RÉPUBLIQUEALGÉRIENNEDÉMOCRATIQUEETPOPULAIRE
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Kasdi Merbah Ouargla

Faculté des Sciences Technologie et Sciences de la Matière -MSTM

Département des Mathématiques et Informatique

N° d'ordre :……

N° de Série :……

Mémoire

Présenté en vue d'obtenir le diplôme de

Magister en informatique

Option : Technologie de l'Information et de Communication (TIC)

PROPOSITION D'UNE ONTOLOGIE FORMELLE POUR LA MODÉLISATION ET LA SIMULATION INTELLIGENTE

Présenté et soutenu publiquement le **28/06/2012**Par :

BEN HEBIRECHE HALIMA

Devant le jury composé de :

Président	**Dr. Said Mohamed SAID**	**Maitre de conférences, université d'Ouargla**
Rapporteur	**Dr. Ahmed KORICHI**	**Maitre de conférences, université de Ouargla**
Examinateurs	**Dr. Driss KORICHI**	**Maitre de conférences, université de Ouargla**
	Dr. Brahim BELATTAR	**Maitre de conférences, université de Batna**

UNIVERSITÉ D'OUARGLA -2012-

Remerciements

Louange à ﷲ, seigneur de l'univers, de m'avoir donné la force, la patience et la volonté à accomplir ce travail.

Je tiens à exprimer mes remerciements et ma profonde reconnaissance au Dr KORICHI Ahmed maitre de conférences à l'université d'Ouargla et mon Directeur de mémoire de magister, pour la pertinence de ses remarques et sa patience pendant ce travail, sa manière de diriger qui fut pour moi une grande source d'inspiration et de motivation, il a su m'encourager et m'a permis de travailler dans une ambiance scientifique exceptionnelle.

Je remercie chaleureusement les membres du jury de mémoire : monsieur Dr. Said Mohamed SAID pour m'avoir fait l'honneur de présider mon jury de thèse, et les messieurs Dr. Driss KORICHI et Dr. Brahim BELATTAR pour avoir accepté de juger ce travail.

Bien sûr, je tiens à remercier ma famille, mes amis et mes collègues à la fois aux départements d'informatique de l'université d'Ouargla pour leur présence dans les moments de joie et leur soutien dans les moments de peine.

Enfin, je remercie toutes les personnes qui m'ont aidée, de prés ou de loin pour la réalisation de ce travail en particulier**.**

BEN HEBIRECHE Halima

Table des matières

Chapitre II : Le domaine de l'ontologie

Chapitre III : Vers un modèle conceptuel de l'ontologie DeMO-E

Chapitre IV : Implémentation

Table des figures

Liste des tableaux

N°Tableau	Titre	Page

Introduction générale

Introduction générale

Problématique

L'être humain ne cesse de chercher à comprendre le monde qui l'entoure. Les différentes phases d'étude nécessaires à cette compréhension ont toujours évolué relativement avec la technologie : observation, constatation, hypothèse, preuve, explication, acquisition, modélisation, etc.

L'avènement de l'informatique et l'apparition de nouveaux formalismes de traitement nous ont permis d'automatiser plusieurs tâches pénibles et fastidieuses, de l'assistance au remplacement, la machine s'est montrée nettement supérieure à l'homme dans les domaines de calcul, et parfois même dans les activités intellectuelles. Donc quel que soit le domaine d'application, l'homme est amené à communiquer avec la machine, cette interaction nécessite un minimum de compréhension mutuelle, le besoin de savoir ce que fait la machine et pourquoi elle le fait s'avère très important; pour ce faire, la machine doit pouvoir expliquer voir même justifier ses comportements, cela constitue une double activité intelligente, du fait que la machine nécessite non seulement un raisonnement sur ses propres comportements, mais aussi de pouvoir gérer un dialogue naturel avec l'humain et de comprendre ce qu'il veut.

Dans le domaine de modélisation et de simulation, un modèle est défini comme une description mentale, ou figurée qui, pour un champ de questions est pris comme représentation abstraite d'une classe de phénomènes, plus ou moins habilement dégagés de leur contexte par un observateur pour servir de support à l'investigation, et/ou la communication, il permet de décrire le fonctionnement d'un système avec un degré de détails nécessaire à la résolution du problème posé.

Actuellement, beaucoup d'outils existent pour assister le modélisateur dans la conception et l'implémentation de leur modèles; mais malheureusement ces outils sont dotés de moyens très limités (généralement visualisation graphique et trace) pour aider l'utilisateur lors de l'exploitation de ces modèles pour lui expliquer ce qui se passe dans le scénario qu'il traite, ces moyens devraient permettre d'atténuer les imperfections de communication et d'améliorer la compréhension d'une situation calculée, donc l'intégration d'un module explicatif (intelligent) est bien souhaitable pour éclaircir l'utilisateur sur la véracité d'une transition, ou éventuellement l'ambigüité d'un comportement attendu.

Le travail présenté dans ce mémoire concerne l'ingénierie ontologique, et plus particulièrement l'utilisation des ontologies dans le domaine de modélisation et simulation intelligente et ce par la construction d'une ontologie «DeMO-E » Discrete event Modeling Ontology-Explication à partir la méta-ontologie DeMO [1] . Après une étape de conceptualisation qui a permis l'identification des connaissances du domaine, une étape d'ontologisation a permis de formaliser les connaissances identifiées (sous forme d'une ontologie) en utilisant un modèle sémantique, une dernière étape appelée opérationnalisation a permis la transcription de l'ontologie dans un langage formel et opérationnel de représentation de connaissances ; pour l'opérationnalisation de l'ontologie, l'éditeur d'ontologie Protégé (version 4.1) est utilisé pour représenter l'ontologie sur machine.

Objectifs spécifiques

Dans ce contexte l'objectif de ce travail est de proposer et de définir un méta-modèle pour la modélisation et la simulation intelligente, pouvant s'exprimer par une ontologie formelle permettant de :

- Donner la possibilité à l'utilisateur de poser des questions chaque fois qu'il est en face d'une situation qu'il ne perçoit pas bien ou qu'il pense qu'elle aurait pu se dérouler autrement.
- Mieux comprendre les modèles de simulation intelligente.
- Augmenter la confiance dans les résultats de la simulation.

[1] **DeMO :** Discrete-event Modeling Ontology

- Fournir un moyen d'acquérir des connaissances du domaine de modélisation et la simulation à événements discrets.

Vis-à-vis nos objectifs, nous identifions les étapes de travail suivantes :

❖ **Analyse des pratiques de la simulation :** Cette étape répond à deux objectifs : tout d'abord, face à l'étendue du problème posé, elle recadre précisément les domaines de notre recherche en se basant sur ce que nous pensons avoir un intérêt à traiter, le second objectif consiste à donner des éléments d'analyse du contexte spécifique à notre étude; le domaine de la modélisation et simulation intelligente, les caractéristiques principales de ce domaine y sont développées, eu égard à ce qui constitue la finalité de notre travail ; la spécification de simulateur intelligent.

❖ **Etude des ontologies** : Pour mieux maîtriser les méthodes de conception et de réalisation de l'ontologie, et en vue de les appliquer dans la phase de conception de notre module d'explication, cette étape va nous permettre de mieux cerner le domaine des ontologies pour atteindre notre objectif qui est la proposition d'une ontologie formelle pour la modélisation et la simulation intelligente. Dans un premier temps, nous maitrisons les concepts liés aux ontologies ;ensuite, nous établissons un état de l'art des outils de conception et de développement d'ontologie en vue d'identifier celles qui conviendraient le mieux compte tenu des objectifs de ce travail de recherche.

❖ **Etude conceptuelle :** Sur la base des deux étapes précédentes et eu égard au contexte spécifique du domaine de simulation étudié. Dans cette étape, nous proposons une conception d'une ontologie formelle pour la simulation en utilisant les méthodes de conception retenue.

❖ **Implémentation**: Dans cette étape nous proposons une implémentation d'ontologie développée sur la base des résultats des étapes précédentes, dans le but de valider les modèles conceptuels proposés.

Organisation de mémoire

Malgré la linéarité imposée par tout processus de rédaction, qui nécessite une importante reconstruction d'acheminement intellectuel effectué, nous nous sommes donc efforcés de préserver la forte interaction qui a existé depuis le début entre les deux grands thèmes exposés dans ce mémoire, qui sont la modélisation et la simulation intelligente d'une part et l'ontologie d'autre part.

La structure en chapitres de ce mémoire reflète notre démarche de recherche :

- Le premier chapitre est consacré à la modélisation et la simulation intelligente, nous détaillons plus particulièrement la modélisation et la simulation à évènement discret.

- Dans le deuxième chapitre en définissant la notion de l'ontologie et d'ingénierie ontologique, nous présentons les éléments constituant une ontologie et leurs différents formalismes de représentation, nous citons par la suite les étapes de construction d'une ontologie, et enfin nous terminons par une présentation de quelque outils de construction d'ontologies utilisés, afin d'implanter des ontologies.

- Dans le troisième chapitre nous exposons l'ontologie DeMO puis et on passe à la conception de notre ontologie DeMO-E, ensuite nous détaillons les éléments constituant cette ontologie.

- Nous présentons dans le quatrième chapitre l'environnement de développement Protégé ainsi que l'implémentation de notre ontologie« DeMO-E».

Enfin, on termine ce mémoire par une conclusion générale, dans laquelle on va donner une présentation du bilan de notre travail ainsi que les perspectives futures de notre ontologie.

Chapitre I

Le domaine de la simulation

I.1 Introduction

La simulation est actuellement le seul outil utilisable lorsqu'on désire appréhender le fonctionnement d'un système complexe que l'on ne parvient pas à formaliser pour utiliser des méthodes analytiques. Les applications de la simulation sont très nombreuses : l'analyse des systèmes de services (Banques, téléphonie, ...), les systèmes naturels (biologiques, écologiques,...), les systèmes informatiques (d'exploitation, de communications, ...)...etc.

Notre réflexion est associée aux notions de système, de modèle, de modélisation et de simulation, ces notions sont dépendantes des techniques et du domaine d'application choisi. Toutes ces entités interviennent dans le processus de modélisation et de simulation, la figure I.1 présente ces différents éléments.

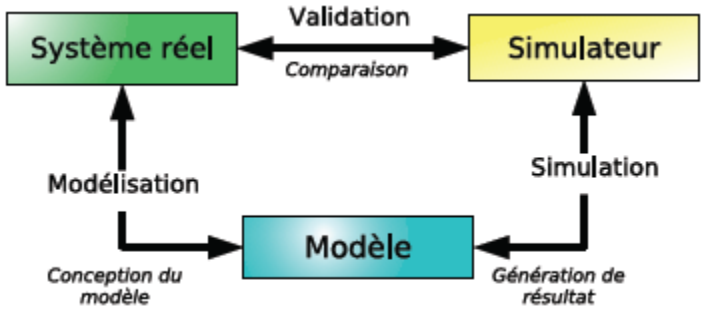

Figure I.1 : La relation modèle, modélisation et simulation. [Antoine, 2008]

La modélisation et la simulation sont des domaines scientifiques faisant un grand usage de l'informatique, et leur mise en œuvre s'effectue au sein de logiciels appelés environnements de modélisation et de simulation.

Dans le présent chapitre, nous présentons un aperçu sur le principe de modèle, de modélisation et de simulation ; ensuite, nous illustrons les étapes d'une étude de simulation et les outils utilisés, puis nous citons les principales approches de modélisation. Finalement, nous donnons une brève définition de l'environnement de modélisation intelligente.

I.2 Modèle et modélisation :

Parfois, il est impossible d'étudier le système directement du fait qu'il soit inaccessible, ou trop coûteux pour qu'on puisse faire dessus des expériences directement, du fait qu'il change trop rapidement ou trop lentement ; dans ces cas, on considère un système de manière globale, par abstraction de certaines contraintes, nous pouvons associer à un système complexe une représentation simplifiée de sa structure et de son fonctionnement, cette représentation plus simple à décrire et à utiliser s'appelle un modèle.

I.2.1 Définition :

La définition de modèle et de modélisation peut varier nettement selon le domaine historique et la discipline étudiée. Il y a une relation forte entre contexte et définition. D'après *P. Fishwich* "*modéliser c'est décrire une réalité sous la forme d'un système dynamique, à l'aide d'un langage de description, à un certain niveau d'abstraction*" et selon *C. Oussalah*" *la modélisation est un épimorphisme entre un système réel et un modèle dont la finalité est de donner une représentation simplifiée et observable de la structure et du comportement du système réel*". [Antoine, 2008]

Dans notre contexte, ceci nous conduit à définir, la modélisation comme un processus d'identification d'un phénomène et l'opération par laquelle on établit son modèle, afin d'en proposer une représentation interprétable, reproductible et simulable. C'est une technique qui consiste à restituer, sous une forme compréhensible par l'ordinateur, un objet ou un phénomène quelconque.

Ce modèle, ainsi constitué, permet d'approcher le fonctionnement ou le comportement du système par l'intermédiaire d'un certain nombre d'hypothèses et de règles qui composent ce qu'on appelle un formalisme de modélisation.

Une définition d'un modèle énoncée par l'AFCET, est la suivante :

"*Un* modèle *est un schéma, i.e. une description mentale (intériorisée), ou figurée(diagrammes, formules mathématiques, etc... qui, pour un champ de questions est pris comme représentation abstraite d'une classe de phénomènes, plus ou moins habilement dégagés de leur contexte par un observateur pour servir de support à l'investigation, et/ou la communication*".[Belattar, 2000]

Un modèle est donc une représentation d'un système (réel ou imaginé) dont le but est d'expliquer et de prédire certains aspects du comportement de ce système. Le processus de construction d'un modèle de simulation peut être schématisé comme suit :

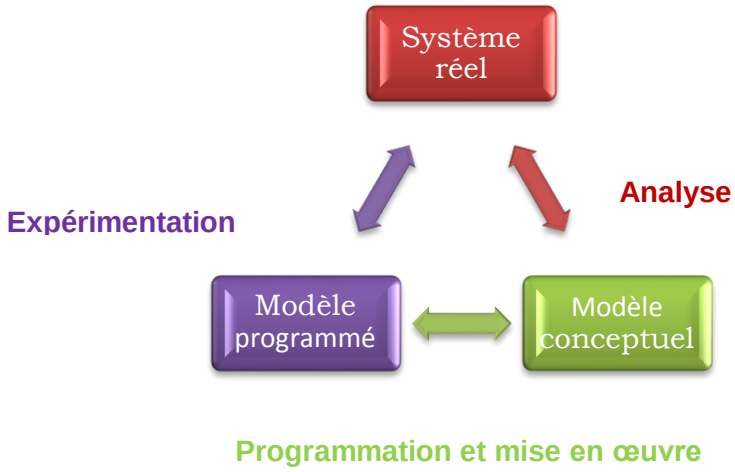

Figure I.2 : Processus simplifié de Modélisation.[Belattar, 2000]

Le modèle conceptuel est une représentation (mathématiques logique, verbale, etc.) du système réel (ou retenu pour une étude particulière), il est obtenu dans une phase d'analyse et de modélisation. Le modèle programmé est la mise en œuvre du modèle conceptuel sur un calculateur, il est obtenu dans une phase de programmation et de mise en œuvre. Les enseignements sur le système réel sont obtenus à la suite d'expérimentations sur le modèle programmé dans une phase d'expérimentation.

I.2.2 Les caractéristiques fondamentales d'un modèle sont :

- Le caractère abstrait, qui doit notamment permettre de faciliter la compréhension du système étudié.
- Il réduit la complexité du système étudié.
- Il permet de simuler le système étudié et reproduit ses comportements.

Modéliser une entité revient donc à définir les concepts qui permettent de la décrire et les relations fonctionnelles qu'entretiennent ces concepts.

La figure suivante illustre qu'un modèle est une représentation d'un certain système.

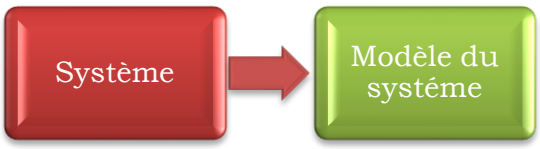

Figure I.3 : Système et modèle

I.2.3 Classification de modèles :

Pour un système donné, il est possible de construire plusieurs types de modèles selon les objectifs poursuivis ou les contraintes à satisfaire. On peut considérer qu'il existe deux grandes catégories de modèles : les modèles analogiques, encore appelés modèles physiques (maquettes du système) et les modèles abstraits. Une représentation possible des différents types de modèles est indiquée sur la figure I.4.

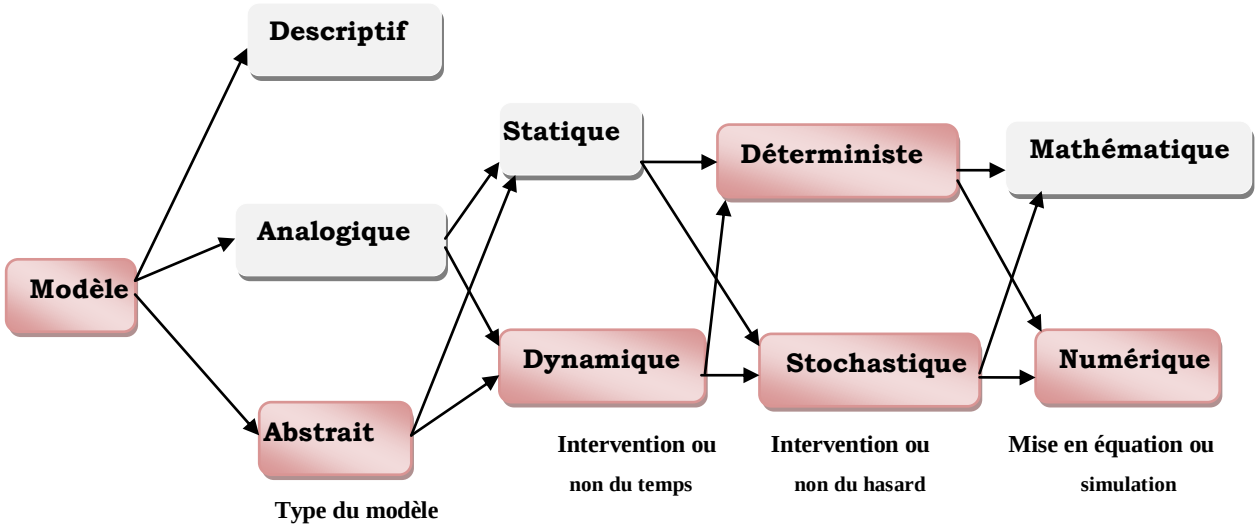

Figure I.4 : Classification des différents types de modèles.[Fleury et al, 2007]

Parmi les modèles analogiques, on peut citer par exemple, une maquette d'avion permettant d'évaluer l'écoulement de l'air sur les ailes. Ce modèle physique est une représentation à une échelle 1/5 de l'avion tel qu'il est imaginé par les ingénieurs, mais il n'intègre pas tous les détails de l'appareil qui volera par la suite.

La réalisation d'un modèle mathématique signifie que tous les composants du système et toutes les connexions entre les composants sont exprimées sous la forme de relations mathématiques.

Évidemment, la conception d'un modèle, quelle que soit sa nature, requiert l'acquisition de certaines compétences dans le domaine du système étudié : productique, physique, biologie, écologie… D'autre part, le modèle hérite des lacunes des connaissances mises en œuvre : lois physiques approchées, collecte difficile de données souvent partielles ou estimées.

I.2.4 Les objectifs de la modélisation :

Les modèles permettent de mieux comprendre le système que l'on développe, ils permettent ainsi d'atteindre plusieurs objectifs parmi lesquels on cite:

- Les modèles nous aident à visualiser un système tel qu'il est ou tel que nous voudrions qu'il soit.
- Les modèles permettent de préciser la structure ou le comportement d'un système.
- Les modèles fournissent un canevas qui guide la construction d'un système.
- Les modèles permettent de documenter les décisions prises.

I.3 La simulation :

La simulation est un outil puissant d'imitation des systèmes de toutes disciplines, dont le développement croissant est largement contribué par l'évolution exponentielle et permanente de la technologie informatique dans tous ses aspects (processeurs, puissance de calcul, mémoires, langages...) ainsi que l'appropriation facilitée par la chute des coûts, donc elle est rapidement devenue incontournable pour la modélisation des systèmes complexes. Elle permet de gérer des modèles afin de produire des données comportementales, c'est-à-dire de faire évoluer les états du modèle dans le temps.

I.3.1 Définition :

Il existe plusieurs définitions de simulation, dont nous présentons ici une synthèse :

- La simulation est l'étude du comportement dynamique d'un système, grâce à un modèle que l'on fait évoluer dans le temps en fonction des règles bien définies, à des fins de prédiction.
- La simulation est une technique expérimentale appropriée à l'étude des systèmes de grande taille composés de plusieurs éléments en interaction. Elle permet ainsi de répondre à certains problèmes à chaque fois que l'expérimentation en grandeur nature se révèle impossible et/ou trop coûteuse.
- Simuler revient à créer une maquette informatique du dispositif étudié, en lui appliquant les règles du système réel, afin de disposer de façon virtuelle d'un prototype de l'usine. A l'aide du modèle défini, on peut mettre au point de véritables plans d'expérience pour faire des tests sur certaines décisions à prendre, certains paramétrages modifiés.

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.

- **L'objectif de simulation :**

L'objectif de simulation sur ordinateur est donc d'évoluer une abstraction d'un système au cours du temps afin d'aider à comprendre le fonctionnement et le comportement de ce système et à appréhender certaines de ses caractéristiques dynamiques pour évaluer certaine décisions.

I.3.2 Système de simulation :

Un système est un regroupement de plusieurs éléments ou entités en interaction. A n'importe quel instant, il est dans un état particulier défini par l'état de ses entités et des relations qui les relient. L'état du système change quand l'état de ses entités change.

Comme définition d'un système nous retenons celle donnée par l'AFCET et qui s'énonce comme suit :

"*Un système est une entité complexe traitée (eu égard à certaines finalités) comme une totalité organisée, formée d'éléments et de relations entre ceux-ci, les uns et les autres étant définis en fonction de la place qu'ils occupent dans cette totalité et cela de telle sorte que son identité soit maintenue face à certaines évolutions.* "

Mais plus particulier un système de simulation est constitué d'un ensemble d'outils qui facilitent la mise en œuvre, l'animation et l'observation des modèles. Le principal constituant d'un système de simulation est le langage de simulation qui permet de décrire en détail les modèles (leur partie statique et dynamique).

Les systèmes de simulation offrent certains nombres d'outils parmi lesquels nous pouvons citer :

En entrée on a:

- Interface graphique de modélisation : qui facilite la mise au point des modèles de simulation sous forme d'automates, blocs, organigrammes ou autres, la génération du programme de simulation est donc automatisée. Si le système de simulation n'offre pas cette interface, le programme de simulation doit être écrit par le programmeur.

En sortie on a :

- Le fichier trace de simulation : en général un fichier contenant l'historique des événements produits durant la simulation et leurs dates d'occurrence.

- Le fichier de résultats (statistiques) : un fichier contenant des statistiques sur les performances du modèle.

- Animation graphique des modèles : une visualisation graphique du comportement dynamique des modèles, en exploitant, en général, le fichier trace de simulation.

- Présentation graphique des résultats de simulation : cette technique exploite les statistiques collectées durant la simulation pour les afficher sous forme graphique (camemberts, courbes, histogrammes.).

- L'interactivité : l'interaction peut être guidée par le système qui passe le contrôle à l'utilisateur à chaque fois qu'une décision devait être prise ou que des informations complémentaires étaient nécessaires à la poursuite de la simulation. Les systèmes actuels sont fondés sur une interaction guidée par l'utilisateur, donc il peut à tout moment interrompre la simulation pour mettre à jour les paramètres ou les modèles et voir l'effet de ces changements instantanément à l'écran ou pour demander la valeur d'un attribut…etc.

I.3.3 Les avantages de simulation :

Lorsque la simulation est utilisée comme un outil de conception et d'analyse des systèmes réels, certains avantages peuvent être avancés :

- La simulation réduit les risques de conception de systèmes qui ne fournissent pas une flexibilité suffisante,
- Un modèle de simulation peut représenter des caractéristiques importantes d'un système et de manière réaliste; il peut notamment incorporer des interactions complexes pouvant exister entre différentes variables du système,
- Différentes alternatives de conception peuvent être facilement évaluées dans un environnement bien contrôlé,
- Un modèle de simulation est capable d'utiliser les mêmes indicateurs de performance qu'un système réel utilise, [Habchi, 2001]
- La simulation permet de mieux comprendre les anciens problèmes et en détecter de nouveaux,
- La simulation est une excellente méthode de comparaison, d'amélioration et d'appréciation des risques,
- La simulation réduit les risques de conception de systèmes qui ne fournissent pas une flexibilité suffisante,
- La simulation permet d'effectuer l'expérimentation sur le modèle sans perturber le bon fonctionnement du système réel,
- Possibilités de vérifier les solutions avec des modèles théoriques,
- Elle permet d'étudier les systèmes à échelle de temps variable, et l'impact des variables sur les performances du système,
- Etude d'un système sans les contraintes matérielles.

I.3.4 Les limites de simulation :

Les limites à considérer avant d'utiliser la simulation sont :

- La simulation ne peut pas optimiser la performance d'un système, elle peut seulement donner des réponses à des questions du genre : " Qu'est-ce qui se passe si ... ?" Comme on l'a déjà dit, une fois que le modèle est programmé et validé, la simulation fonctionne comme une boîte noire en fonction d'un scénario de fonctionnement. Elle ne fait donc que reproduire le comportement du système modélisé,
- Les résultats de simulation sont souvent complexes à interpréter (le modèle doit être ni trop précis parce que on étudie des phénomènes aléatoires).

I.4 Les étapes d'une étude de simulation :

La conduite d'une étude de simulation comprend douze étapes schématisées par l'organigramme de la Figure I.5 : [korichi, 2009]

1. **Formulation du problème** :

Tout projet de simulation commence d'abord par l'énoncé du problème à résoudre, cette phase est un préliminaire indispensable et d'une grande importance, puisque c'est dans cette étape que l'on doit définir précisément ce que l'on veut mettre en évidence avec la simulation, durant cette étape, il est suggéré qu'un ensemble d'hypothèses soit proposées, discutées et acceptées par l'analyste et le client. Enfin, il faut déterminer les questions auxquelles l'étude doit répondre.

2. **Fixation des objectifs :**

Le but de cette phase est d'identifier les objectifs de l'étude afin de fixer clairement les grandes lignes de réflexion, et de déterminer aussi les indicateurs de performance qui vont permettre de vérifier si l'on a atteint les objectifs qu'on s'est fixé. Ceci comprend :

- Les ressources qui seront requis sont essentiellement des ressources humaines et matérielles pour mettre en œuvre l'étude souhaitée.
- Quelles hypothèses veut-on vérifier ? et dans quel contexte ?
- Le personnel.
- Les divers scénarios qu'on veut investiguer, les sorties attendues,
- Les coûts de l'étude ainsi que les temps requis.

3. Construction du modèle :

Cette étape correspond à une première élaboration du modèle où les différentes entités qui composent le système sont identifiées. Il s'agit de réaliser une première abstraction générale (un ensemble de relations mathématiques et logiques) du système où le concepteur détermine les composants, les variables et les interactions entre composants, etc.

4. Collecte des données :

L'objectif de cette étape est de spécifier et de collecter les données dont on a besoin sur le système réel et les soumettre au client pour validation. Il faut remarquer que sur l'organigramme, les étapes de construction du modèle et de collecte des données sont placées au même niveau. Ceci signifie que ces deux étapes peuvent progresser en parallèle.

5. Codage :

A cette étape, il faut choisir une plate forme de simulation qui définira le langage à utiliser pour traduire le modèle conceptuel en modèle opérationnel (acceptable par l'ordinateur).
Le concepteur met en œuvre le modèle du système pour réaliser ses expérimentations.

6. Vérification :

Cette étape permet de vérifier l'exécution du modèle opérationnel dans le simulateur, il s'agit de s'assurer que le modèle s'exécute sans erreurs.

7. Validation :

La validation consiste à s'assurer que le modèle conceptuel est une représentation fidèle du système réel. Il s'agit en fait de savoir si le modèle peut être substitué au système réel pour le but de l'expérimentation.

La façon idéale de valider le modèle conceptuel, dans le cas où le système existe est de comparer ses sorties avec celles du système.

8. Conception d'un cadre d'expérimentation :

La simulation étant vérifiée et validée, elle peut être exécutée pour récupérer les résultats désirés et pour effectuer une analyse de sensibilité du modèle aux paramètres initiaux.

Parmi ces paramètres on peut citer :

- Durée de la simulation
- nombre de simulation à faire (Réplications)
- état initial du modèle
- règles de gestion des files d'attente
- etc.

9. Exécution de la simulation et Analyse des résultats :

Dans cette étape, le modèle programmé sera analysé et interprété par le simulateur qui délivre en sortie des résultats purement statistiques (moyenne, variance, écart type, minimum, maximum,...).L'analyse de ces résultats aura pour objectifs d'estimer les mesures de performances des scénarios qu'on a expérimentés.

10. Exécutions supplémentaires :

Il est suggéré que plusieurs exécutions soient réalisées sur le modèle de simulation. Une analyse de résultats d'exécution doit être faite pour déterminer si d'autres exécutions sont nécessaires ou d'autres scénarios non prévus doivent être expérimentés afin de s'assurer que le modèle répond bien aux objectifs de l'étude.

11. Documentation :

Le concepteur fournit une documentation complète sur le modèle et sa mise en œuvre car il faut toujours associer les résultats au modèle conceptuel et au modèle informatique qui eux mêmes dépendent des objectifs initiaux, en effet, il aura la possibilité de revoir toutes les alternatives prises en considération, les critères de comparaisons qui ont été utilisés et les recommandations faites par l'analyste. Ceci va l'aider énormément dans la prise de décision qui sera principalement basée sur les résultats fournis par la simulation et rapportés dans la documentation.

12. Implémentation :

Dans cette étape, l'analyste choisis parmi les solutions résultantes de simulation celle qui est la meilleure et la justifiera dans la documentation et la propose (et ne l'impose pas) au client. La décision de retenir cette solution pour une éventuelle implémentation reste donc une responsabilité du client, c'est ainsi que la qualité et la richesse de la documentation peut avoir une grande influence sur cette étape. De plus, l'implication du client tout au long de la conduite du projet augmentera les chances d'une implémentation de la solution retenue.

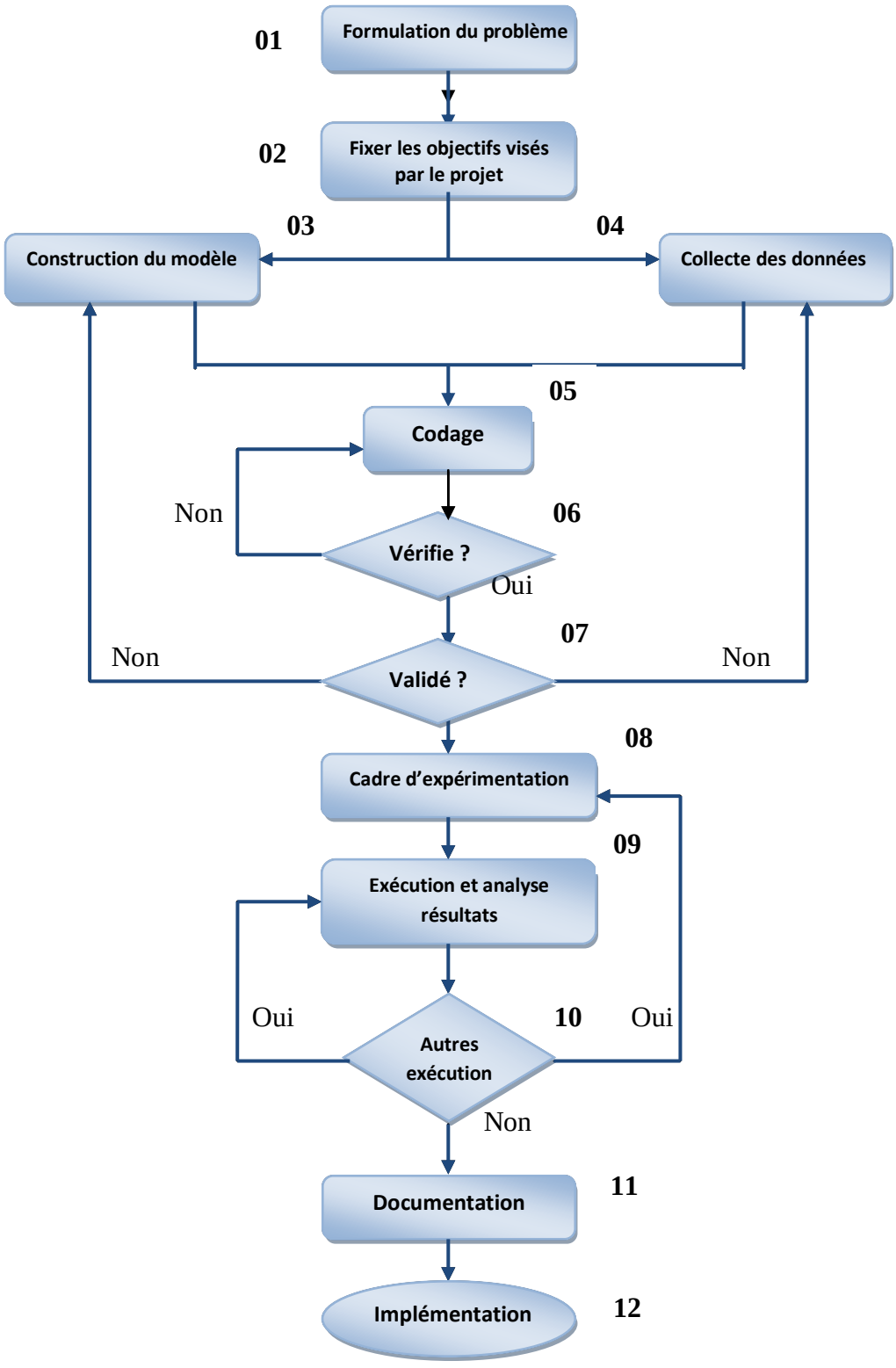

Figure I.5 : Les étapes d'une étude simulation. [Belattar, 2000]

Il faut noter que chaque étape peut être conduite par une ou plusieurs personnes et peut utiliser comme entrée, des résultats des étapes précédentes et que des documents sont produits à chaque étape (diagramme, rapport, code source, etc.) et partagés par différentes personnes et peuvent mener à plusieurs versions.

I.5 Les outils de simulation :

I.5.1 Définition :

On entend par outils de simulation les constituants d'un système, appelé généralement système de simulation, qui fournit les moyens de mettre en œuvre, d'animer et d'observer les modèles, de la même façon que le système d'exploitation d'un ordinateur fournit les moyens de mettre en œuvre et d'exécuter les programmes.[Belattar, 2000]

I.5.2 Classification des logiciels de simulation :

Plus de 100 logiciels de simulation sont disponibles sur le marché avec plusieurs classifications, certain auteurs classent ces logiciels en deux catégories :

- Les logiciels pour la simulation à événements discrets
- Les logiciels pour la simulation à variables continues.

Si le second type est applicable pour un flux continu de produit et d'information, le premier est beaucoup plus applicable dans les systèmes manufacturiers (composants, assemblages...). Mais dans cette étude nous présentons deux autres classifications :

- L'une basée sur la spécialisation du logiciel
- L'autre basée sur l'approche de modélisation. [Habchi, 2001]

I.5.2.1 Classification des langages en fonction de la spécialisation:

Comme nous l'avons déjà cité, le principal constituant d'un système de simulation est le langage de simulation, qui permet de décrire le modèle et les stimuli à lui appliquer au cours du déroulement de la simulation**.** Donc les langages de cette catégorie sont classés en quatre familles qui sont:

- Les langages universels ou évolués,
- Les langages généraux de simulation,
- Les langages spécialisés de simulation,
- Les langages dédiés.

a. Les langages universels ou évolués

Sont des langages de programmation répandus tels que :

VB, FORTRAN, PASCAL, C, ADA, C++, JAVA... ils ont l'avantage de permettre une simulation précise et fiable de tout type de système, quel que soit son niveau de complexité.

Ils permettent de développer des modèles fonctionnels ou logico-mathématiques, grâce à la richesse de programmation qu'ils offrent. Cependant, ils nécessitent des compétences poussées et affirmées en simulation, en informatique et en génie industriel. Ils présentent l'inconvénient du temps de développement et par conséquent du coût de réalisation. Les modèles sont dédiés et généralement inexploitables dans des extensions futures.

b. Les langages généraux de simulation

Sont des langages enrichis par des primitives dédiées à la simulation en général, comme les générateurs de nombres aléatoires, les processeurs de base pour la programmation de certaines entités... ils ont ainsi l'avantage de présenter une couche de base permettant de développer d'autres primitives pour la réalisation de modèles de simulation dédiés. Parmi ces langages on peut citer : *SIM++* et *MODSIMII*.

c. Les langages spécialisés de simulation

Présentent un compromis entre la rapidité de développement du modèle et la rigidité liée aux concepts utilisés, les concepts utilisés sont généralement basés sur la théorie des files d'attente et des processus stochastiques. La modélisation se fait à l'aide d'éléments symboliques, les modèles résultants sont de type réseau *(SLAM)*, des blocs représentatifs de processus *(ARENA, GPSS,WITNESS, SIMNET II)*, ou des diagrammes de cycle d'activités *(HOCUS)*, ces logiciels nécessitent une formation et une expérience, et sont souvent utilisés par les consultants et concepteurs qui ont souvent recours à la simulation pour des systèmes différents. L'utilisation de ces langages est conseillée quand une évolution du modèle est nécessaire.

d. Les langages dédiés

Sont adaptés aux systèmes ayant des caractéristiques identiques, et sont dédiés à un type de système au fonctionnement bien défini ou à une classe de problèmes. La modélisation est basée sur l'aspect topologique (représentation du système physique) à travers des éléments représentatifs du système réel. Ces outils très conviviaux sont limités à un usage répétitif dans un domaine restreint ; l'avantage principal de ce type d'outil est la facilité d'utilisation, alors que l'inconvénient majeur est la rigidité des modèles. Parmi les langages dédiés, citons à titre d'exemple *MAP/1* et *SIMFACTORY*.

I.5.2.2 Classification des langages en fonction de l'approche de modélisation :

Dans cette approche, nous pouvons classer les langages en trois catégories: les langages orientés processus, les langages pilotés par les données et les langages orientés objets :

- **Les langages orientés processus**

Tels que *SLAM II* et *SIMAN* ont respectivement les deux environnements graphiques :*AweSim* et *ARENA*. Ces langages ont réduit avec succès les frontières de programmation et ont probablement fait que la modélisation en simulation soit accessible à plusieurs utilisateurs non informaticiens. Généralement, les langages orientés processus appartiennent à la catégorie des langages spécialisés présentés précédemment.

A. Les langages pilotés par les données

Représentent une abstraction de haut niveau du système. *SIMFACTRY* et *PROMODEL* sont des exemples de langages où le modèle de fond est difficilement modifiable. Ces langages comprennent des objets prédéfinis capables de réaliser une série d'opérations. La modélisation consiste à sélectionner, interconnecter et renseigner ces objets (station, convoyeur...) qui représentent au mieux le système réel. Ce type de langages nécessite peu de programmation de la part de l'utilisateur, mais en revanche, ils sont spécifiques et ont une flexibilité réduite. La plupart des langages pilotés par les données, utilisent un calendrier d'événements qui stocke l'état du système à travers les valeurs associées aux attributs.

B. Les langages orientés objets

Les objets physiques et logiques du système réel(produits, ressources, gammes...) sont représentés par des objets informatiques. L'approche objet cherche à combler l'écart qui existe entre un modèle de simulation classique et le système à étudier. Parmi les langages orientés objets, nous citons *SIMULA, CLOS et MODSIM*.

II.5.3 Les critères de choix d'un outil de simulation :

Le problème majeur réside dans le choix de l'outil ou du langage de simulation. Nous pouvons présenter ici dix critères classés par ordre de priorité qui sont : [BOUROUIS, 2010]

1) Interface H/M conviviale
2) Possibilité de disposer d'une B.D. pour organiser les données
3) Outil de mise au point (Débogueur)
4) Interaction via une souris
5) Section d'assistance dans la documentation du L.S
6) Sauvegarde des modèles de simulation
7) Sauvegarde des résultats
8) Entrée des données et des commandes par clavier
9) Librairie de modules réutilisables
10) Affichage

I.6 Les approches de modélisation :

Les systèmes qu'on modélise ou/et simule peuvent être classés selon plusieurs critères :

- Si l'état du système change avec le temps, il est qualifié de *dynamique*, sinon il est *statique*.
- Si l'état prochain d'un système est prédictible à partir de l'état actuel, alors il est *déterministe*, sinon il est *indéterministe* ou *stochastique*.
- Si le système atteint un état final qui ne changera plus, nous disons que ce système *termine*, le cas échéant le système *ne se termine pas* et change d'état de façon perpétuelle.
- Un système *continu* se caractérise par un changement continu d'état, alors que l'état d'un système *discret* change de façon discrète.

Dans ce mémoire nous nous intéressons à la modélisation et la simulation à évènements discrets car elle permet de ne pas réévaluer l'état du modèle lorsque ce n'est pas jugé nécessaire.

I.6.1. La simulation à évènements discrets :

La simulation à événements discrets consiste à répertorier les différents types d'événements (généralement inconditionnels) et à décrire la procédure de changement d'état correspondante au déclenchement de chaque type d'événement, aucune modification n'est censée avoir lieu entre l'occurrence de deux événements successifs, les événements son ordonnés dans un échéancier selon l'ordre chronologique de leur date d'occurrence.

Donc un modèle à événements discrets est caractérisé par une évolution (changement d'état) discontinue dans le temps, il se compose de deux parties, l'une statique liée à la structure du système et l'autre dynamique liée à l'évolution du système avec le temps [Belattar, 2000]

- **Partie statique :**

Un modèle à événements discrets est composé d'objets ou entités (pièces, machines, etc.) avec lesquelles s'effectuent au cours du temps, des services qui peuvent être actifs (activités ou opérations) ou passifs (attente), et des relations entre ces objets (ex: possibilité pour une pièce de passer sur une machine). On distingue en général deux types d'objets qui sont les clients et les ressources.

Les clients ou usagers : Sur lesquels sont effectués les services (pièces dans un atelier, clients dans un magasin, informations dans un réseau de transmission, etc.) ; souvent, les clients sont des objets circulants (en particulier dans les systèmes de production).

Les ressources : Sont nécessaires pour effectuer les services ; ces ressources peuvent être composées d'une ou plusieurs unités (nombre de machines identiques, de personnel ayant la même compétence, des chariots, des places dans une file d'attente, etc.).

- **Partie dynamique :**

L'aspect dynamique d'un modèle réfère aux mécanismes de changements d'état, lorsque le temps évolue, des interactions se produisent entre les objets qui composent le modèle et ceux-ci changent alors d'état, ces changements d'état peuvent se faire de façon continue, discrète ou une combinaison des deux, c'est ainsi qu'on parle de modèles "continus", modèles à "événements discrets", et modèles "combinés événements discrets- continus".

Le contrôle du temps dans les modèles de simulation se fait soit à pas constant (synchrone) soit à pas variable (asynchrone). La première technique est simple et facile à mettre en œuvre, mais la sélection du pas est très difficile, avec un pas trop grand le risque de sauter des événements persiste et avec un pas trop petit un overhead est toujours présent, c'est à dire le risque de faire plusieurs tests inutiles. La deuxième technique, bien qu'elle soit plus difficile à programmer, s'avère plus économique en temps de calcul, l'incrémentation du temps de simulation se fait d'une date d'événement à l'autre.

I.6.1.1 Les approches de modélisation à événements discrets :

La simulation à événements discrets consiste à reproduire, événement par événement, l'évolution d'un système au cours du temps. L'objectif est donc de faire reproduire à chaque entité du modèle les changements d'état de l'entité correspondante du système réel, il existe trois grandes approches pour modéliser un système à évènements discrets, qui sont :

- L'approche par évènement: Event scheduling, Next Event.
- L'approche par activité : Activity Scanning.
- L'approche par processus: Process interaction.

Ces trois approches de modélisation sont caractérisées par le fait qu'elles conduisent à des modèles ayant une structure hiérarchique à 3 niveaux : [Belattar, 2000]

- **Niveau 1 :** Correspond au programme de contrôle de la simulation appelé aussi exécutif ou noyau.
- **Niveau 2 :** Correspond aux opérations décrivant la dynamique du système simulé, du point de vue de l'utilisateur, ce niveau constitue le modèle de simulation proprement dit.
- **Niveau 3 :** Comprend un ensemble de routines utilitaires.

L'approche par processus est la plus fine, elle est utilisée par la plupart des langages de simulation, puisqu'un processus est une séquence d'opérations pouvant être des évènements ou des activités, cette approche permet de combiner les avantages des autres approches.

I.6.1.2 Composantes d'un modèle à évènements discret :

Les modèles de simulation à événements discrets ont tous plusieurs composantes en communs et l'organisation logique de ces composantes permet d'aider le codage, la mise au point, et la mise à jour du programme traduisant le modèle de la simulation.

En particulier, les composantes suivantes se retrouvent dans la plupart des modèles de simulation à événements discrets adoptant l'approche par « événement »:

- ✓ **L'état du système:** Défini par une collection de variables d'état décrivant l'état du système à un temps particulier.
- ✓ **L'horloge de simulation:** Variable indiquant la valeur courante du temps de la simulation.
- ✓ **La liste des événements:** Une liste qui contient les dates d'occurrence des événements devant avoir lieu dans le futur.
- ✓ **Des compteurs statistiques:** Variables utilisées pour collecter des informations statistiques sur les performances du système.
- ✓ **La routine d'initialisation :** Sous-programme servant à initialiser le modèle de simulation au temps zéro.
- ✓ **Routine de Gestion du temps (contrôle du temps):** Sous-programme qui détermine le prochain événement à partir de la liste des événements et initialise l'horloge de simulation avec la date d'occurrence de cet événement courant.
- ✓ **Routines associées aux événements:** Sous-programme qui met à jour l'état du système quand un type particulier d'événement se produit (il y a une routine d'événement pour chaque type d'événement).
- ✓ **Bibliothèque de routines utilitaires:** ensemble de sous programmes utilisés pour générer des variables aléatoires identifiées comme faisant partie du modèle de simulation.
- ✓ **Générateur des résultats:** Sous-programme qui calcule des estimations (à partir des compteurs statistiques) des mesures de performance désirées et génère en fin de simulation un rapport.
- ✓ **Le programme principal:** Chargé d'initialiser l'état du système au début de la simulation, toutes les variables utilisées au cours de la simulation, il invoque la routine de gestion du temps pour déterminer le prochain événement devant avoir lieu et passe le contrôle à la routine associée à cet événement, il contrôle aussi si la fin de la simulation est atteinte et invoque dans ce cas le générateur de résultats.

I.7 L'environnement de la simulation intelligente :

Un environnement de simulation intelligente[2] est un système de connaissances de grande intégration, qui se compose de plusieurs systèmes de raisonnement symbolique (Lisp, Prolog, etc.) et des logiciels de simulation numérique, au fait, il utilise les techniques d'intelligence artificielle.

L'environnement de modélisation intelligente est un système flexible, intégré et à base de connaissances capable d'apprendre, d'autocorrection et d'auto coordination, Il est capable aussi de s'adapter aux changements nécessaires, l'objectif de ce dernier est de chercher les meilleures solutions au bon moment en se référant aux objectifs souhaités.
Des environnements de simulation intelligente sont également utilisés pour aider l'utilisateur dans la composition d'un scénario, pour lui expliquer ce qui s'y passe, ces moyens devraient permettre d'atténuer les imperfections de communication et d'améliorer la compréhension d'une situation calculée.

Ce dernier avantage est assuré grâce à l'intégration d'un module explicatif, il est souhaitable pour éclaircir à l'utilisateur sur la véracité d'une transition, il vise à donner la possibilité à l'utilisateur de poser des questions chaque fois qu'il est en face d'une situation qu'il ne perçoit pas bien ou qu'il pense qu'elle aurait pu se dérouler autrement, cela permet de :

- Mieux comprendre les modèles de simulation.
- Augmenter la confiance dans les résultats de la simulation.
- Servir comme outil d'apprentissage de la simulation.
- Aider dans le débogage des modèles de simulation.

Donc nous constatons que les demandes d'explication sont variées et nécessitent plusieurs informations et connaissances en plus de la trace d'exécution que l'utilisateur ne peut pas avoir.
Donc, il est nécessaire de concevoir un module intelligent réservé à l'explication, qui à travers un raisonnement sur les modèles de simulation (sous forme de connaissances) pourrait satisfaire les demandes d'explications et de justifications.

[2]Un cadre pour la conception de l'environnement de simulation intelligent est détaillé dans l'article d'Azadeh et Ghaderi [Azadeh et Ghaderi, 2006].

I.8 Conclusion

L'explication en simulation est un sujet de recherche important surtout pour le domaine industriel. La compréhension du résultat de simulation grâce à la simulation intelligente permet de gagner le temps, l'effort et permet de prendre la meilleur décision au bon moment, mais la tâche de l'explication proprement dite est très délicate et nécessite une grande expérience, cette expérience permet de mieux comprendre les modèles de simulation, qui a pour conséquence d'augmenter la confiance dans les résultats de simulation.

Pour se faire, il faut trouver les moyens et les techniques appropriées pour sa capitalisation et sa réutilisation.

Chapitre II

Le domaine de l'ontologie

II.1 Introduction :

L'utilisation des connaissances en informatique a pour but de ne plus faire manipuler en aveugle des informations à la machine mais de permettre un dialogue et une coopération entre le système et les utilisateurs. Pour cela, le système doit avoir accès non seulement aux termes utilisés par l'être humain mais également à la sémantique qui lui associée, afin qu'une communication efficace soit possible, les ontologies visent à représenter cette connaissance en étant à la fois interprétables par l'homme et par la machine.

Actuellement, les ontologies constituent un enjeu stratégique dans la représentation et la modélisation des connaissances, récemment, elles ont été introduites pour formaliser les connaissances dans le domaine de modélisation et de simulation intelligente, elles définissent les primitives indispensables pour leur représentation, ainsi que leur sémantique dans un contexte particulier, elle permettent aussi la recherche d'informations, le traitement du langage naturel, l'intégration intelligente d'informations et la gestion des connaissances. Elles fournissent une connaissance partagée et commune sur ce domaine qui peut être échangée entre des personnes et des systèmes hétérogènes.

A travers cette étude sur les ontologies, nous recherchons à savoir comment en tirer profit afin de formaliser les connaissances que l'on a sur le domaine de modélisation et de simulation intelligente dans le respect du principe de plasticité.

Dans ce chapitre, nous attachons à décrire les différentes définitions du concept d'ontologie en fonction du contexte dans lequel il est utilisé, ainsi que les différents éléments constituant l'ontologie ; ensuite, nous présentons les typologies des ontologies ainsi que les diverses classifications existantes. Un aperçu rapide des formalismes de représentation d'ontologies est ensuite donné, puis, nous passons en revue les différentes étapes intervenant dans la construction des ontologies. Enfin pour implémenter une ontologie, il convient d'utiliser un langage de spécification, ce travail est facilité par de nombreux outils assistant l'utilisateur en phase d'opérationnalisation.

II.2 Notion d'ontologie et d'ingénierie ontologique :

II.2.1 Origine de l'ontologie :

Ontologie est un terme qui a tout d'abord été défini en Philosophie comme une branche de la Métaphysique qui s'intéresse à l'existence, à l'être en tant qu'être et aux catégories fondamentales de l'existant. En effet, ce terme est construit à partir des racines grecques ontos qui veut dire ce qui existe, l'Etre, l'existant, et logos qui veut dire l'étude, le discours, d'où sa traduction par l'étude de l'Etre et par extension de l'existence [Amardeilh, 2007].

Dans la philosophie classique, l'Ontologie correspond à ce qu'Aristote appelait la Philosophie première (protè philosopha), c'est-à-dire la science de l'être en tant qu'être, par opposition aux philosophies secondes qui s'intéressaient à l'étude des manifestations de l'être (les existants) [Psyché, 2007] .

Bien que ce sont les Grecs qui ont inventé cette science, ils ne l'avaient pas appelé «Ontologie», le terme étant beaucoup plus récent (XVIIe siècle) que la discipline qu'il désigne, La discipline elle-même a évolué vers une voie imprévisible, il y a seulement une vingtaine d'années. En effet, d'après [Valéry et al, 2003] l'ontologie a été abordée dans le domaine de l'intelligence artificielle pour la première fois par John McCarthy qui reconnut le recoupement entre le travail fait en Ontologie Philosophique et l'activité de construire des théories logiques de systèmes d'intelligence artificielle, il affirmait déjà en 1980 que les concepteurs de systèmes intelligents fondés sur la logique devraient d'abord énumérer tout ce qui existe.

II.2.2 Définitions :

Ontologie : partie de la métaphysique qui s'attache à l'étude ou à la théorie de l'être dans son essence, indépendamment des phénomènes de son existence. Plusieurs autres définitions du concept ontologie ont été proposées dans le domaine de l'Intelligence Artificielle, ces définitions sont souvent des raffinements de définitions déjà proposées et/ou sont complémentaires avec elles.

Dans le cadre de l'intelligence artificielle, Neeches et ses collègues [Neeches, 1991]furent les premiers à proposer une définition à savoir : «une ontologie définit les termes et les relations de base du vocabulaire d'un domaine ainsi que les règles qui indiquent comment combiner les termes et les relations de façon à pouvoir étendre le vocabulaire».

En 1993 [3], Gruber propose la définition suivante : «spécification explicite d'une conceptualisation»

Par explicite, il faut entendre que le type de concepts utilisés ainsi que leurs contraintes d'utilisation sont définies de façon explicite, qui est jusqu'à présent la définition la plus citée dans la littérature en intelligence artificielle.

Cette définition a été modifiée légèrement par Borst [Borst, 1997] comme «spécification formelle d'une conceptualisation partagée».

Ces deux définitions sont regroupées dans celle de Studer comme «spécification formelle et explicite d'une conceptualisation partagée».

- **Formelle:** l'ontologie doit être lisible par une machine, ce qui exclut le langage naturel.
- **Explicite:** la définition explicite des concepts utilisés et des contraintes de leur utilisation.
- **Conceptualisation:** le modèle abstrait d'un phénomène du monde réel par identification des concepts clefs de ce phénomène.
- **Partagée:** l'ontologie n'est pas la propriété d'un individu, mais elle représente un consensus accepté par une communauté d'utilisateurs.

[3] Dans leur article [Gruber, 1993]

Pour Guarino & Giaretta[4] «Une ontologie est une spécification rendant partiellement compte d'une conceptualisation». Swartout et ses collègues [Swartout et al, 1997] la définissent comme suit :

« Une ontologie est un ensemble de termes structurés de façon hiérarchique, conçue afin de décrire un domaine et qui peut servir de charpente à une base de connaissances». Cette définition se base sur le fait qu'ils construisent des ontologies de connaissances spécifiques à des domaines d'expertise.

La même notion est également développée par Gomez [Gomez 1999] comme : « une ontologie fournit moyens de décrire de façon explicite la conceptualisation des connaissances représentées dans une base de connaissances. ».En conclusion nous pouvons définir une ontologie comme suit :

Une ontologie est une spécification formelle d'une conceptualisation commune d'un domaine indépendamment d'une application particulière, elle est utilisée par des personnes, des bases de données et des applications ayant besoin de partager des informations portant sur un domaine [Lassila et Mcguiness, 2001].L'ontologie est donc le support de l'acquisition des connaissances et elle est aussi un outil utile pour interfacer les agents logiciels et les agents humains.

Pour communiquer, les agents doivent donc partager une ontologie et pour qu'une ontologie soit une connaissance partagée, elle doit être explicite et être exprimée dans un langage ou un formalisme partagé.

II.2.3 Notion d'ingénierie ontologique :

Historiquement, l'ingénierie ontologique a émergé de l'ingénierie des connaissances vu la détection d'un besoin de passer à une ingénierie s'appuyant plus solidement sur des fondements théoriques et méthodologiques, afin d'améliorer la conception des systèmes intelligents, elle permet de spécifier la conceptualisation d'un système, c'est-à-dire, de lui fournir une représentation formelle des connaissances qu'il doit acquérir, sous la forme de connaissances déclaratives exploitables par un agent.

[4]Dans l'article publié en 1995[Guarino et Giaretta, 1995]

L'ingénierie ontologique peut être définie comme une thématique visant à proposer des concepts, des méthodes, des outils et des langages dédiés à la définition, au développement et au déploiement d'ontologies exploitables au sein d'un (ou de plusieurs) Artefact.

II.3 Les objectifs d'ontologie:

L'ontologie permet de fournir un vocabulaire conceptuel dans un domaine donnée afin de permettre la communication au sujet de ce domaine, elle permet de définir les concepts utilisables pour décrire une connaissance consensuelle ainsi que les relations qui existent entre les concepts et leurs contraintes d'utilisation, donc elle fournit un cadre unificateur des concepts manipulés ce qui favorise la communication entre les membres de l'entreprise. L'ontologie peut aussi permettre l'interopérabilité entre différentes méthodes, paradigmes, langage ou outils, elle peut permettre les multipoints de vues sur des connaissances.

En fin, elle permet de favoriser la réutilisabilité, la fiabilité et la spécification d'un système informatique.

II.4 Les composantes d'une ontologie :

Comme nous l'avons abordé, les ontologies fournissent un vocabulaire commun d'un domaine et définissent la signification des termes et des relations entre elles, les connaissances dans les ontologies sont principalement formalisées en utilisant cinq types de composants [Gomez et Benjamin, 1999] à savoir :

- Les concepts (ou classes),
- Les relations (ou propriétés),
- Les fonctions,
- Les axiomes (ou règles),
- Les instances (ou individus).

Ces composants sont illustrés dans la figure suivante.

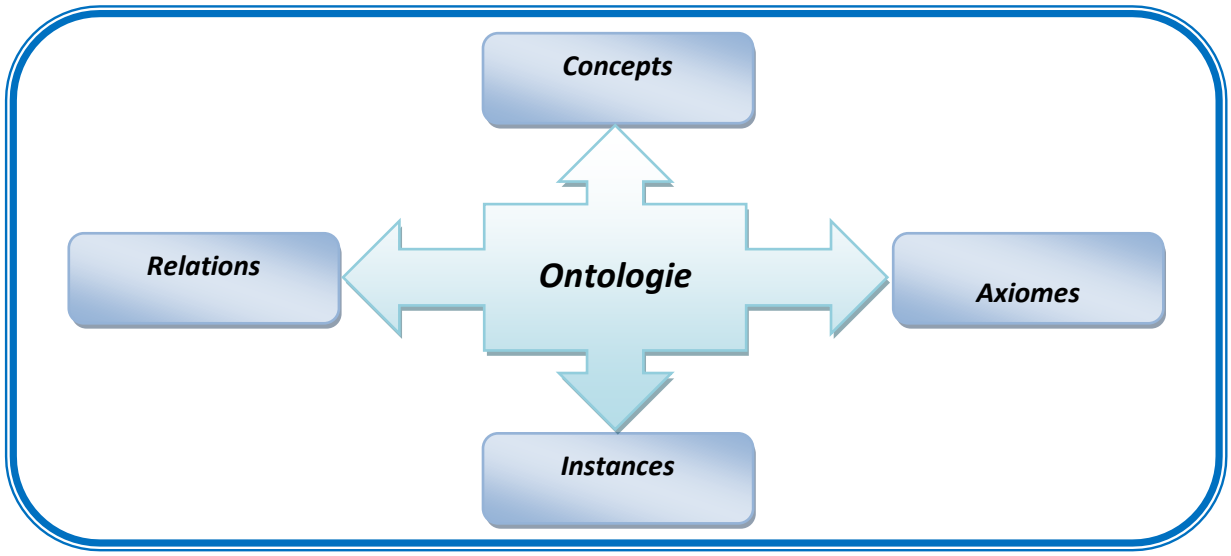

Figure II.1 : Les Composants de l'ontologie.

II.4.1 Les concepts :

Un concept est un constituant de la pensée (un principe, une idée, une notion abstraite) sémantiquement évaluable et communicable. L'ensemble des propriétés d'un concept constitue sa compréhension ou son intention et l'ensemble des êtres qu'il englobe son extension. Ces concepts selon Gomez peuvent être classifiés selon trois dimensions :

- Niveau d'abstraction (concrets ou abstraits).
- Atomicité (élémentaires ou composés).
- Niveau de réalité (réelle ou fictive).

1) Les propriétés portant sur un concept :

- La généricité: un concept est générique s'il n'admet pas d'extension.
- L'identité: un concept porte une propriété d'identité si cette propriété permet de conclure quant à l'identité de deux instances de ce concept.
- La rigidité: un concept est dit rigide si toute instance de ce concept en reste instance dans tous les mondes possibles. Exemple: humain est un concept rigide, étudiant est un concept non rigide.

- L'anti-rigidité: un concept est anti-rigide si toute instance de ce concept est essentiellement définie par son appartenance à l'extension d'un autre concept.

Exemple : étudiant est un concept anti-rigide car l'étudiant est avant tout un humain.

- L'unité: un concept est un concept unité.

2) Les propriétés portant sur deux concepts :

- L'équivalence: deux concepts sont équivalents s'ils ont la même extension.

Exemple : étoile du matin et étoile du soir.

- La disjonction: (on parle aussi d'incompatibilité) deux concepts sont disjoints si leurs extensions sont disjointes.

Exemple : homme et femme.

- La dépendance: Un concept C1 est dépendant d'un concept C2 si pour toute instance de C1 il existe une instance de C2 qui ne soit ni partie ni constituant de l'instance de C1. Exemple : parent est un concept dépendant de enfant (et vice-versa).

II.4.2 Les relations :

Elles représentent des interactions entre les concepts, elles permettent de construire des représentations complexes de la connaissance du domaine, elles établissent des liens sémantiques binaires, organisables hiérarchiquement.

Les propriétés intrinsèques à une relation :

- Les propriétés algébriques : symétrie, réflexivité, transitivité…
- La cardinalité : nombre possible de relations de ce type entre les mêmes concepts (ou instances de concept), l es relations portant une cardinalité représentent souvent des attributs. Exemple: une pièce a au moins une porte.

II.4.3 Les fonctions :

Elles présentent des cas particuliers de relations dans lesquelles le nième élément de la relation est unique pour les n-1 éléments précédents. [Gomez et Benjamin, 1999] Formellement, les fonctions sont définies telles que : $F : c_1 * c_2 * ... * c_{n-1} * c_n$.

II.4.4 Les axiomes (ou Règles) :

Les axiomes sont des expressions qui sont toujours vraies, ils ont pour but de définir dans un langage logique la description des concepts et des relations permettant de représenter leur sémantique. Ils représentent les intentions des concepts et des relations du domaine et, de manière générale, leurs inclusions dans une ontologie peuvent avoir plusieurs objectifs :

- Définir des restrictions sur la valeur des attributs.
- Définir les arguments d'une relation…

II.4.5 Les instances (ou individus) :

Elles constituent la définition extensionnelle de l'ontologie ; elles sont utilisées pour représenter des éléments dans un domaine.

II.5 Classification des ontologies :

Les ontologies peuvent être subdivisées en plusieurs dimensions, parmi celles-ci nous en examinerons cinq, à savoir :

- Selon l'objet de la conceptualisation.
- Selon le niveau de granularité.
- Selon le niveau de formalisation de la représentation des connaissances.
- Selon le niveau de la représentation des connaissances.
- Selon l'information dont l'ontologie a besoin et la richesse de sa structure interne.

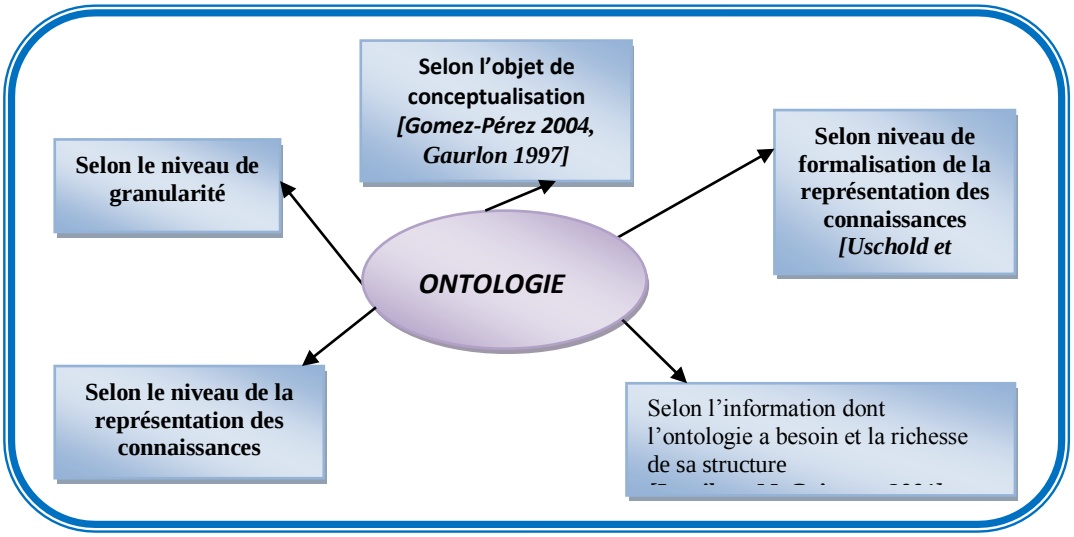

Figure II.2: Typologies des ontologies selon cinq dimensions.

II.5.1 Selon l'objet de la conceptualisation :

Dépendamment de leur objet de conceptualisation, les ontologies sont classifiées de la manière suivante :

- **Ontologie de représentation de connaissances :** Elle modélise les représentations primitives utilisées pour la formalisation des connaissances sous un paradigme donné. Comme exemple : l'ontologie de Frame qui intègre les primitives de représentation des langages base de frames : classes, instances, facettes, propriétés/slots, relations, restrictions, valeurs permises, etc.

- **Ontologie supérieure ou de Haut niveau :** Cette ontologie est une ontologie générale. Son sujet est l'étude des catégories des choses qui existent dans le monde, qui sont les concepts de haute abstraction tels que : les entités, les événements, les états, les processus, les actions, le temps, l'espace, les relations, les propriétés. L'ontologie de haut de niveau est fondée sur : la théorie de l'identité, la méréologie (theory of whole and parts rôle) et la théorie de la dépendance.

- **Ontologie Générique/ méta-ontologies** : Egalement appelée noyau ontologique qui modélise des connaissances moins abstraites que celles véhiculées par l'ontologie de haut niveau mais assez générales néanmoins pour être réutilisées à travers différents domaines. Cette ontologie inclue un vocabulaire relatif aux choses, évènements, temps, espace, causalité, comportement, fonction, etc.

- **Ontologie du Domaine :** Cette ontologie exprime des conceptualisations spécifiques à un domaine, elle est réutilisable pour plusieurs applications de ce domaine. Elle fournit les concepts et les relations permettant de couvrir les vocabulaires, activités et théories de ces domaines. Selon Mizoguchi, l'ontologie du domaine caractérise la connaissance du domaine où la tâche est réalisée. Par exemple, dans le contexte du e-learning, le domaine peut être celui de la formation.

- **Ontologie de Tâches :** L'ontologie de tâches fournit un vocabulaire systématisé des termes employés pour résoudre des problèmes liés aux tâches qui peuvent être ou non du même domaine. Elle fournit un ensemble de termes au moyen desquelles nous pouvons décrire généralement comment résoudre un type de problèmes, elle inclut des noms, des verbes et des adjectifs génériques dans les descriptions de tâches.

- **Ontologie d'application :** C'est l'ontologie la plus spécifique, elle contient des concepts dépendants d'un domaine et d'une tâche particulière, elle est spécifique et non réutilisable. Ces concepts correspondent souvent aux rôles joués par les entités du domaine lors de l'exécution d'une certaine activité [Hernandez, 2005].

- **Ontologie de méthodes :** Ce type d'ontologie modélise les définitions des concepts et des relations pertinentes pour le processus de raisonnement afin d'effectuer une tâche spécifique.

II.5.2 Selon le niveau de granularité (détail) :

Par rapport au niveau de détail utilisé lors de la conceptualisation de l'ontologie en fonction de l'objectif opérationnel envisagé pour l'ontologie, deux catégories au moins peuvent être identifiées [Psyché 2007] :

❖ **Granularité fine :** ce niveau correspond à des ontologies très détaillées, elles possèdent ainsi un vocabulaire plus riche capable d'assurer une description détaillée des concepts pertinents d'un domaine ou d'une tâche. Ce niveau de granularité peut s'avérer utile lorsqu' il s'agit d'établir un consensus entre les agents qui l'utilisent.

❖ **Granularité large :** ce niveau correspond à des vocabulaires moins détaillés. Par exemple : les scénarios d'utilisation spécifiques où les utilisateurs sont déjà préalablement d'accord à propos d'une conceptualisation sous-jacente. Les ontologies génériques possèdent une granularité large, du fait que les concepts qu'elles traduisent sont normalement raffinés subséquemment dans d'autres ontologies de domaine ou d'application.

II.5.3 Selon le niveau de formalisation de la représentation:

Par rapport au niveau de formalisation de la représentation des connaissances de l'ontologie, on peut distinguer quatre sortes d'ontologies : [Psyché, 2007]

- **Informelle :** l'ontologie est exprimée en langage naturel (sémantique ouverte). Cela peut permettre de rendre plus compréhensible l'ontologie pour l'utilisateur mais peut rendre plus difficile la vérification de l'absence de redondances ou de contradictions.
- **Semi-informelle :** l'ontologie est exprimée dans une forme restreinte et structurée du langage naturel, cela permet d'augmenter la clarté de l'ontologie tout en réduisant l'ambiguïté.
- **Semi-formelle :** l'ontologie est exprimée dans un langage artificiel défini formellement.
- **Formelle :** l'ontologie est exprimée dans un langage artificiel disposant d'une sémantique formelle ainsi que des théorèmes permettant de prouver ses propriétés, l'intérêt d'une ontologie formelle est la possibilité d'effectuer des vérifications telles que la complétude, la non-redondance, la consistance, la cohérence, etc.

II.5.4 Selon le niveau de la représentation des connaissances :

L'appellation de niveaux de représentation des connaissances ainsi que leur interprétation peuvent varier selon les auteurs. À titre de comparaison, nous décrivons les typologies de Mizoguchi et de Bachimont, d'une part, Mizoguchi propose une typologie sur trois niveaux très axée sur le processus d'ingénierie ontologique :

- **Niveau 1 (ou niveau conceptuel) :** Il est spécifié en faisant abstraction de toute contrainte informatique et sert donc de support à l'acquisition des connaissances., c'est une collection structurée de termes. La tâche fondamentale dans la construction d'une ontologie est l'articulation du monde d'intérêt, c'est-à-dire l'extraction des concepts et leur identification dans des hiérarchies de type is-a, cette étape est indispensable pour que l'on puisse parler d'ontologie. À ce stade, la modélisation étant informelle, une

spécification en langage naturel complétée de graphiques est acceptable et des définitions minimales des concepts sont faites.

- **Niveau 2 (ou niveau formel) :** Il est formalisé au moyen d'un langage de représentation interprétable par une machine ; en plus des définitions du niveau 1, les définitions formelles sont ajoutées pour empêcher des interprétations inattendues des concepts. De même, les relations ou contraintes nécessaires sont également traduites en langage machine pour former un ensemble d'axiomes ; a ce stade, les relations sont plus riches que celles du niveau précédent. Les définitions sont déclaratives et formelles pour permettre aux ordinateurs de les interpréter, l'interprétation d'une ontologie à ce niveau permet aux ordinateurs de répondre aux questions sur les modèles construits à partir de l'ontologie. La plupart des efforts de construction de l'ontologie aboutissent à ce niveau.

- **Niveau 3 (ou niveau opérationnel) :** Il est encodé au moyen d'un langage de programmation dont la spécification obtenue est exécutable. L'ontologie est exécutable dans le sens où les modèles construits à partir de l'ontologie fonctionnent en utilisant des modules fournis par quelques-uns des codes abstraits associés aux concepts dans l'ontologie. Ainsi, elle peut répondre à des questions à propos de la performance du temps d'exécution des modèles.

D'autre part, Bachimont propose une typologie de trois niveaux très axée sur la construction de la signification :

- **Niveau sémantique (ou interprétatif) :** tous les concepts (caractérisés par un libellé) doivent respecter les quatre principes différentiels :
  - Communauté avec l'ancêtre.
  - Différence (spécification) par rapport à l'ancêtre.
  - Communauté avec les concepts frères (situés au même niveau).
  - Différence par rapport aux concepts frères (sinon il n'aurait pas lieu de le définir).

Ces principes correspondent à l'engagement sémantique qui s'assure que le libellé de chaque concept aura un sens univoque et non contextuel associé. Deux concepts sémantiques sont identiques si l'interprétation du terme/libellé à travers les quatre principes différentiels aboutit à un sens équivalent.

➢ **Niveau formel (ou référentiel) :** En ajout aux caractéristiques énoncées au niveau précédent, les concepts référentiels se caractérisent par un terme/libellé dont la sémantique est définie par une extension d'objets. L'engagement ontologique spécifie les objets du domaine qui peuvent être associés au concept conformément à sa signification formelle. Deux concepts formels sont identiques s'ils possèdent la même extension. Exemple : Les concepts « d'étoile du matin » et « d'étoile du soir » associés à Venus.

➢ **Niveau opérationnel (ou computationnel) :** En plus des caractéristiques énoncées au niveau précédent, les concepts au niveau opérationnel sont caractérisés par les opérations qu'il est possible de leur appliquer pour générer des inférences ou engagement computationnel. Deux concepts opérationnels sont identiques s'ils possèdent le même potentiel d'inférence.

II.5.5 Selon l'information dont l'ontologie a besoin et la richesse de sa structure interne :

Lassila et Mcguiness [Lassila et Mcguiness, 2001] proposent une classification des ontologies selon l'information dont l'ontologie a besoin et la richesse de sa structure interne. Cette classification consiste en un continuum allant d'ontologies légères aux ontologies lourdes:

◆**Vocabulaire contrôlé:** C'est un ensemble de termes définis par un groupe de personnes ou une communauté, la signification des termes n'est pas forcément définie et il n'y a pas d'organisation logique entre les termes. Ce vocabulaire peut être utilisé afin de labelliser des contenus documentaires. Les catalogues sont des exemples de vocabulaires contrôlés.

◆**Glossaire:** C'est un ensemble de termes avec leur signification, la définition de chaque terme est donnée en langage naturel. Cette représentation apporte plus d'informations car une personne peut lire la définition, cependant elle n'est pas interprétable par l'ordinateur.

◆**Thésaurus:** Glossaire contenant des descriptions sémantiques entre les termes. Ils sont principalement utilisés pour assister les documentalistes dans la tâche

d'indexation manuelle de documents, ils sont reconnus pour présenter différents avantages dans ce contexte, ils offrent tout d'abord une vue générale sur les termes et relations d'un domaine, ils définissent ensuite un vocabulaire standardisé pour l'indexation. Ils permettent d'assurer qu'un seul terme d'un ensemble de synonymes soit choisi pour l'indexation (terme dit « à utiliser »).

Les thésaurus sont également utilisés lors de la spécification d'une requête pour spécifier ou généraliser une recherche documentaire à partir des termes dits plus spécifiques ou plus génériques.

- **Hiérarchie informelle de généralisation (is-a) :** sa structure est basée non pas sur des relations de généralisation mais sur la proximité des concepts.
- **Hiérarchie formelle de généralisation (is-a):** sa structure est déterminée par des relations de généralisation.
- **Hiérarchie formelle de généralisation (is-a) avec instances du domaine:** Similaire à la catégorie précédente mais incluant des instances.
- **Frame:** ontologies incluant des classes avec propriétés pouvant être héritées.
- **Ontologies avec restrictions de valeur :** ontologies pouvant contenir des restrictions sur les valeurs des propriétés.
- **Ontologies avec contraintes logiques:** ontologies pouvant contenir des contraintes entre constituants (exemple relations) définies dans un langage logique.

II.6Cycle de vie d'une ontologie :

Étant donné que les ontologies sont destinées à être utilisées comme des composants logiciels dans des systèmes informatiques répondant à des objectifs opérationnels différents, leur développement doit s'appuyer sur les mêmes principes que ceux appliqués en génie logiciel. En particulier, elles doivent être considérées comme des objets techniques évolutifs et posséder un cycle de vie spécifique. Les activités liées à une ontologie peuvent être regroupées en trois catégories [Furst, 2004]:

- Des activités de gestion de projet : planification, contrôle, assurance qualité.
- Des activités de développement : spécification, conceptualisation, formalisation.
- Des activités de support : évaluation, documentation, gestion de la configuration.

La Figure II.3 représente les différentes activités qui expliquent que le cycle de vie préconisé est un cycle par prototypes.

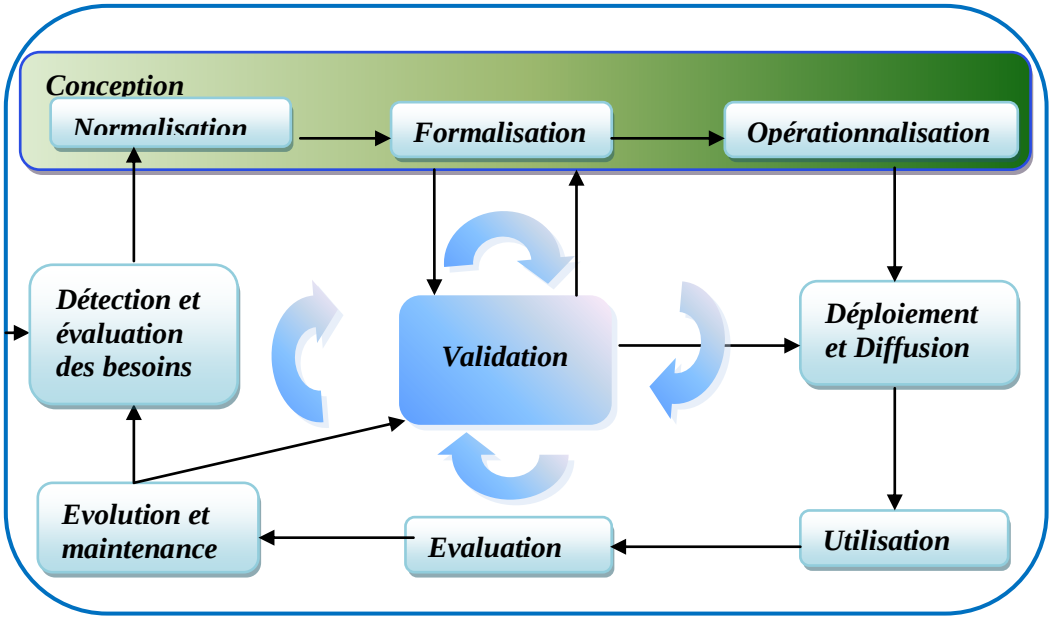

Figure II.3 : Le cycle de vie d'une ontologie [Baneyx, 2007].

Fernandez et ses collègues [5] insistent sur le fait que les activités de documentation et d'évaluation sont nécessaires à l'étape du processus de construction d'ontologie, l'évaluation précoce permettant de limiter la propagation d'erreurs.

II.7Les formalismes de représentation :

Une ontologie, telle qu'elle est décrite dans la section précédente (II.2), a besoin d'être représentée formellement; en plus, elle doit représenter l'aspect sémantique des relations liant les concepts. A cet effet, de nombreux formalismes ont été développés telle que :

[5][Fernandez et al, 1997]

- Les réseaux sémantiques.
- Les schémas.
- les logiques de description.

II.7.1 Les réseaux sémantiques :

Un réseau sémantique est une structure de graphe qui encode les connaissances ainsi que leurs propriétés. Les nœuds du graphe représentent des objets (concepts, situations, événements…etc) et les arcs expriment des relations entre ces objets, ces relations peuvent être des liens " **sorte-de** " exprimant la relation d'inclusion ou des liens " **est-un** " représentant la relation d'appartenance.

Parmi les réseaux sémantiques, très répandus pour la conceptualisation des ontologies, on trouve les graphes conceptuels dont le but fondamental est d'être " *un système de logique hautement expressif, permettant une correspondance directe avec la langue naturelle* "[Sowa, 1992], ce type de graphes constitue un formalisme général de représentation de connaissances fondé sur la logique.

Les graphes conceptuels ont été mis au point par John Sowa pour modéliser une ontologie de haut niveau [Sowa, 2000], un graphe conceptuel est un graphe étiqueté, biparti, connexe et fini. Les sommets représentent les entités, attributs, états ou évènements. Chaque sommet est typé. Ces types sont ordonnés dans une structure de treillis orienté du plus spécifique au plus général avec des relations "***sorte-de*** ".

L'intérêt de ces graphes réside dans leur non-ambiguïté et leur facilité d'utilisation, ceci a incité les concepteurs de plusieurs applications à les utiliser, que ce soit dans l'acquisition des connaissances, la recherche d'informations et le raisonnement sur la connaissance conceptuelle.

II.7.2 Les schémas :

La notion de schéma (ou frame) est apparue en 1932 dans le domaine de la psychologie ; plus tard, les schémas ont été introduits en intelligence artificielle par Minsky afin de résoudre

les problèmes de la vision par ordinateur. D'après la définition dans [Minsky, 1975] un schéma est une structure de données complexe. Il est considéré comme un prototype décrivant une situation ou un objet standard. Il sert de référence pour comparer des objets que l'on désire reconnaître, analyser ou classer.

Les prototypes doivent prendre en compte toutes les formes possibles d'expression de la connaissance. Un schéma est caractérisé par des *attributs*, des *facettes* et des *relations*.

- Les attributs définissent la structure de données ;
- Les facettes définissent la sémantique des attributs et décrivent l'ensemble des valeurs possibles pour cet attribut. Elles peuvent être de deux formes : déclaratives et procédurales.

Les relations expriment la sémantique d'héritage. Elles peuvent être générales (spécialisation, composition) ou spécifiques à une application.

II.7.3 Logiques de descriptions

Les logiques de description (LDs) découlent directement des travaux fondateurs de Bachmann et de son système KL-ONE. Depuis le début des années 90, la recherche en logique de description s'est considérablement développée, elles peuvent être considérées comme un fragment de la logique du premier ordre, dans lequel les formules ont une variable libre pour les descriptions de concept et deux variables libres pour les descriptions de relations[Buchheit et al,1993].

Une LD est composée de deux parties :
Un langage terminologique TBOX et un langage assertionnel ABOX. Le langage assertionnel est dédié à la description de faits et le langage terminologique à la description de concepts et de rôles. La principale tâche de raisonnement au niveau terminologique est de calculer les relations de subsomption entre concepts [Napoli, 1997].

II.8Les étapes suivis lors de la construction d'une ontologie:

L'étape 1 : La conceptualisation

Elle consiste à identifier précisément, à partir du corpus (ensemble de documents généralement exprimés en langage naturel qui doivent couvrir l'ensemble du domaine de connaissances considéré) et à travers des interviews avec les experts du domaine, les objets conceptuels propres au domaine considéré (concepts, relations et axiomes), certaines connaissances implicitement utilisées dans le domaine ne sont cependant jamais exprimées, ni dans le corpus, ni par les experts, car elles sont acquises par l'expérience sensorielle et accumulées différemment. Un des points les plus délicats de la conceptualisation consiste donc à identifier ces connaissances. La mise en évidence de ces connaissances implicites ne peut a priori se faire que lors de l'utilisation de l'ontologie.

On obtient alors un modèle conceptuel informel (car exprimé en langage naturel) ou une ontologie informelle.

L'étape 2 :L'ontologisation

L'ontologisation consiste en une formalisation partielle, sans perte d'information, du modèle conceptuel. Il s'agit de transcrire les connaissances exprimées a priori en langage naturel dans un langage ou paradigme de représentation d'ontologie (le model Frame, le modèle entité relation, le modèle de graphe conceptuel ou réseau sémantique…), afin de respecter les objectifs généraux des ontologies, GRUBER propose 5 critères permettant de guider le processus d'ontologisation [Gruber, 1993]:

1. la clarté et l'objectivité des définitions, qui doivent être indépendantes de tout choix d'implémentation ;

2. la cohérence (consistance logique) des axiomes ;
3. l'extensibilité d'une ontologie, c'est-à-dire la possibilité de l'étendre sans modification ;
4. la minimalité des postulats d'encodage, ce qui assure une bonne portabilité ;
5. la minimalité du vocabulaire, c'est-à-dire l'expressivité maximum de chaque terme.

L'étape 3 :L'opérationnalisation

L'opérationnalisation consiste à l'intégration des connaissances dans un système à base de connaissance donc à outiller l'ontologie pour permettre à une machine (via cette ontologie) de manipuler des connaissances du domaine, cette étape consiste ainsi à formaliser complètement l'ontologie obtenue précédemment dans le cadre d'un langage de représentation de connaissances formel et opérationnel.

Dans le cas où le langage d'ontologisation n'est pas opérationnel, il est nécessaire, soit d'outiller ce langage (dans la mesure du possible) soit de transcrire l'ontologie dans un langage opérationnel. Avant d'être livrée aux utilisateurs, l'ontologie doit bien sur être testée par rapport au contexte d'usage pour lequel elle a été bâtie. [Furst, 2004]

La figure suivante illustre l'enchaînement des trois étapes permettant de passer des données brutes à une ontologie opérationnelle :

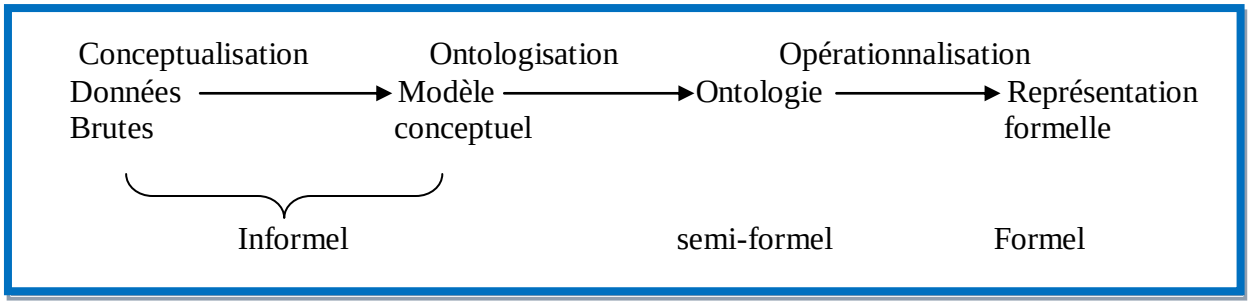

Figure II.4 : Processus de construction d'une ontologie exploitable.

II.9 Evaluation d'une ontologie :

L'évaluation d'une ontologie se fait soit en l'utilisant dans des applications soit en la discutant avec les experts de domaine.

II.9.1 Critères d'évaluation d'une ontologie :

D'après Gruber [Gruber, 1993], cinq critères permettent de mettre en évidence des aspects importants d'une ontologie :

- **La clarté :** la définition d'un concept doit faire passer le sens voulu du terme, de manière aussi objective que possible (indépendante du contexte). Une définition doit de plus être complète (c'est à dire définie par des conditions à la fois nécessaires et suffisantes) et documentée en langage naturel.
- **La cohérence :** rien qui ne puisse être inféré de l'ontologie ne doit entrer en contradiction avec les définitions des concepts (y compris celles en langage naturel).
- **L'extensibilité :** les extensions qui pourront être ajoutées à l'ontologie doivent être anticipées, il doit être possible d'ajouter de nouveaux concepts sans avoir à toucher aux fondations de l'ontologie.
- **Une déformation d'encodage minimale :** une déformation d'encodage a lieu lorsque la spécification influe la conceptualisation (un concept donné peut être plus simple à définir d'une certaine façon pour un langage d'ontologie donné, bien que cette définition ne corresponde pas exactement au sens initial). Ces déformations doivent être évitées autant que possible.
- **Un engagement ontologique minimal :** le but d'une ontologie est de définir un vocabulaire pour décrire un domaine, si possible de manière complète ni plus ni moins. Contrairement aux bases de connaissances par exemple, on n'attend pas d'une ontologie d'être capable de fournir systématiquement une réponse à une question arbitraire sur le

domaine. Une ontologie est la théorie la plus faible couvrant un domaine, elle ne définit que les termes nécessaires pour partager la connaissance liée à ce domaine.

II.9.2 Validation d'une ontologie :

La validation de l'ontologie en amont de son opérationnalisation est souhaitable, la validité des hiérarchies doit donc être testée dès la phase d'ontologisation, aussi bien du point de vue formel que du point de vue sémantique.

***La validation formelle consiste à vérifier que* :**

- Il n'y a pas de cycle (pas de définition en boucle),
- Il n'y a pas redondance de concepts ou de relations,
- Chaque hiérarchie est bien connexe (pas de concept ou de relation isolé des autres et donc sans aucun sens) [Gomez 1999],
- La hiérarchie est conforme aux choix de modélisation (par exemple la détection de l'héritage multiple).

La validation sémantique consiste à vérifier :

- la cohérence sémantique de l'ontologie,
- La validation sémantique permet de contrôler que la structure des hiérarchies est correcte de point de vu sémantique, le sens des concepts ainsi que les liens existant entre ces concepts doivent avoir un sens pour les experts du domaine.

Le deuxième cas nécessitant l'évolution d'une ontologie est celui où les objectifs changent, c'est-à-dire que le contexte d'usage est modifié, ou que le domaine de connaissance est élargi. On peut décider soit de construire une nouvelle ontologie avec les connaissances à ajouter et l'intégrer dans l'ontologie déjà constituée (ce qui revient à fusionner les deux ontologies chose qui reste un thème de recherche encore peu exploré), soit d'agréger directement les nouvelles connaissances dans l'ontologie existante.

II.10 Les outils de construction d'ontologies :

De nombreux outils de construction d'ontologies utilisant des formalismes variés ont été développés, beaucoup d'outils offrent des supports pour le processus d'ontologisation, mais peu d'entre eux offrent une aide à la conceptualisation. Ces outils d'aide à la construction d'ontologie sont plus ou moins indépendants des formalismes de représentation, parmi ces outils, on peut citer :

- **ONTOLINGUA** de l'Université Stanford ; Le serveur Ontolingua est le plus connu , c'est un environnement de construction d'ontologies en langage Ontolingua. Il consiste en un ensemble d'outils et de services qui supportent la construction en coopération d'ontologies, entre des groupes séparés géographiquement.

- **PROTEGE** est le plus connu et le plus utilise des éditeurs d'ontologie. Open-source, développe par l'Université de Stanford, il a évolué depuis ses premières versions (Protege-2000) pour intégrera partir de 2003 les standards du web sémantique RDF (Resource Description Framework), RDFS (Resource Description Framework Schema) et notamment OWL, Prttégé offre de nombreux composants optionnels : raisonneurs, interfaces graphiques. Dans Protégé 2000, le modèle de connaissance est basé sur une hiérarchie de classes explicite ou l'ontologie est décrite de manière déclarative. Protégé est un éditeur d'ontologies pour les différents langages : RDF et OWL, il dispose de plugins pour ces deux langages.

- **WEBONTO** du Knowledge Media Institute de l'Open University ; WebOnto et TADZEBAO sont des outils complémentaires. Tadzebao permet aux ingénieurs des connaissances de tenir des discussions sur les ontologies, en mode synchrone et asynchrone. WebOnto supporte la navigation collaborative, la création et l'édition d'ontologies sur le Web.

- **SWOOP** est un éditeur d'ontologie développé par l'Université du Maryland dans le cadre du projet MINDSWAP ; contrairement à Protégé, il a été développé de façon native sur les standards RDF et OWL, qu'il prend en charge dans leurs différentes syntaxes (pas seulement XML), c'est une application plus légère que Protégé, moins évoluée en termes d'interface, mais qui intègre aussi des outils de raisonnement.

- **ONTOEDIT** (ONTOLOGY EDITOR) développé par la compagnie Ontoprise, c'est un environnement de construction d'ontologies basé sur une méthodologie. Il permet l'édition des hiérarchies de concepts et de relations dans le cadre du paradigme des frames et l'expression d'axiomes algébriques portant sur les relations, et de propriétés telles que la généricité d'un concept. Des outils graphiques dédiés à la visualisation d'ontologies sont inclus dans l'environnement. OntoEdit intègre, dans sa version commerciale, un serveur destiné à l'édition d'une ontologie par plusieurs utilisateurs ainsi qu'un plug-in permettant le test de la cohérence d'une ontologie. Enfin, un plugin nommé ONTOKICK offre la possibilité de générer les spécifications de l'ontologie par l'intermédiaire de questions de compétence. OntoEdit gère de nombreux formats de représentation de connaissance dont DAML+OIL, RDFS et FLogic [Furst, 2004].

II.11 Conclusion :

Dans ce chapitre nous avons étudié les ontologies et leur utilisation. L'ontologie est une conceptualisation d'un domaine à laquelle sont associés un ou plusieurs vocabulaires de termes, elles servent à décrire formellement des concepts et des relations entre elles, donc elles apparaissent désormais comme une clé pour la manipulation automatique de l'information au niveau sémantique

Les apports d'utilisation des ontologies sont divers, elles jouent un rôle important dans les systèmes à base de connaissance, outre la réutilisation et le partage de connaissances, elles permettent de faciliter la communication entre les acteurs de différentes organisations et en particulier, la réalisation de l'interopérabilité entre différents systèmes, elles permettent aussi

non seulement la création de systèmes mais capables de gérer et de raisonner sur ces connaissances.

Elles peuvent être décrites dans plusieurs outils de construction d'ontologie (WEBONTO, ONTOLINGUA, WEBONTO, PROTEGE, etc.) mais nous pouvons dire que l'outil PROTEGE s'impose comme référence.

Chapitre III

Vers un modèle conceptuel de l'ontologie

III.1 Introduction :

Ce chapitre présente notre contribution à la problématique posée dans ce mémoire, à savoir la construction d'une ontologie pour la modélisation et la simulation intelligente, baptisée «DeMO-E» Descret even Modeling Ontology-Explication.

Comme mentionné dans le chapitre précédant, il existe beaucoup de définitions pour le terme ontologie, ici nous parlerons d'une ontologie formelle utilisée dans des applications informatiques dites intelligentes.

On a donc, par opposition aux ontologies couvrant des domaines spécifiques, les ontologies dites supérieures ou méta-ontologie, celles-ci identifient et définissent des concepts généraux. Elles ont plusieurs buts :

- La fondation pour des ontologies plus spécialisées.
- De servir de cadre pour intégrer des ontologies orientées domaine.
- De faire communiquer de multiples ontologies couvrant le même domaine mais usant d'un vocabulaire différent.

C'est ce type d'ontologie que nous allons étudier à travers notre ontologie DeMO-E. Donc nous essayons dans ce chapitre d'enrichir l'ontologie DeMO (Descret even Modeling Ontology) par l'intégration d'explication, cette nouvelle ontologie nous appelons DeMO-E et qui doit servir de cadre conceptuel à une plateforme de modélisation et de simulation intelligente.

Nous commençons d'abords par un état de l'art, ensuite nous présenterons l'ontologie DeMO et ses différents concepts. Enfin, nous présenterons notre ontologie.

III.2 Etat de l'art :

Les recherches dans le domaine d'intégration des ontologies dans la conception des environnements de modélisation et de simulation sont très limitées en nombre et en contenu, nous pensons que cela est dû au fait que le domaine d'ontologie pour la modélisation et la simulation est très jeune d'une part, et d'autre part qu'il n'y a pas beaucoup de groupes de recherches qui ont traité ce sujet.

Les travaux publiés sur ce sujet discutent pour la plupart des quelques impacts du Web sémantique sur la modélisation et la simulation sans donner des orientations sérieuses vers une ontologie de modélisation et de simulation intelligente. Aujourd'hui, il existe un consensus sur les intérêts pratiques d'une telle ontologie, mais des recherches plus approfondies sur ce sujet font toujours défaut.

Depuis ces dernières années, de nombreux travaux ont été publiés dans des conférences dédiées spécialement au thème de la simulation basé sur le Web, parmi ces travaux on peut citer :

En 2000 Page [Page et al, 2000] et [Page et Opper, 2000] examinent dans ces articles la nature de la simulation basé sur le Web. Reed et les autres [Reed et al, 2000] expriment les possibilités d'améliorer le processus de conception des appareils fondés sur le Web de modélisation et de simulation, les simulations pourraient également être utilisées pour l'optimisation, d'autres chercheurs tels que Chen et Yucesan ont enquêté sur cela. Miller et al [Miller et al, 2001] expliquent la technologie derrière la simulation basée sur le Web, et font valoir la nécessité de démontrer l'application de la simulation basée sur le Web pour les grands projets. Kuljis et Paul [Kuljis et Paul, 2001] évaluent les progrès dans ce domaine de la simulation web. Ils font valoir la nécessité d'une simulation basée sur le Web pour se concentrer sur la résolution des problèmes du monde réel afin d'être couronnée de succès. Kimet al [Kim et al, 2002] étudient comment les techniques génèrent le code exécutable à partir de documents mentionnés XML, et qui peut être utilisé pour créer des simulations.

En dépit des tendances actuelles visant à prendre en considération l'utilisation des ontologies dans la modélisation et la simulation, il existe peu de travaux de recherches dans ce domaine. Le célèbre travail de Fishwick et Miller [Fishwick et Miller, 2004] fournit un cadre théorique propice pour examiner l'utilisation d'ontologies pour la modélisation et la simulation, les auteurs ont été impliqués dans le projet RUBE qui a développé un système pour des simulations de combat, illustrée dans [Fishwick et Miller, 2004], dans ce projet ils ont utilisé des standards ouverts et Protégé pour construire l'ontologie.

En 2004 Lacy et Gerber ont examiné comment OWL peut être utilisé pour faciliter la modélisation et la simulation, parce que l'ontologie utilise des standards ouverts, ces simulations pourraient être largement disponibles sur le web. L'article de Miller et Baramidze [Miller et Baramidze, 2005] consiste à expliquer également leurs recherches dans l'ontologie DeMO (Discrete-Event Modelling Ontology) pour la simulation et la modélisation, celui-ci utilise OWL pour définir une hiérarchie de classes de simulation et de modélisation.

Une étude récente a été menée par Silver et el [Silver et al. 2007]qui ont étudié la relation entre l'ontologie et les modèles de simulation exécutable sous le titre :

«*From domain ontologies to modeling ontologies to executable simulation*», une autre étude réalisée aussi par Silver et les autres dans [Silver et al, 2009] explique la possibilité de supporterl'interopérabilité par l'utilisation d'ontologie de modélisation et de simulation à événement discret(DeMO)

Donc, l'utilisation d'ontologie pour la simulation est un domaine de recherche à explorer, il est également nécessaire pour les utilisateurs de simulation pour être en mesure de recueillir des informations auprès des différents niveaux, donc l'objectif est de développer une représentation fidèle de connaissances en utilisant un éditeur d'ontologie.

III.3 DeMO : Ontologie de modélisation et simulation à événements discrets :

Comme les méta-ontologies sont des systèmes complexes, deux éléments cruciaux doivent être pris en compte pour en choisir une :

- L'ontologie doit fournir un ensemble de distinctions conceptuelles riches (au moins, par rapport au domaine de l'application)
- Toutes les caractéristiques dont on a besoin doivent être clairement caractérisées (ou caractérisables).

Nous avons choisi de travailler avec DeMO parce que cette ontologie est la seule dans le domaine de modélisation et simulation à événement discret, et qui fournit un certain nombre de concepts de haut niveau utiles à notre application.

III.3.1 Définition :

DeMO (Discrete-Event Modelling Ontology) est une ontologie pour la modélisation et la simulation à événements discrets, elle a été produite par les systèmes d'information distribuée à grande échelle(LSDIS[6]) du laboratoire à l'Université de la Géorgie.

DeMO est une ontologie dite de niveau supérieur qui englobe les concepts les plus larges et les plus abstraits requis pour la modélisation et la simulation à événements discrets, elle fournit des définitions d'usage universel et se pose comme fondement pour des ontologies de domaines plus spécifiques. Ici le système à événements discrets DES[7] est compris comme un système dirigé par les événements avec des états discrets. L'évolution du système peut donc être décrite par une séquence des paires ordonnées (état, temps): $\{(s_0, t_0), (s_1, t_1), ...\}$ avec l'hypothèse que pour chaque changement d'état s_n à s_{n+1}, il existe un événement e (ou un ensemble d'événements E^*), qui a causé ce changement. Donc comme son nom l'indique, elle est centrée sur les modèles à événements discrets, dans lequel l'état de système modifie discrètement au fil du temps en raison de la survenance des événements, en utilisant le langage OWL pour définir plus de 100 classes et de nombreuses propriétés.

[6] **LSDIS :** Large Scale Distributed Information Systems
[7] **DES:** Discret Event Simulation

III.3.2 L'architecture de DeMO:

Nous donnons ici un aperçu sur l'architecture d'ontologie DeMO parce qu'une description complète des connaissances et des classes proposées par DeMO dépasserait le cadre de ce mémoire.

1. Les éléments de l'ontologie

Dans les ontologies DeMO, il existe cinq types de composants qui sont ;

- Les classes,
- Les relations,
- Les axiomes,
- Les attributs.

Les deux éléments les plus importants sont les classes et les relations qui sont utilisées pour créer des liens entre tous les autres types, chaque type d'éléments joue un rôle syntaxique différent dans les assertions DeMO et peut être distingué et reconnu sur cette base, de plus des conventions ont été adoptées pour différencier ces derniers, ils ne font pas parties de la syntaxe formelle du langage OWL mais ont été mis en place pour l'ontologie DeMO. Le nom de classe commence par une lettre majuscule, mais les attributs commencent par une lettre minuscule l'image de ces exemples :
« DeModel » (classe), « name » (attribut), il faut remarquer que les espaces ne sont pas admis dans les noms des termes (exemples : « ModelComponents »).

a. Les Classes de l'ontologie DeMO :

L'ontologie DeMO se compose de quatre classes (abstraites) principales, elles représentent des connaissances principales du domaine de simulation à événement discret (DES) qui sont:

- DeModel,
- ModelConcepts,
- ModelComponents,
- ModelMechanisms.

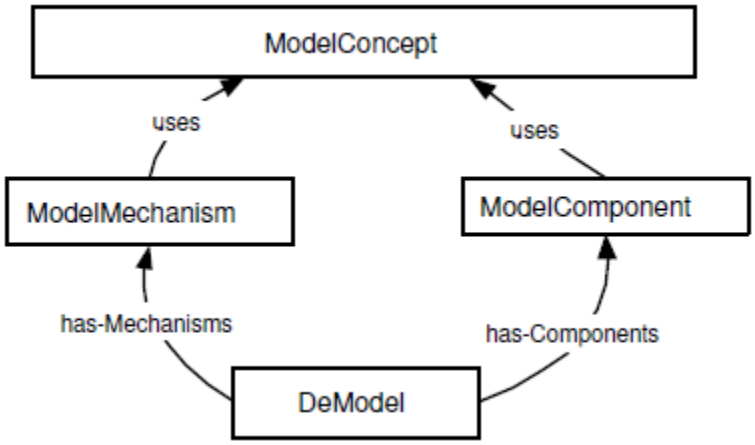

Figure III.1 : Représentation graphique de la conception de DeMO. [Miller et al, 2004]

1) La classe DeModel :

Cette classe représente une racine de taxonomie, comme il a été mentionné plus haut, et le point de départ pour la construction mentale d'ontologie.

La racine dans cette taxonomie est divisée en quatre types différents de modèles à événements discrets en fonction de leur caractérisation structurale(les éléments formels utilisés pour définir la structure du modèle) sont: orientée état , orientée événement, orientée activité et orientée processus comme montre la figure III.2.

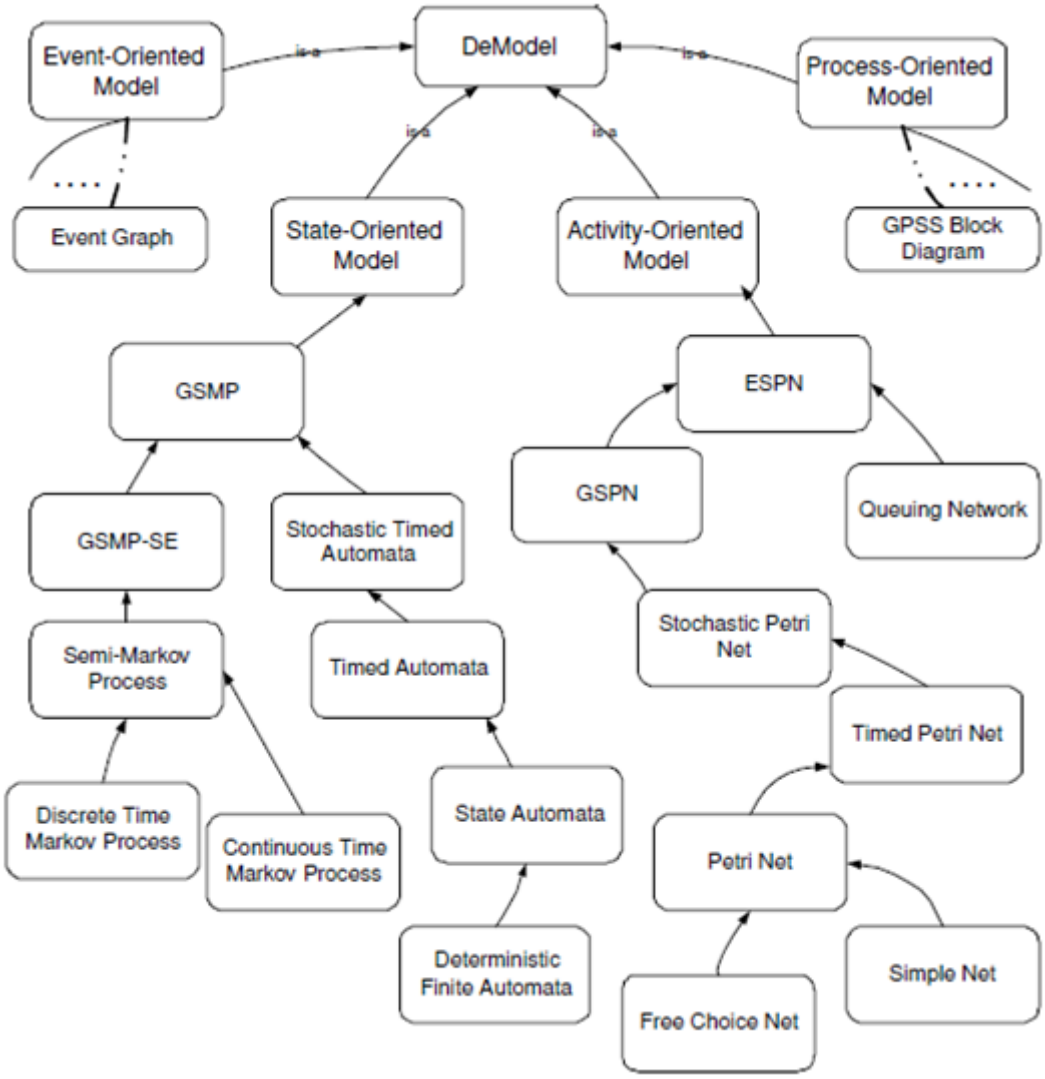

Figure III.2 : Partie de la taxonomie de l'ontologie DeMO. [Silver et al, 2009]

Chacune de ces formalismes de premier niveau définit un ensemble de composants (concepts) utilisés dans la construction des sous-classes inférieures, leurs sous-classes sont définies en imposant des restrictions sur les propriétés des formalismes plus élevés, cette classification de deuxième niveau illustre une approche de subsomption dans la construction d'une taxonomie.

Les sous-classes de ces classes (StateOrientedModel, EventOrientedModel, ActivityOrientedModel et ProcessOrientedModel) représentent les techniques de modélisation existants, tels que :

- Les réseaux de pétri,
- Chaînes de Markov,
- Les graphes d'événements, etc.

2) La classe ModelComponent :

Cette classe décrit les éléments du modèle utilisé comme blocs de construction de modèle DeModel (chaque modèle est relié par la relation has-a avec sa classe ModelComponent).

Les modèles State-Oriented et Event-Oriented ont besoin des composants formels suivants pour être définis:

- StateSpace,
- EventSet,
- TimeSet,
- TransitionFunction,
- ClockFunctionet
- InitialState.

De manière analogue, le modèle Activity-Oriented a besoin des éléments suivants :

- PlaceSet,
- ActivitySet,
- TimeSet,
- IncidenceFunction,
- InitialMarkinget
- ClockFunction.

Ici ClockFunctiona une signification légèrement différente par rapport au formalisme de modélisation Etat-Oriented.

Mais le modèle Process-Oriented utilise les éléments suivant en tant que composants officiels :

- ✓ ProcessSet,
- ✓ ProcessState-Set,
- ✓ TimeSet,
- ✓ TransitionFunction,
- ✓ ClockFunction.

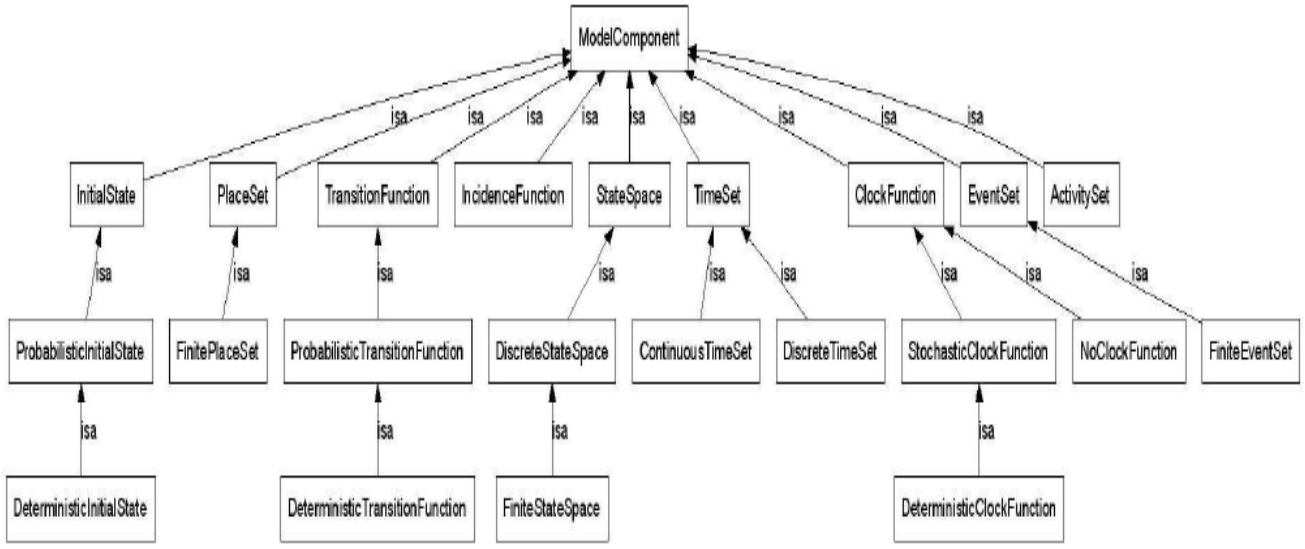

Figure III.3 : La hiérarchie de la classe ModelComponent. [Miller et al, 2007]

3) La classe ModelMechanism :

Cette classe définit comment les composants fonctionnent au sein du modèle, elle contient les sous-classes qui spécifient les différents mécanismes (règles d'engagement) adoptés par des techniques de modélisation, des particuliers :

- EventSchedulingMechanism,
- TransitionTriggeringMechanism,
- horlogerieSettingMechanism, etc.

En substance, les ModelMechanisms expliquent comment faire fonctionner le modèle, au moins abstraitement.

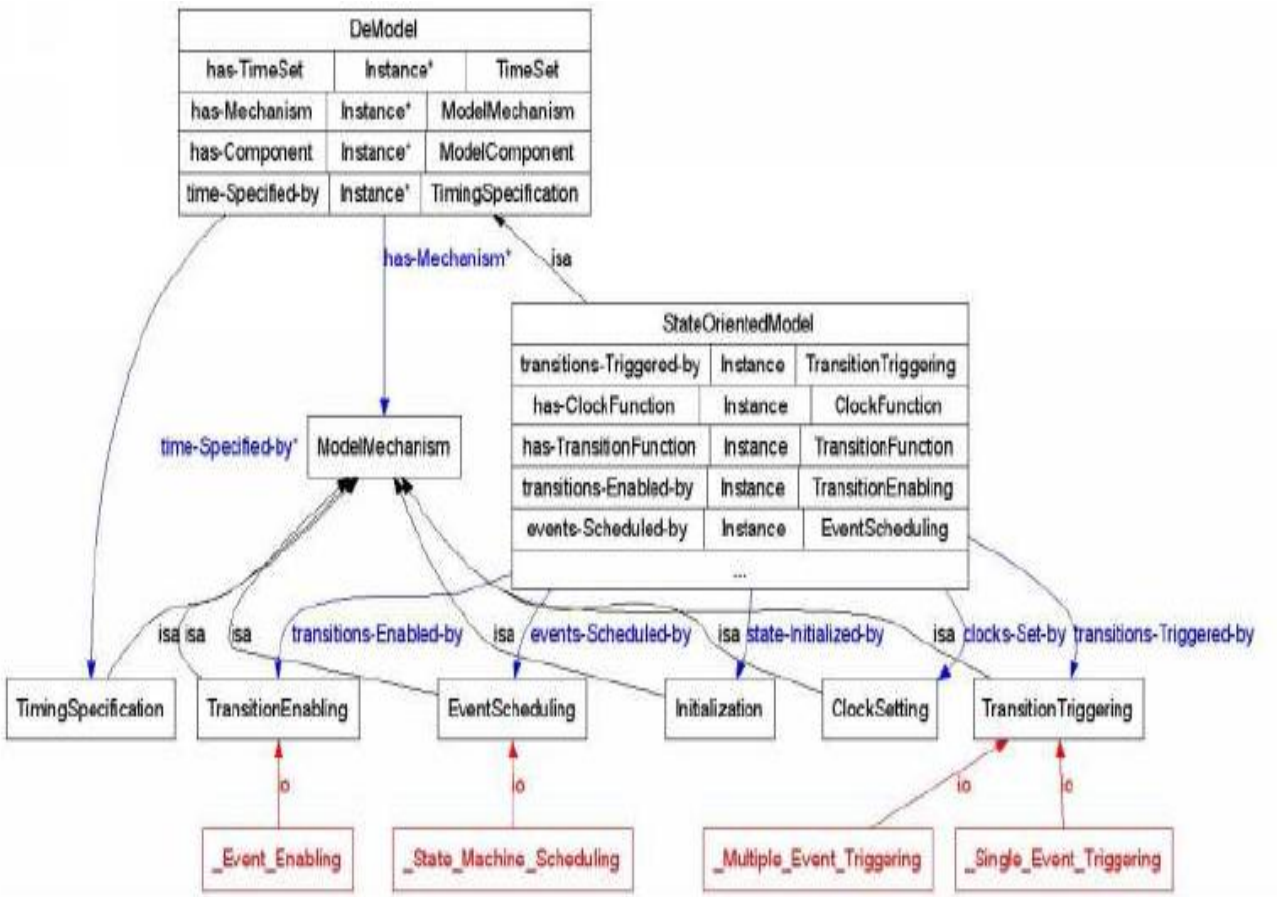

Figure III.4 : La hiérarchie de classe ModelMechanism. [Miller et al, 2007]

4) La classe ModelConcept :

Cette classe représente les concepts les plus fondamentaux utilisés dans l'ontologie DeMO à partir de la quelle les autres concepts sont construits, cette classe a les sous-classes suivantes:

- State,
- Event,
- Time,
- Transition,
- Activity,
- Place,
- Token,
- Entity,
- Process,
- Resource,
- Color, et
- Clock.

Certaines de ces classes ne peuvent pas être formellement définies en termes d'autres concepts (les classes) et souvent seulement des descriptions informelles en anglais sont offertes. Dans un sens, elles servent comme un ensemble de termes de base sur lequel l'ontologie est construite. Afin de fournir une plus grande modularité, la classe ModelConcept avec ses sous-classes peuvent être placée dans une distincte méta-ontologie qui servira un certain type de glossaire utilisé par d'autres ontologies, la figure suivante représente une hiérarchie de classe ModelConcept.

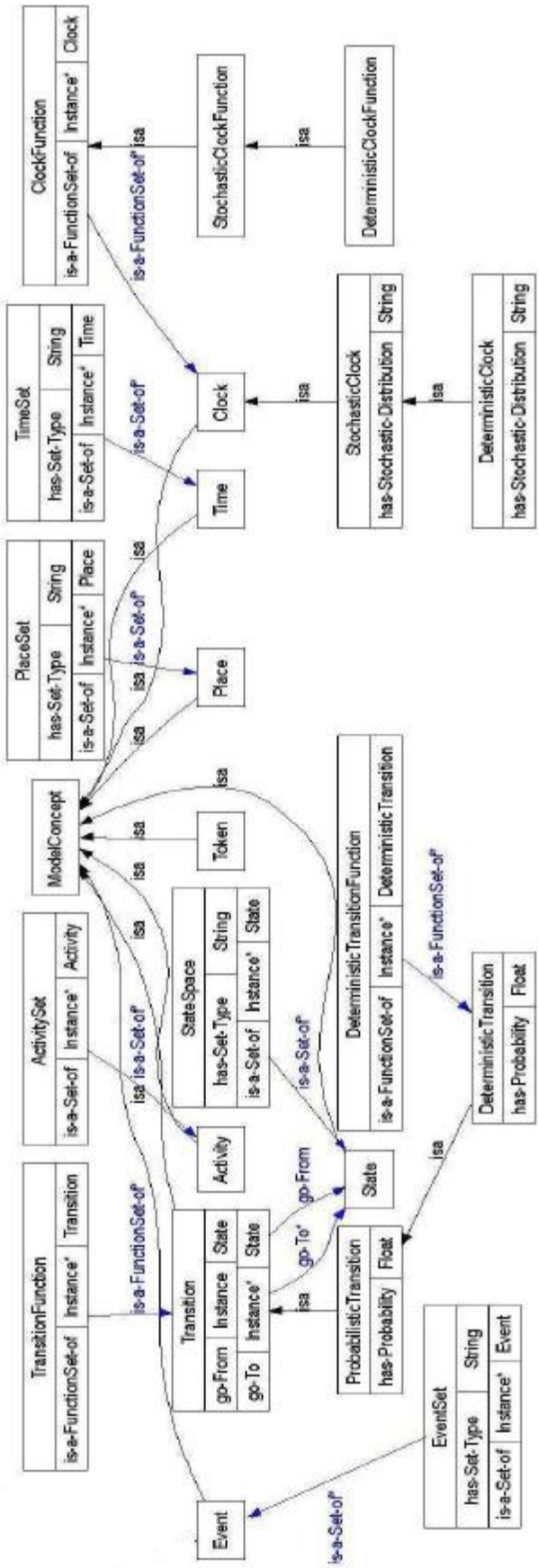

Figure III.5 : La hiérarchie de classe ModelMechanism. [Miller et al, 2007]

III.4 Construction d'une ontologie pour la modélisation et la simulation intelligente :

Dans cette section, nous construisons notre ontologie DeMO-E. qui concerne le domaine de modélisation et simulation intelligente. A cette fin, nous suivrons les étapes du processus de construction d'ontologie développé dans le deuxième chapitre. DeMO-E est une ontologie : **lourde** (avec contrainte logique), **méta-ontologie** pour le domaine de modélisation et simulation intelligente, d'une **granularité ni assez large ni assez fine** et **formelle**. Rappelons que nous ne pouvons pas dissocier les différentes phases de ce dernier.

Il est à espérer que l'ontologie DeMO-E sera utile aux chercheurs et développeurs dans le domaine de la modélisation et la simulation, donc elle permet :

- À l'utilisateur de mieux comprendre le fonctionnement du modèle simulé.
- De favoriser l'interopérabilité des données, la recherche, la récupération de l'information et l'inférence automatisée.
- De définir une structure hiérarchique (dans un certain sens) d'un ensemble de formalismes de modélisation pertinents et les relations entre eux.
- De fournir un point de départ pour d'autres travaux.
- De trouver facilement le meilleur formalisme pour modéliser un système donné et de définir les composants de modélisation, ainsi que de faciliter la méta-modélisation et de la multi-modélisation.
- De localiser des logiciels de modélisation et de simulation.

On commence le processus par des connaissances brutes et on aboutie à une ontologie opérationnelle (pour la modélisation et la simulation intelligente) représentée par le langage OWL. Ce processus est composé de cinq étapes :

- Spécification des besoins.
- Conceptualisation.
- Opérationnalisation et Implémentation.
- Evolution et Test de l'ontologie.

III.4.1 Spécification des besoins :

Une ontologie ne peut être construite qu'après la phase de spécification, il s'agit d'établir un document informel de spécification des besoins, au niveau de ce document, nous décrivons l'ontologie à construire à travers les cinq aspects représentés dans la Figure suivante :

Domaine de connaissance

Domaine de la modélisation et la simulation intelligente.

Objectif

Le but opérationnel de notre ontologie est l'explication de résultat de simulation, elle est aussi utilisée pour capitaliser les connaissances sur la modélisation et la simulation intelligente, pour trouver facilement le meilleur formalisme de modéliser un système donné et peut être aussi utilisée pour la formation…

Utilisateurs

Professionnels de modélisation, Professionnels de simulation, les étudiants, …etc.

Sources d'informations

Documents techniques en rapport avec le domaine de modélisation et de simulation.

Figure III.6 : Document de spécifications de besoin DeMO-E.

III.4.2 Conceptualisation :

Une fois que la majorité des connaissances est acquise, on doit les organiser et les structurer en utilisant des représentations intermédiaires semi-formelles qui sont faciles à comprendre et indépendantes de tout langage d'implémentation. Cette phase contient plusieurs étapes qui sont :

- Construction du glossaire de termes.
- Construction du diagramme de relations binaires et de classification de concepts.
- Dictionnaire de concepts.
- Tableaux des relations binaires.
- Tableaux des attributs.

III.4.2.1 Construction de glossaire de termes :

Construire un glossaire de termes est la première tâche à effectuer dans l'étape de conceptualisation, ce glossaire contient les noms et les descriptions de tous les termes relatifs au domaine (concepts, instances, attributs, relations entre les concepts, etc.) qui seront représentés dans l'ontologie finale, par exemple, dans notre cas les termes Analyser et Generator sont des concepts, has-ActivitySet et uesed-by représentent des relations…etc, le tableau III.1 illustre le glossaire de termes d'DeMO-E.

Nom du terme	Description
QuestionInNaturalLanguage	Un argument ou une donnée qui représenter les questions.
ResponseInNaturalLanguage	Les données représentent les résultats de l'explication.
Analyser	L'analyseur de la question formulée en langue naturelle produit une représentation du sens de la question.
Generator	Produit le texte explicatif final.
ConstructorOfResponse	Il représente les méthodes de construction de la réponse.
ModelConcept	Elle représente les concepts les plus fondamentaux utilisés
State	défini par une collection de variables d'état décrivant l'état du système à a un temps particulier.
EventSet	La liste des événements: Une liste qui contient les dates d'occurrence des événements devant avoir lieu dans le futur.
Clock	variable indiquant la valeur courante du temps de la simulation.
Name	Déterminer le nom du concept.
Type	Déterminer le type d'un paramètre
Analysed-by	Permet d'indiquer que chaque question d'utilisateur doit être analysée par l'analyseur.
Is-a	Est un lien entre un objet spécialisé et un objet générique. Exemple : les entités StateOrientedModel et DeModel sont liés par une telle relation «StateOrientedModel is-a DeModel »
part-of	Permet de définir la classe qui fait parties d'une classe plus spécifique donnée. Exemple : <<Analysor part-of modelExplains>>.
(…)	(…)

Tableau III.1 : La table du glossaire de termes.

Remarque : Lors du développement de cette ontologie, quelques concepts sont inspirés de l'ontologie DeMO.

III.4.2.2 Construction des hiérarchies de concepts :

Chaque hiérarchie organise un groupe de concepts sous forme d'une taxonomie :

La classe ModelExplains:

Cette classe représente le module d'explication qui permet l'explication du résultat de simulation grâce à ces sous classes :

- **Analyser :** C'est l'analyseur de la question d'utilisateur, formulée en langue naturelle, il produit une représentation du sens de la question, cette classe relié avec la classe ConstructorOfResponse par la relation has-besion-of pour dire que le Constructeur de réponse a besoin des résultats de l'analyseur pour construire l'explication.
- **ConstructorOfResponse :** C'est le constructeur de la réponse qui représente le cœur du module d'explication (ModelExplains), il produit une représentation du sens littéral de la réponse.
- **Generator :** Cette classe est le responsable de la mise en langue des résultats du constructeur de la réponse, il permet la génération de surface ou la production de texte explicatif final.

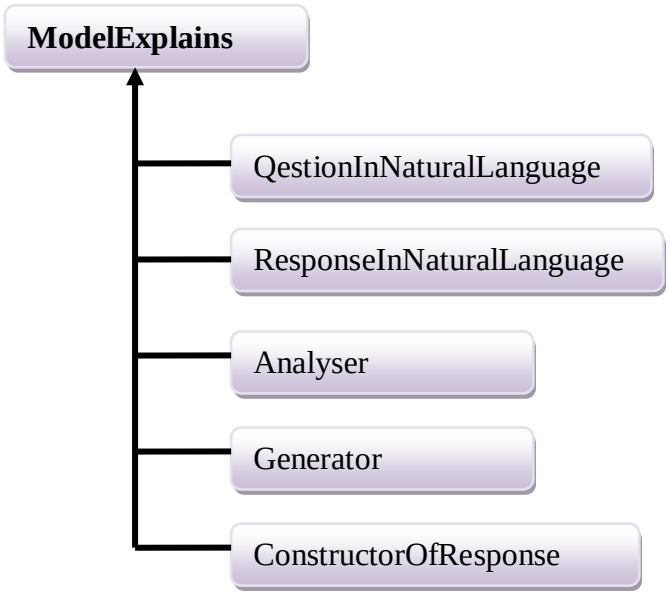

Figure III.7 : Hiérarchie de classe ModelExplains.

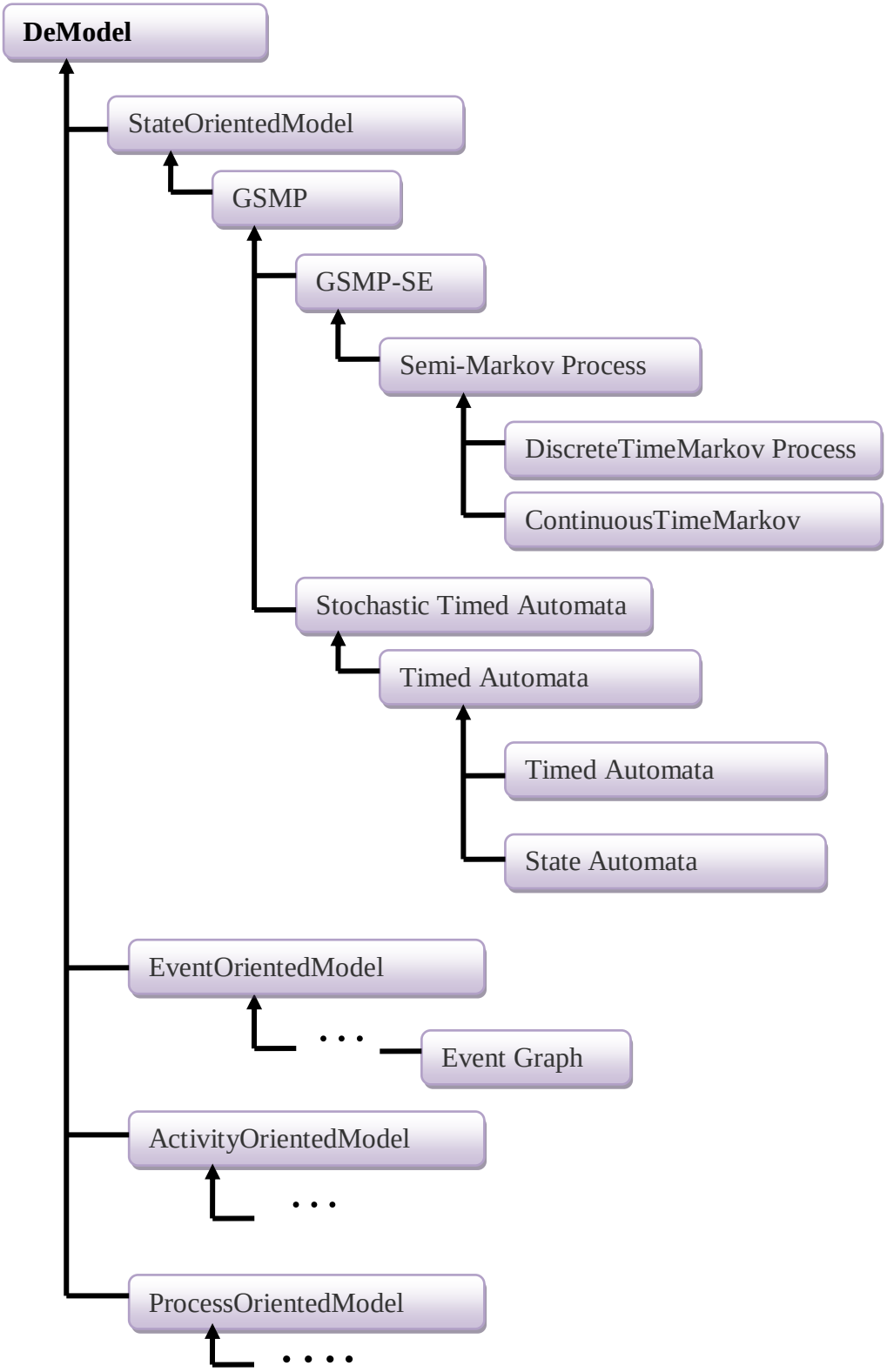

Figure III.8: Partie d'une hiérarchie de classe DeModel.

III.4.2.3 Construction d'un diagramme des relations binaires :

Ce diagramme permet de représenter de manière graphique les différentes relations qui existent entre les divers concepts de même ou de différentes hiérarchies, la figure III.9 illustre le diagramme de relation de notre ontologie.

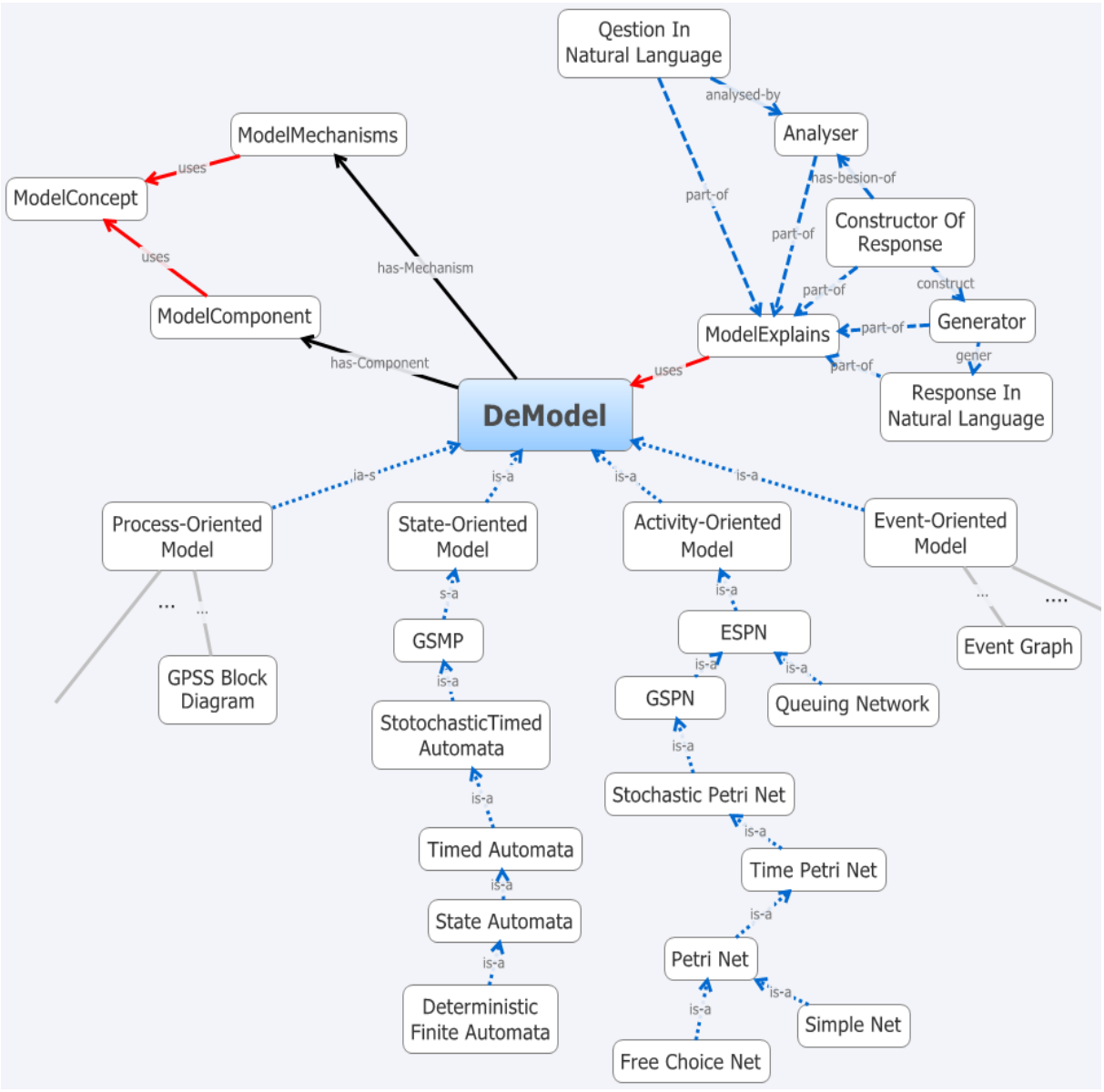

Figure III.9 : Le diagramme de relation de l'ontologie DeMO-E.

III.4.2.4 Construction de la table des relations binaires :

La table des relations binaires définie pour chaque relation utilisée dans le diagramme précédent : le nom de la relation, le nom des concepts sources et cibles, le nom de la relation inverse et les cardinalité source et cible.

Nom de la relation	*Concept source*	*Cardinalité-S*	*Conceptcible*	*Cardinalité-C*	*Relation Inverse*
analysed-by	QestionInNaturalLanguage	1, n	Analyser	1, 1	Analysed
Gener	Generator	1, n	ResponseInNaturalLanguage	1, 1	-
Construct	ConstructorOfRespons	1, n	Generator	1, 1	used-by
has-Component	DeModel	1, n	ModelComponent	1, n	-
Has-Mechanism	DeModel	1, n	ModelMechanism	1, n	-
part-of	Analyser	1, n	ModelExplains	1, 1	-
…	…	…	…	…	…

Tableau III.2 : La table des relations binaires de l'ontologie DeMO-E.

III.4.2.5 Construction de la table des attributs des concepts :

Les attributs sont des propriétés qui prennent leurs valeurs dans les types prédéfinis (String, Integer, Boolean, Date…), pour chaque attribut nous spécifions : le nom, la description, les concepts qu'il contient, le type, l'intervalle de ses valeurs possibles, la cardinalité et sa valeur par défaut. Le tableau (Tableau III.3) représente la table des attributs des concepts de notre ontologie DeMO-E.

Attribut	*Description*	*Concept*	*Type*	*Cardinalité*
name	Le nom du concept	Tous les concepts	chaine	1,1
synonyme	Les synonymes d'un concept		chaine	0, n
mode of operation	Mode de fonctionnement	Analysor, Generator, constructor of response	chaine	1, n
description	La description des termes	Tous les concepts	chaine	0, n
…	…	…	…	…

Tableau III.3 : La table des attributs des concepts.

III.5 Conclusion

Nous avons présenté dans ce chapitre le modèle conceptuel de l'ontologie DeMO-E, cette dernière est une ontologie pour la modélisation et la simulation intelligente qui permet de montrer comment expliquer le résultat de simulation, nous avons proposé à ce titre un méta-modèle d'explication (ModelExplains). Ce modèle est décrit à l'aide de l'ontologie; il est incorporé à l'ontologie DeMO du domaine de modélisation et de simulation en suivant la méthode de construction ontologique de l'université de Standford. A notre connaissance, c'est la première conception ontologique intégré l'explication dans la simulation.

À travers tous ce que nous avons réalisé dans ce chapitre, nous pouvons dire que le processus suivi pour la construction de l'ontologie DeMO-E nous a permis de réussir finalement à construire une méta-ontologie pour la modélisation et la simulation intelligente.

Le chapitre suivant proposera la description de l'étape d'opérationnalisation de l'ontologie DeMO-E ainsi que l'implémentation de cette ontologie.

Chapitre IV

Implémentation

IV.1 Introduction:

A ce niveau, une ontologie est une formalisation systématique des concepts, définitions, rapports et règles qui capturent la teneur sémantique d'un domaine dans un format compréhensible par une machine, on parle ici de langage formel.

La construction de l'ontologie formelle est une tâche complexe, même pour implémenter une petite ontologie, celle-ci va prendre plusieurs lignes de code et nécessite un grand effort et du temps; notamment, si cette ontologie est codée directement par le développeur en langage d'ontologie sans faire recours à aucun outil. Pour cela, les éditeurs d'ontologie sont conçus pour libérer l'ontologiste de cette complexité et générer automatiquement la structure de l'ontologie créée en langage de spécification (XML, DAML, OWL…).

Dans ce chapitre nous allons présenter le travail d'implémentation qu'on a fait, et qui consistait premièrement à l'édition de notre ontologie sous l'outil Protégé. Nous commençons par une présentation de cet outil, ensuite nous détaillons les étapes d'opérationnalisation et d'implémentation de notre ontologie que nous avons baptisé « DeMO-E » (Descret even Modeling Ontology-Explication).Enfin, nous illustrerons la phase de vérification par le biais du moteur d'inférence RACER.

IV.2 Opérationnalisation et implémentation :

Apres avoir conçu le modèle d'ontologie dans le chapitre conception, nous passons maintenant à sa opérationnalisation et implémentation. Ceci se fait à travers l'éditeur d'ontologie Protégé version4.1.

IV.2.1 Choix d'un langage de spécification :

Différents langages ont été proposés pour représenter les ontologies; notre choix a été orienté vers OWL car ce langage est apparu plus tard, qui est le langage standard de représentation et de spécification de l'ontologie, c'est un dialecte XML fondé sur une syntaxe RDF, il se différencie du couple RDF / RDFS par le fait que c'est un langage d'ontologie, contrairement à RDF.

Si RDF et RDFS apportent à l'utilisateur la capacité de décrire les classes et les propriétés, OWL intègre, en plus, des constructeurs de comparaison des propriétés et des classes : identité, équivalence, contraire, cardinalité, symétrie, transitivité, disjonction, etc. Ainsi, OWL offre aux machines une plus grande capacité d'interprétation du contenu web que RDF et RDFS, grâce à un vocabulaire plus large et a une vraie sémantique formelle. Et on a aussi par comparaisons à ses prédécesseurs RDFS et DAML-OIL qui ont souffert des autres limites, comme l'impossibilité de raisonner et de mener des raisonnements automatisés sur les modèles de connaissances établis à l'aide de ces langages, en plus, le codage d'une ontologie sous forme OWL présente l'avantage de la rendre réutilisable. [Xavier, 2005]

IV.2.2 choix d'un outil d'implémentation :

Différents outils ont été proposés dans le deuxième chapitre pour aider à l'implémentation des ontologies, ces outils permettent d'éditer une ontologie, d'ajouter des concepts et des relations, etc. Ils intègrent différents langages de spécification (RDF,OWL).

L'implémentation de notre ontologie DeMO-E s'est effectuée à travers l'éditeur d'ontologiesProtégé_4.1, plusieurs raisons ont motivé notre choix :

- Protégé est un éditeur d'ontologies open source et gratuit.
- Il possède une interface modulaire, ce qui permet son enrichissement par des modules additionnels (plugins).
- Il permet l'édition et la visualisation graphique d'ontologies.
- Il permet de contrôler la cohérence de l'ontologie par la vérification des contraintes.
- Protégé est fourni une API écrite en JAVA, qui permet de développer des applications pouvant accéder aux ontologies de Protégé et de les manipuler.
- Il permet d'importer et d'exporter des ontologies dans les différents langages de spécification d'ontologies (RDF-Schéma, OWL, DAML, OIL,...etc.).
- Exécuter des raisonneurs.

IV.2.3 Présentation de l'éditeur Protégé :

L'outil Protégé a été développé par le Stanford Medical Informatics au sein de l'Université de Stanford, qui fournit un environnement graphique permettant l'édition, la visualisation et le contrôle (vérification des contraintes) d'ontologies.

Le modèle de représentation de connaissances de Protégé est issu du paradigme des frames et contient des classes (pour modéliser les concepts), des slots (pour modéliser les propriétés des concepts) et des facettes (pour définir les valeurs des propriétés et des contraintes sur ces valeurs), ainsi que des instances des classes, il introduit la notion de méta-classes, dont les instances sont des classes. Il est également une librairie Java qui peut être étendue pour créer de véritables applications à base de connaissances en utilisant un moteur d'inférence pour raisonner et déduire de nouveaux faits par application des règles d'inférence aux instances de l'ontologie et à l'ontologie elle-même (méta-raisonnement).

L'architecture de Protégé consiste en deux parties : **le model** et **la vue**. Le model est le mécanisme de représentation interne pour les ontologies et la vue fournit une interface utilisateur pour l'affichage et la manipulation du model sous-jacent.

L'interface de Protégé complet ainsi que l'architecture logicielle ouverte et extensible permettant l'insertion de plug-ins, ont grandement participé au succès de Protégé. Ces plug-ins pouvant apporter de nouvelles fonctionnalités entre autre OntoGraf permet de gérer des représentations sous forme graphique, et PROMPT qui est dédié à la gestion de plusieurs ontologies, en particulier à leur fusion et un plugin dédié à OWL.

IV.2.4 Edition de l'ontologie DeMO-E :

1) Création d'une nouvelle ontologie OWL :

Lors du premier démarrage de « Protégé 4.1 », une boite de dialogue (*Welcome to Protégé*) s'ouvre comme montre la figure IV.1. Afin de créer une nouvelle ontologie OWL, nous devons cliquer sur le bouton « Create new OWL ontology » de ce fait, une autre boite de dialogue (Create ontology wizard) se crée, sur laquelle plusieurs choix sont offerts, permettant de créer de nouveaux ontologie selon plusieurs format.

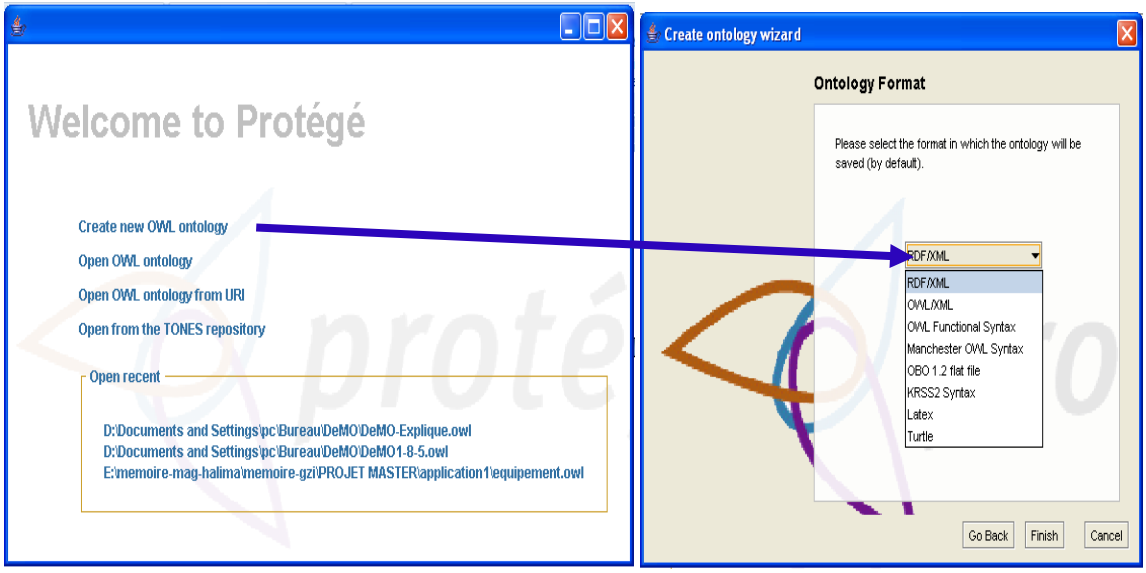

Figure IV.1 : Création d'un nouveau projet dans Protégé 4.1.

Dans notre cas il s'agit d'un format OWL « OWL Files (.owl or .rdf) ». Une fois que ce choix est validé, l'interface de l'outil s'affiche permettant d'éditer, de visualiser et d'enregistrer des ontologies en OWL. La figure IV.2 présente l'interface de Protégé 4.1.

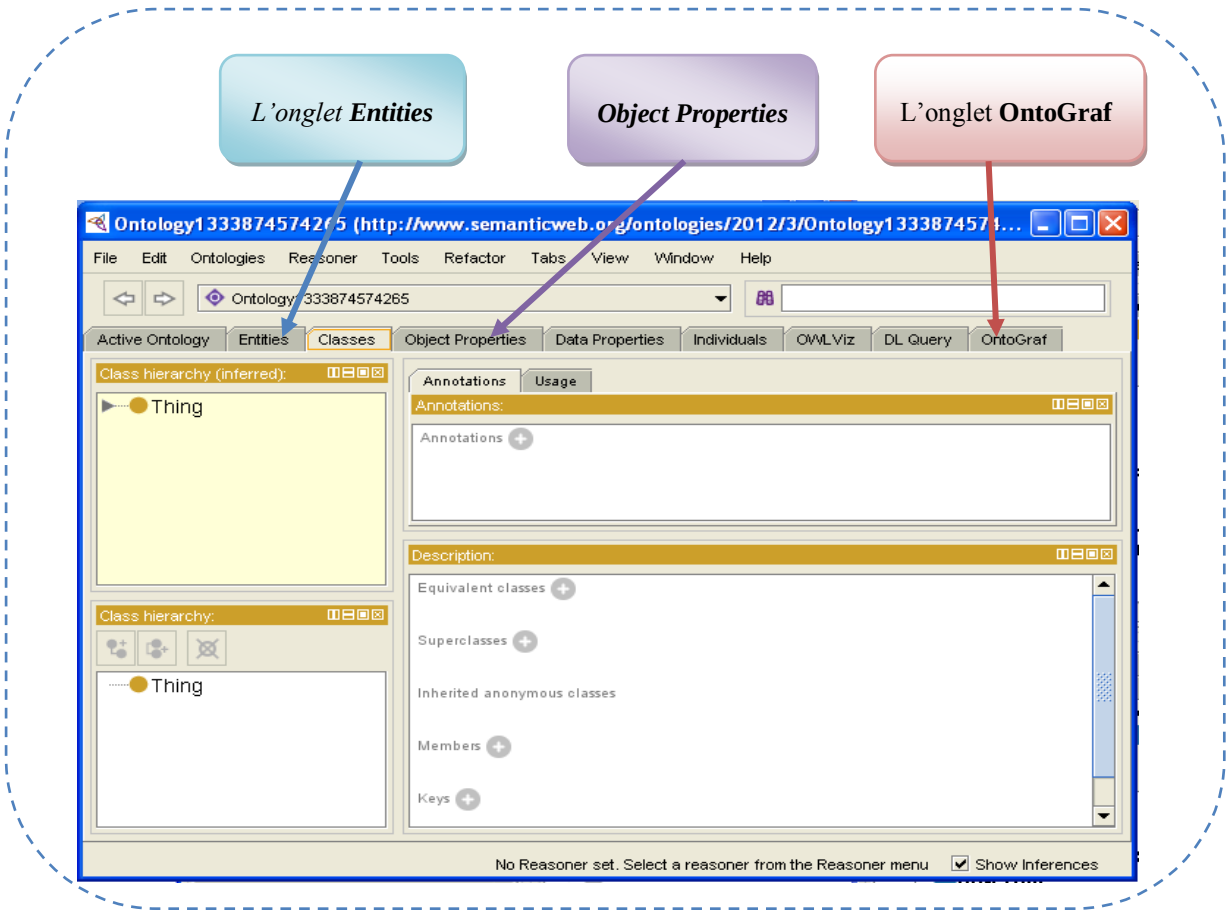

Figure IV.2 : Interface de l'outil Protégé (version 4.1).

Cette interface est composée de plusieurs onglets :

- L'onglet «Active ontologie» permet de visualiser les informations relatives au projet telle que l'URI de l'ontologie et les espaces de nom associés et d'ajouter des métadonnées par le biais d'annotations.
- L'onglet «Entities» permet de définir les concepts de l'ontologie et leurs conditions/restrictions.
- L'onglet «Object Properties» permet de définir les relations entre les différents concepts.
- L'onglet «Data Properties» permet de définir les attributs de concepts. leurs conditions/restrictions.
- L'onglet «Individuals» permet de peupler l'ontologie.
- L'onglet «OntoGraf» permet de visualiser le graphe de l'ontologie.

D'autres onglets (comme OWLViz ou DL Query) peuvent être ajoutés par l'intermédiaire de plugins supplémentaires.

2) Création des classes et la hiérarchie des classes

Nous avons par la suite introduit les concepts du modèle d'ontologie conçu précédemment. La spécification de cette ontologie par les concepts relatifs au domaine de modélisation et de simulation aboutit à une ontologie générique.

Le procédé que nous avons choisi pour définir la hiérarchie des classes est le procédé de haut en bas, c'est-à-dire que l'on a commencé par définir les concepts les plus généraux, nous les avons par la suite spécialises.

Lors de l'introduction des concepts de l'ontologie, nous avons choisi de respecter certaines conventions :

- Tous les noms des classes commencent par une lettre majuscule.
- Tous les noms des attributs commencent par une lettre minuscule.
- Les noms des attributs composés de plusieurs mots sont séparés par un tiret, exemple : analysed-by, is-a.
- Tous les noms sont au singulier. Exemple : Analyser, State.

Comme illustre la figure IV.2 l'interface utilisateur de Protégé contient divers onglets associés à des tâches spécifiques ; parmi ces onglets, nous présentons ceux utilisés dans l'implémentation de l'ontologie DeMO-E.

L'onglet **Classes** permet de créer les classes (concepts) et la hiérarchie de classes, une classe universelle **Thing** est utilisée comme racine pour cette hiérarchie, et la création des sous-classes se fait par le choix de la classe mère, suivi par une simple cliquesur le bouton de création des sous-classes comme montre la figure IV.3.

Dans cet onglet, les classes disposant des sous-classes apparaissent sur l'onglet sont précédées par le signe (▼), alors que les classes disposant des sous-classes qui n'apparaissent pas sur l'onglet sont précédées par le signe (▶). Ces sous-classes peuvent être montrées par une simple clique sur ce signe, l'ensemble de concepts créés se synthétise dans une hiérarchie

comme montre la figure suivante. Dans ce qui suit des captures écran des éléments de l'ontologie DeMO-E.

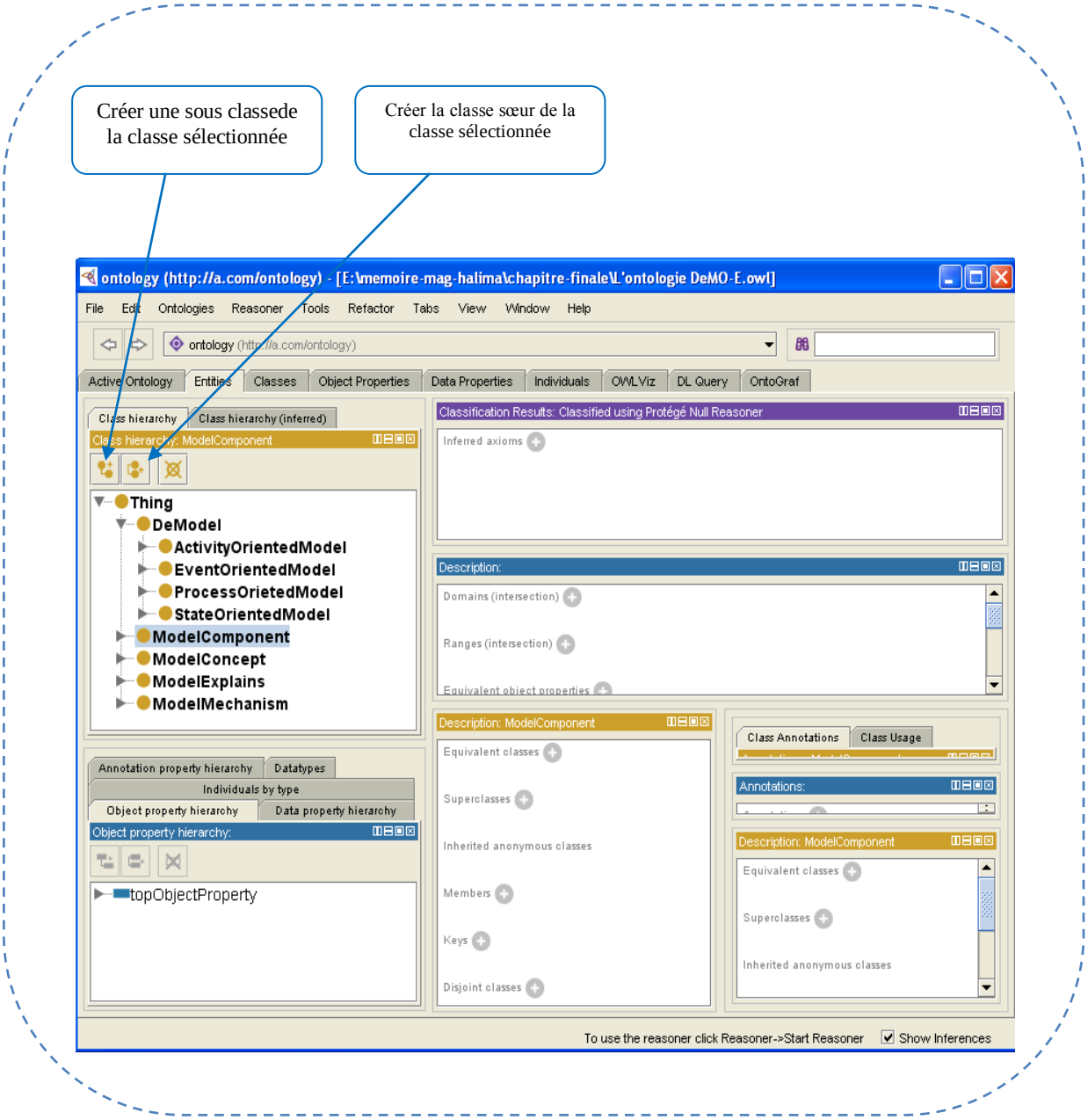

Figure IV.3 : Création des classes dans Protégé 4.1.

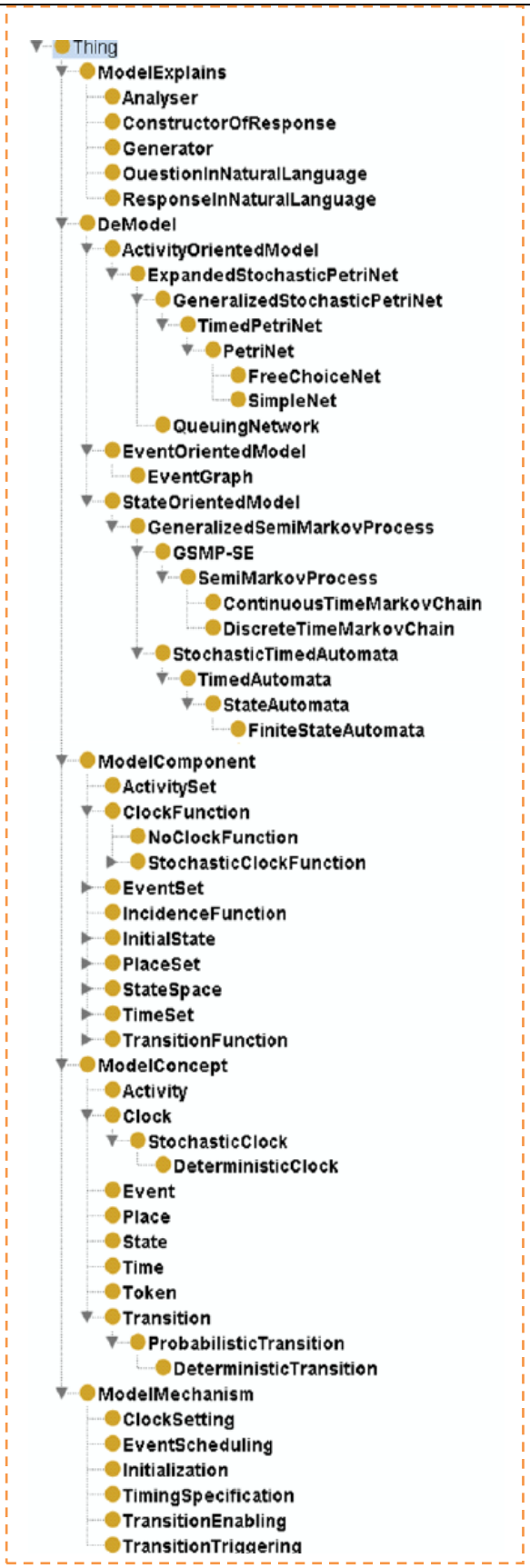

Figure IV.4 : Hiérarchie de l'ontologie DeMO-E sous Protégé**.**

Nous donnons dans la figure IV.5 une capture d'écran de l'ontologie DeMO-E édité sous Protégé.

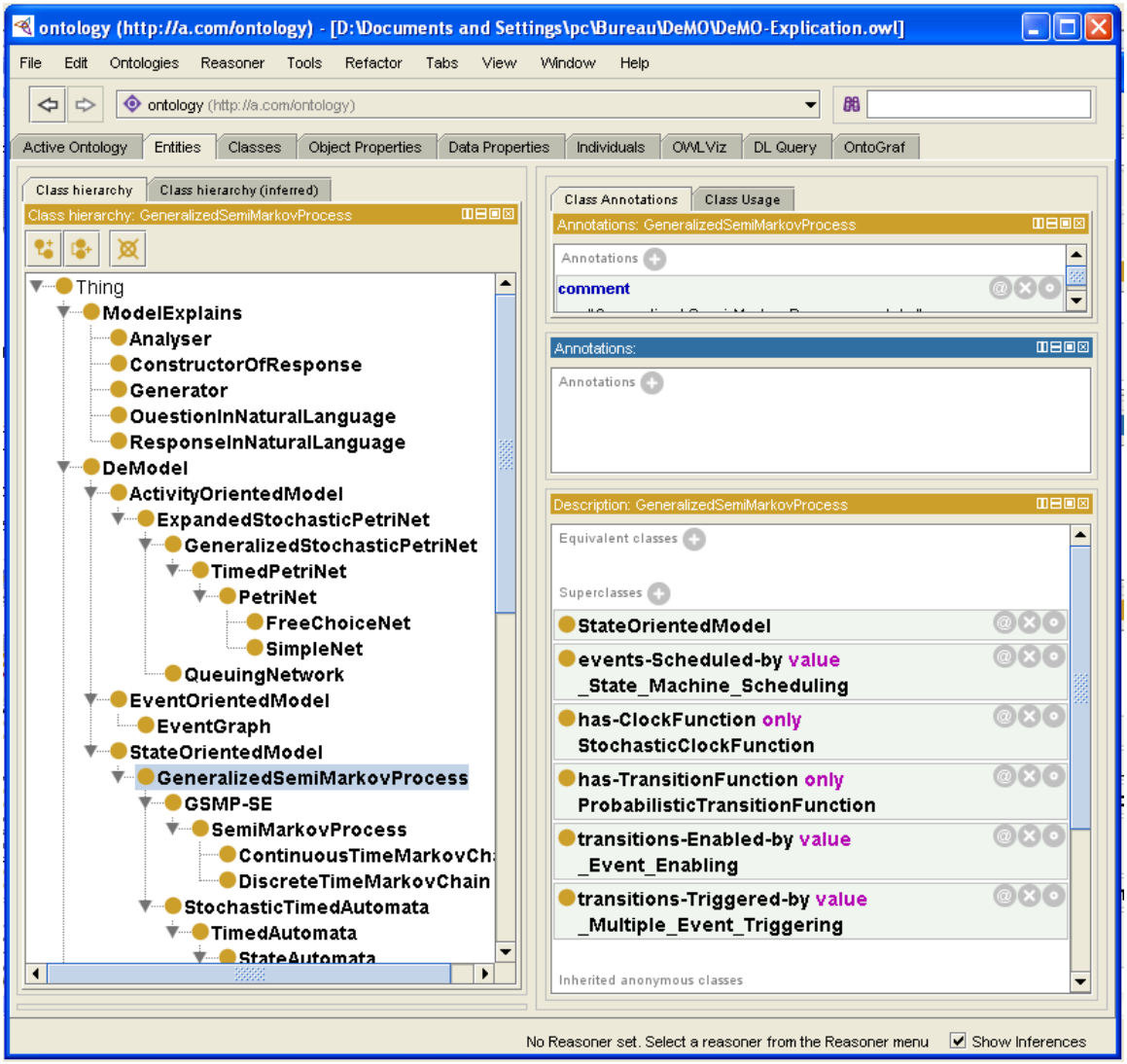

Figure IV.5 : L'ontologie DeMO-E édité sous l'outil Protégé.

3) Définition des propriétés des classes

Après avoir construit les concepts, nous allons maintenant créer les propriétés pour chacun d'eux, les attributs vont être créés sous Protégé 4.1 par l'onglet **dataTypeProperty**et les relations par l'onglet **objectProperty**.

Les propriétés d'une classe sont les propriétés héritées de sa super classe, plus ses propres propriétés privées, la figure IV.6 montre les potentialités de Protégé pour la création des propriétés.

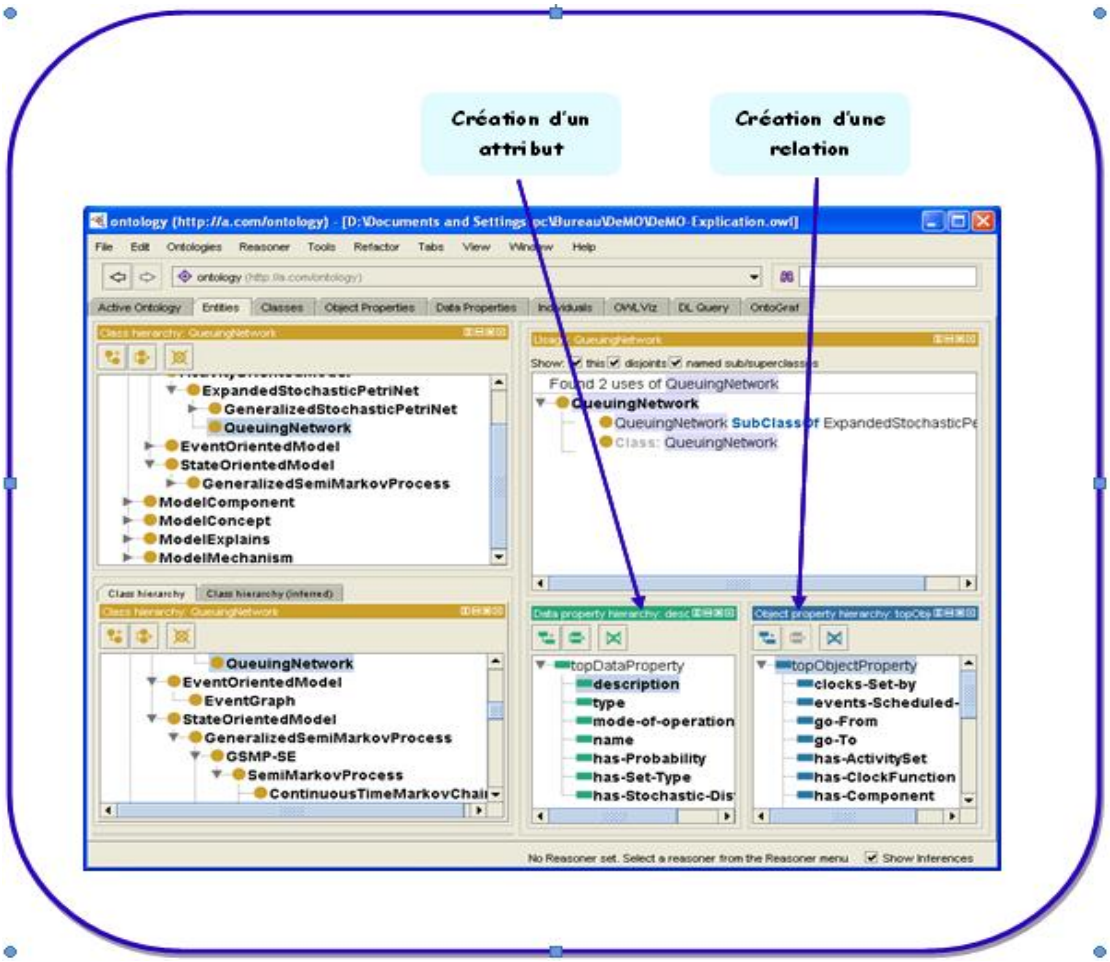

Figure IV.6 : Création de propriétés pour une classe.

IV.2.5 Génération du code OWL:

L'outil Protégé a été conçu pour dégager et libérer le développeur de la complexité du codage, même pour implémenter une petite ontologie, celle–ci va prendre plusieurs lignes du codes et nécessite un grand effort, ce que nous pouvons constater après la génération du code de notre ontologie, une partie de ce code de l'ontologie DeMO-E sera présentée dans la figure suivante.

```
<!-- http://a.com/ontology#clocks-Set-by -->
<owl:ObjectProperty rdf:about="http://a.com/ontology#clocks-Set-by">
<rdf:type rdf:resource="&owl;FunctionalProperty"/>
<rdfs:range rdf:resource="http://a.com/ontology#ClockSetting"/>
<rdfs:domain rdf:resource="http://a.com/ontology#StateOrientedModel"/>
</owl:ObjectProperty>
<!-- http://a.com/ontology#events-Scheduled-by -->
<owl:ObjectProperty rdf:about="http://a.com/ontology#events-Scheduled-by">
<rdf:type rdf:resource="&owl;FunctionalProperty"/>
<rdfs:range rdf:resource="http://a.com/ontology#EventScheduling"/>
<rdfs:domain rdf:resource="http://a.com/ontology#StateOrientedModel"/>
</owl:ObjectProperty>
<!-- http://a.com/ontology#gener -->
<owl:ObjectProperty rdf:about="http://a.com/ontology#gener"/>
<!-- http://a.com/ontology#go-From -->
<owl:ObjectProperty rdf:about="http://a.com/ontology#go-From">
<rdf:type rdf:resource="&owl;FunctionalProperty"/>
<rdfs:range rdf:resource="http://a.com/ontology#State"/>
<rdfs:domain rdf:resource="http://a.com/ontology#Transition"/>
</owl:ObjectProperty>
<!-- http://a.com/ontology#go-To -->
<owl:ObjectProperty rdf:about="http://a.com/ontology#go-To">
```

Figure IV.7 : Un extrait du code OWL de DeMO-E généré par Protégé.

IV.2.6 Visualisation de l'ontologie :

La visualisation du graphe de l'ontologie s'effectue grâce à l'onglet «OntoGraf» qui existe dans Protégé 4.1.

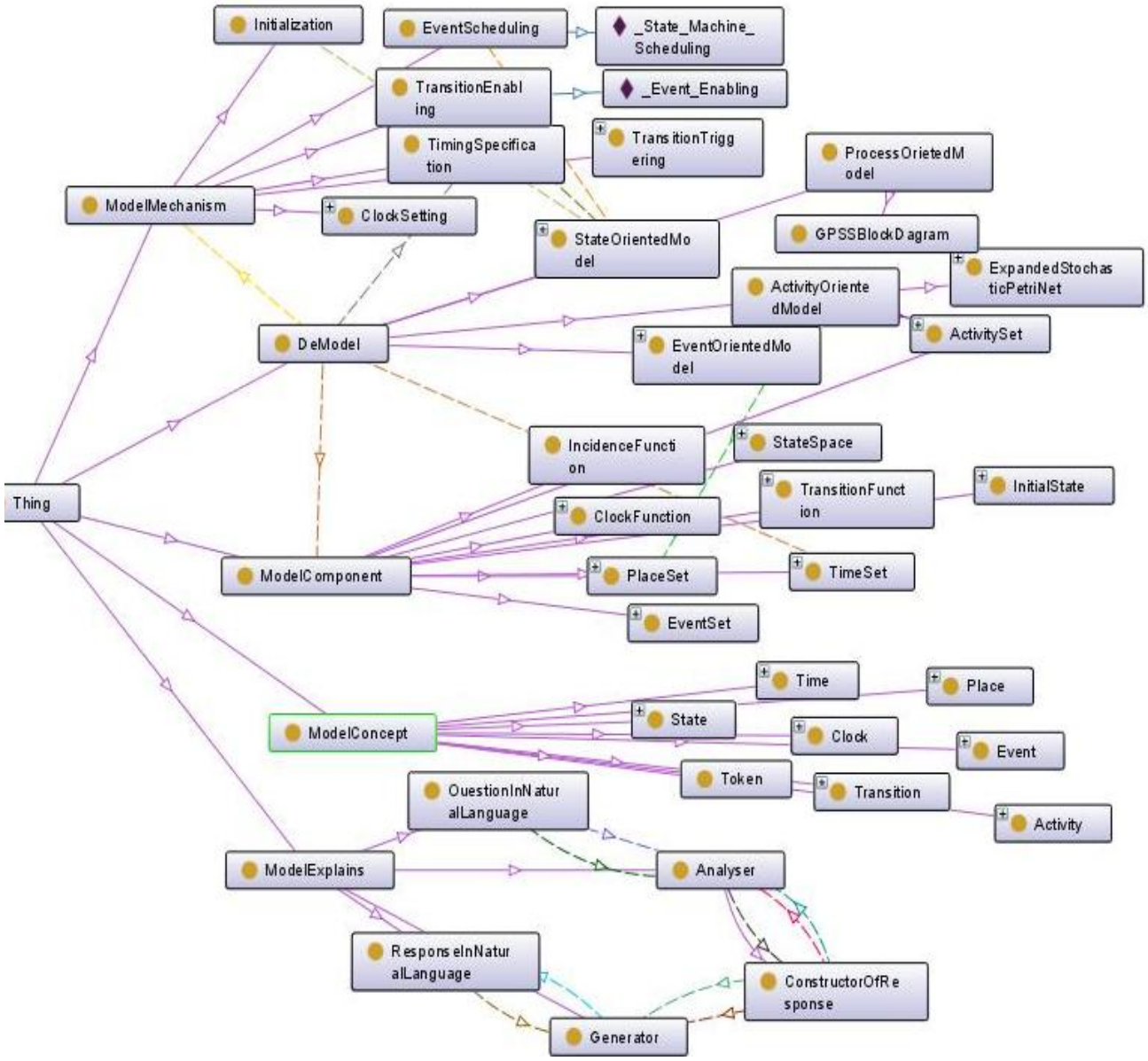

Figure IV.8 : La visualisation du graphe de l'ontologie.

IV.3 Évaluation de l'ontologie :

Un des grands avantages de l'utilisation de Protégé est la possibilité de faire des raisonnements sur les données présentes dans l'ontologie que l'on a créée grâce aux moteurs d'inférence afin de vérifier la cohérence de l'ontologie [8] et de déduire de nouvelles connaissances ou encore pour faire de la classification.

Pour pouvoir faire cela, nous utilisons le raisonneur Racer[9] (la configuration du Racer est détaillée en annexe A) pour tester l'ontologie DeMO-E, les principaux services offerts par Racer sont :

- Le test de consistance (satisfiabilité, cohérence).
- Le test de classification (subsomption).

Racer fournit les justifications des inférences qu'il génère et extrait pour certaines incohérences détectées, un ensemble unique composé des axiomes ayant causé l'incohérence [Racer, 2012], cependant, il ne propose pas de résolutions.

IV.3.1 Test de consistance :

Le test de consistance fournit par Racer est effectué en se basant sur la description des classes (conditions), il permet de s'assurer qu'aucune définition d'une classe n'est contradictoire avec une autre (l'inexistence des classes contradictoires) c'est à dire vérifier que pour chaque classe, il faut qu'il existe au moins un individu membre de cette classe.

La figure IV.9 présente le résultat de ce test, qui indique que toutes les classes sont consistantes.

[8] C'est à dire que l'ontologie créée ne contient pas des définitions contradictoires.

[9] Mais il en existe d'autres tels que Jess, Hermit, ...

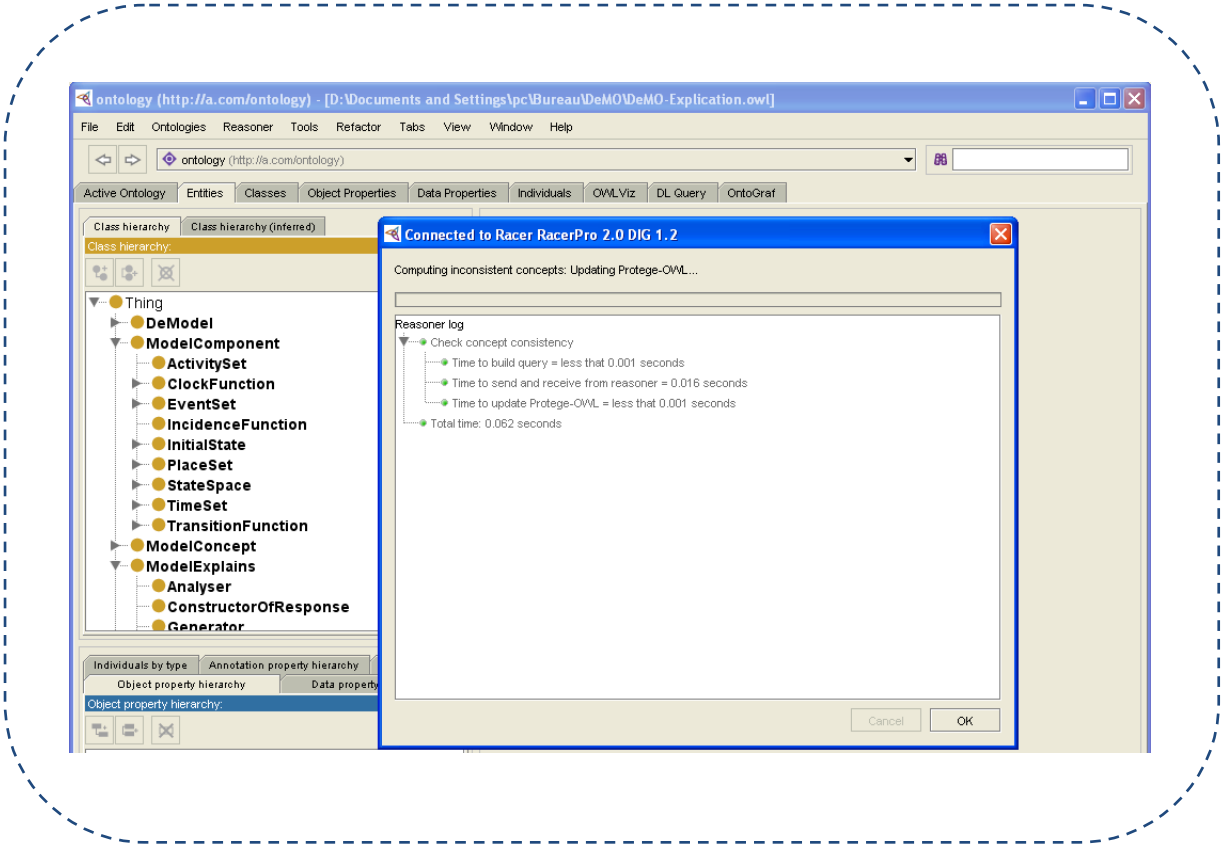

Figure IV.9 : Vérification de l'inconsistance des concepts.

IV.3.2 Test de classification :

Ce test permet de vérifier si une classe est une sous classe d'une autre classe ou non, lorsque ce test est invoqué, le test de consistance est d'abord effectué pour toutes les classes de l'ontologie parce que les classes inconsistantes ne peuvent pas être classées correctement. Le résultat de ce test est affiché graphiquement par Protégé. Il consiste en une «Class hierarchy inferred» calculée à partir de l'hiérarchie construite manuellement par l'ontologiste. Cette hiérarchie de classe inférée est une hiérarchie où les classes sont classifiées selon la relation superclasse/sous-classe.

La différence entre les deux hiérarchies est affichée en rouge par le système si elle existe (dans cette hiérarchie les classes inconsistantes apparaîtront dans une nouvelle classe Nothing), comme montre la figure suivante, la nouvelle hiérarchie est visualisée juste à coté de «Class hierarchy». A partir du résultat obtenu par ce test nous remarquons qu'aucune suggestion n'est produite par le raisonneur et que «Class hierarchy» et «Class hierarchy inferred» sont identiques.

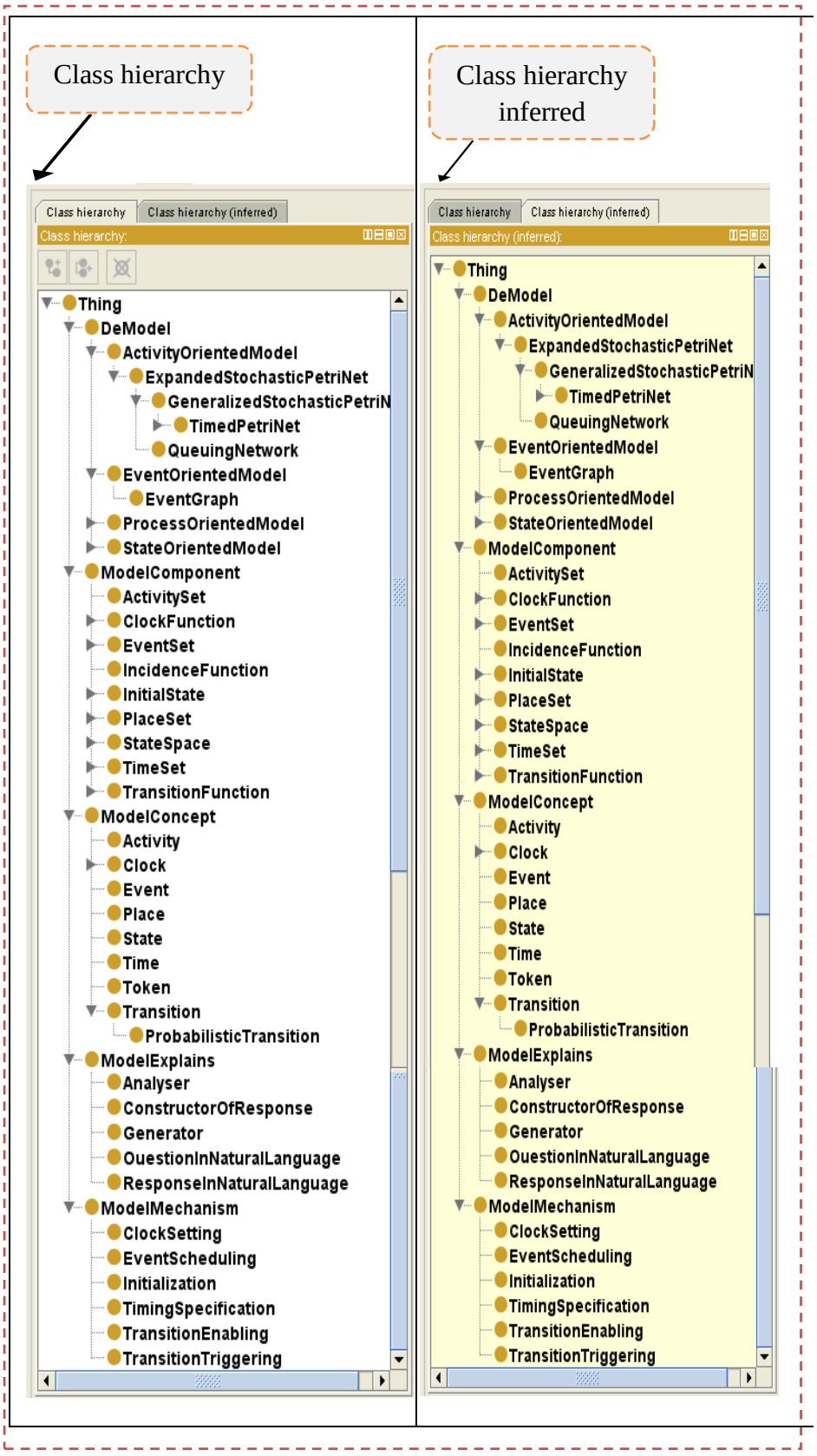

Figure IV.10 : Vérification de la classification.

IV.4 Conclusion :

Dans ce chapitre nous avons présenté l'implémentation de notre ontologie DeMO-E, nous avons tout d'abord présenté l'environnement de développement Protégé 4.1, puis nous avons donné une description détaillée de la phase d'implémentation à travers des captures d'écran de l'ontologie sous Protégé et on termine ce chapitre par donner des illustrations des tests faits sur DeMO-E pour les valider.

L'atout majeur d'DeMO-E consiste à sa cohérence ainsi que sa consistance montrée par les tests effectués sur l'ontologie, de ce fait, elle est prête à une future évaluation au sein de n'importe quelle application en rapport avec le domaine qu'elle modélise.

À travers tous ce que nous avons réalisé dans ce chapitre, nous pouvons dire que le processus suivi pour la construction de l'ontologie DeMO-E nous a permis de réussir finalement à construire une méta-ontologie formelle pour la modélisation et la simulation intelligente.

Conclusion générale

Conclusion générale et perspectives

Le travail réalisé dans le cadre de ce mémoire s'inscrit dans le domaine de l'Ingénierie Ontologique, et se voulant essentiellement une contribution à la problématique de la gestion des connaissances dans le domaine de la modélisation et simulation intelligente.

Actuellement, les ontologies constituent un enjeu stratégique dans la représentation et la modélisation des connaissances. Récemment, elles ont été introduites pour formaliser les connaissances dans le domaine de modélisation et simulation intelligente.

Le déploiement d'ontologie en modélisation et simulation permet l'intégration intelligente d'information et la gestion des connaissances, de fournir une connaissance partagée et commune sur ce domaine, elles définissent les primitives indispensables pour leur représentation, ainsi que leur sémantique dans un contexte particulier, outre la réutilisation et le partage de connaissances, elles permettent de faciliter la communication entre les acteurs de différentes organisations et en particulier, la réalisation de l'interopérabilité entre les différents systèmes, elles permettent non seulement la création de systèmes mais aussi la capacité de gérer des connaissances mais aussi de raisonner sur ces connaissances et, pourquoi pas, d'en produire de nouvelles.

Dans ce cadre, nous avons tout d'abord étudié les notions issues du domaine de modélisation et de simulation afin de mieux comprendre les fondements théoriques ainsi que les outils de mise en œuvre sur ordinateur. Ensuite, nous avons discuté des ontologies et leur utilisation, nous avons commencé par la définition de la notion d'"ontologie", nous avons présenté les trois principaux formalismes de présentations à savoir les schémas, les réseaux sémantiques et les logiques de descriptions. Nous avons découvert, les langages de leur implémentation et enfin certains outils servant leur exploitation.

Une fois l'ontologie conceptuelle mise au propre, nous avons passé à son opérationnalisation avec l'outil Protégé qui nous a permis de générer automatiquement le code OWL de notre ontologie. Enfin, les tests de consistance et de classification ont été appliqués sur l'ontologie opérationnelle par le biais du raisonneur RACER, cela a donné naissance à une ressource ontologique cohérente prête à une évaluation et une exploitation par les différents acteurs de la modélisation et de la simulation.

À travers tous ce que nous avons réalisé dans ce mémoire, nous pouvons dire que le processus suivi pour la construction de notre ontologie, nous a permis de réussir finalement à construire une méta-ontologie pour la modélisation et la simulation intelligente qu'on a baptisée «DeMO-E» Descret even Modeling Ontology-Explication, cela permet aux utilisateurs de mieux comprendre le fonctionnement du modèle simulé et l'explication du résultat de simulation, de favoriser l'interopérabilité de données, la recherche, la récupération de l'information, et l'inférence automatisée, elle permet aussi de définir une structure hiérarchique (dans un certain sens) d'un ensemble de formalismes de modélisation pertinents et les relations entre eux, de trouver facilement le meilleur formalisme pour modéliser un système donné et de définir les composants de modélisation.

Comme perspectives du travail réalisé dans ce mémoire:

- De compléter ce travail du point de vue architecture logiciel et implémentation.
- D'évaluer notre ontologie DeMO-E à la limite par les experts du domaine, leur point de vue concernant le contenu de l'ontologie a un impact très important pour vérifier sa complétude.
- Par la suite, la mettre au point au sein d'une application concrète de la modélisation et de la simulation, il est fort probable que l'ontologie soit révisée, raffinée et complétée.
- Créer un outil de navigation et de visualisation de l'ontologie.

En effet l'ontologie que nous avons proposée peut être largement exploitée dans un autre créneau, comme celui de l'enseignement de la simulation.

Bibliographie

Références bibliographiques :

[Amardeilh, 2007] Amardeilh F. (2007). W*eb sémantique et informatique linguistique: propositions méthodologiques et réalisation d'une plateforme logicielle*. Thèse de doctorat Université Paris X Ŕ Nanterre.

[Antoine, 2008] Paul Antoine. (*2008*). *Approche de modélisation approximative pour des systèmes à événements discrets : Application à l'étude de propagation de feux de forêt.* Thèse de doctorat de l'université Corse- Pasquale Paoli. 178pages.

[Azadeh et Ghaderi, 2006] Ali Azadeh et Farid Ghaderi. (2006).*A Framework for Design of Intelligent Simulation Environment*. Journal of Computer Science 2 (4): 363-369. ISSN 1549-3636. University of Tehran-Iran.

[Bachimont, 2000] Bachimont B. (2000). *Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie des connaissances*. In Ingénierie des connaissances. Évolution Récentes et nouveaux défis. Paris: Eyrolles.

[Buchheit et al, 1993] M. Buchheit, F. Donini et A. Shaerf. (1993).*Decidable reasoning in terminological knowledge representation systems.* Journal of artificial intelligence research. P 109-138.

[Baneyx, 2007] Baneyx A. (2007). *Construire Une Ontologie De La Pneumologie : Aspects Théoriques, Modèles Et Expérimentations.* Thèse de doctorat, Université Pierre et Marie Curie - Paris 6, Laboratoire Inserm UMR_S 872 – Santé Publique et Informatique Médicale, France.

[Belattar, 2000] Belattar B. (2000). *Simulation & modélisation.* Support de Cours. Département d'Informatique, faculté des sciences de l'ingénieur, Université de Batna (Algérie). 167 pages.

[Benjamin et al, 2005] Benjamin P, Akella K V, Fernandes R. (2005). *An ontology-driven framework for* process-oriented applications, Proceedings of Winter Simulation Conference, P 2355-2363.

[Borst, 1997] Borst W. N. (1997). *Construction of Engineering Ontologies*. Center for elematica and Information Technology, University of Tweenty, Enschede, NL.

[Bourouis, 2010] Bourouis A, (2010). *Une approche de modélisation des systèmes à évènement discrets utilisant les CONCEPT-MAPS*. Thèse de doctorat de l'université El Hadj Lakhdar - Batna. 147 pages.

[Erard et Deguenon, 1999] Erard, P.J. et Deguenon P. (1999) *Simulation par événements discrets*. P.P.U.R, 1er édition, Lausanne. 417 pages.

[Fayez et al, 2005] Fayez M, Rabelo L et Mollaghasemi M. (2005), *Ontologies for supply chain simulation modeling*, Proceedings of Winter Simulation Conference, P 2364-2370.

[Fernandez et al, 1997] Fernandez M., Gomez-Pérez A., Juristo N. *METHONTOLOGY: From Ontological Art Towards Ontological Engineering*. Proceedings of the AAAI-97 Spring Symposium Series on Ontological Engineering, Stanford, CA, USA, 1997.

[Fishwick et Miller, 2004] Fishwick P et Miller J. (2004). *Ontologies for modeling and simulation: Issues and approaches.*Proceedings of the 2004 Winter Simulation Conference.P 259-264.

[Fishwick, 1996] Fishwick, P.A. (December 1996). *Web-Based Simulation: Some Personal Observations*. In: Proceedings of the 1996 Winter Simulation Conference, Coronado, CA, 8-11. P 772-779.

[Fishwick et Miller, 2004] Paul A Fishwick et John A Miller (2004). *Ontologies for modeling and simulation: issues and approaches*. Proceedings of the of the 2004 Winter Simulation Conference. P 259-264.

[Fleury et al, 2007] Gérard Fleury, Philippe Lacomme et Alain Tanguy. (2007). *Simulation à événements discrets.* Éditions Eyrolles.

[Furst, 2004] Furst F. (2004). *Contribution à l'ingénierie des ontologies : une méthode et un outil d'opérationnalisation*. Thèse de doctorat, Université de Nantes. 184 pages.

[Gaëlle, 2002] Gaëlle Lortal. (2002). *État de l'art ontologies et intégration/fusion d'ontologies*.

[Gomez, 1999] Gomez Pérez A. (1999). *Développements récents en matière de conception, de maintenance et d'utilisation d'ontologies.* 3èmes rencontres Terminologie et intelligence artificielle TIA.

[Gomez et Benjamin, 1999] Gomez Pérez et Benjamins V R. (1999). *Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods*. Proceedings of the IJCAI-99 workshop on Ontology and Problem-Solving Methods (KRR5).

[Gruber, 1993] Gruber T, (1993). *A translation approach to portable ontology specifications, KnowledgeAcquisition*. 5(2), P 199-220.

[Guarino, 1998] Guarino N. (1998). *Formal Ontology and Information Systems.* IOS Press.

[Guarino et Giaretta, 1995] Guarino N., et Giaretta P. (1995). Ontologies *and Knowledge Bases: Towards a Terminological Clarification*. In Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing, Mars N. J. I., Amsterdam: IOS Press.

[Habchi, 2001] Georges Habchi, (2001). *Conceptualisation et modélisation pour la simulation des systèmes de production.* L'habilitation a dirigé des recherches. Université de Savoie.

[Hernandez, 2005] Hernandez N. (2005). *Ontologies de domaine pour la modélisation du contexte en recherche d'information*. Thèse de doctorat, université Paul Sabatier de Toulouse.

[Kassel, 2002] Kassel G. (2002). *OntoSpec : une méthode de spécification semi-informelle d'ontologies*. In Actes des journées francophones d'Ingénierie des Connaissances.

[Kim et al, 2002] Kim T, Lee T, Fishwick P. (2002).*A Modeling Two Stage and Process Simulation for Web-Based Modeling and Simulation*. ACM Transactions on Modeling and Computer Simulation, 12 (3). P 230-248.

[korichi, 2009] korichi A. (2009). *TCAO et Simulation : Vers une plate-forme d'analyse et de conception de systèmes de production orientée groupe*. Thèse de doctorat de l'université El hadj lakhdar - Batna. 217 pages.

[Kuljis et Paul, 2001] Kuljis J, Paul RJ. (2001). *Une évaluation des web-based simulation: où nous errons?.*Simulation Practice and Theory, 9. P 37-54.

[Lassila et Mcguiness, 2001] Lassila O et Mcguiness D. (2001). *The role of frame-basedrepresentation on the semantic Web*. Rapport technique KSL-01-02. Knowledge Systems Laboratory, Stanford University.

[Miller et al, 2001] Miller J, Fishwick PA, Taylor SJE, Benjamin P, Szymanski B. (2001).*La recherche et des opportunités commerciales dans Web basées sur des simulations.*Simulation Practice and Theory. 9, P 55-72.

[Miller et al, 2004] Miller, Baramidze, Fishwick et Amit P. (2004). *Investigating Ontologies for Simulation Modeling*. Proceedings of the 37th Annual Simulation Symposium (ANSS'04), Arlington, Virginia April. P 55-71.

[Miller et al, 2006] Miller J, Baramidze G T, SHETH A et al. Ontologies for Modeling and Simulation: An Initial Framework. University of Georgia. 47 pages.

[Miller et al, 2007] Miller J, He C, Couto J I. (2007) Impact of the Semantic Web on Modeling and Simulation, Department of Computer Science University of Georgia. 37 pages.

[Miller et Baramidze, 2005] Miller J et Baramidze G. (2005). *Simulation and the semantic web*, Proceedings of Winter Simulation Conference. P2371-2377.

[Minsky, 1975] Marvin Minsky. (1975).*A framework for representing knowledge*. In P.M WINSTON, réd, The Psychology of Computer Vision. McGraw Hill, New York. P 211–277.

[Mizoguchi et Ikeda, 1996] Mizoguchi R et Ikeda M. (1996). *Towards Ontological Engineering* (AI-TR-96-1), Osaka: ISIR.

[Napoli, 1997] Napoli A. (Décembre 1997). *Une introduction aux logiques de descriptions*. N° 3314.

[Neeches et al, 1991] Neeches R., Fikes R. E., Finin T., Gruber T. R., Senator T. et Swartout W. R. (1991). *Enabling technology for knowledge sharing*. AI Magazine.

[Page et al, 1998a] Page, E.H., Buss, A., Fishwick, P.A., Healy, K.J., Nance, R.E. and Paul, R.J.(1998). *The Modeling Methodological Impacts of Web-Based Simulation*. In:Proceedings of the 1998 SCS International Conference on Web-Based Modeling and Simulation.San Diego, CA, 11-14 January. P 123-128.

[Page et al, 2000] Page, E.H., Buss, A., Fishwick, P.A., Healy, K.J., Nance, R.E. et Paul, R.J. (2000). *Web-Based Simulation: Revolution or Evolution*. Submitted to: ACM Transactions on Modeling and Computer Simulation, February.

[Page et Opper, 2000] Page E.H., Opper J.M. (2000). *Investigating the Application of Web-Based Simulation Principles within the Architecture for a Next-Generation Computer Generated Forces Model*. Future Generation Computer Systems. P.159-169.

[Psyché, 2007] Psyché V. (2007). *Rôle des ontologies en ingénierie des EIAH : Cas d'un système d'assistance au design pédagogique.* Thèse de doctorat, Université du Québec-Montréal.

[Reed et al, 2000] Reed JA, Follen GJ, Afjeh AA. (2000). *Amélioration de l'Aircraft Design Process Using Web-Based Modeling and Simulation*. ACM Transactions on Modeling and Computer Simulation. P 58-83.

[Shen, 1998] Shen, C. (11-14 January 1998). *Discrete-Event Simulation on the Internet and the Web*. In Proceedings of the 1998 SCS International Conference on Web-Based Modeling and Simulation, San Diego, CA. P. 57-62.

[Silver et al, 2007] Silver G, Al-Haj Hassan O, Miller J.(2007). *From domain ontologies to modeling ontologies to executable simulation models.* Proceedings of Winter Simulation Conference, P 1108 -1117**.**

[Silver et al, 2009] Silver G, Miller J, Bellipady K R et York W. (2009).*Supporting interoperability using the discrete-event modeling ontology (demo*). Proceedings Winter Simulation Conference, P.1399-1410.

[Sowa, 1992] John SOWA. (1992). *Conceptual graphs summary*. P3-51.

[Sowa, 2000] John SOWA. (August 2000) *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole.

[Staab et Maedche, 2000] Staab S, Maedche A. (2000). *Axioms are objects too: Ontology engineering beyond the modeling of concepts and relations*. Research report 399, Institute AIFB, Karlsruhe.

[Swartout et al, 1997] Swartout B, Patil R, Knight K et Russ T. (1997). *Towards Distributed Use of Large Scale Ontologies*. Spring Symposium Series on Ontological Engineering, Stanford University, CA.

[Valéry et al, 2003] Valéry Psyché, Olavo Mendes et Jacqueline Bourdeau. (2003). *Apport de l'ingénierie ontologique aux environnements de formation à distance*. Article de recherche, Revue STICEF, Volume 10, ISSN : 1764-7223.

[Wladimir et al, 2004] Wladimir, A. F., Hirata C. M., Edgar T. Y. (2004). *GroupSim: A Collaborative Environment for Discrete Event Simulation Software Development for the World Wide Web*. *SagePub*, Vol. 80, Num 6. P 257-272.

[Xavier, 2005] Xavier Lacot. (2005). *Introduction à OWL, un langage XML d'ontologies Web.*

Sites WWW

[DeMO, 2011] DeMO (Discrete Event Modeling Ontology), http://www.cs.uga.edu/~jam/jsim/DeMO/DeMO_intro.html(consulté le 22/01/2011).

[Protégé, 2011] Protégé «The Protégé Ontology Editor and Knowledge Acquisition System», http://protege.stanford.edu (consulté le 14/09/2011).

[Protégé Tutorial, 2011] Protégé OWL Tutorial, http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/(consulté le 22/01/2011).

[Racer, 2012] Le système Racer « Présentation et téléchargement du logiciel RacerPro, en anglais », http://www.racer-systems.com/(consulté le 22/01/2011).

Abréviations

Abréviations

A

AS : Activity-Scanning

D

DAML : DARPA Agent Markup Language

DEM : Discrete-Event Modeling

DeMO : Discrete event Modeling Ontology

DeMO-E : Discrete event Modeling Ontology-Explication

DL : Description Logic

E

EG : Event Graphs

ES : Event-Scheduling

ESPN : Extended Stochastic Petri Net

G

GC : Graphe Conceptuel

GPSS : General Purpose Simulation System

GMSP : General Semi-Markov Processes

I

IA : Intelligence Artificielle

IC : Ingénierie des Connaissances

IO : Ingénierie Ontologique

L

LD Logique de Description

LSDIS: Large Scale Distributed Information Systems

O

OIL : Ontology Inference Layer

OWL : Web Ontology Language

P

PI : Process Interaction

R

RDF : Resource Description Framework

RDF-S : Resource Description Framework-Schema

RACER: Renamed Abox and Concept Expression Reasoner

S

SMP : Semi-Markov Processes

U

URI : Uniform Resource Identifier

W

W3C : World Wide Web Consortium

X

XML : eXtensible Markup Language

XMLS : eXtensible Markup Langage Schema

Annexe

ANNEXE : La configuration de moteur d'inférences Racer

RACER (Renamed Abox and Concept Expression Reasoner) [HM01] est le moteur d'inférence sans doute le plus connu et l'un des plus utilisés dans le domaine d'ontologie pour tester ces performances et sa stabilité. Racer travaille sur les ontologies modélisées par son langage, mais il accepte des ontologies décrites en RDF ou OWL, ces dernières étant traduites vers le langage utilisé par Racer, c'est un fichier exécutable disponible sous la forme d'un serveur pour Linux et Windows.

Racer peut être utilisé dans Protégé pour raisonnement et contrôle de consistance de l'ontologie afin d'obtenir une ontologie cohérente, il peut être lancé directement par un invité de commande ou par un double click sur l'icône du programme.

Le raisonneur RACER est accessible par les protocoles standards TCP ou http, mais avant d'y accéder, il faut d'abord configurer la connexion au serveur qui l'héberge. Dans notre cas, nous allons tester l'ontologie localement ; pour ce faire, il faut connecter RACER à Protégé en suivant les étapes suivantes :

- Lancer l'outil Protégé,
- Activer le menu *OWL* de Protégé,
- Sélectionner l'option *Préférences*,

Dans la fenêtre de dialogue qui s'affiche (*OWL préférences*, voir la figure A.1) il faut s'assurer que l'URL du raisonneur est réglé sur *'http://localhost:8080'* ou *'http://127.0.0.1:8080'* qui est habituellement la valeur par défaut. Après ça, et pour le valider en cliquant sur le bouton 'Close' comme la montre la figure ci-dessous.

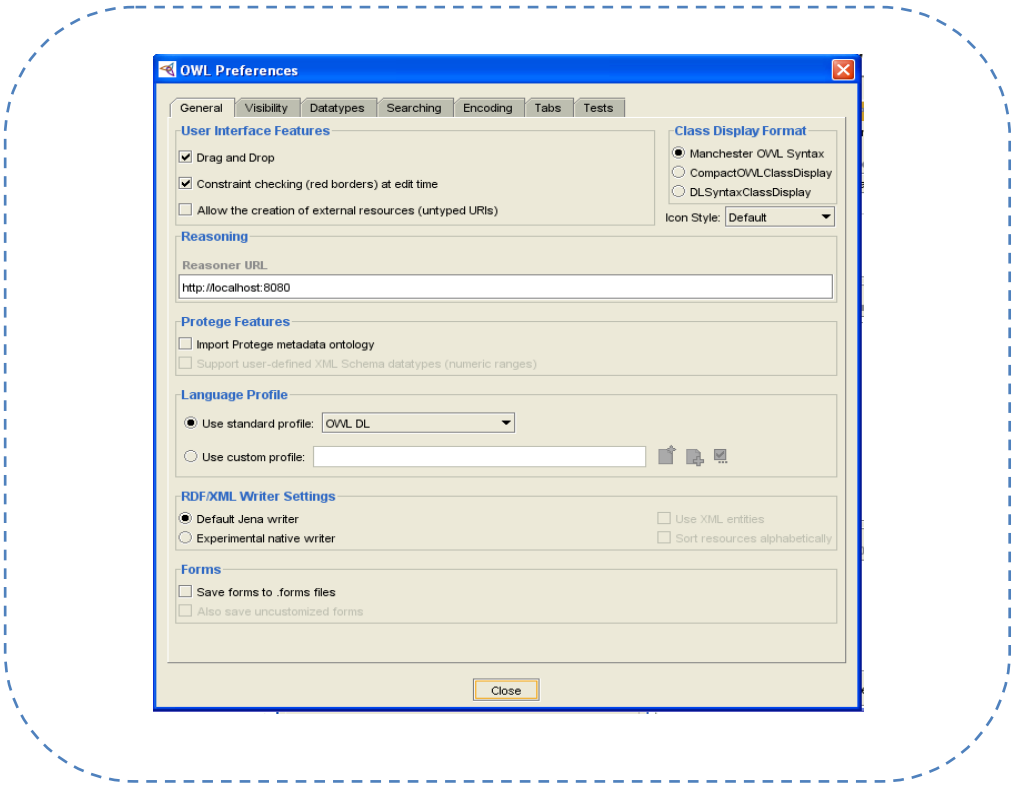

Figure A.1 La fenêtre de Préférences OWL.

Télécharger et exécuter RACER localement sous Windows, les services HTTP et TCP vont être activés sur le port 8080 de la machine local (localhost), comme illustré dans la figure A.2.

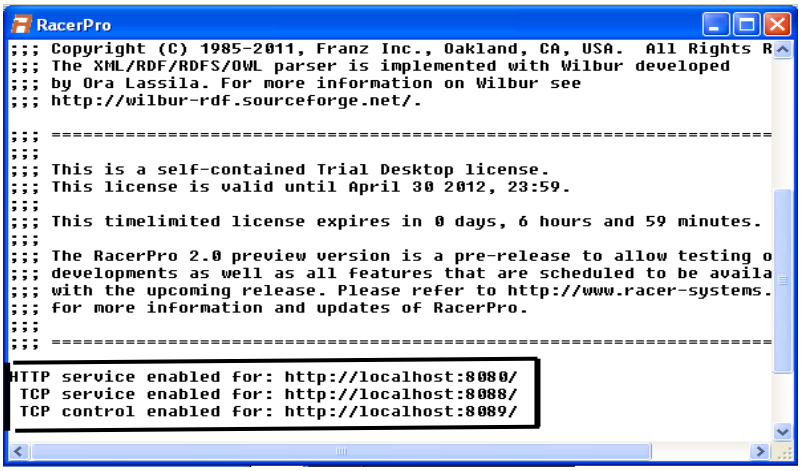

Figure A.2 L'exécutable de RACER sous Windows.

Résumé-Abstract

Résumé

Le travail réalisé dans ce manuscrit se situe au croisement de deux domaines : l'ontologie et le domaine de modélisation et simulation. La contribution dans ce travail de recherche est essentiellement la proposition d'une ontologie formelle pour la modélisation et la simulation intelligente (plus particulier la modélisation et la simulation des systèmes à évènements discrets.) qu'on a baptisée «DeMO-E» Descret even Modeling Ontology-Explication. Nos avons orienté nos recherche vers la définition d'un environnement de simulation intelligente, d'une ontologie formelle, puis vers la spécification de solutions ontologiques permettant d'assister les spécialistes de domaine de simulation dans la construction des logiciels de simulation intelligente. Notre recherche a permis, d'une part, de fournir une connaissance partagée et commune sur le domaine de la simulation en nous rapprochant d'une typologie de modélisation standard, et d'autre part, de proposer une solution technique basée sur les offres actuelles du domaine d'ontologie.

Mots Clés *: Ontologie, Simulation, modélisation, DeMO-E, simulation intelligente, Explication*

Abstract

The research accomplished in this thesis is set in the crossing of two fields: the ontology and the domain of modeling and Simulation. The contribution in this research work is essentially the proposition of the formal ontology for Modeling and intelligent Simulation (more particularly the modeling and simulation of discrete event systems.) we have called «DeMO-E» Even Descret Modeling Ontology-Explanation.Thisresearchis turned towards to the definition of an Intelligent Simulation Environment, the formal ontology, then to the specification of ontological solutions allowing to assist the specialists of a simulation, our research has; firstly, to provide a common and shared knowledge on the field of simulation in bringing us closer to a typology of standard modeling, and secondly, to propose a technical solution based on current offers the domain ontology.

Keywords:*Ontology, Simulation, modeling, DeMO-E, Intelligent Simulation, Explication.*