bel, 2001].

Dans ce travail, nous avons choisi le DEVS car il est plus approprié pour représenter la dynamique des systèmes industriels. Ce type de systèmes est asynchrone, dans lequel ses étatssont initiés par ses événements qui provoquent des changements à des moments:

- DEVS intègre naturellement la notion de temps où d'autres nécessitent une extensiondu formalisme pour les supporter.
- DEVS propose une définition formelle séparée du simulateur du modèle.
- DEVS donne un cadre plus général pour la modélisation et la simulation de systèmes à événements discrets.
- DEVS permet de définir des multi-modèles formels, exécutables et numériques, en intégrant des formalismes hétérogènes et des modèles exécutables.

Chapitre 01 :

Modélisation et Simulation des Systèmes

Introduction

La démarche de modélisation et de simulation est une approche scientifique qui a pour but de produire de la connaissance sur des phénomènes naturels ou artificiels. Cette démarche est utilisée pour comprendre, prédire voire contrôler des objets ou des phénomènes qu'ils soient déjà existants ou encore au stade de la conception. Le principe est de créer une simplification (un modèle) des phénomènes ou de l'objet étudie, par exemple une maquette réduite, un prototype ou un modèle numérique. En étudiant le comportement du modèle dans différentes situations il est possible de répondre aux questions que l'on se pose sur le phénomène initial. Ce chapitre constitue une introduction à notre travail par la présentation de quelques notions, qui nous semblent essentielles, sur la modélisation et la simulation d'une façon générale.

1. Modélisation des systèmes:

1.1 Objectif de la modélisation : Le but principal d'une modélisation est souvent de rendre clair et explicite un ensemble d'informations largement implicites. Lors de la construction du modèle, un grand nombre d'ambiguïtés doiv ent être levées. Modéliser un système avant sa réalisation permet de mieux comprendre le fonctionnement du système. C'est également un bon moyen de maîtriser sa complexité et d'assurer sa cohérence. Dans le domaine de l'ingénierie du logiciel, le modèle permet de mieux répartir les tâches et d'automatiser certaines d'entre elles. C'est également un facteur de réduction des coûts et des délais. Le modèle est aussi indispensable pour assurer un bon niv eau de qualité et une maintenance efficace.

1.2. Notion du système : Le terme système est défini comme un ensemble d'éléments qui interagissent pour atteindre un objectif, il est largement utilisé dans tous les domaines de la v is et du savoir par exemple : le système circulatoire, le système informatique, le système scolaire, le système économique,…etc. Ces exemples montrent qu'un système n'est pas un assemblage disparate d'éléments. Il consiste en un groupement de composants bien déterminés qui fonctionne ensemble pour atteindre un but ou un objectif commun. A n'importe quel instant, le système est dans un état particulier défini par l'état de ses composants et des relations qui les relient.

L'état du système change lorsque l'état de ses composants et / ou l'état des relations qui les relient change.

1.3. Types de systèmes : Il existe plusieurs types de systèmes : statique ou dynamique, déterministe ou stochastique, continu ou discret.

1.3.1. Système déterministe : Il fonctionne de manière prévisible, l'interaction entre les différentes parties est connue av ec exactitude. Si l'on possède une description de l'état du système à un moment donné, le prochain état du système peut être donné exactement et sans erreur.

1.3.2. Système probabiliste (stochastique) : Ce modèle est basé sur l'élaboration d'équationsprécises, le système est alors totalement prévisible.

1.3.3. Système continu : Les changements de l'état du système se font permanences av ec le temps(dépend d'une variable liée au temps).

1.3.4. Système discret (discontinu) : Le système est caractérisé par des événements qui surv iennent à des instants non fixes et causent un changement de l'état jusqu'au prochain événement.

1.3.5. Système mixte : La séparation entre système continu et discret est en quelque sorte artificielle, car en réalité la plupart des systèmes possèdent des composantes continues et discrètes à la fois. De tels systèmes sont dits mixtes.

1.4. Notion de modèle : D'une façon générale un modèle est une représentation d'un système réel (physique, économique, humain,…etc) réalisé dans le but de mieux étudier ce système est expliqué certains de son comportement.de façon un peu plus particulière, on peut utiliser la définition de P.hubert: « un modèle et défini comme un objet ou une personne à imiter, un exemple ou un archétype. Il est devenu dans le domaine scientifique une construction matérielle ou abstraite "ressemblant" à l'objet modélisé, selon un certain nombre de caractéristiques pertinentes eu égard aux données disponibles et à l'objectif poursuivi».

Parfois, il est impossible d'étudier le système directement du fait qu'il soit inaccessible (système solaire), trop couteux, il change trop rapidement (tir nucléaire) ou lentement (mouvement d'une comète), dans ce cas, l'étude est faite sur un modèle, c'est-à-dire un deuxième système original et aussi parfait que l'étude l'exige.

1.5. Types de modèles : On distingue plusieurs catégories de modèles On selon la nature dusystème étudié : Iconiques ou physiques. Symboliques ou abstraits.

1.5.1. Les modèles iconiques ou physiques : Sont une représentation du système réel par une maquette à l'échelle réduite qui est encore très utilisée par exemple (construction automobile, architecture…).

1.5.2. Les modèles symboliques ou abstraits : Sont une représentation mathématique et logique d'un problème pouvant manipuler de façon expérimentale sur un ordinateur. Exemple : le modèle mathématique du système solaire. L'ensemble des symboles et des équations mathématiques décrivent le comportement dynamique du soleil et de ses planètes à un niveau purement abstrait. Seuls les deux derniers modèles concernent la simulation.

1.6. Modélisation informatique : Le modèle informatique est le plus utilisé actuellement dans certains systèmes complexes, comme c'est le cas dans la construction d'av ions ou d'automobiles par exemple, on utilise, généralement des prototypes grandeur nature du système réel sur lesquels sont effectué des simulations par informatique.

1.7. Les outils de modélisation informatique : Il existe plusieurs outils pour la modélisation informatique : les systèmes à file d'attente, les réseaux de pétri,…etc. Ces outils sont à la base de différents logiciels de simulation.

1.7.1. Les files d'attente : Une file d'attente est un outil important utilisé pour modéliser certains types de systèmes stochastiques, elle est constituée d'un ensemble d'individus (ou objets) qu'on appelle clients qui viennent acquérir un service auprès d'un ou plusieurs serveurs.

Les réseaux de files d'attente sont fréquemment utilisés pour modéliser des systèmes à événements discrets. Elle permet la modélisation du partage des ressources.

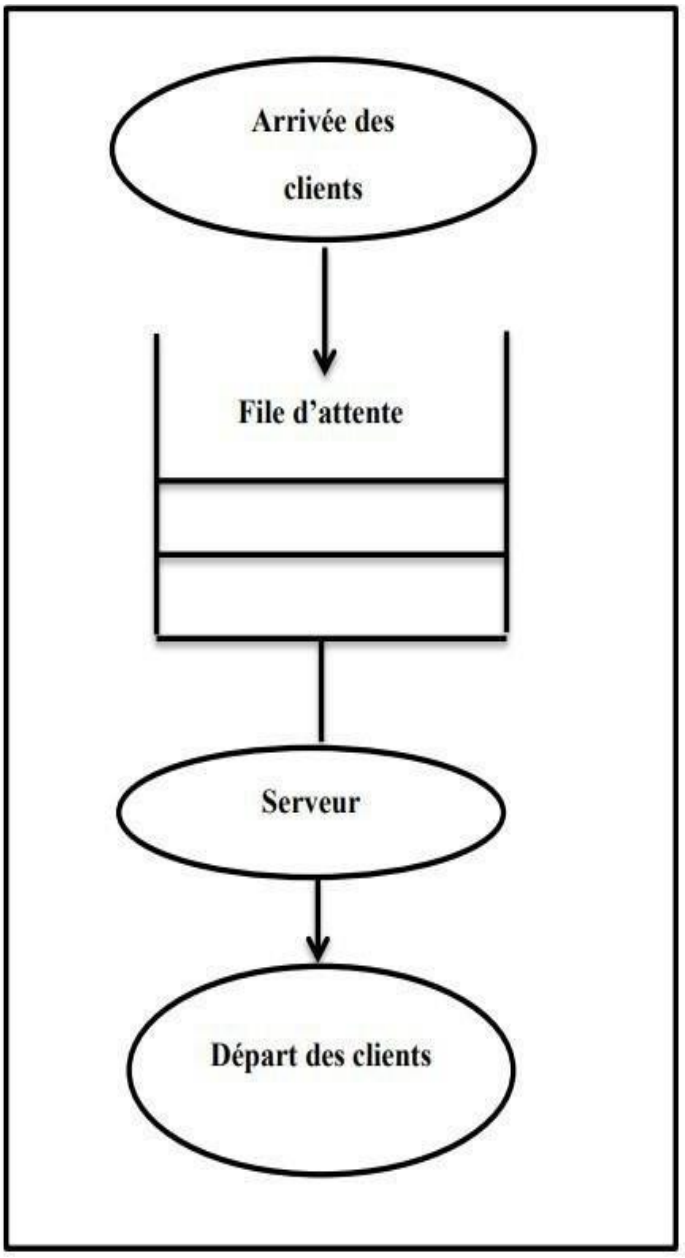


Figure1.1 : Représentation graphique d'une file d'attente primaire

La représentation ci-dessus (Figure1) peut être étoffée par l'ajout de plusieurs utilisateurs (serveurs multiples) au bout de la file d'attente et d'une capacité limitée de la file d'attente.
Le système entré-sortie des clients peut être contrôlé (FIFO, LIFO, Random…). Les clients eux-mêmes peuvent être classés, puis ensuite répartis dans la file selon ce classement. Finalement le réseau de files d'attente est créé par la connexion d'un ensemble de file simple.

1.7.2. Les réseaux de pétri : Le réseau de pétri est un modèle mathématique à variables discrètes, créés en 1962 par Carl Adam Pétri. Il se représente par un graphe biparti (composé de deux types de nœuds) orienté (composé d'arc(s)) reliant des places et des transitions (les nœuds). Deux places ne peuvent pas être reliées entre elles, ni deux transitions. Les places peuvent contenir des jetons (pouvant par la suite prendre différentes valeurs), représentant généralement des ressources disponibles.

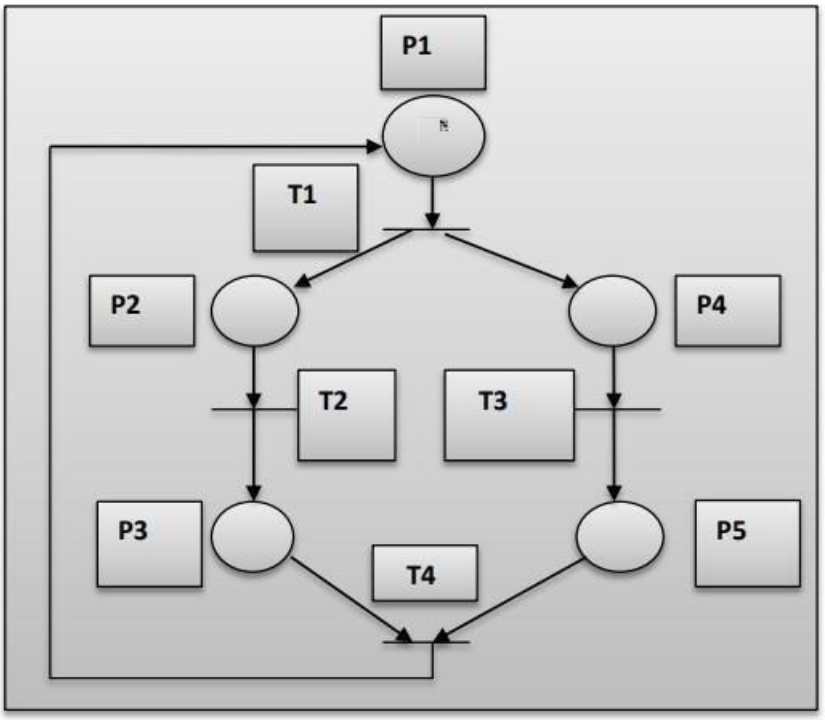


Figure1.2 : Typologie de la modélisation graphique des réseaux de Pétri.

1.7.3. Modélisation orientée objet : Dans la Modélisation orientée-objet, les systèmes sont uniquement constitués d'entités appelées objets. Ces objets sont définis par des types qui définissent de façon syntaxique et sémantique les propriétés que présenteront les objets du type, ces propriétés permettent de délimiter, de qualifier les objets et de savoir comment on peut communiquer avec eux.

L'implémentation de ces propriétés est ensuite fournie par une classe, structure de données qui lie à unepropriété une implémentation possible.

L'implémentation des propriétés par une classe peut être représentée soit sous forme d'emplacement mémoire (champs de données) appelé attribut, soit sous forme de calcul (opération) appelé méthode. En orientée-objet, si un type définit donc l'interface de l'objet,

la classe lui fournit une implémentation. A ce titre, la classe est le moule par lequel est construit un objet. On dit alors d'un objet qu'il est une instance de telle classe. Les objets interagissent via des envois de messages.

1.7.4. Modélisation orientée agent : La modélisation OA permet une factorisation (décomposition en sous-systèmes) et une délégation naturelle des tâches, buts et/ou rôles à sous-systèmes multi-agent et leurs agents. Cette décomposition permet de concevoir plus facilement des systèmes complexes en décomposant le problème en sous-problèmes. Les sous-systèmes sont séparés sur différentes machines et les communications et interactions s'effectuent via des langages de communication qui sont la plupart du temps basés sur la théorie des actes de langages.

2. Simulation : La simulation c'est la démarche scientifique qui consiste à réaliser une reproduction artificielle, appelée modèle, d'un phénomène réel que l'on désire étudier, à observer le comportement de cette reproduction lorsqu'on en fait varier certains paramètres, et à en induire ce qui se passerait dans la réalité sous l'influence de variations analogues. La démarche de simulation passe donc par trois étapes distinctes: l'étape de modélisation, qui consiste à construire le modèle du phénomène à étudier, l'étape d'expérimentation, qui consiste à soumettre ce modèle à un certain type de variations, et l'étape de validation, qui consiste à confronter les données expérimentales obtenues avec le modèle à la réalité.

Selon Shannon , la simulation peut être définie de la façon suivante : " **the process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behavior of the system and/or evaluating various strategies for the operation of the system**".

Pour Fishwick, la simulation peut être définie de la façon suivante: "**Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analyzing the execution output**".

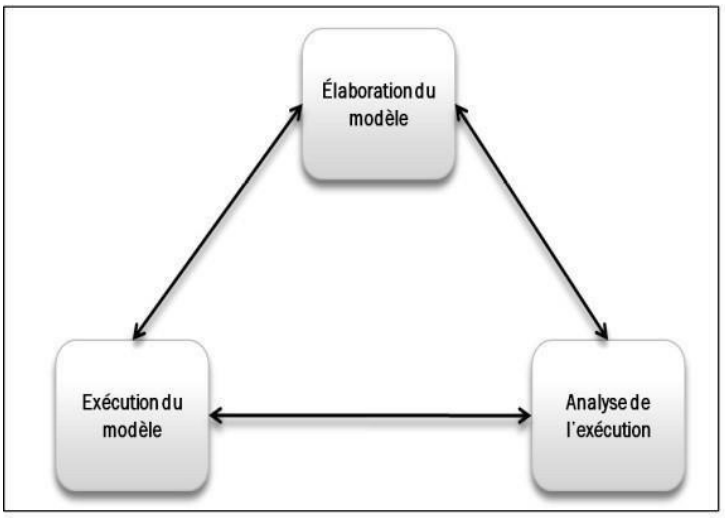


Figure1.3 : La simulation informatique.

2.1. La simulation informatique : La simulation informatique est l'un des outils permettant de simuler des phénomènes réels, le but principal de la simulation est de :

- Étudier un système réel de manière à comprendre son fonctionnement interne et/ou à en prévoir son évolution sous certaines conditions.
- Cette étude se fait nécessairement à travers un modèle du système réel qui est utilisé pour réaliser les expérimentations.

La théorie de la simulation :

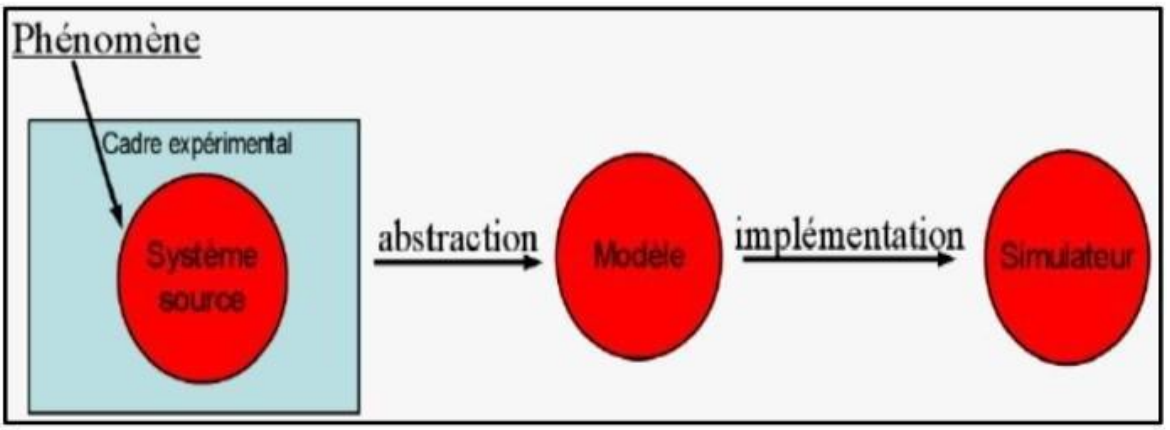


Figure 1.4 : Vue d'ensemble de la simulation informatique.

Système source : le phénomène que l'on souhaite étudier.

Cadre expérimental : spécifications des conditions d'observation du système et des objectifs dela simulation.

Modèle : l'ensemble des instructions qui permettent de générer - à l'aide d'un programmeinformatique - le comportement du système au cours du temps.

Simulateur : le programme informatique capable d'exécuter le modèle et de produire son comportement.

2.2. Catégories de simulation informatique : On peut distinguer trois catégories de simulations :

2.2.1. La simulation continue : S'applique à la classe des systèmes dynamiques, caractérisés par des changements qui se font en permanence avec le temps. Ces types de changements peuvent être présentés par des équations différentielles qui v ont théoriquement permettre aux variables d'être calculées à tout instant.

2.2.2. La simulation discrète : Dans laquelle le système est soumis à une succession d'évènements qui le modifient. Ces simulations ont v ocation à appliquer des principes simples à des systèmes de grande taille. La simulation discrète se div ise en deux grandes catégories : Asynchrone ou time-slicing : on simule à chaque fois le passage d'une unité de temps sur tout le système. Synchrone ou event- sequencing : on calcule l'arrivée du prochain événement, et on ne simule qu'événement par év énement, ce qui permet souvent des simulations rapides, bien qu'un peu plus complexes à programmer.

2.2.3. La simulation par agents : Où la simulation est segmentée en différentes entités qui interagissent entre elles. Elle est surtout utilisée dans les simulations économiques et sociales, où chaque agent représente un individu ou un groupe d'individus. Par nature, son fonctionnement est asynchrone

2.3. Les avantages de la simulation :

La simulation est souvent moins chère que l'expérimentation et comporte beaucoup moins de risques lorsque l'homme fait partie du système étudié. Les résultats peuvent être obtenus beaucoup plus rapidement.

La simulation permet d'effectuer des recherches sur un système isolé, en faisant varier les paramètres un à un et en recommençant avec les mêmes conditions initiales.

La simulation informatique est un outil qui a eu rapidement une bonne réputation dans la conceptionet d'analyse des systèmes.

La simulation aide les chercheurs, les ingénieurs et bien d'autres professionnels à prendre dans le temps des décisions intelligentes concernant la conception et l'opération d'un système.

La simulation est utilisée pour investir une large variété de questions de type "WHAT IF" (QUOI SI)à propos d'un système réel.

La simulation permet le contrôle du temps

La simulation peut être utilisée même si les données considérées sont en quelque sorte superficielle et sommaire.

La simulation permet d'avoir un aperçu des variables les plus importantes sur la performanced'un système ainsi que les interactions entre elles.

3. Domaine d'application de la simulation : Le tableau ci-dessous présente quelques domainesd'application de la simulation :

Région	**Domaines d'application de la simulation**
Système informatique	Les composants « hardwares », les logiciels, le réseau du « hardware », les bases de données et la gestion, le processus d'information, la fiabilité des « hardwares » et des logiciels, etc.
Domaine manufacturiers	Systèmes de manutention, les lignes d'assemblage, les installations de production automatisées, les installations de stockage, les systèmes de contrôle d'inventaire, l'étude de fiabilité et de maintenance, le plan d'aménagement, le design des machines, etc.
Les affaires	Analyse des stocks et des commodités, la politique des prix, stratégies de marketing, les études d'acquisition, les « cash-flow», les prév isions, les alternatives de transport, la planification de la main d'œuvre, etc.
Gouvernement	Les armes militaires et leurs utilisations, les stratégies militaires, planification de la population, l'utilisation des terres, la distribution des soins médicaux, la protection contre les feux, services de polices, etc.
Bio-Science	Les analyses des performances du sport, le contrôle desmaladies, les cycles de v ie biologiques, les études biomédicales, etc.

Tableau.1 : Domaine d'application de la simulation.

4. Buts de la modélisation et de la simulation : Plusieurs buts peuvent être réalisés par la simulation :

- Meilleure compréhension du système.
- Mesure de performance du système.
- Dimensionnement/optimisation de systèmes de production (avant/après réalisation).

- Expérimentation du comportement dynamique des systèmes à l'aide de modèlesinformatiques.

- Réalisation d'un modèle est son utilisation économique (plus simple, moins chère, moins dangereuse, possible, etc.) à la place du système.
- Identification des facteurs critiques (Panne, Blocage).
- Réponse aux questions « que se passe-t-il…si …? ».
- Outil d'aide à la décision en « temps réel » (durant exploitation).
- Outil de formation du personnel (simulateurs de vol, simulateurs de choix…)

Conclusion

Le modèle peut être caractérisé par quelques paramètres physiques, et il s'adapte aux données expérimentales. La simulation nous permet de v isualiser le comportement d'un système et de l'analyser. Ceci dit, il ne faudrait pas oublier que les résultats obtenus par simulation, dépendent des hypothèses retenues pour construire les modèles.

Chapitre 02 :

Le Formalisme DEVS

2.1 Introduction

DEVS (Discrete Event System Specification) est un formalisme proposé par le professeur B.P Zeigler en 1976. Ce formalisme repose sur un ensemble de bases mathématiques pour la modélisation et la simulation des systèmes à événements discrets [Zeigler *et al.*, 2000b]. Aujourd'hui, une grande communauté de chercheurs s'intéresse à ce formalisme. Aussi, ses spécifications ont évolué, de nouvelles considérations ont ainsi été ajoutées pour l'améliorer et l'adapter à la modélisation de phénomènes et de systèmes particuliers tels que les éco- systèmes, les systèmes naturels, les systèmes industriels,…La version originale de DEVS est appelée actuellement DEVS classique. D'autres versions plus récentes de DEVS telles que Parallel DEVS [Zeigler *et al.*, 2000b; Chow et Zeigler, 1994; Vangheluwe, 2004] ont été introduites pour prendre en charge un certain nombre de concepts qui n'existent pas de façon explicite dans DEVS classique.

Ainsi, dans Parallel DEVS, le cas de conflit occasionné par l'occurrence de transitions simultanées est pris en considération, chose qui n'existe pas dans DEVS classique. Ce cas est traité dans Parallel DEVS grâce à la fonction de transition-collision appelée λ (*confluent transition function)* qui agit au niveau des modèles atomiques.

Il existe plusieurs autres extensions en plus de DEVS classique et de Parallel DEVS. Parmi les plus significatives, Cell-DEVS [Wainer et Giambiasi, 2001a; Wainer et Giambiasi, 2001b] et Dynamic Structure DEVS (DSDEVS) [Barros, 1997; Uhrmacher, 2001]. Cell-DEVS améliore DEVS en le rendant plus efficace dans la modélisation des phénomènes utilisant un partitionnement des états dans l'espace. L'espace est décomposé en cellules, le fonctionnement d'un tel système est bien adapté aux automates cellulaires. DSDEVS adapte DEVS pour la modélisation de situations qui incluent des structures à changement dynamique. L'un des avantages de DEVS est qu'il procure non seulement un cadre modulaire et hiérarchique, mais aussi le fait qu'il détermine le concept de simulateur abstrait (c.à.d. une sémantique opérationnelle) grâce à laquelle le comportement du model peut être régénéré.

Il existe plusieurs implémentations différentes de ces formalismes qui reposent sur DEVS et qui en reprennent les fondements. Ainsi plusieurs outils DEVS pour la modélisation et la simulation sont disponibles actuellement [Zeigler, 2005; Bolduc et Vangheluwe, 2002; Nutaro, 2003], chacun ayant ses propres caractéristiques et fonctionnalités. Nous en présentons, dans ce qui suit, quelques-uns en les commentant de façon succincte.

2.2 Le formalisme DEVS

Le formalisme DEVS [Fishwick 1995; Vangheluwe, 2001] est une approche de modélisation basée sur la théorie générale des systèmes. Il s'agit, plus précisément, d'un formalisme modulaire et hiérarchique pour la modélisation basée sur le concept d'état.

DEVS a été élaboré et adopté depuis plus de quarante ans par une communauté internationale de chercheurs [Vangheluwe, 2001; Sarjoughian et Zeigler, 1999; Kofman *et al.*, 2000; Hild,

2000; Anglani *et al.*, 2000a; Filippi *et al.*, 2002a; Jacques and Wainer, 2002; Hamri *et al.*, 2006]. Ces travaux sont basés sur le développement d'architectures logicielles permettant d'une part de faciliter les étapes de modélisation, de simulation et de validation et d'utiliser, d'autre part, le même environnement de multi modélisation pour analyser les systèmes résultant des différents champs afin de générer automatiquement les algorithmes de simulation.

DEVS peut être considéré comme un environnement de multi-modélisation qui permet de rassembler (voire de fédérer) de manière cohérente d'autres formalismes de modélisation qui se basent eux-mêmessur la théorie générale des systèmes. C'est, en effet, un formalisme adapté à un grand nombre de champs d'application [Barros, 1995; Uhrmarcher, 2001; Ntaimo et Zeigler, 2004; Troccoli et Wainer, 2003].

2.2.1 La modélisation DEVS

La multitude d'outils de modélisation et le désir de réutiliser des modèles existants ont motivé les chercheurs à orienter leurs travaux vers un objectif de normalisation de ces outils. Le formalisme DEVS semble être bien adapté à cette mission. La force du formalisme DEVS pourrait se résumer dans sa capacité à exprimer, grâce à la notion d'abstraction appliquée à chaque niveau, et ce, à partir des modèles atomiques ou couplés, la collaboration d'un ensemble de modèles où chaque entité interagit avec les autres. Bien qu'indépendante de l'implémentation, DEVS assure une vision modulaire et hiérarchique des modèles dynamiques. Les événements générés par un modèle peuvent prendre des valeurs dans différents domaines. Ainsi, et d'après BP Zeigler [Zeigler *et al.*, 2000a], nous pouvons prouver qu'il existe un modèle DEVS pour l'ensemble des systèmes à événements discrets. Mais nous pouvons aller plus loin. En effet, DEVS peut être «universel» [Touraille *et al.*,

2010], ce qui permet le couplage des modèles et formalismes décrits comme étant des

paradigmes hétérogènes. L'idée cruciale est que les modèles sont considérés comme des boîtes noires qui n'ont de liens avec le monde extérieur qu'à travers les ports d'entrée et de sortie. Ces ports leur permettant d'échanger des événements et des valeurs. Grâce à cette fonctionnalité d'abstraction, plusieurs modèles peuvent être couplés tout en tirant profit de la réutilisation des modèles existants. Il est également possible de mener une vérification formelle des modèles DEVS, qui est une assistance précieuse dans la conception des systèmes [Freigassner *et al.*, 2000].

2.3 DEVS classique

2.3.1 Spécification formelle d'un modèle atomique DEVS classique

Un modèle atomique DEVS est décrit par l'équation suivante :

$$AtomicDEVS = <X, Y, S, \delta_{int}, \delta_{ext}, t_a, \lambda> \quad (2.1)$$

X est l'ensemble des entrées externes. Y est l'ensemble des sorties du modèle.

S représente l'ensemble des états. Deux variables d'état sont habituellement présentes, " phase " et " sigma " (en l'absence d'événements externe le système reste dans la " phase " courante pendant le temps donné par " sigma").

δ_{int}: $S \rightarrow S$ est la fonction de transition interne qui fait évoluer le système d'un état à un autre de manière autonome.

δ_{ext}: $Q \times X \rightarrow S$ est la fonction de transition externe qui se produit lorsque le modèle reçoit un événement externe. Elle renvoie le nouvel état du système basé sur l'état actuel.

$Q = \{(s, e) \mid s \in S, 0 \leq e \leq ta(s)\}$ ensemble total des états. e : est le temps écoulé depuis la dernière transition.

λ: $S \rightarrow Y$: fonction de sortie du modèle. Elle est activée lorsque le temps écoulé dans un état donné est égal à sa durée de vie.

$t_a(s)$ indique la durée de vie d'un État "s" du système. C'est le temps pendant lequel le modèle demeurera dans cet état si aucun événement externe ne se produit. Quand la variable

d'état de " sigma " est présente, ceci la fonction renvoie juste la valeur du " sigma ".

Un événement émis en sortie d'un modèle est interprété comme un stimulus par les modèles qui sont connectés à cette sortie. Ce stimulus est spécifié par la fonction de transition externe. La fonction de transition interne, quant à elle, est évaluée si la durée de vie d'un état notée *ta* est atteinte. La fonction de sortie est activée lors des transitions internes. B.P. Zeigler propose dans [Zeigler *et al.,* 2000b] principalement deux formes de DEVS : classique ou avec ports. Nous emploierons dans ce qui suit, uniquement la définition de DEVS avec ports ce qui permet de mieux découper les modèles. Un modèle DEVS atomique peut, ainsi, être représenté de façon graphique par une boîte noire contenant la structure et le comportement du système. La fonction δ_{con} qui est responsable de la gestion des conflits peut être éventuellement utilisée pour compléter la liste des fonctions DEVS lorsque deux événements surviennent en même temps. Cette fonction fait partie d'une variante de DEVS appelé Parallel-DEVS [Zeigler *et al.,* 2000b].

La figure 3.1 illustre un modèle DEVS en action.

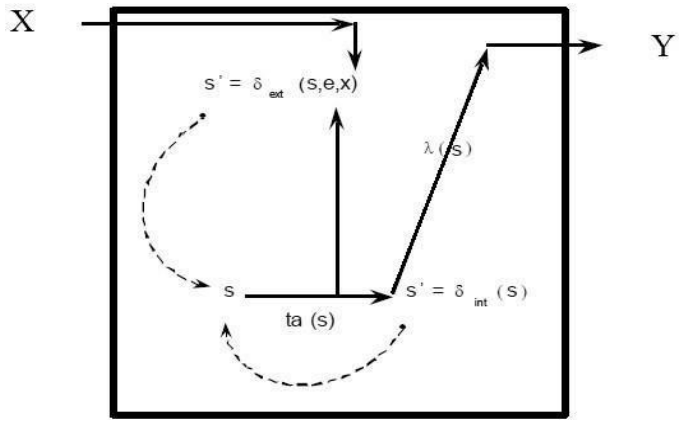


Figure 2.1 : Modèle atomique en action.

A un instant *t,* le modèle atomique est dans un état *s.* Si aucun événement externe n'intervient, le système restera dans cet état pendant durant le temps donné par la fonction ta(s). Lorsque le temps de vie dumodèle expire, i.e. lorsque qu'il s'est écoulé e = ta(s), le système active sa fonction de sortie λ(s) et change d'état grâce à l'exécution de la fonction de transition interne δint (s). Si un événement externe x ЄX intervient avant que le temps ne soit expiré, i.e. quand le système est dans l'état total (s,e) avec e ≤ ta(s), le système change d'état grâce à l'exécution de la fonction de transition externe δext (s,e,x). Dans les deux cas, le système est alors dans un nouvel état (s') avec un nouveau temps restant ta (s') et ainsi de suite.

Le temps de vie du modèle peut être quelconque entre zéro et l'infini. Dans le premier cas, on dit de s qu'il est dans un état transitoire (aucun événement externe ne peut intervenir

avant l'arrivée du prochain changement d'état) dans le second cas, le système restera dans l'état s indéfiniment si aucun événement externe ne vient l'interrompre. Nous disons dans ce cas que s'est un état passif.

2.3.2 Spécification formelle d'un modèle DEVS couplé classique

Un modèle couplé DEVS est un modèle construit en employant d'autres modèles atomiques ou couplés. Ainsi, chaque modèle atomique DEVS peut être combiné avec un ou plusieurs modèles afin de construire un modèle Couplé. Cette opération peut être répétée afin d'obtenir une hiérarchie de modèles couplés. Ainsi, un modèle DEVS couplé peut être considéré comme un réseau hiérarchique de composants atomiques et couplés (cf. Figure 3.2). Le réseau encore appelé structure du modèle est caractérisé par les ports d'entrées et de sorties des modèles qui le constituent et les connections internes entre ces modèles.

$$CoupledDevs = <X_{self}, Y_{self}, D, \{Md/d \in D\}, EIC, EOC, IC> \quad (2.2)$$

Self: est le modèle lui-même.

X_{self}: est l'ensemble des entrées du modèle couplé.

Y_{self}: est l'ensemble des sorties du modèle couplé.

D est l'ensemble des noms associés à des éléments du modèle, *self* n'est pas en D.

{Md / j Є D} est l'ensemble des composants du modèle couplé.

EIC, EOC et IC définissent la structure de couplage dans le modèle couplé.

EIC est l'ensemble des couplages externes en entrée. Ils relient les entrées du modèle couplé à celles de ses propres composants.

EOC est l'ensemble des couplages externes en sortie. Ils relient les sorties des composants à celles des modèles couplés.

IC définit le couplage interne. Il relie les sorties des composants aux entrées provenant d'autres composants dans le même modèle couplé. Toutefois, aucune rétroaction directe des boucles n'est autorisée. Ce qui signifie qu'un port de sortie d'un composant (modèle) ne peut pas être connecté à un port d'entrée du même composant.

La figure 3.2 présente un modèle couplé.

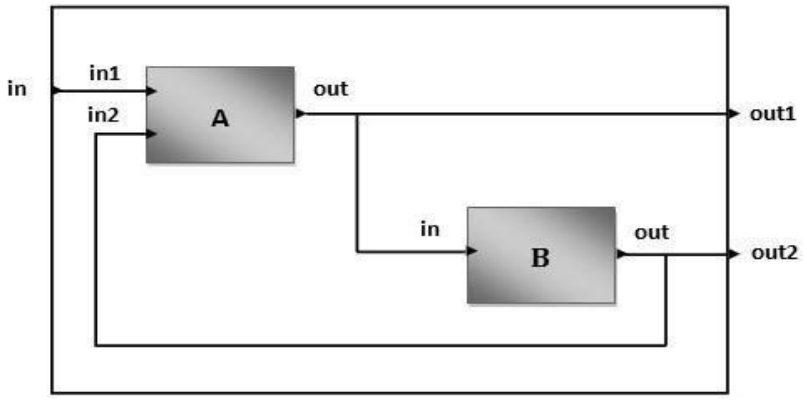


Figure 2.2 : Représentation graphique d'un modèle couplé (b) se composant de deuxmodèles atomiques A et B.

2.3.3 Simulation DEVS

Un des points forts de la modélisation/simulation DEVS réside dans la notion de simulation abstraite.

B.P. Zeigler a défini un simulateur abstrait indépendant d'une quelconque implèmentation et apte à simuler tout modèle dècrit selon le formalisme DEVS. Il est ainsi nécessaire, pour établir une simulation, que le comportement du système à simuler soit déterminé de façon précise. De même, les différentes interactions qui existent entre les différents entités le composant doivent, elles aussi, être déterminées. Une fois ce travail fait, le principe consiste en l'élaboration automatique des algorithmes de simulation correspondants. Dans DEVS, la simulation a pour objectif de generer un ensemble d'évènements de sortie à partir d'un ensemble d'évènements d'entrée issus d'un système déterminé. L'architecture du simulateur abstrait, developpé par B.P. Zeigler repose sur le formalisme DEVS et met en œuvre un ensemble d'algorithmes qui permettent d'implémenter les instructions implicites du modèle considéré pour en élaborer le comportement [Zeigler,
1984; Zeigler, 1990]. C'est là un interet fondamental étant donné que la construction du simulateur est indépendante du modèle. De plus, l'architecture du simulateur abstrait telle qu'elle est proposée sépare la partie modélisation de la partie simulation au niveau de la réalisation. A chaque composant du modèle correspond un composant du simulateur.

Dans le simulateur abstrait, on trouve deux sortes d'éléments de simulation: les coordonnateurs et les simulateurs. Ces deux éléments sont appelés : les processeurs. Les simulateurs sont responsables des modèles couplés qui leurs sont liés, tandis que les coordonnateurs sont, quant à eux, responsables du contrôle des modèles atomiques qui leurs sont assujettis.

Le coordonnateur Root (racine) est un coordonnateur particulier qui contrôle tout le

procédé de simulation et est lié au coordinateur du modèle couplé, de plus haut niveau. La structure de l'ensemble des processeurs représente un graphe qui est nommé ‘arbre de simulation’ (figure 2.3). Cet arbre a en charge l'exploitation du modèle.

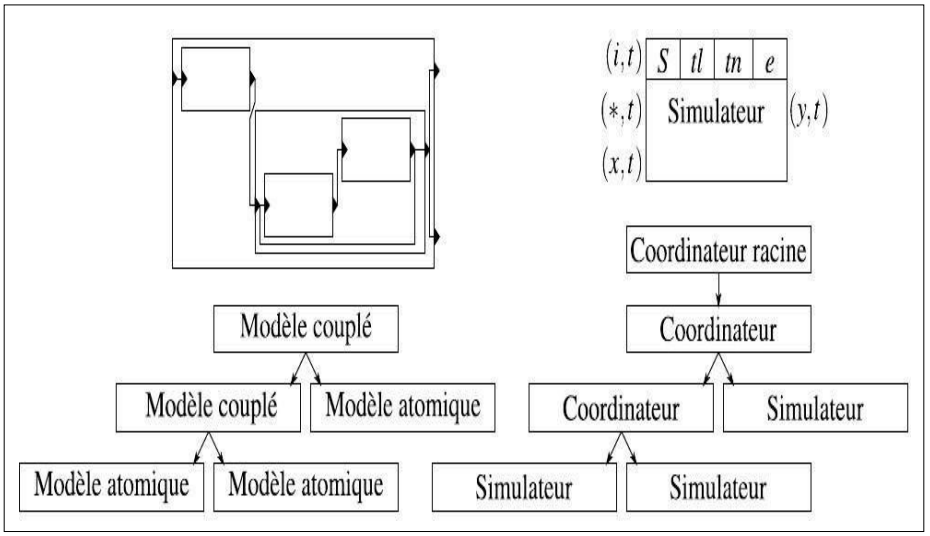


Figure 2.3 : Arbre de simulation

Chaque élément communique grâce à plusieurs types de messages : les "i"-messages initialisent les variables des processeurs, les "x"-messages symbolisent l'arrivée d'un évènement d'entrée externe sur le module, les "y"-messages symbolisent la réponse d'un module par un évènement de sortie, les "*"- messages permettent de prendre en compte les évènements internes qui vont modifier l'état du module associé au processeur concerné par le message, les "d"-messages signalent qu'une transition d'état du module concerné vient de s'achever ; le temps véhiculé par ce type de message permet la synchronisation de la simulation. On trouvera dans [Zeigler,
1976 ; Vangheluwe, 2001] leur enchainement. Un échéancier, qui est une structure de données stocke les événements générés par les messages et les classe par ordre chronologique. La tête de l'échéancier représente le futur immédiat, et la queue le futur plus lointain. La simulation à pour rôle de faire évoluer le temps et à provoquer les changements d'états en fonction des évènements.

2.3.3.1 Algorithme d’un composant Simulateur

Nous considérons ci-dessous un composant simulateur DEVS qui utilise deux variables temporelles t_l *(last time)* et t_n *(next time)*. *(last time)* et t_n *(next time)*. t_l *(last time)* correspond au temps de simulation du dernier événement et t_n *(next time)* au temps

d'apparition du prochain événement. En considérant la définition du temps d'avancement *ta ;* on a : $tn = tl + ta$. Lorsque, de plus, le temps de simulation t est connu, le composant simulateur peut calculer le temps écoulé depuis le dernier événement : $e = t - tl$ et le tempsqui reste avant l'apparition du prochain événement : $s = tn - t = ta - e$. Le temps tn est envoyé au composant coordinateur parent afin de lui permettre d'effectuer une bonne synchronisation des événements.

Algorithme 2.1 : Algorithme du simulateur DEVS.

Variables :

parent // *coordinateur parent*

tl // *temps du dernier événement*

tn // *temps du prochain événement interne*

DEVS =< *X,Y,S,* δint , δext , λ, *ta*>//

*modèle associé*y // *sortie courante du*

modèle **Réception i-message (i,t) au**

temps t : $tl = t - e$

$tn = tl + ta(s)$

Réception *-message (*,t) au temps t :

si ($t \neq tn$) alors

Erreur : mauvaise synchronisation

$y = \lambda(s)$ *//stockage de la sortie avant changement d'état*

envoie y-message (y,t) au parent coordinateur

$s = \delta int(s)$

$tl = t$

$tn = tl + ta(s)$

Réception x-message (x,t) au temps t avec x en entrée:

si $!(tl \cdot t \cdot tn)$ alors

Erreur : mauvaise synchronisation

$e = t - tl$

$s = \delta ext(s,e,x)$

$tl = t$

$tn = tl + ta(s)$

Algorithme 2.2: Un composant Coordinateur

Un composant coordinateur a pour tâche d'assurer le bon fonctionnement et la synchronisation des ses subordonnés *(simulateurs et/ou coordinateurs qui le composent)*. Dans un but de synchronisation, ce composant utilise une liste d'événements $liste_{event} = \{(d, t_{nd}) \mid d \in D, t_{nd} \in R+\}$. Le premier élément de la liste détermine, alors, le prochain événement du coordinateur. Le temps minimum, $t_n = min\ \{t_{nd} | d \in D\}$ est envoyé aux parents du coordinateur comme étant le temps du prochain événement. De manière similaire, le temps de l'événement précédent du coordinateur est calculé par : $t_l = max\ \{t_{nd} | d \in D\}$.

La fonction $Z_{i,j}$ permet la transmission des valeurs entre les ports des modèles i et j. Par exemple, si la valeur y_i est issue d'un port du modèle i et que ce port est relié à un port du modèle j qui attend une valeur x_j, on a alors : $Z_{i,j}(y_i) = x_j$. L'ensemble I_i est composé des modèles qui influencent le modèle i. Si un modèle i a deux ports d'entrées reliés à deux modèles j et k, on a : $I_j = \{ j,k\}$.

Algorithme 2.2 : Algorithme du coordinateur DEVS.

Variables :
parent // *coordinateur parent*
t_l // *temps du dernier événement*
d //fils imminent sélectionné*
Réception i-message (i, t) **au temps t :**
pour chaque modèle d dans D faire :
envoi d' un i-message (i, t) au fils d
$t_l = max\ \{t_{ld} | d \in D\}\ t_n = min\ \{t_{nd} | d \in D\}$

$t_n = min\{t_{nd} | d \in D\}$

Réception x-message (x, t) **au temps t avec x en entrée :**
si !($t_l \leq t \leq t_n$) alors :

Erreur : mauvaise synchronisation

//*consultation du couplage externe pour obtenir les fils influencés receveur* =

$\{r \mid r \in D, MC \in I_r$,

$Z_{MC,r}(x) \neq 0\}$ pour

chaque r dans

receveur :

envoie x-message (x_r, t) avec $x_r = Z_{MC,r}(x)$ à r

$tl = t$

$tn = min\{tnd | d \in D\}$

Réception y-message ($yd*, t$) **au temps t de la part de** $d*$**:**

si $d* \in IMC$et $Zd*,MC(yd*) \neq 0$ alors :

envoie y-message (yMC, t) avec $yMC = Zd*,MC(yd*)$ au parent

receveur = $\{r \mid r \in D, d* \in Ir, Zd*,r(yd*) \neq 0\}$

pour chaque r dans *receveur* :

envoie x-message (xr, t) avec $xr = Zd*,r(yd*)$ à r

Algorithme 2.3 : Algorithme du coordinateur *"Root"*.

Variables :

t //*temps de*

fils //*subordonné*

t_n // temps du prochain événement du fils envoie

uni-message (i,t) au fils

boucle jusqu'à la fin de la simulation :

envoie un *-message(*,t) au

$t = t_n$

2.3.4 Limites de DEVS classique

Dans un schéma général comportant un ensemble de modèles DEVS, une transition interne pourrait survenir à une même date dans plusieurs de ces modèles. Cependant, dans DEVS classique, en cas d'*événements simultanés* le lien de causalité entre modèles peut être brisé. Ce cas de figure peut s'expliquer aisément de la manière suivante : Si un modèle est influencé par plusieurs autres, la fonction 'select' qui interdit la simultanéité des évènements aura pour effet que les événements de sortie des influenceurs ne seront pas reçus par l'influencé à la même date. De même, un événement externe peut apparaître sur les ports d'entrées d'un modèle à la même date que celle prévue pour la transition interne. La prise en

compte de ce conflit est fortement dépendante du modèle considéré. Pour pallier ces limites, le formalisme DEVS parallèle (Chow et Zeigler, 1994).

2.4 DEVS parallèle

Le formalisme DEVS parallèle (*Parallel Discrete EVent system specification* PDEVS), proposée par [Chow et Zeigler, 1994], est un formalisme qui reprend les principes de base de DEVS classique. Ce formalisme étend DEVS classique à la prise en compte de conflits entre les fonctions de transitions internes et externes. PDEVS donne la possibilité au modélisateur de spécifier une fonction de conflit pour traiter les transitions internes et externes simultanées. De plus, afin de pallier les problèmes liés aux *événements simultanés* dans la version classique, on dispose dans PDEVS de *bags* d'événements destinés à la fonction de transition externe. Ces *bags* permettent de collecter les événements émis à une même date. Ainsi, avec PDEVS, plusieurs événements externes peuvent être réalisés à la même date. Dans la pratique, Ces événements sont collectés et stockés comme des ensembles d'événements survenus à la même date. Les sorties réalisées par les modèles concernés sont stockées dans des *bags* d'entrées notés I^b. Chacun des événements du bag est identifié par sa date d'occurrence. Ainsi, aucune relation d'ordre n'est préconisée pour les événements appartenant à un même bag. On peut ainsi autoriser et dénombrer les évènements simultanés sur chaque port d'entrée. La prise en compte des événements du bag n'est autorisée qu'après les transitions internes de tous les modèles concernés. De ce fait, les transitions externes sont réalisées par des *bags* représentant ainsi la réponse agrégée des événements simultanés.

2.4.1 Spécification formelle d'un modèle atomique DEVS parallèle

Le modèle DEVS atomique parallèle M est décrit par un tuple

$$M = <X, Y, S, \tau, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda> \qquad (2.3)$$

X est l'ensemble des ports x et des valeurs d'entrées, Y est l'ensemble des ports y et des valeurs de sorties, S est l'ensemble des états partiels du système, $\tau : S \rightarrow R0^+$ est la fonction d'avancement du temps, $\delta_{int} : S \times S$ est la fonction de transition interne,

$\delta_{ext} : Q \times X^b \rightarrow S$ est la fonction de transition externe, X^b est l'ensemble des *bags* des entrées X,

Q est l'ensemble des états totaux, $Q = \{(s, e) | s \in S, (0 \leq$

$e \leq \tau(s)\}$,

e est le temps écoulé depuis la dernière transition, $\delta con : S \times X^b \rightarrow S$ est la fonction de conflit,

$\lambda : S \rightarrow Y^b$ est la fonction de sortie

De même manière que pour la DEVS classique, en l'absence d'événements sur les ports d'entrées, le système conserve un état passif jusqu'à la prochaine transition interne. Une sortie est alors réalisée suivie de la transition interne. Si un événement externe apparait sur l'un des ports d'entrée avant la date prévue pour la transition interne, l'état du système passe à $\delta ext\ (s,\ e,\ x^b)$. Dans PDEVS, à la différence de DEVS classique, la transition externe se base sur les *bags* d'événements provenant d'un ou de plusieurs modèles. Si un événement apparait sur X^b à $e = \tau(s)$, le système passe dans l'état $\delta con\ (s,\ e,\ x^b)$. Le comportement de la fonction de conflit est défini par le modélisateur. Par défaut $\delta con = \delta ext\ (\delta int(s,\ e),\ 0,\ x^b))$, donnant ainsi la priorité à la transition interne lors d'un conflit.

2.4.2 Spécification formelle d'un modèle couplé PDEVS

Un modèle couplé dans l'extension DEVS parallèle a une structure quasiment identique à celle de laversion classique. Les modèles atomiques sont tous parallèles.
Un modèle DEVS couplé M est décrit par un tuple

$$M = \langle X, Y, D, \{M_n\}, \{I_n\}, \{Z_{n,n'}\}\rangle \qquad (2.4)$$

où
X est l'ensemble des ports et des valeurs d'entrées,
Y estl'ensemble des ports et des valeurs de sorties,
D est l'ensemble des identifiants des modèles constituant le modèle couplé M, y compris

l'identifiant « self » de M lui même,

M_n est un modèle DEVS (atomique/couplé) constituant du réseau hiérarchique et indexé par $n \in D$, I_n est l'ensemble des modèles qui influencent le modèle n,

$Z_{n,n'}$ est une famille de fonctions de transfert telles que : $Z_{n,n'} : Y_n \rightarrow X_{n'}$ si n, n' ∈ D et n ∈ $I_{n'}$,

$Z_{self,k} : X \rightarrow X_k$ si k ∈ D et self ∈ I_k, $Z_{k,self} : Y_k \rightarrow Y$ si k ∈ D et k ∈ I_{self} ,

Une conséquence directe de la suppression de la fonction de sélection est la prise en compte des modèles concernés. En effet, tous les modèles concernés sont autorisés à effectuer une sortie.

2.5 Cell-DEVS

Une autre extension de DEVS classique est Cell-DEVS qui est née de la constatation que de nombreux modèles font intervenir des espaces discrets et utilisent des formalismes tels que les automates cellulaires. Cette extension a été développée par Wainer et Giambiasi dans [Wainer et Giambiasi, 2001a]. Le but de cette extension est de pouvoir décrire et simuler des modèles à base d'automates cellulaires multidimensionnels et à événements discrets. La dynamique des cellules est temporisée dans le sens où l'état d'une cellule sera modifié en fonction de l'état de son voisinage mais cet état ne sera connu des cellules voisines qu'après un certain délai. Cell-DEVS fournit un mécanisme simple de définition de la synchronisation des cellules. L'extension du formalisme se résume à l'ajout de variables supplémentaires, de leurs sémantiques et d'un simulateur abstrait.

Un modèle Cell-DEVS est basé sur la même structure de base qu'un modèle DEVS classique. On retrouve les modèles atomiques pour les cellules, les éléments de base d'un réseau et les modèles couplés pour les réseaux eux-mêmes. Néanmoins, la structure proposée par Zeigler [Zeigler *et al.,* 2000b] se voit augmentée d'attributs.

Le modèle DEVS d'une cellule est définie par la structure suivante :

$$TDC = <X, Y, I, S, N, delai, d, \delta_{int}, \delta_{ext}, \tau, \lambda, t_a> \qquad (2.5)$$

où :

– *X* est l'ensemble des ports et des valeurs d'entrée ;
– *X* est l'ensemble des ports et des valeurs de sortie) ;

– *I* l'interface de la cellule ;
– *S* l'ensemble des états de la cellule ;
– *N* l'ensemble des états des cellules voisines ;
– *delai* le type de délai utilisé ;
– *d* la durée du délai ;
– δ_{ext}et δ_{int}la fonction de transition externe et interne ;

– τla fonction locale de calcul ;
– λla fonction de sortie ;
– *ta*la fonction d'avancement du temps.

Le voisinage d'une cellule ainsi que ses connexions en termes de ports sont déterminés par son interface *I.*Il y a autant de ports d'entrée que de voisins. La fonction τ modélise la fonction de calcul de l'état de la cellule en fonction de l'état de son voisinage. Cet état d'une cellule n'est effectif (stable) pour les cellules voisines qu'au bout du délai d'attente *d*. La figure 2.4 représente ce fonctionnement.

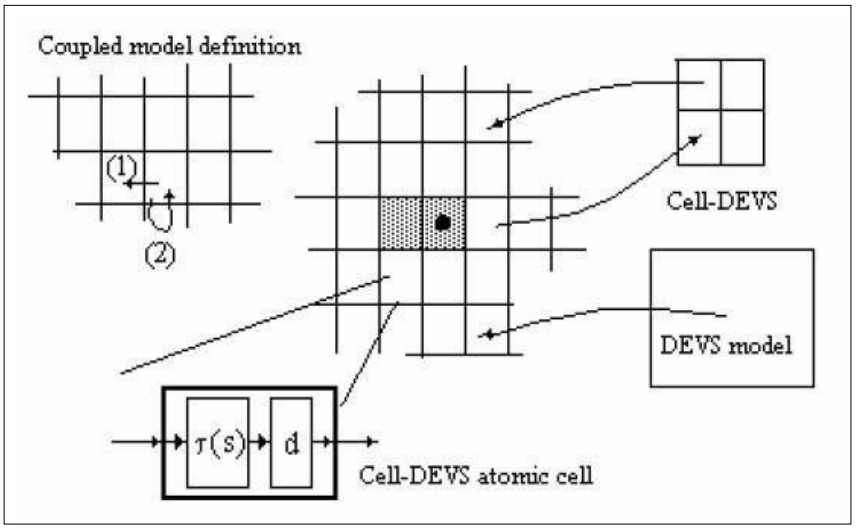


Figure 2.4 : Description d'un modèle Cell-DEVS [Wainer, 2000]

La définition d'un modèle couplé complète la définition des cellules. Ainsi, Le modèle d'une cellule détermine son interface vue de l'extérieur mais ne précise pas la forme des connexions. C'est le modèle couplé qui définit les connexions entre les cellules :

$$GCC = <Xlist, Ylist, I, X, Y, n, \{t1, ..., tn\}, N, C, B, Z> \quad (2.6)$$

Dans cette équation, on a *Xlist* et *Ylist* qui déterminent les listes des modèles (cellules) du réseau qui possèdent des ports d'entrée et des ports de sortie non connectées en interne et, par conséquent, disponibles pour une connexion avec un autre modèle. Dans DEVS, c'est cet ensemble qui détermine toutes les connexions entre les ports d'entrée et les ports de sortie avec les ports du modèle couplé.

I est l'interface *I* du réseau qui réunit au niveau d'une même structure les éléments

de définition de l'interface du réseau vers l'extérieur.

Z met en relation les ports de sortie qui sont des éléments de *Ylist* et les ports d'entrée de *Xlist* d'un autre réseau. *X* et *Y* : ensemble des événements d'entrée et de sortie.

Pour simplifier la définition des connexions entre les cellules du réseau (communément notées connexions internes dans les modèles couplés), un «*pattern*» de voisinage, noté *N*, est défini. Ce

«*pattern*» spécifie pour toute cellule n'appartenant pas à la bordure *B* la position relative de ses voisins.

Cell-DEVS est un outil puissant utilisant DEVS dans le domaine des automates cellulaires. Cell- DEVS offre aussi des simulateurs abstraits afin de préciser le comportement d'un modèle.

2.6 Formalismes DEVS pour la Modélisation approximative

2.6.1 Fuzzy-DEVS

Le formalisme fuzzy-DEVS introduit par Y. Kwon dans [Kwon *et al.,* 1996] dérive du formalisme DEVS classique et en conserve la sémantique, les concepts et la modularité. Il est basé sur la logique Floue et sur la règle "Max-Min". Ce formalisme permet la modélisation et l'analyse de systèmes complexes à évènement discret avec une structure et un comportement partiellement inconnus.

Dans Fuzzy-DEVS, indique pour chaque état où le système à analyser se trouve, l'ensemble des évolutions possibles du système (le modèle garde la trace des évolutions possibles) [Anglani *et al.,* 2000a]. La structure du modèle de base "modèle atomique flou" est :

$$A\,MF = <X, Y, S, \delta int, \delta ext, \lambda, \tilde{t}a> \qquad (2.7)$$

Avec :

- *X, Y* : L'ensemble des évènements d'entrées sorties ;
- *S* : l'ensemble des états séquentiels ;
- $\delta int : S \times S \rightarrow [0, 1]$: fonction floue de transition interne ;
- $\delta ext: Q \times X \times S \rightarrow [0, 1]$: 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 defuzzyfication de $ta(s)$;

- $\lambda : S \times Y \rightarrow [0, 1]$: fonction floue de sortie ;
- $ta: S \times \tilde{N} \rightarrow [0, 1]$: fonction floue d'avancement du temps, où .*N* est l'ensemble des nombres flous appartenant à $R[0, \infty[$.

fuzzy-DEVS étend les fonctions caractéristiques de DEVS aux ensembles flous.

(1) Les fonctions floues de transition interne et externe (δint , δext) représentent les possibilités de transition entre chaque état (passage de st à st+1), elles sont exécutées pour δint lorsque le temps (ta) est écoulé et pour δext lorsque un évènement externe arrive avant que le temps (ta) ne soit écoulé.

(2) La fonction floue de sortie λ génère les valeurs possibles de sortie Y_t, d'un état s_t.

(3) La fonction floue d'avancement du temps peut être représentée par des variables linguistiques (on introduit un flou au niveau de la date de l'évènement)

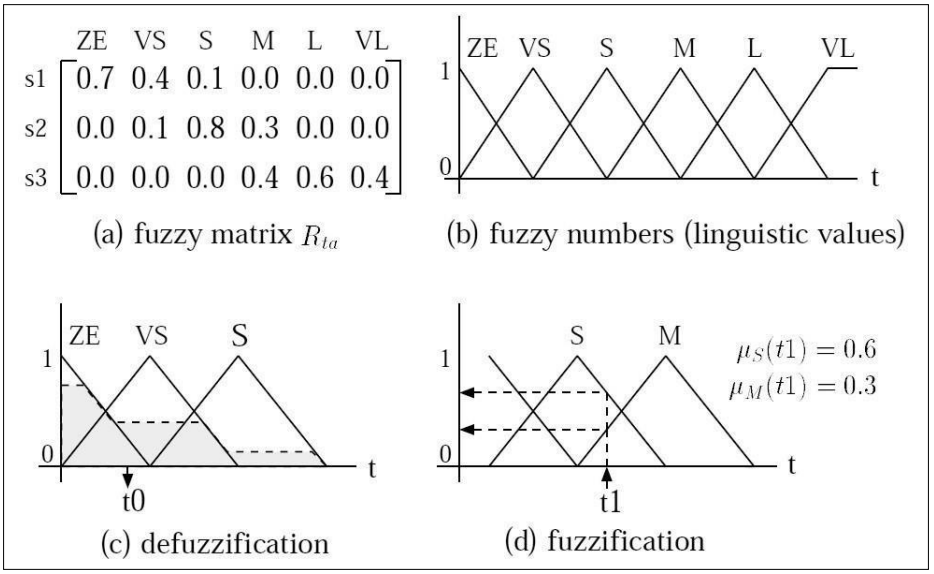


Figure 2.5 : Exemple de données fuzzy-DEVS [Kwon *et al.*, 1996]

L'examen de la figure (2.5, a) fait ressortir les valeurs d'avancement dans le temps des états s1, s2 et s3 sous forme matricielle. On associe à chaque valeur, dans le temps, un terme linguistique (figure 3.5, b) (ZE : Zero, VS : Very Small, S : Small, M: Medium, L : Large, VL : Very Large). Si l'on prend l'exemple de la probabilité qu'à l'état s3, on ait la valeur 'L'est 0,6. Afin d'obtenir des valeurs réelles telles que représentées sur la figure 3.5, c ; on utilise la méthodede "defuzzication" et inversement la méthode de "fuzzification" nous permet d'obtenir valeurs floues (figure 2.5,d).

Contrairement au modèle atomique DEVS, Le modèle *A MF*, est non déterministe, en ce sens qu'il nerépond pas aux deux conditions suivantes :

(1) A l'expiration du temps, la fonction de transition interne est exécutée (δ_{int} (s_t) = s_t+1). De même ;lorsqu'un évènement externe arrive avant que n'a pas expiré ; la fonction de transition externe (δ_{ext} (s_t,X_t)

= s_t+1) est exécutée.

(2) Quand la durée de vie d'un état est finie ; la fonction de sortie (λ (s_t) = Y_t) est exécutée.

La règle "Max-Min" détermine, dans fuzzy-DEVS l'état suivant s_t+1 et non avec δ_{int} et δ_{ext} [Kwon

et al., 1996].

On distingue deux types de modèles dans Fuzzy-DEVS tout comme dans DEVS-classique, un modèle atomique et un modèle couplé, ce dernier ayant une structure identique à celle du modèle couplé DEVS :

$$CM = <X, Y, M, EIC, EOC, IC, SELECT> \quad (2.8)$$

Les différentes possibilités d'entrées, de sorties et changements d'état dans fuzzy-DEVS sont représentées par des matrices. Des arbres de probabilités, représentent les évolutions des modèles [Kwon *et al.,* 1996 ; Anglani *et al.,* 2000a]. Les arbres de probabilités génèrent toutes les trajectoires possibles, les algorithmes qui les mettent en œuvre ne sont pas toujours très performants et constituent des domaines de recherche très actifs. Ce formalisme est appliqué dans plusieurs domaines tels que : analyses de marché(prévision de prix ou de cours), aide à la décision [Anglani *et al.,* 2000a, Anglani *et al.,* 2000b]. Cependant, cette approche qui utilise la règle "Max-Min", les fonctions "fuzzyfication" et "defuzzification", ne semble pas, pour beaucoup de chercheurs totalement cohérente avec le formalisme DEVS. D'autant plus qu'un modèle fuzzy-DEVS ne possède pas la propriété de fermeture sous composition du formalisme DEVS classique. Pour exécuter une simulation, les paramètres flous doivent être remplacés par des paramètres réels (defuzzyfication). Pour effectuer la procédure inverse, on procède par la méthode de fuzzyfication.

2.6.2 Min-Max-DEVS

Le formalisme Min-Max-DEVS [Giambiasi et Ghosh, 2001 ; Hamri *et al.,* 2006] introduit par N. Giambiasi est une suite des travaux de M.Smaili [Smaili, 1994] concernant la modélisation et la simulation de circuits logiques à retard flou. Le but poursuivi par Min-Max-DEVS est la modélisation et la simulation des systèmes réels pour lesquels les valeurs

des retards ne sont pas connues avec exactitude. Ce formalisme est applicable dans les systèmes pour 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, la seule différence réside dans la représentation de la fonction d'avancement du temps *ta*. Dans Min- Max-DEVS elle est définie comme suit :

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

– $ta(si) = (dmin, dmax)$; Où *dmin* est le temps minimum pendant lequel le modèle reste dans le même état *si*, *dmin* représente l'évolution la plus rapide du système ; *dmax* est le temps maximum pendant lequel le modèle reste dans le même état *si*, *dmax* représente l'évolution la plus lente du système.

On distingue deux modifications importantes relativement au formalisme DEVS classique.

Le premier concerne l'évolution du modèle ; dans Min-Max-DEVS, les évènements externes sont choisis en fonction du temps minimum ; en l'absence d'évènement externe, un évènement interne est déclenché au temps maximum. Le second changement concerne les algorithmes de simulation, la fonction *ta* ayant été modifiée, il y'a lieu d'en tenir compte au niveau de la simulation. La fonction *ta* fixe tous les temps correspondants aux évènements internes. Un évènement est déclenché en fonction du temps. Ainsi, deux nouvelles variables de temps ont été définies : *ts* représente le temps maximum avant le prochain évènement et *tf* représente le temps minimum avant le prochain évènement. Les algorithmes de simulation ont été modifiés pour prendre en compte ces évolutions. [Giambiasi et Ghosh, 2001].

Min-Max-DEVS présente des pistes intéressantes pour la représentation et la prise en compte des imprécisions au niveau du temps de vie de l'état *ta*. . Cependant, ceci est applicable essentiellement dans un nombre restreint de domaines tels que la détection de fautes ou la prise en compte de retards flous dans les circuits numériques.

2.7 Real-Time DEVS (RTDEVS)

Le formalisme en temps réel de DEVS (RTDEVS) est une *extension* de formalisme classique de DEVS seulement dans les modèles atomiques de DEVS. Le formalisme RTDEVS pour les modèles couplés reste le même comme l'original sauf qu'un modèle couplé de RTDEVS n'a aucune spécification. C'est parce qu'une horloge de simulation dans

RTDEVS n'est plus une horloge virtuelle mais une horloge en temps réel, qui n'est pas commandée par un algorithme de simulation. Un modèle atomique de RTDEVS, RTAM, est défini comme suit :

La structure d'un RTDEVS atomique est donnée par :

$RTAM = <X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta, ti, \psi, A>$ *(2.9)*

X est l'ensemble d'événements d'entrée Y est l'ensemble d'événements de sortie S est l'ensemble d'états séquentiels

$\delta_{ext} : Q \times X \rightarrow S$ est la fonction de transition externe d'état
$\delta_{int} : S \rightarrow S$ est la fonction de transition interne d'état$\lambda: S \rightarrow Y$ est la fonction de sortie
$ta : S \rightarrow R0^{+}U\infty$est la fonction d'avancement de temps

$ti : S \rightarrow R^{+}0,\infty UR^{+}0,\infty$est la fonction d'avancement d'intervalle du temps

ψ : s → A est la fonction de mappage de l'activité A : est l'ensemble d'activités avec des contraintes

Dans le formalisme DEVS classique, le temps de simulation avance seulement lorsqu'un simulateur appelle la fonction d'avancement de temps 'ta' du modèle associé. La fonction d'avancement de temps 'ta' dans le formalisme de RTDEVS se comporte la même que cela dans le formalisme DEVS classique sauf que la fonction calcule l'événement suivant des nombres entiers, qui est un nombre réeldans le formalisme DEVS classique. Le temps calculé par la fonction d'avancement de temps aussi synchronisée avec le temps de l'horloge murale.

2.8 Formalismes DEVS à structures dynamiques

2.8.1 Le formalise DS-DEVS

Lorsque l'on s'intéresse à l'étude des systèmes complexes, ces derniers sont généralement décomposés en sous-systèmes plus simples à manipuler. Jusqu'ici, nous avons présenté les formalismes DEVS pour les systèmes dont la structure est définie par une composition hiérarchique fixe de modèles atomiques et/ou couplés. Or, parmi les reproches faits à ces

formalismes c'est justement leurs incapacités à modifier dynamiquement leurs structures en cours de simulation. En effet, le comportement d'un modèle couplé est déterminé par celui des modèles atomiques le constituant. La structure du système est statique et est indépendante de son comportement. Cette caractéristique est très limitante notamment pour les applications du monde réel.

Plusieurs approches [Barros, 1995; Uhrmacher, 2001] ont été proposées afin de dépasser cette limite. [Barros, 1995] propose le formalisme *Dynamic Structure DEVS* (DS-DEVS [Barros,

1995, 1996, 1997, 2003] basé sur DEVS classique et qui fournit des mécanismes pour le changement dynamique de la *structure* d'un modèle DEVS. Cette spécification n'effectue aucune modification de spécification des modèles atomiques. Un modèle couplé particulier appelé *exécutive* est prévu dans le formalisme DS-DEVS.A chaque état du modèle couplé *exécutive* est associé à une structure du modèle couplé dont la dynamique dépend des transitions internes et externes de l'exécutive.

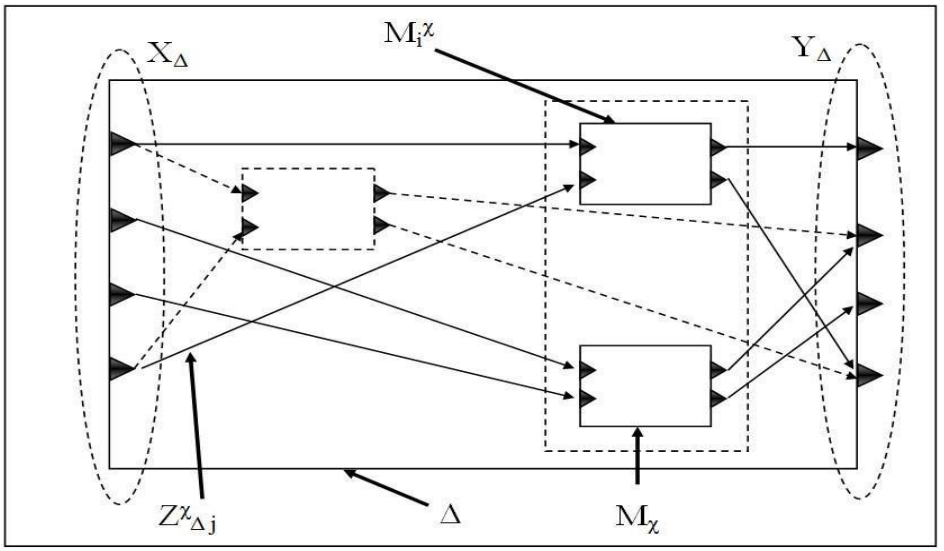


Figure 2.6: Représentation graphique d'un modèle DS-DEVS.

Le changement dynamique de structures se base sur des réseaux de modèles DEVS atomiques. Ainsi, à la différence des modèles couplés DEVS, les listes de connexions et de modèles du réseau peuvent changer au cours du temps. Le réseau à structure dynamique *DSDEVN* est défini par un tuple

$DSDEVN = <X_{\Delta}, Y_{\Delta}, \chi, M_{\chi}>$ *(2.10)*

où

Δ est l'ensemble des modèles atomiques activables, χ est le nom de l'exécutive et $M\chi$ le modèle associé à l'exécutive χ. Ce couple détermine la manière dont la structure change au cours du temps. Le modèle $M\chi$ est défini par un tuple

$$M\chi = <X\chi, Y\chi, S\chi, s^0\chi, \tau\chi, \gamma, \Sigma^*, \delta int,\chi, \delta ext,\chi, \lambda\chi> \qquad (2.11)$$

où

$X\chi$ est l'ensemble des ports et des valeurs d'entrées, $Y\chi$ est l'ensemble des ports et des valeurs de sorties, $S\chi$ est l'ensemble des états partiels du système,

$s^0\chi$ est l'état partiel initial du système. Cet état se compose de :

- $D\chi$: l'ensemble des composants (nom des modèles composants le modèle dynamique) actifs
- M^{χ}_i : modèle du ième composant actif
- I^{χ}_i : l'ensemble des composants sous l'influence du ième composant actif
- $Z^{\chi}_{i,j}$: fonction de traduction entre les ports du modèle du composant i et ceux du modèledu composant j
- Select : fonction de gestion des conflits entre
- composants θ^{χ} : ensemble de variables d'état du

modèle global

$\tau\chi : S\chi \rightarrow R^+0$ est la fonction d'avancement du

temps,$\gamma : Q\chi \rightarrow \Sigma^*$ est la fonction de structure,

Σ^* est l'ensemble des structures,

$\delta int,\chi : S\chi \times S\chi$ est la fonction de transition interne,

$\delta ext,\chi : Q\chi \times X\chi \rightarrow S\chi$ est la fonction de transition externe où :$Q\chi$ est l'ensemble des états totaux,
$Q\chi = \{(s\alpha,\chi, e)|s\alpha,\chi\ 2\ S\chi, 0 \leq e \leq \tau(s\alpha,\chi)\}$

e est le temps écoulé depuis la dernière transition,

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

Nous remarquons qu'à l'exception de Σ* et γ, cette structure est assez proche de celle des modèles atomiques DEVS classiques. L'état partiel $s^{\alpha}\chi \in S\chi$ contient des informations relatives

au comportement et à la structure du réseau de modèles. Si l'on considère un état partiel, $s^{\alpha}\chi$,

la structure du réseau de modèles $\Sigma^{\alpha} \in \Sigma^*$ est définie par $\Sigma^{\alpha} = \gamma\ (s^{\alpha}) = \chi$ ($X^{\alpha}self$, $Y^{\alpha}self$, D^{α},

$\{M^{\alpha}d/d \in D\}$, $(EIC)^{\alpha}$, $(EOC)^{\alpha}$, $(IC)^{\alpha}$), ces paramètres sont les mêmes que ceux décrits dans la définition du modèle DEVS couplé . Le réseau est défini par une composition hiérarchique de modèles DEVS atomiques et couplés. Tout changement provoqué par l'appel de $\gamma(s^{\alpha}\chi)$ se traduit par l'ajout ou la suppression de modèles et le changement de connexions entre modèles. Par exemple, la valeur X^{α}définit l'ensemble de ports d'entrées actifs. Cet ensemble est un sous- ensemble de $X\chi$. S'il est modifié, cela signifie que les événements admissibles par le modèle changent.

2.8.2 Le formalisme DSDE

Le formalisme DSDE pour *parallel Dynamic Structure Discrete Event system specification* [Barros, 1997, 1998] est une version parallèle de DS-DEVS. Comme dans *PDEVS*, à chaque modèle est associée une fonction de conflit capable de prendre en compte des transitions interne et externe simultanées. Ce formalisme intègre également la notion de *bags* d'événements permettant ainsi de mieux appréhender la collecte et la gestion des événements qui sont émis à une même date. DSDE peut également piloter des modèles atomiquesPDEVS.

2.9 Plateformes et environnements basée DEVS

2.9.1 JDEVS

JDEVS est un logiciel développé par le laboratoire Systèmes Physiques de l'Environnement de l'Université de Corse, principalement, par J. B. Filipi [Filippi et Bisgambiglia, 2003]. Il est issu d'un travail de thèse portant sur le développement d'une plate-forme basée sur un environnement de modélisation et de simulation de systèmes naturels complexes. JDEVS propose un ensemble de solutions pour la communication mettant en jeu des données fournies par des composants externes comme les systèmes d'informations géographiques (SIG). Les modèles intégrés et proposés par la plate-forme sont les automates cellulaires et les réseaux de neurones [Filippi *et al.*, 2002b].

2.9.2 ADEVS

Adevs (*A Discrete EVent System simulator)*, a été développé par J. Nutaro [Nutaro, 2003] à l'Université d'Arizona. C'est une bibliothèque dédiée à la construction de simulations basées sur DEVS parallèle et *Dynamic* DEVS (dynDEVS). Adevs constitue une extension pour le support de la gestion de structures dynamiques de modèles basée sur *Dynamic DEVS* [Uhrmacher, 2001].

2.9.3 DEVS/C++

Cette bibliothèque constitue l'une des premières plates-formes basées sur le formalisme DEVS. Elle est considérée comme une implémentation efficace des simulateurs abstraits et propose différents mécanismes d'optimisation [Zeigler *et al.,* 1996]. Elle a été développée à l'Université d'Arizona par Hyup.

J. Cho et Young K. Cho avec la possibilité d'utiliser les extensions de structures dynamiques et d'automates cellulaires en laissant la possibilité d'être distribuable sur plusieurs calculateurs. Plusieurs versions existent de DEVS/C++.

2.9.4 CD++

CD++ est un ensemble de bibliothèques développé en 2002 par G.Wainer [Wainer, 2002] et les Universités de Carleton, au Canada et Buenos Aires, en Argentine. CD++ permet la définition de modèles DEVS et Cell-DEVS, par l'utilisation d'un langage de spécification de haut niveau par la manipulation de graphes d'états. Certaines versions de CD++ incorporent

les extensions DEVS parallèle et DEVS temps réel [Glinsky et Wainer, 2002]. Dans CD++, les modèles peuvent se baser sur les réseaux de Petri, les automates cellulaires, ou les graphes d'états DEVS. Le logiciel DEVSView permet aux utilisateurs de créer des visualisations à partir des sorties de CD++ avecdes rendus des états et des événements sous forme d'animation.

2.9.5 ATOM^3

Atom^3 (A TOol for Multi-formalism and Meta-Modeling) est une plate-forme de modélisation et de simulation développée par Juan De Lara à l'Université Autonome de Madrid(UAM) et Hans Vangheluwe à l'Université Mc. Gill au Canada, au laboratoire Modelling, Simulation and Designa. Dans Atom^3, les modèles et les formalismes, sont décrits sous forme de graphes [de Lara et Vangheluwe, 2002]. À partir d'une description des

formalismes, Atom^3 propose des outils de manipulation des modèles décrits dans ces formalismes. Les méta-modèles fournis sont, entre autres, les réseaux de Petri, les automates à états finis, les diagrammes de flux… Les modèles peuvent alors être traduits vers le formalisme DEVS automatiquement par des routines de traduction [Vangheluwe, 2000]. DEVS est alors utilisé comme formalisme unificateur en transformant les modèles sous forme DEVS. Le simulateur DEVS attaché est écrit en Python et se nomme PyDEVS (pour Python DEVS).

2.9.6 DEVS/JAVA

DEVS/Java a été développé à l'Université de l'Arizona [Sarjoughian et Zeigler, 1998], il en existe plusieurs versions. DEVS/Java supporte plusieurs extensions de DEVS et permet l'extension de la modification de la structure des modèles atomiques et couplés tels que l'ajout d'autres modèles couplés ou non et leur suppression. Les modifications des ports de ces modèles peuvent aussi être modifiés. D'autres extensions sont également incluses comme l'intégration de systèmes d'équations différentielles et des automates cellulaires. Cette plateforme permet le développement collaboratif par composition de manière synchrone ou asynchrone. Cette plate-forme propose également le support de la norme High Level Architecture, HLA, pour la distribution de simulation [Zeigler *et al.,* 1999].

2.9.7 MOOSE

MOOSE (Multi-model Object Oriented Simulation Environment) [Fishwick, 1998] est un autre environnement de modélisation et de simulation qui a été développée à l'université de Floride. MOOSE comprend une interface graphique, une bibliothèque de modèles et un moteur de modélisation et de simulation. Un aspect important de l'environnement MOOSE concerne le fait qu'il supporte l'intégration de plusieurs types de modèles tels que les modèles en diagrammes de blocs, les modèles à équation sous contraintes et les modèles à base de règles. Ces modèles peuvent être connectés de manière récursive et hiérarchique.

2.9.8 ECLPSS

Eclpss (Ecological Component Library for Parallel Spatial Simulation) [Woodbury *et al.,* 2002] est un environnement basé sur le langage Java. Cet environnement est orienté vers la simulation d'écosystèmes prenant en considération de nombreuses échelles de temps et d'espace. Trois entités composent les modèles Eclpss: un ensemble de variables, un ensemble de composants avec état modifiable et un ensemble de simulateurs pour ces composants. Les composants Eclpss sont développés en code java et stockés dans une bibliothèque pour réutilisation. L'intérêt principal de cet environnement est la possibilité

d'évaluer des modèles sur machines parallèles.

2.9.9 Small DEVS

SmallDEVS est un environnement expérimental de simulation basé sur le formalisme DEVS développé par Vladimir Janousek et ElodKironsky à l'université de Brno de Technologie (République de Tchèque). Il est orienté objet basé sur les prototypes et tient compte de l'aspect classe d'objet, aussi bien que la construction de modèles orienté objet. Ses composantes offrent la manipulation interactive avec des modèles et des simulations possibles. Cet outil supporte bien la modélisation et la simulation interactives.

2.10 Conclusion

Nous avons présenté dans ce chapitre le formalisme DEVS développé par B.P. Zeigler ainsi que des extensions DEVS qui permettent de prendre en compte parallélisme, imprécision et incertitude sur les événements ou sur les états d'une part. Où si l'incertitude intervient au niveau des changements d'états, alors que l'imprécision intervient sur le temps ou les variables des événements. Dans les deux cas, ces changements modifient la structure des modèles et les algorithmes de simulation. Et la dynamique de structures en cours de simulation d'autre part, ensuite nous avons brossé une vue d'ensemble de plateformes de ces formalismes.

Chapitre 03 : Procédé de compression la réfrigération de propane

3.1 Introduction

Un procédé industriel est un procédé de nature mécanique ou chimique destiné à synthétiserdes produits chimiques, en garde quantité et dans des conditions acceptables.

Dans ce chapitre, nous allons présenter les concepts de base qui constitue presque tous les procédés industriels, ensuite nous allons étudier un système industriel de compression de la réfrigération de propane en décrivant ses boucles de régulation et en lui proposant un modèle basé le formalisme DEVS pour sa simulation.

3.2 Les procédés industriels

Un procédé industriel est un procédé de nature mécanique ou chimique destiné à fabriquer ouà synthétiser des produits dans des conditions techniquement et économiquement acceptables, de façon sécuritaire et dans le respect de l'environnement. Ce profil de compétences inclut la conception et la modification de procédés industriels de transformation et de conditionnement ainsi que le soutien à l'exploitation.

Il existe plusieurs types de procédés industriels et ils sont :

- Procédés de fabrication continus et discontinus.
- Procédés mono variable et multi variable.
- Procédés stables et instables.

3.2.1 Paramètres caractéristiques de la réponse d'un procédé

3.2.1.1 Caractéristiques dynamiques d'un procédé :

- **Temps de réponse** C'est l'aptitude du système à suivre les variations de la grandeur réglant. Dans le cas d'un échelon de la grandeur réglant, la croissance de la grandeur réglée définit les différents temps de réponse.

- **Dépassement** Le premier dépassement permet de qualifier la stabilité d'un système. Plus celui-ci sera important, plus le système sera proche de l'instabilité. Dans certaine régulation, aucun dépassement n'est toléré. Dans la réponse indicielle de la figure précédente le premier dépassement est de 20 %.

3.2.1.2 Caractéristiques statiques d'un procédé :

La caractéristique statique est la courbe représentative de la grandeur de sortie S en fonction dela grandeur d'entrée E : **S = f(E).**

Remarque : On ne peut tracer la caractéristique statique que d'un système stable.

- **Gain statique :** La linéarisation de la caractéristique statique du procédé est réaliséepar la vanne de réglage.

- **Erreur statique** Si le système est stable, l'erreur statique E est la différence entre la consigne W et la mesure de la valeur réglée X.

E = W – X

- **Linéarisation de la caractéristique statique :** La linéarisation de la caractéristique statique du procédé est réalisée par la vanne de réglage.

3.3 La régulation

La régulation de procédé existe depuis plus de 2000 ans alors que les Grecs et les Babyloniens utilisaient ce principe pour réguler les niveaux d'eau dans leur canalisation. La régulation de procédés industriels, quant à elle, est apparue au cours des années 1930 et c'est la compagnie «Taylor Instrument Companies » qui l'utilisa dans la commercialisation du «Fullscope», le premier contrôleur ayant de véritables fonctionnalités proportionnelle, intégrale et dérivée : le contrôleur PID. L'application de la régulation de procédé s'est étendue aujourd'hui aux secteurs manufacturiers: pharmaceutique, agroalimentaire, robotique, électronique et bien plusencore.la majorité des processus industriels nécessitent de contrôler un certain nombre de paramètres: température, pression, niveau, débit, pH, concentration d'oxygène, etc.

Toute chaîne de régulation (ou d'asservissement) comprend trois maillons indispensables : l'organe de mesure, l'organe de régulation et l'organe de contrôle. Donc La régulation des procédés industriels regroupe l'ensemble des moyens matériels et techniques mis en œuvre pourmaintenir une grandeur physique à régler, égale à une valeur désirée, appelé consigne.

Lorsque des perturbations ou des changements de consigne se produisent, la régulation provoque une action correctrice sur une grandeur physique du procédé appelée grandeur réglante.

Dans le cas de la régulation, la consigne est fixée et le système doit compenser l'effet des perturbations, à titre d'exemple, le réglage de la température dans un four, de la pression dans un réacteur, le niveau d'eau dans un réservoir.

Pour le technicien de régulation le terme procédé désigne une partie ou un élément d'une unité de production industrielle ; par exemple un échangeur thermique qui comporte une régulation de température ou un ballon dont le niveau est régulé. Procédé et régulation forment un tout indissociable.

Le choix des éléments de la chaîne de régulation est dicté par les caractéristiques du processusà contrôler, ce qui nécessite de bien connaître le processus en question et son comportement.

3.4 Les régulateurs:

Un régulateur est un mécanisme automatique qui élabore un signal de commande U en fonctionde l'écart de réglage **M-C** selon un algorithme donné **f : U=f((M-C))**

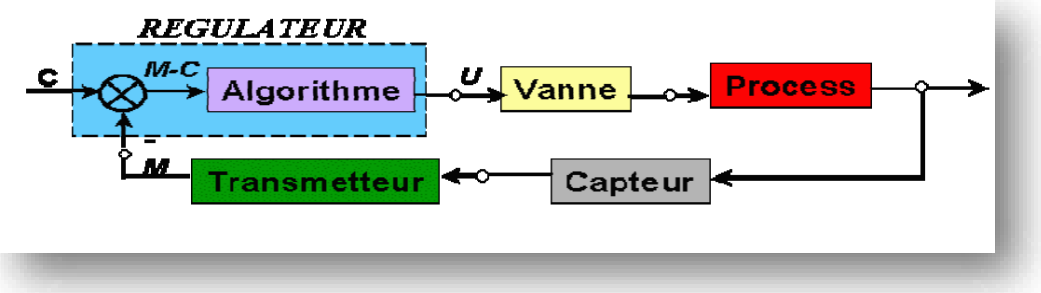


Figure 3.1: le mécanisme automatique de régulation

3.4.1 Classification des régulateurs

Les régulateurs sont classés comme suit :

3.4.1.1 Pneumatique

Le régulateur indicateur pneumatique série 80 est utilisé pour la régulation de la pression ou dela température sur tous types de procédés. L'appareil mesure directement la pression ou la température du fluide, affiche cette valeur sur un cadran, la compare à une consigne et délivre un signal pneumatique de régulation 3-15 psi en standard. Ce signal commande l'appareil final de régulation. Quatre courbes de régulation sont disponibles :

Tout ou Rien, Proportionnel, Proportionnel+Intégral, Proportionnel+Intégral+Dérivé.

- **caractéristiques :**

- Bonne réponse dynamique
- Maintenance réduite
- Faible consommation en air
- Haute fidélité

- Interchangeabilité facile des composants
- Compatible avec tout type de vanne
- Bande proportionnelle ajustable de 00 jusqu'à 200%

3.4.1.2 Numérique :

Un régulateur numérique est une petite unité numérique qui agit comme un contrôleur de système. Selon les besoins en termes de contrôle, un régulateur numérique peut prendre la forme d'un ordinateur de bureau ou d'un microcontrôleur. La transformée de La place est remplacée par une transformée en Z dans le régulateur numérique où un signal de temps discret sous la forme d'une séquence de nombres complexes ou réels est converti en une représentationcomplexe de domaine fréquentiel.

Les régulateurs numériques sont essentiellement utilisés en tant que système de retour d'informations. Un régulateur numérique classique dispose des composants suivants :

- Un convertisseur A/D– Les entrées analogiques sont converties au formatnumérique
- Un convertisseur D/A– Les sorties numériques sont converties en entrées analogiques pour l'appareil.
- Un programme reliant les sorties et les entrées.

➢ **Caractéristiques :**

- Coût peu élevé.
- Un petit microcontrôleur peut être acheté pour seulement 5$.
- Ils offrent la flexibilité de pouvoir être facilement configurés en utilisant un logiciel.
- Vous pouvez modifier les paramètres et adapter les performances selon l'appareil.
- Moins sensibles aux impacts environnementaux que les régulateurs analogiques.

3.4.1.3 Electronique :

Sortie 4-20 mA utilisent des signaux analogiques à base d'amplificateurs opérationnels.
Ces régulateurs utilisent une électronique analogique, à base d'amplificateurs opérationnels, ils cèdent le pas à la technologie numérique, mais sont encor nombreux dans l'industrie.

➢ **Caractéristiques :**
- Très faible encombrement
- Pas d'entretien périodique
- Plus économique qu'une installation avec réservoir hydrophore

- Lutte contre les coups de bélier (pour les régulateurs avec petit réservoir intégré)
- Simplicité d'installation et d'intervention en cas de panne.

3.4.1.4 PID

Le PID reste de loin la technique de régulation la plus répandue. Lorsque sont apparus les régulateurs numériques, on aurait pu penser que le PID allait en prendre un coup, car les microprocesseurs sont capables d'exécuter toutes sortes d'algorithmes et donc de mettre en œuvre des techniques de régulation beaucoup plus sophistiquées que le PID. Le PID avait cependant des arguments à faire valoir : il donnait en général satisfaction et il était bien connu des régleurs. Cela lui a permis de résister et de rester omniprésent, même si certaines autres techniques sont apparues ces derniers temps, notamment pour mettre en œuvre la régulation auto-adaptative.

PID, faut-il le rappeler, est un sigle décrivant les trois actions de base du régulateur :

Proportionnelle, Intégrale et Dérivée. Il est important de connaître les effets de ces troisactions.

3.4.2 Actions des Régulateurs PID

3.4.2.1 Action proportionnelle

La sortie S(t) du régulateur proportionnel s'exprime par la relation :$S(t) = \pm Gr.\varepsilon(t) + S0$Où :

- Gr est le gain du régulateur
- S0 la sortie du régulateur lorsque $\varepsilon(t) = 0$, c'est-à-dire

Lorsque la mesure est égale à la consigne. Le signe "±" présent dans l'équation précédente indique que l'on peut choisir le sens d'action du régulateur, ce sens étant choisi en fonction dela vanne et du processus à commander ; dans le sens direct (signe "+"),la sortie S(t) et lamesure M(t) varient dans le même sens alors que dans le sens inverse (signe "–"), elles varient en sens inverse.

L'action proportionnelle du régulateur s'exprime soit par le gain Gr (on emploie aussi K et Kp), soit par la bande proportionnelle BP (également appelée PB, XP % et P %).

C'est la variation en % de l'entrée du régulateur nécessaire pour que la sortie varie de 100%. Elle est d'autant plus faible que le gain est élevé; on a donc :

BP%=100/Gr.

BP est de l'ordre de 3 à 400% dans les régulateurs électroniques.

3.4.2.2 Action intégrale

La régulation proportionnelle s'avère insuffisante chaque fois que l'on souhaite régler la mesure avec une assez bonne précision. D'autre part, dans le cas de procédés industriels où l'onveut une marge de stabilité assez grande, il faut s'imposer des gains faibles, ce qui a pour conséquence d'engendrer des écarts ε importants. La fonction intégrale, utilisée seule ou associée à la fonction proportionnelle, permet d'éliminer cet écart. L'action intégrale agit proportionnellement à la surface de l'écart, et elle poursuit son action tant que l'écart n'est pas nul. On dit que l'action intégrale donne la précision statique. Malgré tout, elle contribue à accélérer le comportement global de la boucle. Comme dans le cas de l'action proportionnelle, un dosage trop important de l'action intégrale engendre une instabilité de la boucle de régulation. Pour son réglage, il faut là aussi trouver un compromis entre la stabilité et la rapidité.

La sortie du module intégrateur varie proportionnellement à la surface de l'écart :
Dans laquelle Ti est le cœfficient de dosage de l'action intégrale, appelé aussi temps d'action intégrale. Les unités utilisées sont des minutes ou des secondes.

Sur certains régulateurs, le dosage de l'action intégrale est exprimé en répétitions par minutes (n, avec n = 1/Ti).

Dans la pratique, l'action intégrale (I) est associée à l'action proportionnelle (P), car ces deux actions sont complémentaires. L'action intégrale améliore le comportement de la régulation et permet d'obtenir des résultats satisfaisants dans des applications courantes set classiques.

3.4.2.3 L'action dérivée

La régulation PI permet d'obtenir dans la majorité des cas un comportement satisfaisant de la régulation. Malgré tout, pour certaines applications, notamment lorsque les variations de la grandeur contrôlée sont rapides et les temps de réponse de la régulation plutôt longs, il est possible d'améliorer la régulation en ajoutant aux actions PI la fonction dérivée D. Le rôle de l'action dérivée est de compenser en partie le temps mort existant sur le procédé. Lorsqu'elleest correctement dosée, l'action dérivée a également un effet stabilisateur, c'est-à-dire qu'elle procure un amortissement plus rapide du régime transitoire lorsqu'il se produit une perturbationou lors d'une modification de la valeur de la consigne. On montre également par une étude théorique de la stabilité que l'action dérivée produit une légère augmentation de

l'actionproportionnelle, ce qui a pour effet d'obtenir une réponse plus rapide.

Comme pour les deux autres actions (proportionnelle et intégrale), si on dose exagérément l'action dérivée, on peut compromettre la stabilité.

L'action dérivée a pour effet de délivrer une sortie variant proportionnellement à la vitesse de variation del'écart (lorsque l'écart reste constant, la sortie du module dérivée est nulle).

3.5 La régulation des vannes

La vanne de régulation est utilisée comme organe de réglage dans différents types de boucles de régulation. Elle permet de contrôler le débit dans une canalisation, en modifiant les pertesde charges de celle-ci.

3.5.1 Classification des régulateurs

Il existe plusieurs représentations d'une vanne :

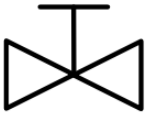
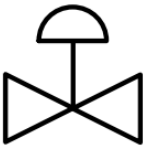
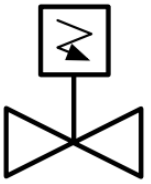
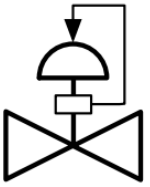
Vanne simple	Vanne manuelle	Vanne Pneumatique	Electrovanne	Vanne pneumatique avec positionneur
				

Figure 3.2: Les types de vanne

Mais dans nos recherches, nous n'avons mentionné que deux types, qui sont :

3.5.1.1 Vanne Pneumatique:

Une vanne pneumatique est un robinet à commande motorisé dont le mécanisme de fermetureet d'ouverture est actionné par un servomoteur à gaz comprimé (habituellement de l'air).

Les vannes pneumatiques entrent dans l'élaboration de systèmes d'automatisation ou à commandes déportées. À ce titre, on les retrouve dans tous les secteurs de l'industrie, de la machine-outil, de la chimie, de la pétrochimie, du médical, de la pharmacie, de l'agroalimentaire, des transports, etc.

Il existe de nombreux systèmes de manœuvres, progressifs ou contrôlés, mais la majorité des vannes pneumatiques fonctionne selon le principe du tout ou rien (fermé ou ouvert).

3.5.1.2 Electrovanne :

Une électrovanne ou électrovanne est une vanne à commande électrique. Grâce à cet organe,il est possible de travailler sur le débit de fluide dans le circuit avec un signal électrique (Par exemple, il y a une électrovanne dans une machine à laver.)

Les électrovannes ont deux états possibles :

- Entièrement ouvertes
- Entièrement fermées

L'état change suivant qu'elles soient alimentées électriquement ou non (sont entièrement fermées en l'absence d'alimentation électrique et qui s'ouvrent lorsqu'elles sont alimentées).

3.5.2 Calcul de Cv

Les formules ci-dessous, modifiées permettent de déterminer directement le coefficient de débit CV à partir des conditions effectives d'utilisation du fluide, exprimées en unités métriques.

Pour déterminer le diamètre nominal d'une vanne (ou bien le diamètre de passage intérieur pour les vannes à passage réduit) on doit calculer le coefficient du débit CV en fonction des conditions effectives du fluide et, en fonction du type de clapet choisi, on détermine le diamètre de passage à l'aide des tableaux des CV correspondants aux diverses versions de vannes.

3.5.2.1 Liquides

La formule suivante est variable pour des liquides qui ne présentent pas de phénomènes de ré vaporisation

$CV=1,17Q\sqrt{(d/\Delta P)}$

- Q : Débit en m3.h-1 ;
- d : masse volumique du liquide en kg.dm-3 à la température de fonctionnement ;
- ΔP : pression différentielle en bar.

3.5.2.2 Vapeur

Premier cas : pression aval absolue supérieure à 50% de la pression absolue d'entrée dans la vanne.

$CV=Q/16(\sqrt{(\Delta P.P1)})$

- Q : Débit en kg.h-1 ;
- ΔP : pression différentielle en bar ;
- P1 : pression absolue de la vapeur à l'entrée de la vanne en bar.

Deuxième cas : pression aval absolue inférieure à 50% de la pression absolue d'entrée dans la vanne.

CV=Q/(10.P1)

- Q : Débit en kg.h-1 ;

P1 : pression absolue de la vapeur à l'entrée de la vanne en bar.

3.5.2.2 Gaz

Premier cas : pression aval absolue supérieure à 50% de la pression absolue d'entrée dans la vanne.

CV= (Q/380)√((d.T)/(ΔP.P2))

- Q : Débit en kg.h-1 ;
- ΔP : pression différentielle en bar ;
- P2 : pression absolue du gaz en aval de la vanne en bar ;
- d : densité du gaz par rapport à l'air ;
- T : température absolue en K.

Deuxième cas : pression aval absolue inférieure à 50% de la pression absolue d'entrée dans la vanne.

(Q/205.P1)√(d.T)

- Q : Débit en kg.h-1 ;
- ΔP : pression différentielle en bar ;
- P1 : pression absolue de la vapeur à l'entrée de la vanne en bar ;
- d : densité du gaz par rapport à l'air ;
- T : température absolue en K.

3.6 Le système industriel à simuler :

Comme cas d'application, nous avons choisi un système industriel qui est constitué par de plusieurs boucles de réfrigération du propane pour un projet du nouveau train de GNL Skikda, Algérie

3.6.1 Description du système à simuler

Le système de compression de la réfrigération de propane de l'unité 16 fournit le refroidissement ou la condensation au système réfrigération MR de l'unité 16, aux refroidisseurs d'alimentation de la colonne d'épuration de l'unité 15 et au pré refroidisseur d'alimentation du sécheur de l'unité 13.

Le propane venant du bac de stockage de propane 76-MF01 (d'une capacité de 40 tonnes) est utilisé pour remplir 40-50 % de l'accumulateur de propane 16-MD06. Le premier remplissage se fait à partir du 76-MF01 ou du stockage existant. L'accumulateur est utilisé alors pour charger le système de réfrigération de propane. Si on veut, on peut utiliser le ballon de transfert de propane16-MD07 ou la pompe de transfert de propane 16-MJ05 pour collecter et transférerle propane liquide vers le bac de stockage ou dans le circuit de réfrigération.

Le compresseur PR 16-MJ04 est entrainé par une turbine à gaz 16-MJ04-GT frame 7. Cette même turbine à gaz entraine aussi le compresseur HP MR qui est accouplé au même arbre. Un moteur auxiliaire 16-MJ04-M de 17 MW est accouplé aussi à cet arbre pour aider le compresseur au cas où la turbine à gaz ne peut pas fournir, toute seule, l'énergie requise par la

charge du procédé GNL. Lors d'une opération normale, tous ces éléments tournent à 3600 rpm sauf pour le moteur de démarrage qui va débrayer quand la charge du moteur d'assistance au démarrage augmente afin d'accélérer la turbine à gaz (GT) d'environ 360 rpm à environ 3400 rpm. Le système de compression de la réfrigération de propane de l'unité 16 fonctionne tel un système fermé de telle façon que le propane est continuellement en circulation en une boucle fermée, comme indiqué sur les PFD's PR-16-PR32-004-1 et 2 et se référer au schéma simplifié de la page suivante.

La vapeur de propane entre dans le compresseur à travers quatre (04) ballons d'aspiration qui fonctionnent à quatre (04) niveaux de pression (0,05 ; 0,87 ; 2,52 et 5,30 bar eff).

La vapeur de propane chaud (après son passage à travers le compresseur) quitte le refoulement du compresseur (à environ 16,8 bar eff et 70 ºC) puis elle est condensée en liquide (à environ 15,9 bar eff) dans le condenseur de propane 16-MC09 avant d'être collectée dans l'accumulateur de propane 16-MD06. Le condenseur de propane est constitué de trente

(30) batteries contenant soixante (60) faisceaux et quatre-vingt-dix (90) moto-ventilateurs.
Le propane liquide est sous-refroidi de 49°C à 31°C dans le sous-refroidisseur de propane 16-MC10, puis il est pressurisé vers chacun des huit (08) refroidisseurs du système selon la demande du procédé.

Le sous-condenseur de propane 16-MC10, est constitué de six (06) batteries contenant douze

(12) faisceaux et dix-huit (18) moto-ventilateurs.

Le niveau de propane liquide dans chaque refroidisseur est contrôlé par un régulateur de niveausur chaque entrée du refroidisseur. Le régulateur de niveau permet au propane liquide (dans le côté calandre de l'échangeur) de maintenir le niveau du liquide au moment de sa

vaporisation par le gaz chaud (dans le côté tube). La vapeur générée dans chaque échangeur est retournée vers l'un des ballons d'aspiration du compresseur pour compléter le circuit.

Au cas où la charge du procédé GNL ne satisfait pas la demande du compresseur en vapeurs, chaque étage possède un régulateur de débit de vapeurs anti-retour (anti pompage) pour maintenir les besoins minimums en débit pour le compresseur et empêcher le pompage du compresseur.

Les régulateurs anti-pompage fonctionnent automatiquement pour empêcher le risque de pompage pour chaque étage du compresseur.

Un refroidisseur de recyclage de compresseur de propane 16-MC11 (en amont des régulateurs anti-pompage) refroidit le gaz de recyclage anti-pompage pendant l'utilisation de ce mode.

Le 16-MC11 est constitué de trois (03) batteries contenant six (06) faisceaux et neuf (09) moto-ventilateurs.

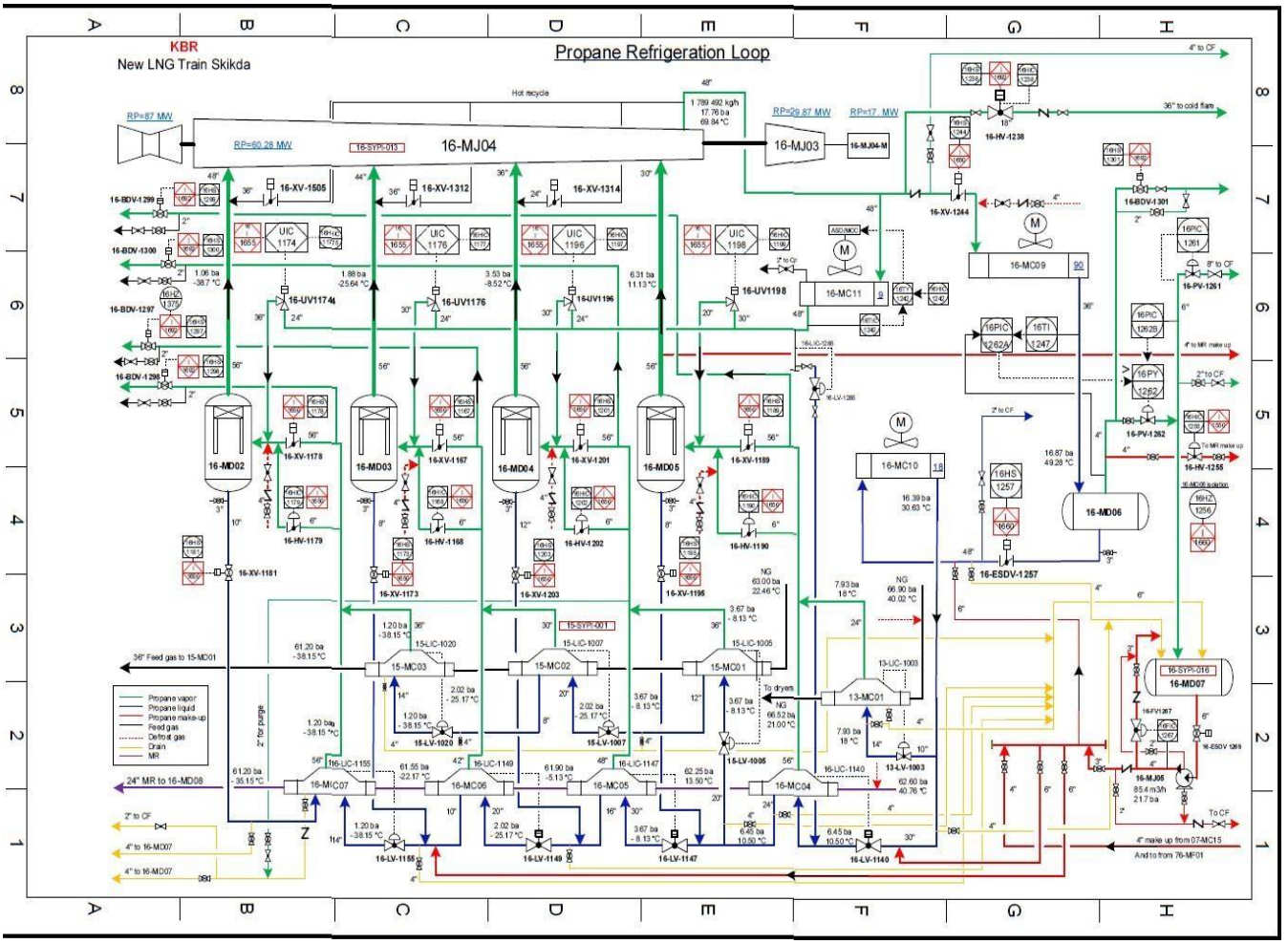


Figure3.3 : Schéma technique de la boucle de réfrigération au propane

3.6.2 Exploitation de système avec de régulateur automatique :

3.6.2.1 Accumulateur de propane MP 16-MD06

Le liquide de propane condensé dans le condenseur de propane 16-MC09 entre dans l'accumulateur de propane 16-MD06. Cet accumulateur fonctionne avec un niveau variable.

Afin de minimiser les vapeurs inertes dans l'accumulateur, le régulateur de pression envoie les vapeurs vers la ligne de vapeurs du ballon de transfert de propane 16-MD07.

Le point de consigne de ce régulateur est calculé dans le DCS (en ce basant sur l'indicateur de température du liquide "sortie" du condenseur 16-TI-1247).

Point de consigne pour le 16-PIC-1262A = 1,013 + 6,9871e (0,0203T).

La pression est en bar eff et la température en ºC. Ce régulateur maintient la pression du ballonde l'accumulateur au-dessus du point d'ébullition du propane condensé.

3.6.2.2 Ballon de transfert de propane 16-MD07

Le régulateur de pression 16-PIC-1261 dégaze l'excès de pression venant du 16-MD07 vers le collecteur torche froide. Le régulateur de pression 16-PIC-1262B maintient une pression d'interagie de propane dans la partie supérieure du ballon en introduisant les vapeurs à partir del'accumulateur de propane 16-MD06. Ce cas peut arriver seulement si la pompe de transfert depropane 16-MJ05 est en marche.

Note : Le point de consigne du 16-PIC-1261 doit toujours être plus grand que le point de consigne du 16-PIC-1262B, ainsi les soupapes 16-PV-1261 et 16-PV-1262 ne sont pas ouvertessimultanément.

Si elles sont toutes les deux ouvertes en même temps, l'inventaire de propane sera perdu vers latorche.

3.6.2.3 Refroidisseur de recyclage du compresseur de propane 16-MC11

La principale fonction du 16-MC11 est de refroidir le refoulement du compresseur quand il est en opération de recyclage et quand plus de quatre (04) vannes anti pompage sont en opération.

Pendant une opération normale, et quand les vannes anti-pompages sont fermées, une petite quantité de vapeurs au refoulement du compresseur entre en contact avec le refroidisseur et se condense. Le pot séparateur sous le 16-MC11 collecte le liquide qui se forme puis l'envoie versle refroidisseur de propane MR HHP 16-MC04, afin de compléter l'apport de réfrigérant de propane venant du sous-refroidisseur de propane 16-MC10. Le régulateur de niveau 16-LIC- 1285maintient un niveau de liquide de propane constant dans le pot séparateur 56" qui setrouve sous le refroidisseur, en manipulant la vanne sur la ligne de refoulement, et ce aprèsune réduction à une ligne 8" qui fournit du propane liquide au refroidisseur de propane MR HHP 16-MC04.

Le refroidisseur de recyclage est constitué de trois (03) batteries contenant six (06) faisceaux échangeurs et neuf (09) ventilateurs (03 par batterie). Deux (02) des moto-ventilateurs sont à vitesse-fixe et un moto-ventilateurs dans chaque batterie possède un pilote "vitesse variable", contrôlé par le régulateur de température 16-TIC-1242.

L'opérateur démarre et arrête les neuf (09) moto-ventilateurs à partir du DCS. Le régulateur de température 16-TIC-1242 maintient une température de vapeurs froides dans la ligne de recyclage anti-pompage venant du refoulement des compresseurs vers les quatre (04) ballons d'aspiration de propane 16-MD02/03/ 04/ 05. Le débit dans cette ligne est contrôlé par des régulateurs de recyclage anti-pompage mentionnés sur le tableau ci-dessus.

Normalement, il n'y aura pas de débit de recyclage anti-pompage dans cette ligne, la "sortie" du régulateur est faible et les ventilateurs à vitesse variable sont éteints. Si l'opérateur veut démarrer les moto-ventilateurs à vitesse variable pour fournir plus de refroidissement au propane qui sort du refroidisseur, il peut utiliser la commande de chargement manuelle 16-HIC-1242. Cette commande manuelle règle les vitesses du moto-ventilateur à "vitesse variable"à travers le sélecteur "haut" 16-TY-1242.

Note: Le régulateur 16-TIC-1242, indique et contrôle seulement les températures de recyclage anti-pompage qui retournent vers les ballons d'aspiration, mais ni indique, ni contrôle la température au refoulement du compresseur.

3.6.2.4 Refroidisseurs/Condenseurs Réfrigération MR

- Envoyer une petite quantité de liquide propane à l'intérieur du **13-MC01** à travers la13-LV-1003. Quand le13-LIC-1003 atteint environ 20%, le mettre enautomatique.
- Envoyer une petite quantité de liquide propane à l'intérieur du **16-MC04** (àtravers la 16-LV-1140 et quand le 16-LIC-1140 atteint environ 20%, le mettre en automatique), puis le **16-MC05** (à travers la 16-LV-1147 et quand le 16-LIC-1147 atteint environ 20%, le mettre en automatique), ensuite le **16-MC06** (à travers la 16- LV-1149 et quand le 16-LIC-1149 atteint environ 20%, le mettre en automatique) et enfin le 16-MC07 (à travers la 16-LV-1155 et quand le régulateur 16-LIC-1155 atteint environ 20%, le mettre en automatique).

3.6.2.5 Chillers d'alimentation de la colonne d'épuration

- Envoyer une petite quantité de liquide propane à l'intérieur des équipements suivants :**15-MC01**(à travers la 15-LV-1005 et quand le 15-LIC-1005 atteint environ 20%, le mettre en automatique), puis le **15-MC02** (à travers la 15-LV- 1007 et quand le 15-LIC-1007 atteint environ 20%, le mettre en automatique), et enfin le **15-MC03** (à travers la 15-LV-1020 et quand le 15-LIC-1020 atteint environ 20%, le mettre en automatique).
- Maintenir des niveaux bas dans chaque échangeur (20-30%) et remplacer le propane

liquide perdu dans le **16-MD06**, ainsi le niveau de l'accumulateur va rester entre 40 à 60%.

3.7 Spécification DEVS proposé :

Pour gérer les contraintes de temps telles que la date limite, l'heure de début et l'heure de fin, nous définissons certains attributs liés à la spécification temporelle. Ainsi, nous propose de délimiter le temps d'exécution d'une activité, où:

SR Soon Recepte: avant cette temps, aucune activité ne peut être reçue.

SS Soon Send : avant cette temps, aucune activité ne peut être envoyé.

LS Late Send: la dernière fois peut envoi une activité. Passé ce délai, ne peut envoi aucune activité.

LT Late Terminate: est temps la plus éloignée à laquelle l'activité devrait se terminer.

Si l'activité termine son exécution après LT, le modèle déclenchera une erreur.

ES End Send : La vanne ne doit pas démarrer tant que le temps ES écoulé n'est pas écoulé (c'est à dire, après avoir envoyé le régulateur de l'événement « request to send»).

3.7.1 Vanne

Formellement, la vanne est défini par l'équation du modèle atomique DEVS suivante :

Vanne = (X, Y, S, δint, δext, λ, ta)

Où:

X = {In_r ∪ Ini i ∈ [0, n]}, "In_r" événements = {authorize to recepte, authorize to send,authorize to terminate}.

Y = {Out_r, Out}, "Out_r" événements = {request to recepte, Recepte, request to send, Send,request to terminate, Terminate}.

S = {Recepting, Recepted, Open the valve, Close the valve, Sending, Sended, Terminated, Terminating, Waiting, Error}.

δext La Fonction externe:

δext (Recepting, in_r?request to recepte) = Recepted
δext (Terminating, in_r?request to terminate) = Terminated
δext (Sending, in_r?request to send) = Sended

δint La fonction interne:

δint(Waiting) = Recepting
δint(Recepting) = δint(Sending) = δint(Terminating) = Error
δint(Terminated) = δint(Error) = Waiting

δint(Recepted) = Open the valve
δint(Recepted) = Close the valve

δint (Open the valve) = δint(Close the valve) = Sending

λ Fonction de sortie:

λ(Recepted) = Out_r!recepte
λ(Sended) = Out_r!send
λ(Terminated) = Out_r!terminate

ta La fonction d'avance de temps:

ta(waiting) = ∞, si la vanne en état finis ta(waiting)

= ES, si la vanne en état début

ta(s) = 0 Where s ∈ S ∩ {Recepted, Open the valve, Close the valve, Error}ta(Terminated) = r (termination)

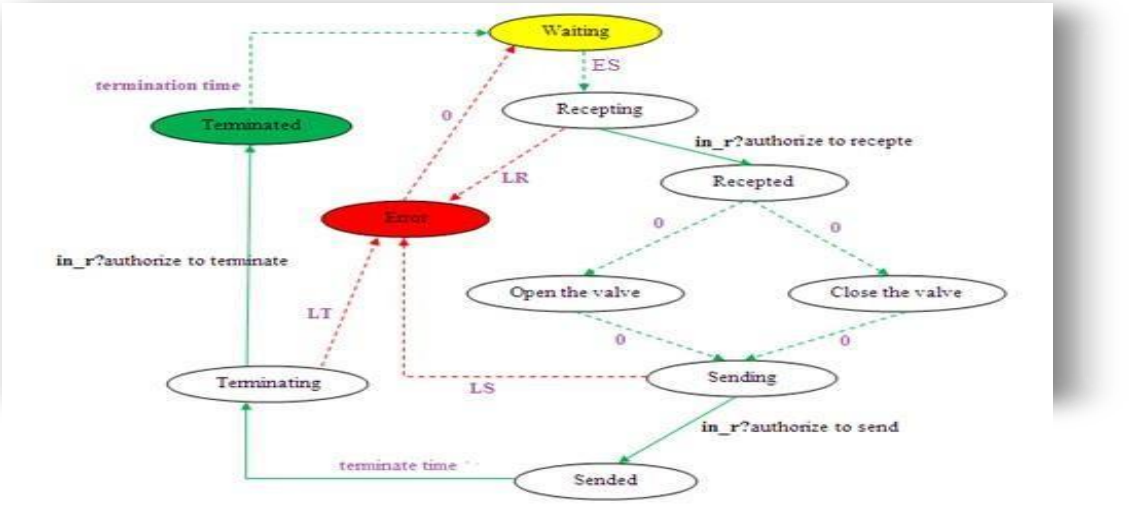


Figure 3.4: diagramme de transition des états de vanne au fil du temps

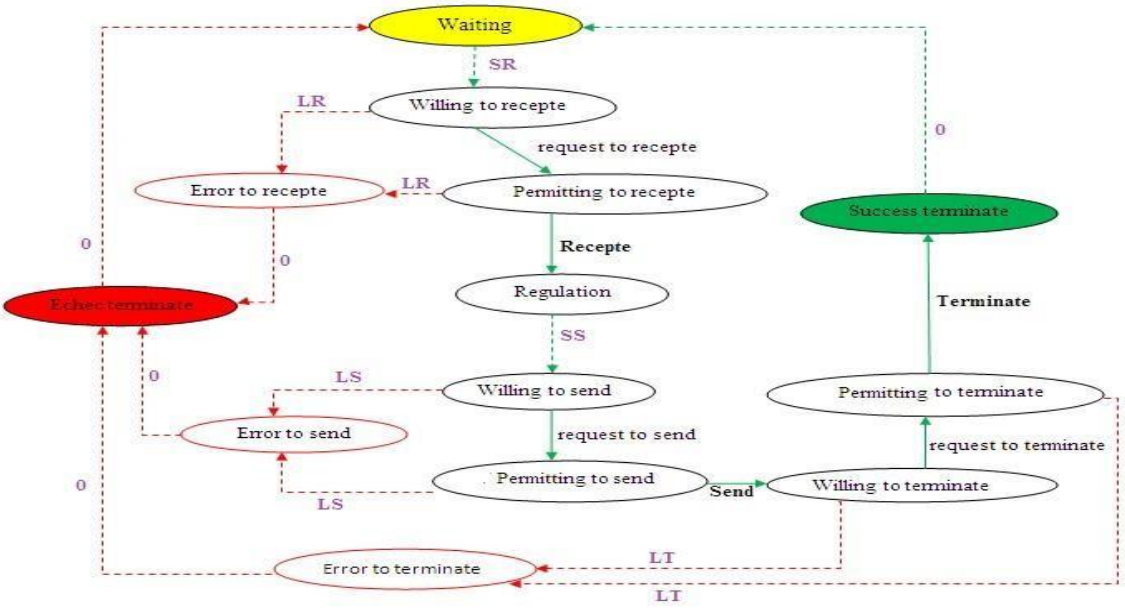


Figure 3.5 : Le diagramme de transition des états de régulateur au fil du temps

3.7.2 Le régulateur:

Formellement, le régulateur est défini par l'équation du modèle atomique DEVS suivante :
Régulateur = (X, Y, S, δint, δext, λ, ta)
Où:
X = {In}, "In″ événements = {request to recepte, Recepte, request to send, Send, request to terminate, Terminate}.

Y = {Out}, "Out″ évenements = {authorize to recepte, authorize to send, authorize to terminate}.

S = {Waiting, Willing to recepte, Permitting to recepte, Regulation, Willing to send, Permitting to send, Willing to terminate, Permitting to terminate, Error to recepte, Error to send, Error to terminate, Echec terminate, Success terminate}.

δint La fonction interne:
δint (Waiting) =Willing to recepte
δint (Regulation) = Willing to send
δint (Willing to recepte) = δint (Permitting to recepte) = Error to recepte
δint (Willing to send) = δint (Permitting to send) = Error to send
δint(Willing to terminate) = δint(Permitting to terminate)=Error to terminate
δint (Error to recepte) = δint (Error to send) = δint (Error to terminate)= Echec terminate
δint (Echec terminate) = δ int (Success terminate) = Waiting

δext La fonction externe:
δext (Willing to recepte, request to recepte) = Permitting to recepte
δext (Permitting to recepte, Recepte) = Regulation
δext (Willing to send, request to send) = Permitting to send
δext (Permitting to send, send) = Willing to terminate

δext (Willing to terminate, request to terminate) = Permitting to terminate
δext (Permitting to terminate, terminate) = Success terminate
Fonction d'avance de temps:
ta(Waiting) = t, where t < SR ta(Willing to Recepte) = LR – SR ta(Permitting to recepte) = LR − (ev)request to recepte.Timeta(Willing to send) = LS − SS

ta(Permitting to send) = LS – (ev)request to send.Timeta(Willing to terminate) = LT – (ev) Send. Time
ta(Permitting to terminate) = LT – (ev)request to terminate. Time ta(Regulation) = SS – (ev)Recepte. Time
ta(Error to recepte) = 0 ta(Error to send) = 0 ta(Echecterminate) = 0 ta(Success terminate) = 0
λ Fonction de sortie :
λ (Permitting to recepte) = out!authorize to recepte
λ (Authorizing to send) = out!authorize to send
λ (Authorizing to terminate) = out!authorize to terminate

3.7.3 La dynamique du modèle proposé :

La figure 3.4 Le diagramme de transition des états de vanne au fil du temps et la figure 3.5 Le diagramme de transition des états de régulateur au fil du temps. En fait, les états sont maîtrisésà la fois par les fonctions de transition internes et externes. Les arêtes continues représentent les transitions externes δext. Les arêtes pointillées représentent les transitions

δint. Les arêtes rouges représentent les chemins pour gérer les erreurs par rapport aux contraintes de temps. Les arêtes vertes représentent le chemin d'exécution réussi sans erreurs déclenchées par rapport aux contraintes de temps.

L'état initial est coloré en jaune. L'état de terminaison qui a succès est coloré en vert, l'état de terminaison qui a échoué est coloré en rouge. Les états qui bordés le rouge représentent lesétats intermédiaires qui conduisent à l'état de terminaison infructueuse.

- **Pour le régulateur (voir figure 3.5) :**

L'exécution du système démarre à partir de l'état «Waiting», ce qui signifie le système est en attend. Si l'heure SR est atteinte, alors l'état passe à «Willing to recepte» (cette transition est effectuée par la fonction de transition interne δint qui change les états lorsque leurs échéances sont atteintes). Cependant, Cet état vivra également jusqu'à l'heure «SR», la vanne devrait envoyer l'événement (message) «request to recepte» au régulateur demandant une autorisation d'envoyer. A partir de ce moment (SR), l'état du régulateur est «Willing to recepte». Lorsqu'il reçoit l'événement «request to recepte», il change son état en «Permitting to recepte». La fonction de transition externe δext, qui gère cet événement et modifie l'état du régulateur en conséquence. A ce moment, le régulateur envoie l'événement «authorize to recepte» au vanne (par la fonction λ). Les deux états «Willing to recepte» et «Permitting to recepte» conduisentà l'état «Error recepte» si l'heure LR est encore, alors le modèle se termine à l'état «Error terminate».

Cependant, si elle reçoit l'événement «recepte» avant l'heure LR, alors la fonction δext change l'état pour être «regulation» (Dans ce état, le régulateur calcule le débit ou la pression).

L'état du régulateur passe de « regulation» à «Willing to send» ceci est fait par la fonction de transition interne δint Si l'heure SS est atteinte.

Lorsqu'il reçoit le régulateur l'événement «request to send», il change son état en «Permittingtosend». La fonction de transition externe δext, qui gère cet événement et modifie l'état de régulateur en conséquence. A ce moment, le régulateur envoie l'événement «authorize to send»au vanne (par la fonction λ). Les deux états «Willing to send» et «Permitting to send» conduisent à l'état «Error send» si l'heure LS est encore, alors le modèles termine à l'état «Error terminate». Cependant, si elle reçoit l'événement «Send» avant l'heure LS, alors la fonction δext change l'état pour être «Willing to terminate». Lorsqu'il reçoit le régulateur l'événement «request to terminate», il change son état en «Permitting to terminate». La fonction de transition externe δext, qui gère cet événement et modifie l'état du régulateur en conséquence. A ce moment, le régulateur envoie l'événement «request to terminate» à la vanne(par la fonction λ). Les deux états «Willing to terminate» et «Permitting to terminate»

conduisent à l'état «Error to terminate» si l'heure LS est encore, alors le modèle se termine à l'état «Error to terminate». Cependant, si elle reçoit l'événement «terminate» avant l'heure LT, alors la fonction δext change l'état pour être «Success terminate».Les deux états «success terminate» et «Echec terminate» vont directement à l'état «Waiting» ceci est fait par la fonction de transition interne δint.

- **Pour la vanne (voir figure 3.4) :**

L'exécution du système démarre à partir de l'état «Waiting», ce qui signifie la vanne est en attend. Si l'heure FS est atteinte, alors l'état passe à «recepting» (cette transition est effectuée par la fonction de transition interne δint qui change les états lorsque leurs échéances sont atteintes). Cependant, Cet état vivra également jusqu'à l'heure «FS», le régulateur devrait envoyer l'événement (message) «authorize to recepte» au la vanne. A partir de ce moment (FS),l'état de vanne est «authorize to recepte».

- Lorsque la vanne reçoit l'événement «Request to recepte» de régulateur, elle change son étaten «Recepting» et envoie l'événement «authorize to recepte» au régulateur et change son état en «Recepted» » (cette transition est effectuée par la fonction de transition externe δext).
- La vanne change son état en «Close the valve» ou «Open the valve», cela dépend de la valeurde débit ou de pression envoyée par le régulateur où, la vanne s'ouvre si les valeurs sont élevées, ou elle se ferme si les valeurs sont bas (cette transition est effectuée par la fonction de transition interne δint).
- Lorsque la vanne reçoit l'événement «Request to send» de régulateur, elle change son état en «Sending» et envoie l'événement «authorize to send» au régulateur et change son état en «Sended» (la dernière transition est effectuée par la fonction de transition externe δext).
- Lorsque la vanne reçoit l'événement «Request to terminate» de régulateur, elle change son état en «Terminating» et envoie l'événement «authorize to terminate» au régulateur et change son état en «Terminate» » (cette transition est effectuée par la fonction de transition externe δext).

- Les états «Recepting» et «Sending» et «Terminating» conduisent à l'état «Error» si les heures LR, LS et LT ont encore, alors le modèle revient à l'état «Waiting».

3.7 Conclusion

Nous avons proposé dans ce chapitre un système industriel (boucles de réfrigération au propane pour le nouveau projet de train Skikda LNG, Algérie) et représenté les composants automatiques dans le système proposé avec un diagramme de transition des états au fil du temps.

Chapitre 04 : Implémentation

4.1 Introduction

Après avoir présenté notre procédé industriel, dans ce chapitre nous passons à la réalisation du notre simulateur en utilisant la plateforme DEVS JAVA et l'outil DEVS-Suite qui forme notre environnement de développement dont les caractéristiques seront citées et que nous jugeons performant pour notre solution.

4.2 Outils de programmation

Pour réaliser notre application on a utilisé trois outils de développement :

- Le langage Java : notre travail est articulé sur ce langage.
- Eclipse : environnement intégré de développement (IDE) pour le langage Java (etd'autres langages).
- Plate-forme DEVS-Suite: c'est l'outil le plus utilisé. car il a servi à la création du système multi-agents et à l'établissement de la communication entre agents, et vul'importance de cet outil nous allons l'expliciter encore plus.

4.3 Présentation des outils :

4.3.1 Langage de programmation Java

Figure 4.1 : logo de Java

Java est à la fois un langage de programmation informatique orienté objet et un environnement d'exécution informatique portable créé par James Gosling et Patrick Naughton employés de Sun Microsystems avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld.

Il est caractérisé par sa portabilité, multiplateformes, orienté objet ainsi que la richesse de sa bibliothèque.

4.3.2 JDK (Java Development Kit)

Le Java Development Kit, communément appelé JDK, est le kit de développement de base que propose gratuitement la société Oracle. Le Kit de développement comprend plusieurs outils, parmi lesquels :

- **Javac:** le compilateur Java.
- **Java :** un interpréteur d'applications (machine virtuelle).
- **applet viewer**: un interpréteur d'applets.
- **jdb:** un débogueur.
- **javap :** un dé compilateur, pour revenir du bytecode au code source.
- **javadoc:** un générateur de documentation.
- **jar :** l'éditeur d'archives Java.

Dans notre travail, La version utilisée pour l'implémentation est JDK11.

Figure 4.2- Logo Eclipse

4.3.3 Eclipse

Est un environnement de développement intégré (IDE) open source pris en charge par IBM. Eclipse et populaire pour le développement d'applications Java (Java SE et Java EE) et les applications Android. Il prend également en charge C/C++, PHP, Python, Perl et d'autres développements de projets web via des plug-ins extensibles. Eclipse est une plateforme universelle d'intégration d'outils de développement.

Eclipse est multiplateforme et fonctionne sous Windows, Linux et MacOs.

4.3.4 La plateforme DEVS-Suite

4.3.4.1 Définition

DEVS-Suite est un simulateur DEVS parallèle prenant en charge l'automatisation de la conception d'expériences en combinaison avec les modèles animés et la génération de trajectoires de données à l'exécution.

Figure 4.3 : logo de DEVS-Suite.

4.3.4.2 L'architecture de la plate-forme DEVS-Suite

Il s'agit d'un environnement de simulation libre, discret et polyvalent qui peut être exécuté sur un ordinateur personnel ainsi qu'en ligne via DEVS-Suite Web Start qui permet l'apprentissageen ligne à l'aide de la technologie Java Web Start. Les principaux modules de DEVS-Suite sont DEVSJAVA, DEVS tracking Environment et timeview, comme illustré à la Figure

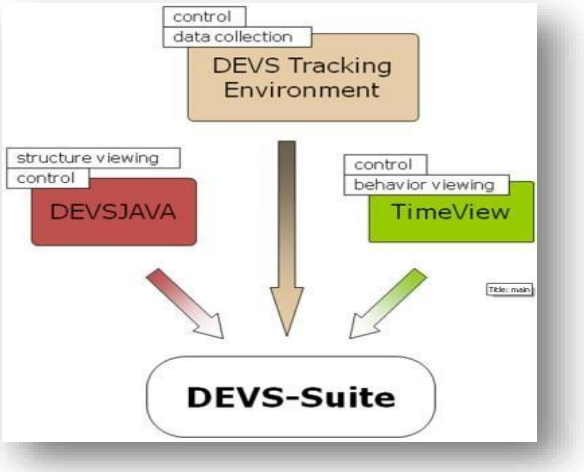


Figure 4.4 : Architecture de simulateur DEVS-Suite.

Basé sur un style d'architecture logicielle Model-View-Façade-Control, DEVS-Suite introduit de nouvelles fonctionnalités pour développer et expérimenter des modèles de simulation DEVS parallèles hiérarchiques basés sur des composants [11]. Dans le formalisme de spécification du système Discrete Event, on spécifie

• Les modèles atomiques dictent la façon dont les entrées/états sont transformées en sorties.

• Les modèles couplés sont définis en termes de modèles atomiques ou d'autresmodèlescouplés et de relations de couplage.

4.3.4.3 Intégration de la plateforme DEVS-Suite:

Guide d'installation et d'exécution

1) Décompressez le fichier DEVS_Suite_3.0.0_executable.zip dans un répertoire de votre choix. Assurez-vous que l'exécutable DEVS_Suite_3.0.0_executable_win_64_lib.jar et le

Le dossier DEVS_Suite_3.0.0_executable_win_64_lib se trouve dans le même dossier.

2) Après avoir lancé le simulateur DEVS-Suite, et ouvert le menu "Load Model ..." dans le menu déroulant Fichier, la boîte de dialogue "configuration du modèle"apparaîtra. Dans cette boîte de dialogue

- Définissez le "Chemin d'accès aux packages de classes de modèles (à partir du dossier actuel)"sur

l'emplacement absolu du paquet que l'on veut charger, '[chemin vers le dossier décompressé]/DEVS_Suite_3.0.0_executable_win_64_lib,

- Définissez le "Chemin d'accès aux packages de fichiers source du modèle (à partir du dossieractuel)"

vers et

- Définissez les "Noms de packages de modèles (un par ligne)" sur SimpArcMod. SimpArcMod a quelques modèles de base. Un modèle de base pour commencer est "efp". L'emplacement pour le package "SimpArcMod" est '[chemin vers le fichier décompressé dossier]/DEVS_Suite_3.0.0_executable_win_64_lib.jar'.

3) Si un fichier jar d'un autre modèle DEVS-Suite existe, il doit être ajouté à DEVS_Suite_3.0.0_executable_win_64_lib. Définissez le "Chemin d'accès aux packages declasses de modèles (du dossier actuel)" à l'emplacement absolu du package que l'on veut charger, '[chemin vers le dossier décompressé]/ DEVS_Suite_3.0.0_executable_win_64_lib/yourmodel.jar' dans le "modèleboîte de dialogue de configuration ».

4.4 Implémentation du système

Pour illustrer la simulation DEVS nous allons suivre un exemple pour fournir des Capturesd'écrans qui décrivent les différentes étapes de la simulation.

La simulation du procédé présenté dans le chapitre précédent a été réalisée dans lesimulateurDEVS-Suite qui est basé sur JAVADEVS, comme illustré dans la figure 4.5.

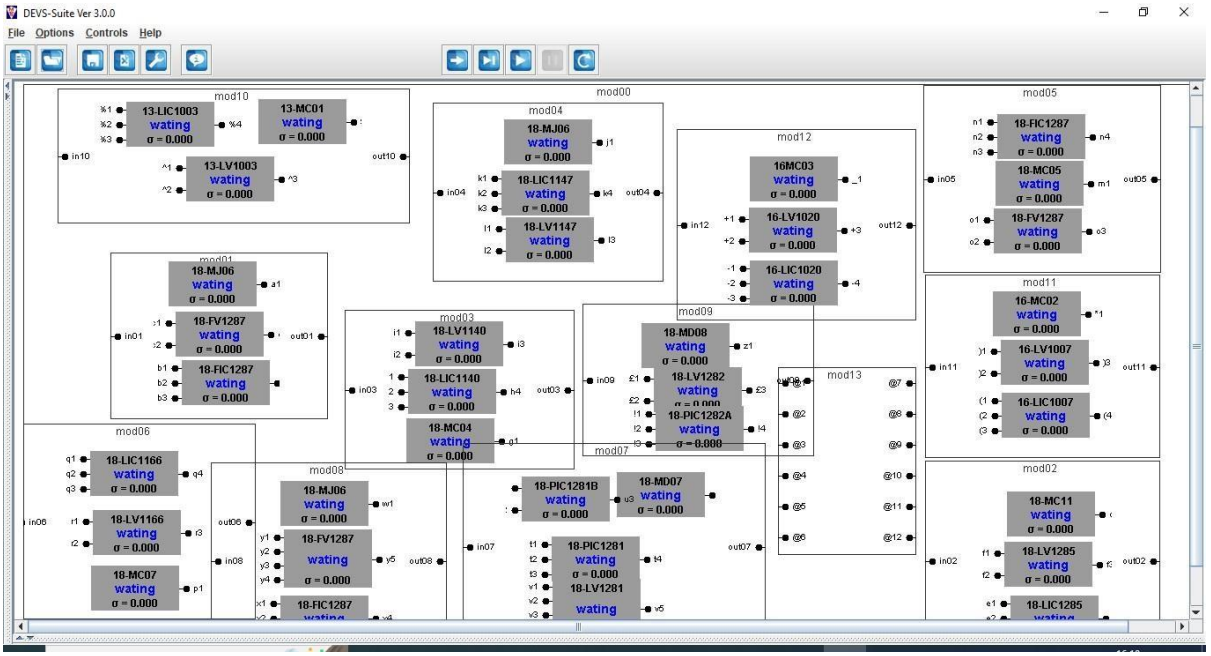


Figure 4.5- L'interface graphique de simulateur du système de compression de laréfrigération de propane en DEVS-Suite

Cette interface graphique représente le système de compression de la réfrigération de propane à travers des modèles Atomiques et couplés. Dans notre simulation nous avons appliqué un plan de simulation dans deux scénarios. Le premier est une exécution réussie dans laquelle nous définissons les activités avec des temps d'exécution appropriés à ce qui est défini dans les contraintes de temps.

4.5 Conclusion

Nous avons utilisé la plateforme DEVS-Suite afin de créer et d'implémenter notre système. Nous avons constaté que cette plateforme est appropriée dans le domaine de simulation informatique et permet de faciliter la tâche aux développeurs débutants dans ce domaine.

Conclusion générale

Le travail que nous avons présenté se situe dans le cadre de la modélisation et la simulation des systèmes industriels. Dans ce travail, nous avons effectué une simulation d'un système industriel (boucles de réfrigération au propane pour le nouveau projet de train Skikda LNG, Algérie), où nous avons opté pour une méthode à événement discret basée sur le formalisme DEVS dans le but de profiter de la puissance formelle de ce formalisme pour la vérification et la validation des systèmes industriels qui sont classés comme des systèmes dynamiques à événements discrets (DEDS) et sont caractérisés par une structure complexe et hiérarchique. Notre démarche de simulation nous permet de répondre mieux à certains besoins fondamentaux de l'industrie : la conception de stratégies de contrôle commande, la conception des unités nouvelles, la formation et l'entraînement des opérateurs…etc. Finalement, nous avons soutenu notre approche par une exemple application, dans le cas de la régulation du procédé de compression de la réfrigération de propane. Notre système a été implémenté sur la plateforme JAVADEVS et en utilisant l'outil DEVS - Suite. Dans de futurs travaux, nos perspectives portent sur la généralisation de l'approche développée dans ce mémoire et nous visons à proposer une méthode générale pour construire des simulateurs capables de fonctionner sur plusieurs types de procédés des systèmes industriels.

Bibliographie

[Hubert, 1969] P. Hubert, Le Glossaire International d'Hydrologie, novembre, 1996.

[Shannon, 1998] R. E. Shannon, Introduction to the art and science of simulation. Proceedings of the 30th conference on Winter simulation. IEEE Computer Society Press, 7- 14, 1998.

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

[Filippi et al., 2002b] J.B. Filippi, P. Bisgambiglia et M. Delhom, Neuro-devs, an hybrid methodology to describe complex systems. Dans Processings of the Society for Computer Simulation International European Symposium on Simulation 2002 conference on simulation in industry, volume 1, 2002.

[Filippi, 2003] J.B. Filippi, Une architecture logicielle pour la multi-modélisation et la simulation à évènements discrets de systèmes naturels complexes. Thèse de doctorat, Université de Corse, 2003.

[Filippi et Bisgambiglia, 2004] J.B. Filippi et P. Bisgambiglia, JDEVS: An implementation of a DEVS based formal framework for environmental modelling.http://www citeseer.ist.psu.edu/filippi03jdevs.html, 2003.

[Fishwick 94]. Computer simulation: Growth through extension. In Society for Computer Simulation, pages 3–20, 1994.

[Fishwick, 1995] P.A. Fishwick, Simulation Model Design and Execution. Digital Worlds, Prentice Hall, 1995.

[Giambiasi et Ghosh, 2001] N. Giambiasi et 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.

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

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

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

[Karsky 02] Karsky M., La dynamique des systèmes complexes, cours ECP, 2002.

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

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

[Leroudier 80] Leroudier J., Simulation à évènements discrets, Editions Hommes et Techniques 1980

[Mouis 84] Mouis F. (épouse Keller), SSIGMA : un outil de représentation et de simulation adapté à la conception de systèmes industriels, thèse, Châtenay-Malabry, 278 p

[Nikiforov 99] Nikiforov I., Simulation des systèmes continus, Printemps 1999 (wwwlm2s.utt.fr/~nikiforo/sy15.pdf)

[Nutaro, 2003] J. NUTARO, adevs, A Discrete EVent System simulator. http: //www.ornl.gov/1qn/adevs/, 2003.

[Riat 92] Riat J. C., Validation par simulation d'architecture de commande d'atelier dans l'industrie de production manufacturière : Application aux systèmes automatisés de production d'un grand constructeur automobile, 287p, Th. : Productique, Automatique et Informatique industrielle : Lille : 1992.

[Sarjoughian et Zeigler, 1998] H. Sarjoughian et B. Zeigler, DEVSJava : Basis for a DEVS-based collaborative ms environment. Dans Actes de la conférence 1998 Society of Computer Simulation International Conference on Web-Based Modelling and Simulation, volume 5, pages 29–36, San Diego. SCS The Society for Modelling and Simulation International, 1998.

[Sarjoughian et Zeigler, 1999] S.H. Sarjoughian et B.P. Zeigler, The role of collaborative Devs Modeler in federation development. In Proceedings of the 99 System Interoperability Workshop, 1999.

[Sarjoughian,2011] Fatemeh Orooji1, Hessam S. Sarjoughian Fattaneh Taghiyareh:Modeling & Simulation of Educational Multi-Agent Systems in DEVS-Suite.

[Seck and Honig, 2012] Seck, M.D. and Honig, H.J. (2012) 'Multi-perspective modeling of complex phenomena', Computational and MathematicalOrganization Theory, Vol. 18, No. 1, pp.128–144.

[Seddari15] Seddari N., Outils formels et opérationnels pour la modélisation et la simulation des systèmes complexes 2015.

[SmallDEVS,2003] SmallDEVS, Disponible sur

ttp://perchta.fit.vutbr.cz:8000/projekty/10. Stéphane Garredu: Approche de méta-modélisation et transformations de modèles dans le contexte de la modélisation et simulation à évènements discrets : application au formalisme DEVS.

[Uhrmacher, 2001] A. M. Uhrmacher, Dynamic structures in modeling and simulation: a reflective approach. ACM Trans. Model. Comput. Simul., 11 :206–232, ISSN 1049-3301. April, 2001. (Cité page 193.)

[Vangheluwe, 2000] H. Vangheluwe, DEVS as a common denominator for multiformalis hybrid systems modelling. Actes de la conférence IEEE International Symposiumon Computer-Aided Control System Design, 1(1) :129–134, 2000.

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

[Zeigler 1976] B.P. Zeilger, Theory of Modeling and Simulation, New-York: WileyInterscience, 1976

[Zeigler 1984] Bernard Zeigler, Multifacetted Modeling and Discrete Event Simulation, Academic Press, London; Orlando, 1984 (ISBN 978-0-12-778450-2)

[Zeigler et al., 2000a] B.P. Zeigler, H. Praehofer et T. G. Kim, Theory of Modeling and Simulation. Academic Press, ISBN 0127784551, 2000.

[Zeigler et al., 2000b] B.P. Zeigler, D. Kim et H. Praehofer, Theory of modeling and simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press, 2000.

[Zeigler et al., 2000] ZEIGLAR, B. P. et SARJOUGHIAN, H. Introduction to DEVS modeling and simulation with JAVA—a simplified approach to HLA-compliant distributed simulations, ACIMS—Arizona Center of Integrative Modeling and Simulation. 2000.