si l'on se concentre directement sur le comportement des agents. Cette approche est indispensable à la conception de simulations large échelle, *i.e.* de simulations contenant un grand nombre d'agents se comportant différemment et entretenant un grand nombre d'interactions variées.

Expertise en informatique : Les éléments du modèle nécessitant une expertise en informatique apparaissent lors des descriptions fines du modèle et figurent donc à la fin du processus de conception. Cela permet d'impliquer des experts du domaine dans le processus de conception plus longtemps qu'avec une spécification directe du comportement et réduit donc les problèmes liés à l'interprétation de leurs propos pour construire le modèle.

Révisions de modèle : On passe graduellement de descriptions macroscopiques du phénomène, qui sont observables et donc relativement objectives, à des descriptions microscopiques, qui relèvent des hypothèses relatives à l'origine du phénomène simulé. **Puisque les hypothèses sont les éléments les plus susceptibles d'être modifiés lors des itérations du processus de simulation, leur introduction tardive dans le modèle réduit les efforts de révision du modèle.**

Réutilisation logicielle : La séparation entre ce que les agents sont capables de faire (*i.e.* les interaction) et le processus qu'ils utilisent pour décider de ce qu'ils font (*i.e.* la sélection d'interaction) favorise la réutilisation logicielle à deux niveaux :

- une interaction peut être réemployée chez des agents ayant des processus décisionnels différents ;
- une interaction peut être réutilisée dans d'autres simulations du même domaine. Des bibliothèques d'interactions sont ainsi conçues au fil des simulations.

De plus, les révisions du modèle sont facilitées puisqu'une modification du processus de décision des agent n'implique pas systématiquement un révision des actions qu'ils sont capables de faire.

Automatisation de l'implémentation : Il est possible d'automatiser partiellement ou entièrement la transformation du modèle en une implémentation, réduisant ainsi la possibilité d'introduire de mauvaises interprétations du modèle.

Vérification et validation : Dans le cas où le simulateur ne fournit pas les résultats escomptés, la conservation de la structure du modèle lors de l'implémentation favorise l'interprétation dans le modèle d'une erreur identifiée dans le programme.

4 Une pyramide d'outils conceptuels et logiciels

Afin de répondre à la problématique posée, nous décrivons dans ce manuscrit l'**approche transversale de conception intitulée IODA** (*Interaction Oriented Design of Agent simulations*), qui facilite la construction d'un vaste ensemble de simulations à l'aide d'une représentation des connaissances **centrée sur les interactions**. Nous explorons de plus selon la perspective de IODA certaines problématiques simulation et en dégageons des extensions facilitant la conception de certaines catégories de simulations. Nous définissons ainsi une approche reposant sur un **cœur générique pouvant être complexifié par une ou plusieurs extensions lorsque la complexité du phénomène simulé le nécessite**.

À cet effet, nous définissons d'une part le cœur de l'approche IODA (décrit dans la partie II), qui facilite la construction d'un vaste ensemble de simulations à l'aide d'une représentation des connaissances centrée sur les interactions.

D'autre part, en nous appuyant sur cette caractérisation « minimale » de IODA, nous explorons trois problématiques de simulation selon la perspective d'une approche centrée sur les interactions :

1. Faut-il nécessairement des interactions complexes afin de modéliser des comportements complexes ?
2. Comment utiliser IODA afin d'éviter d'introduire des biais dans une simulation ?
3. Comment adapter le concept d'héritage des langages objets à la simulation, afin de faciliter l'ingénierie des connaissances ?

Nous utilisons ces études afin de définir quatre extensions disjointes de IODA (décrites dans la partie III) qui fournissent des concepts et des outils méthodologiques traitant efficacement de ces problèmes.

Ce manuscrit suit cette logique, que nous schématisons sur la figure 1.

4.1 Cœur de l'approche

Dans IODA, nous cherchons à cumuler à la fois les avantages des approches transversales qui permettent de concevoir progressivement des modèles contenant un grand nombre d'informations, mais aussi des approches accordant un sens plus large à la notion d'interaction.

Les travaux réalisés dans cette thèse concrétisent les efforts initiés en 2001 par Jean-Christophe Routier, Philippe Mathieu et Sébastien Picault [MRU01, MP05, DMR05, MP06, MPR07] pour définir des modèles, des algorithmes et une méthodologie généraux pour la conception de simulations.

Le cœur de l'approche définie dans cette thèse, que nous décrivons dans la partie II, est caractérisée par trois éléments différents, résumés dans la figure 2.

1. La **base théorique de IODA** composée :
   - d'**un modèle formel** décrivant les divers concepts de l'approche IODA à l'aide du formalisme mathématique ;
   - d'**une méthodologie de conception** fournissant des outils graphiques permettant de construire progressivement un modèle pouvant contenir un grand nombre d'agents et d'interactions différents ;
   - d'**algorithmes** exploitant les informations du modèle afin de fournir une implémentation univoque de la simulation.
2. La **plateforme de simulation** appelée JEDI (Java Environment for the Design of agent Interactions) qui constitue **une implémentation fidèle des concepts** de IODA dans le langage de programmation JAVA ;
3. L'**environnement de développement intégré** nommé JEDI-BUILDER qui constitue un prototype d'implémentation de la méthodologie de conception IODA. Ce dernier permet de construire un modèle IODA dans une interface graphique et d'en générer automatiquement le code pour la plateforme de simulation JEDI.

JEDI et JEDI-BUILDER confirment donc la faisabilité de l'approche IODA.

4.2 Exploration du potentiel applicatif

L'approche IODA permet de concevoir des simulations en centrant la conception sur les interactions. Ce changement de point de vue de modélisation ouvre un grand nombre de perspectives que nous explorons selon trois thématiques différentes. Nous utilisons ces études afin de définir quatre extensions disjointes de IODA.

Premièrement, nous questionnons la nécessité d'utiliser des interactions complexes afin de modéliser des comportements complexes. Cette étude a abouti à l'identification de **patrons de conception** permettant de décomposer la plupart des interactions complexes en un ensemble d'interactions simples, ainsi qu'une extension de IODA à un certain type d'interaction complexe ne pouvant être décomposée simplement.

Deuxièmement, nous explorons comment utiliser notre représentation centrée-interaction afin d'**éviter d'introduire des biais** dans une simulation. Pour cela, nous cherchons à faire apparaître explicitement des choix de conception usuellement implicites dans la plupart des autres approches de conception. L'apparition explicite de ces choix dans le modèle permet de décrire de manière formelle et univoque certaines décisions d'implémentation. Deux extensions du modèle formel, de la méthodologie et des algorithmes de simulations de IODA, découlent de cette étude. La première porte sur le problème de la participation simultanée d'un agent à plusieurs interaction. La seconde porte sur le problème de la concrétisation de comportements stochastiques chez des agents réactifs, dans une architecture comportementale générique.

Enfin, nous étudions comment **transposer les concept de l'héritage** des langages orientés-objets à notre approche, afin de favoriser la factorisation d'éléments sémantiques corrélés du modèle et donc mieux supporter la conception de simulations large échelle. A cette occasion, nous nous reposons sur

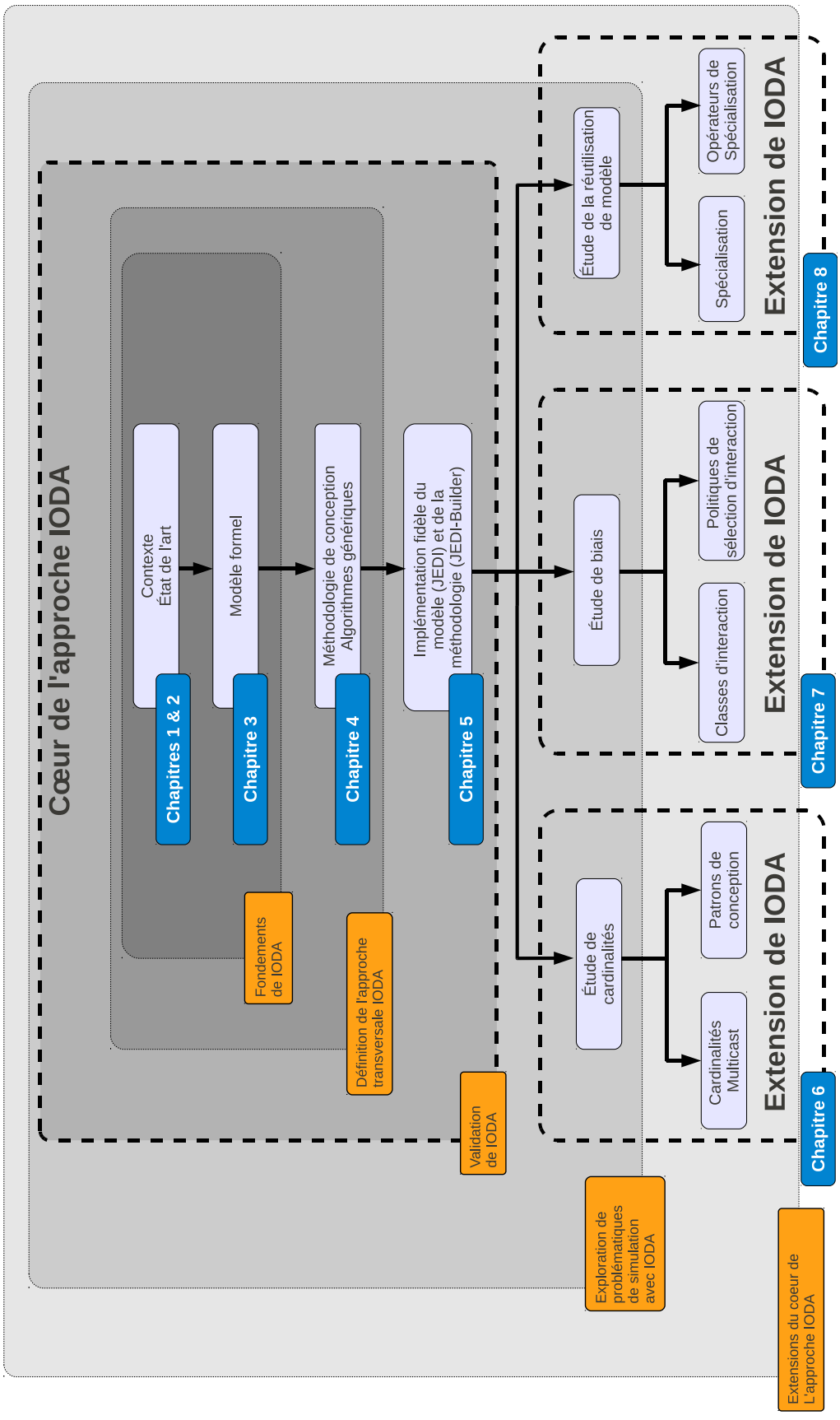

FIGURE 1 – Organisation du manuscrit de thèse et de ses contributions.

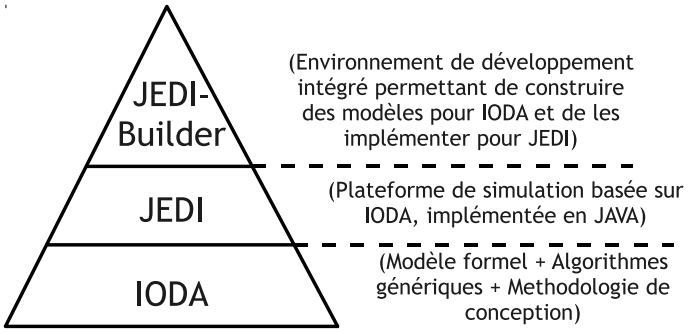

FIGURE 2 – La première contribution de cette thèse : élaboration d'un modèle formel centré sur les interactions, accompagné d'une méthodologie facilitant leur conception, d'algorithmes assurant leur implémentation univoque (IODA) et élaboration d'une implémentation fidèle du modèle (JEDI) complétée par un IDE de conception (JEDI-BUILDER).

notre approche centrée-interaction afin de mieux prendre en compte certains problèmes d'ingénierie des connaissances. Nous définissons pour cela une relation de spécialisation entre agents, qui permet d'exprimer :

- qu'un agent A est un agent B particulier sachant en plus effectuer d'autres interactions. Cette relation est similaire à la notion d'héritage trouvée dans les langages orientés-objet ;
- qu'un agent A est un agent B particulier ne sachant pas effectuer certaines interactions de B.

Cette approche est particulièrement intéressante pour concevoir des agents dont le comportement constitue une exception à un ensemble d'agents, par exemple pour spécifier qu'un individu aveugle est un individu particulier ne sachant pas lire.

5 Organisation du manuscrit

Ce manuscrit est organisé en trois parties, contenant au total huit chapitres.

Première partie Nous consacrons la première partie de ce manuscrit à l'introduction du contexte dans lequel se placent nos travaux, à la présentation de travaux connexes, et à l'identification des problématiques que nous traitons.

Le chapitre 1 justifie l'utilisation du paradigme agent et des systèmes multi-agents dans notre approche afin de faciliter la conception de simulations large échelle. Pour cela, nous identifions dans un premier temps les différents types de simulations informatiques existants, ainsi que le type de simulations informatiques que nous considérons, nommé couramment « simulations explicatives ». Dans un second temps, les problématiques inhérentes à ce type de simulations sont présentées, et nous décrivons en quoi l'usage de systèmes multi-agents contribue en partie à leur résolution.

Dans le chapitre 2, nous étudions plus précisément les différentes approches existantes pour construire une simulation multi-agents et, plus particulièrement, pour aboutir à un simulateur, *i.e.* un programme informatique permettant de réaliser des expériences. Nous dégageons de cette étude l'importance d'utiliser une approche transversale de conception, *i.e.* une approche qui :

- accompagne les concepteurs des premières étapes de la simulation à un simulateur concret ;
- permet de construire graduellement un modèle.

Nous caractérisons de plus les propriétés nécessaires à la définition de telles approches. Dans un second temps, nous étudions quels sont les apports des approches existantes vis-à-vis de des propriétés identifiées et quelles sont leurs lacunes. Nous en dégageons alors l'importance d'utiliser une approche de conception des simulations centrée sur les interactions entre les agents d'une simulation.

Deuxième partie En réponse aux divers problèmes de conception identifiés dans la partie précédente de cette thèse, nous proposons une approche transversale de conception de simulations, que nous centrons sur les interactions entre agents plutôt que sur leur comportement. Nous appelons cette approche Interaction Oriented Design of Agent simulations (IODA).

Un processus transversal est caractérisé par quatre éléments : la structure des modèles spécifiés, la structure de l'implémentation des modèles, la méthodologie permettant de spécifier un modèle et le processus permettant de passer de modèle à implémentation. Cette seconde partie de ce manuscrit, nous caractérisons ces quatre aspects de l'approche IODA, en trois chapitres (voir le résumé de la figure 3).

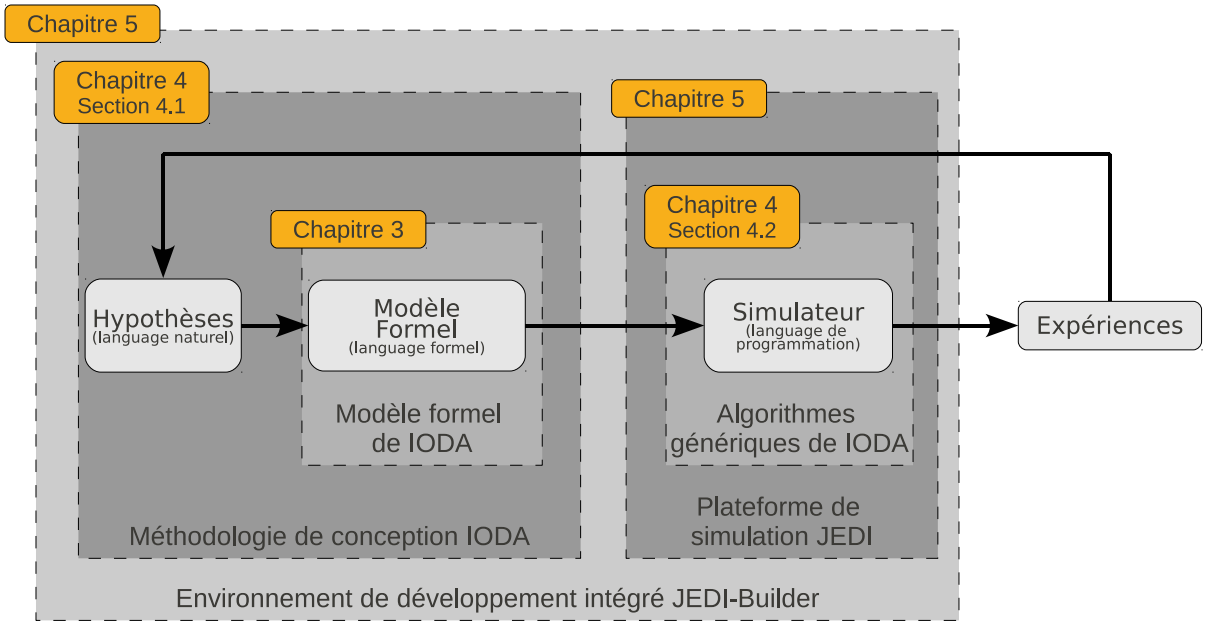

FIGURE 3 – Résumé de l'objet des chapitres 3, 4 et 5 (deuxième partie de ce manuscrit).

Le chapitre 3 détermine comment réifier la notion d'interaction dans une simulation, afin de bénéficier des avantages énoncés dans la partie précédente. Nous décrivons pour cela la structure du modèle de l'approche IODA, à l'aide du formalisme mathématique, ainsi que des méta-modèles et d'exemples.

Le chapitre 4 montre comment construire une approche transversale à l'aide du modèle IODA. Nous y définissons pour cela des outils permettant à la fois de construire graduellement un modèle (une méthodologie de conception), mais aussi de l'implémenter (des algorithmes de simulation). Nous analysons à cette occasion quatre problèmes de conception récurrents selon la perspective de IODA, et en dégageons des avantages tels que l'unification de la représentation des entités d'une simulation.

Le chapitre 5 confirme la faisabilité de l'approche IODA, en décrivant la plateforme de simulation JEDI (*Java Environment for the Design of agent Interactions*), qui implémente fidèlement les concepts de modèles IODA et en décrivant l'environnement de développement intégré JEDI-BUILDER, qui fournit une implémentation fidèle à la méthodologie IODA.

Troisième partie Le changement de perspective induit par notre approche centrée sur les interactions permet d'étudier les problèmes de simulation sous un angle nouveau. Dans cette partie, nous nous sommes appuyons sur IODA, JEDI et JEDI-BUILDER pour explorer des problématiques inhérentes à la simulation informatique, mais n'apparaissant pas explicitement dans les approches existantes, ou y étant traitées de manière non satisfaisante. Ces études ont abouti à l'identification de principes permettant d'étendre IODA et d'y intégrer des outils méthodologiques facilitant la conception de certaines catégories de simulations.

Nous avons mené l'exploration en considérant trois problématiques de simulation différentes :

1. Faut-il nécessairement des interactions complexes afin de modéliser des comportements complexes ?
2. Comment utiliser IODA afin d'éviter d'introduire des biais dans une simulation ?
3. Comment transposer le concept d'héritage des langages orientés-objets à IODA afin de faciliter l'ingénierie des connaissances ?

Puisque les actions impliquant simultanément plusieurs agents ne sont pas modélisées sous la forme d'interactions dans les approches actuelles, ces dernières éludent un problème pourtant réel, apparaissant explicitement dans notre approche centrée interaction : « Des interactions simples suffisent-elles à exprimer des comportements complexes ? » Dans le chapitre 6, nous analysons ce problème à l'aide de deux cas d'étude, ayant abouti à l'identification :
- de deux patrons de conception permettant de modéliser la plupart des interactions impliquant plus de deux agents à l'aide d'interactions en impliquant au plus deux ;
- d'un nouveau type d'interactions appelées « interactions multicast » permettant de modéliser simplement certaines interactions complexes ne pouvant être décomposées.

Afin d'éviter d'introduire des biais dans une simulation, les choix de conception doivent être faits sciemment par les concepteurs. Dans le chapitre 7, nous montrons que pour faire apparaître de tels choix, trois unités fonctionnelles sous-jacentes à toute simulation doivent être finement spécifiées. En menant des études selon cette séparation et en focalisant la conception sur les interactions, nous avons dégagé deux extensions de IODA permettant de rendre explicites les choix de conception relatifs à :
- la participation simultanée à plusieurs interactions ;
- la spécification de comportements stochastiques.

La construction de simulations large échelle ne peut être envisagée sans profiter d'outils d'ingénierie logicielle similaires à l'héritage des langages orientés-objet. En effet, ces outils permettraient d'éviter de fournir une spécification systématiquement exhaustive de la source et la cible de chaque interaction. Dans le chapitre 8, nous étudions sous quelles conditions il est possible d'intégrer des concepts proches de l'héritage dans l'approche IODA afin de faciliter l'ingénierie des connaissances. A cette occasion, nous définissons une extension de IODA fondée sur la relation de spécialisation qui facilite l'ingénierie des connaissances dans une simulation.

Ces trois chapitres identifient en tout quatre extensions possibles de l'approche IODA, qui fournissent des concepts et des outils méthodologiques permettant de traiter plus efficacement certains problèmes de conception spécifiques. L'approche IODA peut ainsi être utilisée dans sa forme « minimale » (*i.e.* sans utiliser les extensions) afin de spécifier des simulations, ou dans une de ses formes étendues, en utilisant une ou plusieurs extensions que nous décrivons. Nous définissons ainsi une approche de conception paramétrable, dont la complexité peut être adaptée à la complexité des problèmes traités.

Simulation multi-agents

Plan du chapitre :

Dans cette thèse, nous cherchons à faciliter la conception de simulations informatiques explicatives, et plus précisément des simulations large échelle, faisant intervenir un grand nombre d'entités entretenant des interactions variées. Pour cela, nous définissons une approche multi-agents centrée sur les interactions.

Ce chapitre justifie notre proposition d'une approche multi-agents de la conception de simulations dites explicatives. Pour cela, nous décrivons l'objet de nos travaux, à savoir la simulation informatique (section 1.1), nous en identifions les principes, ainsi que les principales problématiques (section 1.2). Nous décrivons ensuite les concepts et les propriétés principales du paradigme agent et des systèmes multi-agents (section 1.3) et montrons en quoi ils contribuent à la résolution des problématiques identifiées (section 1.4).

1.1 Contexte général : Simulation informatique de systèmes complexes

L'utilisation de simulations multi-agents (MABS [1]) est un phénomène de plus en plus répandu. Son essor est lié à sa représentation individu-centrée du phénomène, qui contribue grandement à la résolution de problématiques inhérentes à la simulation informatique. Cette première section identifie ces problématiques, en s'appuyant sur les définitions existantes de la simulation. Nous considérons en particulier la définition de Shannon [Sha98] qui suit :

> « [Simulation is] the process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behavior of the system and /or evaluating various strategies for the operation of the system. » [Sha98]

Cette définition fait apparaître quatre grand thèmes : *les objectifs* (*i.e.* « the purpose »), *le modèle*, *les expériences* et *le processus* (voir figure 1.1). Nous traitons ces quatre thèmes séparément pour caractériser le type de simulation informatique étudié dans cette thèse.

Simulation = Objectifs + Modèle + Processus + Expériences

FIGURE 1.1 – Schématisation des principales notions trouvées dans la définition de la simulation proposée par Shannon [Sha98].

1. pour : *MultiAgent Based Simulation*. Nous n'utilisons pas l'acronyme français SMA pour qu'il ne soit pas confondu avec *Système Multi-Agents*.

1.1.1 Terminologie utilisée

La simulation est un outil au service d'objectifs variés et au service de domaines d'application très différents (voir section 1.1.2). Cette variété fait qu'aucun consensus de terminologie n'existe sur le sujet [Öre87, Fis94]. Par conséquent, ce document de thèse s'attache aux *concepts* plus qu'aux termes en eux-mêmes. Nous invitons le lecteur à se remémorer ceci, à chaque fois que l'utilisation de certains termes dans certains contextes lui semble choquante.

1.1.2 Simulation informatique : Quels objectifs ?

L'unique point commun de toute simulation est la volonté de reproduire artificiellement un phénomène (qu'il soit réel ou non) à l'aide d'un programme informatique. Par la suite, nous nous référons au phénomène imité par simulation [2] par le terme « phénomène ».

Selon [Axe97, Edm05], les usages de la simulation informatique diffèrent par les objectifs qu'ils cherchent à atteindre. Ces usages incluent [3] :

1. « la prédiction » (« prediction ») : la simulation prend en paramètres des données issues d'un phénomène et produit en sortie une estimation de l'état du phénomène dans le futur. Les applications de ce type de simulation incluent la prédiction de l'évolution des populations d'un écosystème [Vol28], la prédiction de l'évolution des conditions atmosphériques, la prédiction des performances d'un processeur, *etc.*
2. « la substitution » (« performance ») : la simulation imite le comportement d'un humain pour reproduire une tâche. Ces tâches incluent le diagnostic médical automatisé, la reconnaissance vocale, *etc.* De telles simulations sont aussi utilisées pour éprouver (*i.e.* tester, vérifier ou valider) un système informatique, en reproduisant artificiellement le comportement d'un utilisateur humain ;
3. « l'entraînement » (« training ») : la simulation est utilisée pour immerger un utilisateur dans un environnement artificiel, dynamique et interactif, à des fins d'entraînement. Les applications de ce type incluent l'apprentissage du pilotage d'un avion, l'apprentissage de la conduite, l'apprentissage de la chirurgie, *etc.*
4. « le divertissement » (« entertainment ») : la simulation est utilisée pour immerger un ou plusieurs utilisateurs dans un environnement virtuel à des fins ludiques. Les applications de telles simulations incluent les effets spéciaux dans les films ou les jeux vidéos, l'intelligence artificielle de personnages dans un jeu vidéo, *etc.*
5. « l'éducation » (« éducation ») : la simulation est utilisée comme moyen pédagogique pour mettre en pratique des notions théoriques. Par exemple Clim'way [Sci10] propose une simulation de type SimCity pour sensibiliser à l'impact écologique de l'urbanisme, *etc.*
6. « la preuve » (« proof ») : la simulation fournit la preuve de l'existence d'un cas se conformant à certaines hypothèses. Par exemple, le jeu de la vie [PW84] prouve que des comportements excessivement complexes peuvent être le résultat de règles très simples ;
7. « la découverte » (« discovery ») : la simulation a un objectif similaire à *la preuve* mais, contrairement à cette dernière, les résultats obtenus ne constituent pas une preuve. Ils permettent seulement de conforter ou d'affaiblir les hypothèses émises. Par exemple, le modèle de ségreggation de Schelling [Sch71] conforte l'hypothèse selon laquelle des communautés de personnes peuvent se former dans une ville, même si toutes ces personnes déménagent en se basant sur un critère pouvant sembler tolérant. En effet, avec un critère « déménager uniquement si moins de 30% des voisins sont de mon ethnie » [4], la simulation se termine dans une situation où chaque personne a en moyenne plus de 70% de voisins de la même ethnie qu'elle.

En tant que méthode scientifique, la simulation est principalement utilisée pour la *prédiction*, la *preuve* et la *découverte*.

2. aussi appelé « source system » [ZKP00], « system » [Axe97], « real system » [Sha98], « real or proposed system » [Rob06], « actual or theoritical physical system » [Fis94], « target system or phenomena » [Edm05], ...

3. les exemples proposés ne figurent pas tous dans [Axe97, Edm05]

4. Le critère de l'ethnie est un exemple. La simulation décrite par Schelling reste valide pour tout autre type de dichotomie : pauvre/riche, français/anglais, *etc.*

Dans cette thèse, nous nous intéressons principalement aux simulations utilisées pour la *découverte*. Nous les appelons par la suite *simulations explicatives* (d'après l'appellation « explanatory » d'Edmonds [Edm05]).

Avant de passer à la description de l'élément suivant composant les simulations (le modèle), il nous semble important de caractériser plus précisément la différence entre entre simulations utilisées pour la preuve, pour la prédiction et pour la découverte. En effet, bien qu'elles soient utilisées comme méthodes scientifiques, ces simulations sont conçues et utilisées de manière radicalement différentes [Edm05].

La prédiction

La simulation prédictive est utilisée comme alternative à la résolution de modèles mathématiques, dans le cas où il est analytiquement difficile, voire impossible de les résoudre :

> « A computer simulation is any computer-implemented method for exploring the properties of mathematical models where analytical methods are unavailable » [Hum91] (d'après [AS07])

Dans de tels cas, la simulation permet de faire évoluer les différents paramètres de la simulation progressivement au fil du temps. Une estimation des résultats du modèle est ainsi obtenue sans pour autant le résoudre analytiquement.

De telles simulations ne peuvent être effectuées que si un modèle mathématique peut être formulé. Elles sont donc principalement utilisées dans les cas où les phénomènes étudiés sont pleinement (ou majoritairement) compris, par exemple en sciences physique [AS07].

La découverte

La découverte diffère de la simulation prédictive principalement par l'absence, ou la carence en théories précises permettant de décrire le phénomène. Ses domaines d'application incluent la biologie, ou les sciences sociales. Ces simulations sont utilisées pour reproduire le comportement du phénomène si jamais il fonctionnait selon les hypothèses émises. Si les résultats de simulation sont similaires aux propriétés observées du phénomène, alors les hypothèses émises offrent une explication candidate du mécanisme à l'origine du phénomène. En effet, la simulation ne permet pas de prouver que le phénomène fonctionne réellement tel qu'énoncé. Elle permet seulement de renforcer ou, si les résultats ne sont pas concluants, d'affaiblir les hypothèses émises.

Sauf mention contraire, dans la suite de ce document le terme générique « simulation » est utilisé dans son sens « simulation explicative ».

1.1.3 Simulation informatique : Le rôle du modèle

La diversité de ses applications fait qu'il n'existe aucune définition précise et communément acceptée de la simulation informatique, ni aucune description de la façon dont la simulation est conduite. Dans le cas de son utilisation comme méthode scientifique et en particulier dans son utilisation pour la découverte, certaines définitions permettent toutefois d'en avoir une idée générale.

Les définitions de Shannon [Sha98] citée en section 1.1, de Zeigler [ZKP00] (voir figure1.2) et de Fishwick [Fis94] caractérisent la simulation comme suit : une simulation a pour but d'imiter *un phénomène* dans un certain *cadre expérimental* (*i.e.* un objectif que la simulation cherche à atteindre), en se basant sur une représentation formelle de la façon dont le phénomène est supposé fonctionner, appelée *modèle du phénomène*, qui est traduit par la suite en *simulateur* permettant d'effectuer des expériences *in sillico*.

Le modèle est central à toute simulation : il sert de pivot entre les experts du domaine (*i.e.* les biologistes, les économistes, les sociologues, *etc.*) qui disposent des connaissances permettant d'abstraire le phénomène et les experts en informatique qui disposent des connaissances permettant de construire le simulateur. Il décrit de plus toutes les hypothèses sous lesquelles sont conduites les expériences. Il est donc indispensable pour communiquer et interpréter les résultats expérimentaux obtenus.

Un modèle constitue une représentation abstraite *particulière* d'un phénomène : un phénomène peut être abstrait en modèles très différents. La construction d'un modèle nécessite donc de faire des choix, qui ont des conséquences sur les résultats expérimentaux obtenus. En effet, un modèle peut reproduire

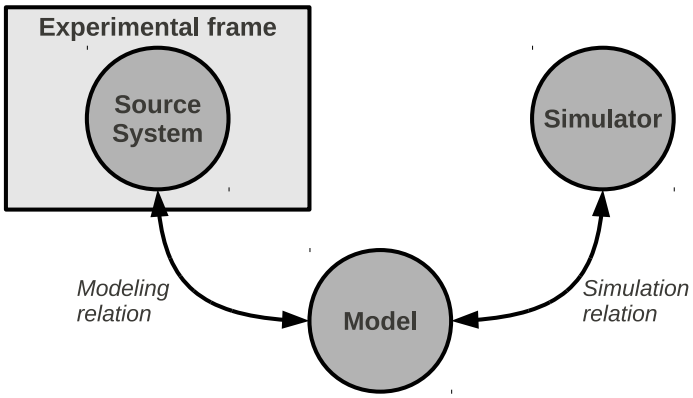

FIGURE 1.2 – Entités fondamentales constituant la simulation informatique et relations qu'elles entretiennent selon Zeigler [ZKP00]

correctement certains aspects d'un phénomène sans pour autant répondre aux hypothèses émises lors de la description du cadre expérimental. Par exemple, le modèle de Lotka-Volterra [J.L25, Vol28] permet de prédire l'évolution des populations de proies et prédateurs d'un écosystème, mais ne permet pas de connaître l'espérance de vie des prédateurs. Un modèle peut aussi répondre correctement aux hypothèses émises, en faisant toutefois défaut au principe de parcimonie.

Le principe de parcimonie

L'objectif d'une simulation n'est pas simplement de découvrir les causes du phénomène étudié, mais aussi de trouver le modèle qui l'explique le mieux. Ce principe, appelé principe de parcimonie, veut que, pour avoir des résultats concluants, il est nécessaire d'identifier les hypothèses minimales (et donc le modèle minimal) permettant de reproduire le phénomène.

La pratique du principe de parcimonie a pour conséquence une ou plusieurs itérations lors de la conception d'une simulation : un modèle est conçu, testé, puis simplifié jusqu'à obtenir un modèle minimal ou inversement, conçu le plus simple possible puis augmenté.

1.1.4 Le processus de simulation

Bien que la grande majorité des définitions de la simulation s'accordent sur le fait que leur conception fait partie d'un processus particulier, aucune définition détaillant ce processus ne fait consensus [And74, GT05, LM91, Sha98]. L'étude de différentes publications traitant du sujet permet toutefois d'en identifier les principales étapes. Nous étudions ici deux de ces définitions, qui décrivent le processus de simulation selon deux perspectives différentes : la réalisation d'expériences et la conception du programme de simulation.

Puisque le terme « simulation » peut être interprété comme « programme informatique permettant d'exécuter le modèle » (*i.e.* simulateur), nous mentionnons par la suite explicitement « processus de simulation » pour éviter toute ambiguïté

Un processus expérimental

Shannon [Sha98] décrit le processus de simulation informatique comme un processus expérimental composé des 12 étapes qui suivent (voir figure 1.3) :

1. « Problem definition » : Définir précisément l'objectif de la simulation, *i.e.* pourquoi étudie-t-on ce phénomène et à quelles questions souhaite-t-on répondre ;
2. « Project planning » : Déterminer si les moyens techniques et logistiques à disposition permettent d'entreprendre la résolution d'un tel problème ;
3. « System definition » : Déterminer les informations pertinentes à prendre en compte dans le modèle ;

4. « Conceptual model formulation » : Développer un modèle préliminaire de la simulation décrivant les composants [5], les variables ainsi que les interactions constituant le système simulé de manière graphique ou dans un pseudo-code ;
5. « Preliminary Experimental Design » : Déterminer sur quels critères la qualité de la simulation est évaluée, quels paramètres faire varier, comment les faire varier et combien d'expériences il est nécessaire d'exécuter ;
6. « Input Data preparation » : Identifier et sélectionner les valeurs pertinentes utilisées initialement pour chaque paramètre de la simulation ;
7. « Model Translation » : Implémenter le modèle dans un langage de simulation particulier ;
8. « Verification and Validation » : Confirmer que l'implémentation est conforme au modèle (vérification) et que les résultats obtenus par une simulation sont qualitativement et quantitativement conformes à ce qui est attendu ;
9. « Final experimental design » : Affiner les critères décrivant la qualité d'une simulation, ainsi que les jeux de paramètres utilisés et le nombre d'expériences à exécuter ;
10. « Experimentation » : Exécuter l'ensemble des expériences prévues et collecter les résultats obtenus ;
11. « Analysis and Interpretation » : Analyser les résultats obtenus et déterminer si la solution proposée répond au problème initialement posé ;
12. « Implementation and Documentation » : Documenter la simulation effectuée et publier les résultats obtenus.

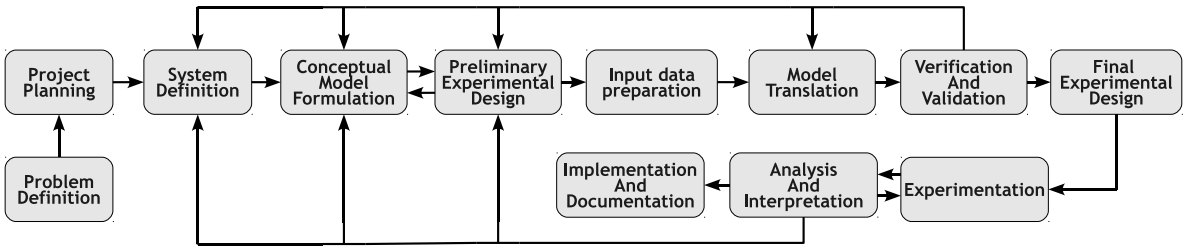

FIGURE 1.3 – Schéma du processus de conception de simulations défini par Shannon dans [Sha76] adapté à sa proposition dans [Sha98].

Cette définition est fortement inspirée des protocoles expérimentaux : seules les étapes de « Project planning », « Model Translation » et « Verification » (mais pas « Validation ») sont propres aux simulations informatiques.

> Puisque nous nous intéressons uniquement aux problématiques liées à la conception de simulations informatiques, ce travail de thèse étudie un sous-ensemble du processus de simulation, s'arrêtant à l'étape de « Verification ». Nous appelons ce sous-ensemble le « processus de conception de simulations ».

Notre volonté n'est pas ici de minimiser les difficultés liées aux autres étapes, mais de restreindre le cadre de l'étude menée dans cette thèse. Ces autres étapes posent également des problèmes non triviaux, tels que la reproduction d'une simulation (aussi appelé problème de divergence implémentatoire par Michel [Mic04]), le problème de la diffusion des résultats de simulation, *etc.*

Un processus de conception

D'autres définitions se focalisent sur une vision orientée « experts en informatique » du processus de simulation [ZKP00, MEU04, Edm05, GII+09] . Ces définitions, en particulier celle de Galan [GII+09]

5. « component » dans le texte. L'auteur utilise ce terme en son sens général. Il ne fait pas référence au paradigme de conception par composants.

(voir figure 1.4), sont particulièrement intéressantes, car le changement de perspective permet de mettre en valeur l'un des problèmes fondamentaux de la simulation explicative. Dans la définition de Galan, le processus de simulation ne repose plus sur un seul, mais sur plusieurs modèles :

1. « Modèle non-formel » : formulé à l'aide du langage naturel, il est écrit par des experts du domaine afin de définir l'objectif de la simulation (décrits dans la section 1.1.2), le but de la simulation (par exemple « reproduire les variations des populations d'un écosystème »), les principaux éléments du phénomène, leurs connexions, ainsi que les principales relations causales ;
2. « Modèle formel » (aussi appelé « modèle conceptuel » [Rob06, Sar98]) : Ce modèle correspond à la réécriture du « modèle non-formel » en utilisant un formalisme particulier. Lors de sa conception, les choix effectués par les experts du domaine dans le « modèle non-formel » sont complétés pour qu'ils puissent être exprimés dans le formalisme choisi ;
3. « Modèle exécutable » (aussi appelé « modèle computationel » [EH03] ou « modèle de la simulation » [Sar98]) : Ce modèle complète le « modèle formel », en lui ajoutant tous les éléments qui permettront son implémentation ;
4. « Programme informatique » : Comme son nom l'indique ce dernier modèle est une implémentation du « modèle exécutable » s'appuyant sur un langage de programmation ou une plateforme de simulation particulière.

Chaque modèle est décrit en fonction des informations extraites du modèle précédent et se rapproche progressivement du programme informatique. Ils permettent donc de passer graduellement d'une représentation abstraite du phénomène réel à son implémentation. Selon Galan, ces modèles peuvent être en pratique disjoints, ou décrire différentes parties d'un même modèle général. De plus, il est intéressant de noter qu'il décrit aussi différents rôles joués par le concepteur durant la construction de la simulation. Chaque rôle ne peut être joué qu'avec des compétences adéquates, variant grandement d'un rôle à un autre.

Dans la section qui suit, nous nous appuyons sur les définitions de Shannon et Galan afin d'identifier les problématiques liées à la conception de simulations.

1.2 Principales problématiques liées au processus de conception de simulations

Les deux définitions présentées dans la section précédente permettent de caractériser les problématiques fondamentales (voir tableau 1.1(a)) rencontrées en simulation informatique en deux thèmes principaux : *l'obtention de résultats erronés* et *la révision du modèle*.

Afin de clarifier nos propos, nous nommons $\mathcal{P}prior_i$ les problèmes de modélisation et d'implémentation menant à des résultats erronés, $\mathcal{P}post_i$ les problèmes liés à la détection de résultats erronés et $\mathcal{P}rev_i$ les problèmes liés aux révisions du modèle.

1.2.1 Obtention de résultats erronés

En sciences expérimentales, il est commun d'obtenir des résultats différents de ceux attendus. Ces résultats signifient que le phénomène réel fonctionne différemment des hypothèses émises par l'expert du domaine.

En simulation informatique, cela n'est pas nécessairement le cas. L'implémentation d'une simulation peut différer de ce qui était imaginé par l'expert du domaine. En effet, comme l'illustre la figure 1.4, l'implémentation d'une simulation informatique est le fruit d'un ensemble non négligeable d'étapes, pouvant faire intervenir des personnes aux compétences très différentes. Chaque étape est sujette à différents types d'erreurs [GII+09], liées à la traduction d'un modèle source en un modèle cible (par exemple transformer le modèle formel en modèle exécutable) :

- le modèle cible peut refléter quelque chose de différent de ce que pense son créateur (le formalisme du modèle cible est mal utilisé) ; ($\mathcal{P}prior_1$)
- une ambiguïté des informations fournies par le modèle source a abouti à un mauvais choix dans le modèle cible ; ($\mathcal{P}prior_2$)

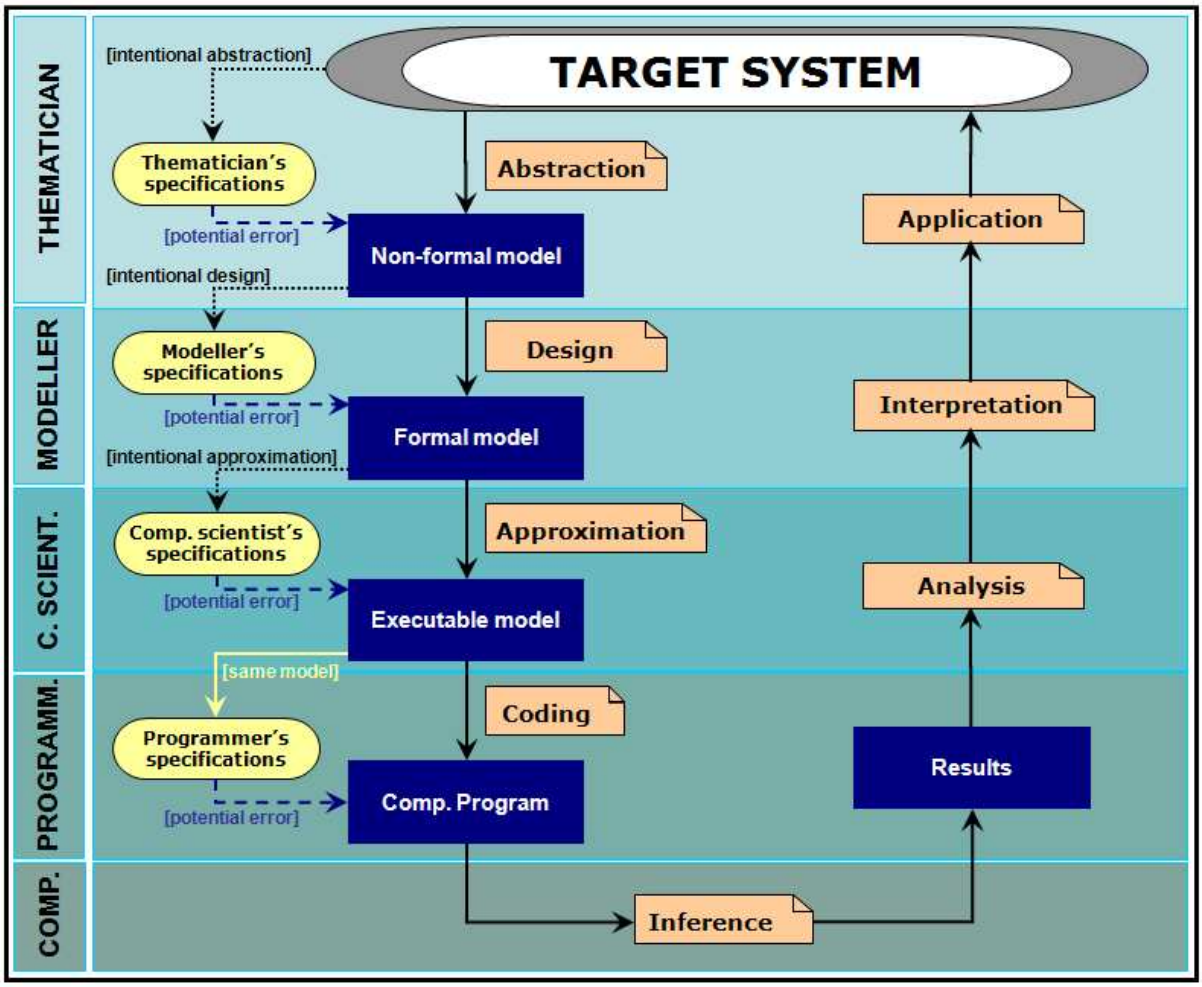

FIGURE 1.4 – Les différentes étapes du processus de conception de simulations, selon Galan [GII+09].

– l'absence d'une information dans le modèle source a abouti à un choix arbitraire dans le modèle cible ; ($\mathcal{P}prior_3$)
– indépendamment des informations fournies par le modèle source, un choix involontaire et implicite a été fait dans le modèle cible. ($\mathcal{P}prior_4$)

Si à un moment ou un autre, une personne concevant la simulation se retrouve dans un de ces cas de figure, le modèle (qu'il soit non-formel, formel, exécutable, ou le simulateur) et par extension la simulation deviennent biaisés. Les conséquences d'un modèle biaisé sont diverses et dépendent de la nature du biais (voir le chapitre 7 pour une description plus détaillée de ce problème). Dans le meilleur des cas, les biais ne concernent que des aspect mineurs de la simulation et n'en altèrent pas les résultats ($\mathcal{P}post_1$). Les biais peuvent aussi aboutir à des résultats de simulation erronés ($\mathcal{P}post_2$). Ces résultats inattendus vont occasionner une nouvelle itération du processus de conception de simulations, qui va donner l'opportunité de détecter l'erreur et de corriger le modèle. Dans le pire des cas, une situation critique survient : le modèle ne décrit pas correctement le phénomène réel et devrait donc produire des résultats erronés, mais son implémentation biaisée compense ce phénomène et donne l'illusion d'avoir des résultats concluants ($\mathcal{P}post_3$).

Il est donc primordial de s'assurer que les choix effectués lors de la constitution des modèles et de l'implémentation sont conformes à l'abstraction du phénomène que l'expert du domaine souhaite obtenir.

1.2.2 Révisions du modèle

Comme le montre sa représentation schématique sur la figure 1.3, le processus de simulation n'est pas strictement séquentiel. Des itérations surviennent lors de trois étapes : l'étape intitulée « Preliminary Experimental Design », l'étape intitulée « Verification and Validation » et l'étape intitulée « Analysis and Interpretation ».

L'étape intitulée « Preliminary Experimental Design » a pour but de raffiner le modèle afin de prendre en compte les différents paramètres utilisés dans les expériences. Puisque l'étape d'implémentation (intitulée « Model Translation ») n'est pas encore atteinte, la prise en compte de cette itération se fait uniquement au niveau modèle. Ce dernier doit être suffisamment souple pour prendre en compte les changements sans avoir à modifier profondément sa structure. Dans le cas contraire, un changement risque de causer l'un des problèmes $\mathcal{P}prior_1$, $\mathcal{P}prior_2$, $\mathcal{P}prior_3$ ou $\mathcal{P}prior_4$. Ce problème est particulièrement vrai dans des modèles équationnels, où certains changements peuvent nécessiter une mise en équation complète du système. ($\mathcal{P}rev_1$)

L'étape intitulée « Verification and Validation » a pour but d'identifier les situations $\mathcal{P}post_1$, $\mathcal{P}post_2$ et $\mathcal{P}post_3$. Si une de ces situation est détectée, une révision du modèle s'impose, *i.e.* une modification du modèle permettant de corriger les erreurs identifiées. La révision n'est possible que s'il y a identification préalable de ce qui a causé l'erreur détectée et donc détermination de quels éléments du modèle (niveau microscopique) sont à l'origine des résultats erronés de simulation (niveau macroscopique). ($\mathcal{P}rev_2$)

L'étape intitulée « Analysis and Interpretation » permet de déterminer si la simulation est considérée comme utile (ayant atteint ses buts et ses objectifs) ou pas. Dans le cas où elle ne le serait pas, il faudrait alors explorer une autre piste, et donc produire un modèle différent du phénomène. Cette étape est propre aux protocoles expérimentaux et n'introduit de problématique spécifique à la simulation, mis à part celle découlant de la révision de modèle, que nous décrivons ci-après.

Toute révision du modèle peut s'avérer coûteuse en termes d'implémentation. Bien que les modèles diffèrent d'une itération à l'autre, ils ont, dans une certaine mesure, une base commune puisqu'ils représentent le même phénomène. Il serait donc possible de réutiliser certains éléments de l'implémentation, afin de réduire les coût induits par le développement de la simulation. ($\mathcal{P}rev_3$)

1.2.3 Comment les résoudre ?

Différentes approches sont utilisées pour résoudre les problèmes mentionnés dans la section précédente. Nous n'en fournissons ici qu'un aperçu, qui nous guidera par la suite dans l'état de l'art.

L'obtention de résultats inattendus est gérée par deux types d'approches : les approches *a priori* et les approches *a posteriori*. Le principe de base des approches *a priori* est d'éviter, à l'aide de divers moyens, les situations pouvant aboutir aux problèmes $\mathcal{P}prior_1$, $\mathcal{P}prior_2$, $\mathcal{P}prior_3$ et $\mathcal{P}prior_4$. Ces solutions se

focalisent donc sur la structure du(des) modèle(s), la façon dont est représentée la connaissance, ainsi que la façon dont les modèles sont construits. Elles incluent :

- favoriser l'intervention des experts du domaine dans le processus de conception de simulations, pour éviter ces problèmes lors de la transition entre abstraction du phénomène et modèle formel ;
- favoriser l'identification des problèmes et choix relatifs à l'implémentation, lors de la transition entre modèle formel et modèle exécutable ;
- fournir des représentations visuelles favorisant la communication entre les différents acteurs du processus de conception de simulations ;
- favoriser la relation de morphisme [ZKP00] entre les différents modèles, *i.e.* une relation d'équivalence qui met en correspondance les éléments de deux spécifications différentes ;
- aider le choix d'un formalisme de modélisation et d'une plateforme d'implémentation à l'aide de critères de comparaison.

A l'opposé, les approches *a posteriori* ont pour principe de fournir des outils permettant d'identifier les situations où la simulation n'est pas valide ($\mathcal{P}post_1$, $\mathcal{P}post_2$ et $\mathcal{P}post_3$), qui sont le symptôme des problèmes $\mathcal{P}prior_1$, $\mathcal{P}prior_2$, $\mathcal{P}prior_3$ et $\mathcal{P}prior_4$. Ces solutions se focalisent sur l'analyse du modèle, de l'implémentation, et des résultats expérimentaux. Elles incluent :

- favoriser le test unitaire de l'implémentation ;
- comparer les résultats expérimentaux à des résultats extraits du phénomène, obtenus d'une simulation prédictive du même phénomène, ou par une résolution analytique d'un modèle mathématique équivalent ;
- comparer les résultats expérimentaux obtenus de modèles et de simulateurs différents ;
- analyser la sémantique du modèle.

Le problème de la révision de modèle est lui géré par des approches favorisant le génie logiciel lors de la conception de la simulation. Ces solutions se focalisent sur la structure et l'architecture de l'implémentation. Elles cherchent en particulier à :

- rendre l'architecture de la simulation modulaire ;
- construire des librairies d'éléments réutilisables ;
- favoriser le morphisme entre modèle et implémentation, afin d'identifier plus facilement la part du modèle à l'origine d'une erreur identifiée dans l'implémentation.

Dans ce document, nous nous intéressons en particulier aux solutions *a priori*, ainsi qu'aux solutions liées aux révisions de modèle. Pour traiter ces problématiques, nous nous focalisons sur la représentation des connaissances dans le modèle, ainsi que sur le processus de conception.

Le paradigme « multi-agents » fournit une représentation des connaissances qui contribue grandement à l'ensemble des problèmes mentionnés dans cette section. La section qui suit décrit ces différents apports, après avoir présenté les principaux concepts des systèmes multi-agents.

1.3 La conception orientée-agent

La simulation multi-agents s'appuie sur le paradigme des systèmes multi-agents. Nous commençons donc naturellement par la présentation des différents concepts liés à ce paradigme.

1.3.1 Les agents

L'étendue des domaines d'application des systèmes multi-agents fait qu'aucune définition de la notion « agent » n'est communément acceptée. En effet, selon les domaines, certaines caractéristiques des agents sont contestées [Woo01].

Selon Wooldridge :

> « An *agent* is a computer system that is *situated* in some *environment*, and that is capable of *autonomous action* in this environment in order to meet its design objectives. » [Woo01]

Notre interprétation de la définition précédente, conduite en la confrontant avec les définitions trouvées dans [Fer99, WJ95], nous amène à considérer qu'un agent est un système informatique qui est :

– « capable d'agir dans son environnement » : un agent peut modifier son environnement au travers des actions qu'ils effectue. Environnement est à comprendre en son sens systémique, c'est à dire « tout ce qui est extérieur à l'agent lui-même » ;
– « situé dans un environnement » : un agent n'est pas omniscient. Il n'agit qu'en fonction de la partie de son environnement qu'il *perçoit* et ne peut agir que sur un sous-ensemble de son environnement ;
– « autonome » : le comportement d'un agent (*i.e.* les actions qu'il choisit d'entreprendre) n'est pas le fruit d'un programme tiers : l'agent choisit lui même comment il agit.

Cette interprétation se retrouve dans [Mic04], qui la schématise d'ailleurs sous une forme très proche de la systèmique (voir figure 1.5). Comme l'indique cette figure, un agent dispose d'une architecture interne

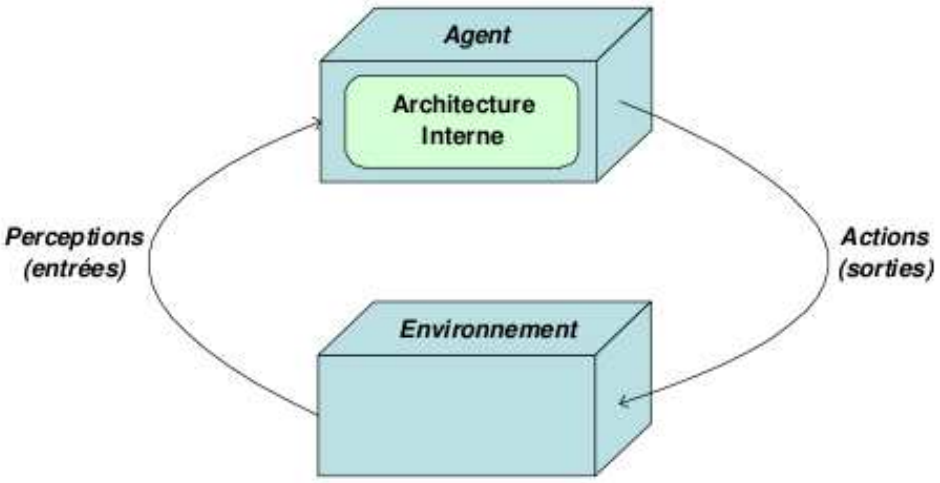

FIGURE 1.5 – Représentation classique d'un agent et de son environnement (d'après [Mic04]).

permettant de décider quelle action effectuer, en fonction des éléments qu'il perçoit. Nous revenons sur cette notion d'architecture interne des agents dans le chapitre 2.

D'autres propriétés peuvent compléter cette définition, mais sont contestées selon les domaines d'application. Parmi les plus communes figurent :

– « la possession d'un état interne » ([Woo01]) : un agent peut avoir un état interne, qui conditionnera ses actions conjointement à ce qu'il perçoit de l'environnement ;
– « la proactivité » ([WJ95]) : un agent n'agit pas uniquement en réaction à ce qu'il perçoit. Il agit aussi en fonction de buts qui lui sont propres ;
– « la communication » ([WJ95]) : un agent peut interagir avec d'autres agents à l'aide d'un langage de communication (un ACL, *i.e.* Agent Communication Language) ;
– « les croyances » ([RG91]) : un agent n'agit pas directement en fonction de ce qu'il perçoit dans l'environnement, mais en fonction de ce qu'il croit savoir de l'environnement ;

Ces différences sont le fruit de l'hétérogénéité des domaines d'application des systèmes multi-agents. En effet, leur utilisation pour concevoir des applications réparties nécessite une structure et une architecture différente de celle utilisée pour la résolution distribuée de problèmes, ou de celle utilisée pour la simulation. Nous détaillons ces problèmes dans le chapitre 2.

1.3.2 Les systèmes multi-agents

Les agents font partie d'un système multi-agents (que l'on notera par la suite MAS [6]), dont la définition est tout aussi contestée. À notre connaissance, toutes les définitions ne s'accordent que sur le fait qu'un système multi-agents ne se résume pas à un système ne contenant qu'un ensemble d'agents.

La définition proposée par Demazeau dans son approche Voyelles et sa plateforme Volcano [Dem95] est particulièrement intéressante. En effet, non seulement elle caractérise (sans faire consensus) ce qu'est

6. Cet acronyme est issu du nom anglais « MultiAgent System »

un système multi-agents, comment ils sont conçus, mais illustre aussi parfaitement le potentiel de ce paradigme du point de vue génie logiciel.

Selon Volcano, un système multi-agents est composé d'un ensemble d'*agents* potentiellement *organisés* qui *interagissent* dans un *environnement* commun. Cette approche repose sur trois caractérisations élémentaires d'un MAS, ainsi que sur une quatrième caractérisation optionnelle :

1. Agents : comment les agents décident-ils des actions qu'ils effectuent ;
2. Environnement : dans quel milieu évoluent les agents ;
3. Interaction : quels sont les moyens utilisés par les agents pour influer sur les actions effectuées par les autres agents ;
4. Organisation (optionnel) : comment structurer en ensembles les agents du système.

La conception du MAS se fait selon un processus reposant l'abstraction du système en un (ou plusieurs) modèle(s), qui sont ensuite implémentés sur une plateforme particulière. Chaque caractérisation est spécifiée dans une « brique » indépendamment des autres, autant dans le modèle qu'à l'implémentation (voir figure 1.6). Ainsi, le système multi-agents est conçu à l'aide de modèles multiples et indépendants permettant de réutiliser et modifier aisément divers éléments logiciels.

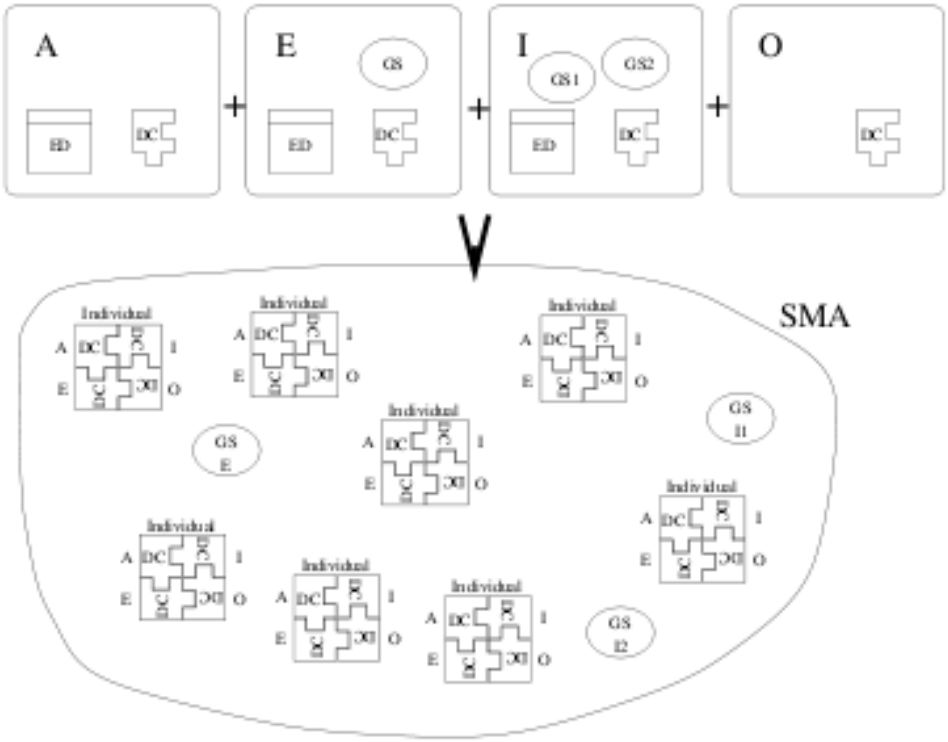

FIGURE 1.6 – La construction d'un MAS avec l'approche Volcano.

Cette définition est toutefois incomplète, car elle postule que toutes les entités contenues dans l'environnement sont des agents, *i.e.* des entités au comportement autonome. Pourtant, d'autres types d'entités peuvent exister dans l'environnement sans pour autant avoir un comportement autonome. Par exemple, une base de données dans une application répartie, un objet physique tel qu'une chaise dans un environnement virtuel, ou un caddie utilisé par des clients dans une simulation. En complément à la définition fournie par Demazeau, nous considérons donc un sous-ensemble de la définition de Ferber [Fer99] qui offre une caractérisation de ces entités. Selon lui, un système multi-agents est caractérisé entre autres par :

– Un environnement E ;
– Un ensemble d'objets [7] O. Ces objets sont situés, c'est-à-dire que tout objet a une position dans E.

7. Ce terme est utilisé en son sens général et indépendant de celui trouvé en programmation orientée-objet.

Ces objets peuvent être perçus, créés, détruits et modifiés par les agents ;
- Un ensemble A d'agents, qui sont des objets particuliers ($A \subseteq O$), lesquels représentent les entités actives du système ;
- Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux. Cette notion est utilisée pour définir les accointances d'un agent, c'est à dire les autres objets de l'environnement qu'il peut percevoir.

Ce complément permet d'identifier les propriétés des entités n'ayant pas de comportement autonome dans la simulation, et en quoi elles participent au comportement des agents.

1.4 La simulation multi-agents

Plusieurs raisons font que les systèmes multi-agents sont devenus très populaires en tant qu'outil de conception de simulations.

Les agents constituent une abstraction anthropomorphique similaire à la notion d'entité (par exemple une molécule, un animal, un humain) dans un phénomène. Ainsi, la description de la dynamique du phénomène ne se fait pas en déclarant des équations portant sur des propriétés macroscopiques de la simulation (par exemple des équations exprimant l'évolution de la population d'une espèce animale dans [Vol28]), mais sur le comportement individuel de chaque entité. Une telle métaphore est proche de la description naturelle d'un phénomène et est donc simple à utiliser par les experts du domaine. Ils effectuent donc des choix moins sujets aux problèmes $\mathcal{P}prior_1$, $\mathcal{P}prior_2$, $\mathcal{P}prior_3$ et $\mathcal{P}prior_4$ lors de la constitution du modèle formel à partir du modèle non-formel. Comme les agents sont aussi des entités logicielles, cette abstraction est conservée dans le modèle exécutable et dans l'implémentation, réduisant d'autant les chances d'obtenir des résultats erronés ($\mathcal{P}post_1$, $\mathcal{P}post_2$ et $\mathcal{P}post_3$).

De plus, la conservation d'une partie de la structure du modèle à l'implémentation favorise l'identification de l'origine de résultats erronés, en particulier lors de la vérification de l'implémentation. En effet, une fois l'erreur repérée dans l'implémentation d'un agent, son interprétation dans le modèle se fait aussi dans le modèle de l'agent ($\mathcal{P}prev_2$).

Les agents sont aussi un paradigme de programmation, proposant un ensemble d'outils logiciels favorisant la réutilisation [Jen99]. Les révisions du modèle sont ainsi rendues plus simples, car une grande partie de l'implémentation peut être réutilisée ($\mathcal{P}prev_3$). De plus, la division de la simulation en différentes parties facilite les tests unitaires, et facilite donc en partie l'étape de vérification ($\mathcal{P}post_1$, $\mathcal{P}post_2$ et $\mathcal{P}post_3$).

1.5 Synthèse

Dans cette thèse, nous cherchons à faciliter la conception de simulations informatiques, modélisant des phénomènes faisant intervenir un grand nombre d'entités entretenant des interactions variées (simulations large échelle). À cette fin, nous avons caractérisé dans ce chapitre l'objet de nos études (la simulation informatique), les problématiques lui étant inhérentes ; Nous avons de plus montré la contribution des notions d'*agent* et de *systèmes multi-agents* à leur résolution.

Parmi les différents sens pouvant être donnés à la « simulation informatique », **nous nous attachons dans cette thèse aux simulations dites explicatives**. Ces simulations visent à expliquer l'apparition d'un phénomène macroscopique observé (par exemple une termitière formant une arche), en décrivant uniquement le comportement individuel des entités présentes dans la simulation (par exemple le comportement de termites).

Pour conduire de telles simulations, une explication candidate au phénomène doit être formulée à l'aide d'une représentation abstraite du phénomène, appelée **modèle**, puis implémentée en un **simulateur**, afin d'être éprouvée par des **expérimentations**. Nous couvrons dans cette thèse un sous-ensemble de ce processus, appelé **processus de conception de simulations**, allant de la construction du modèle à son implémentation.

Le procédé permettant d'exprimer et valider l'explication fournie à un phénomène est sujet à divers problèmes, que nous résumons dans la table 1.1(a) page 33. Les principes permettant de résoudre ces

TABLE 1.1 – Description des problèmes rencontrés lors de la conception de simulations (a), et des différents principes facilitant leur résolution (b). Chaque principe décrit dans la table (b) concerne un ou plusieurs problèmes mentionnés dans la colonne « problème traité ». Ce tableau n'est pas exhaustif : il fournit uniquement une synthèse des solutions évoquées dans ce chapitre.

(a)

Origine	Nom	Description du problème
Construction d'un modèle biaisé	$\mathcal{P}prior_1$	Le modèle (ou l'implémentation) reflète quelque chose de différent de ce que pense son créateur (le formalisme du modèle est mal utilisé).
	$\mathcal{P}prior_2$	Une ambiguïté des informations fournies a abouti à une mauvaise interprétation et donc à un mauvais choix lors de la construction d'un modèle ou de son implémentation.
	$\mathcal{P}prior_3$	L'omission d'une information a abouti à un choix arbitraire lors de la construction d'un modèle ou de son implémentation.
	$\mathcal{P}prior_4$	Indépendamment des informations fournies, un choix involontaire et implicite a été fait lors de la construction d'un modèle ou de son implémentation.
Identification de simulations erronées	$\mathcal{P}post_1$	La simulation fournit les résultats attendus, malgré des erreurs/biais concernant des aspects mineurs du modèle ou de l'implémentation.
	$\mathcal{P}post_2$	La simulation ne fournit pas les résultats attendus, à cause de biais/d'erreurs contenus dans le modèle ou l'implémentation.
	$\mathcal{P}post_3$	La simulation fournit les résultats souhaités uniquement à cause des biais/erreurs concernant des aspects majeurs du modèle ou de l'implémentation.
Révisions du modèle	$\mathcal{P}rev_1$	La prise en compte de nouveaux paramètres change profondément la structure du modèle et de son implémentation.
	$\mathcal{P}rev_2$	Une erreur identifiée dans le simulateur doit pouvoir être interprétée au niveau du modèle de la simulation.
	$\mathcal{P}rev_3$	Les révisions de modèle induisent une ré-implémentation complète de la simulation.

(b)

Origine	Description de la solution	Problème traité
Construction d'un modèle biaisé	Utiliser dans le modèle des notions issues des phénomènes simulés.	$\mathcal{P}prior_1$
	Favoriser l'identification des problèmes et choix relatifs à l'implémentation lors de la construction du modèle.	$\mathcal{P}prior_2$, $\mathcal{P}prior_3$, $\mathcal{P}prior_4$
	Fournir des représentations visuelles favorisant la communication entre les différents acteurs du processus de conception de simulations.	$\mathcal{P}prior_1$, $\mathcal{P}prior_3$
	Aider le choix d'un formalisme de modélisation et d'une plateforme implémentation à l'aide de critères de comparaison.	$\mathcal{P}prior_1$
Identification de simulations erronées	Favoriser les tests unitaires d'implémentation.	$\mathcal{P}post_1$, $\mathcal{P}post_2$, $\mathcal{P}post_3$
	Comparer les résultats expérimentaux à des résultats extraits du phénomène, obtenus d'une simulation prédictive du même phénomène, ou d'une résolution analytique par modèle mathématique équivalent.	$\mathcal{P}post_1$, $\mathcal{P}post_2$, $\mathcal{P}post_3$
	Comparer les résultats expérimentaux obtenus de modèles et de simulateurs différents.	$\mathcal{P}post_1$, $\mathcal{P}post_2$, $\mathcal{P}post_3$
	Analyser la sémantique du modèle.	$\mathcal{P}post_1$, $\mathcal{P}post_2$, $\mathcal{P}post_3$
Révisions du modèle	Rendre l'architecture de la simulation modulaire.	$\mathcal{P}rev_1$, $\mathcal{P}rev_3$
	Construire des librairies d'éléments réutilisables	$\mathcal{P}rev_1$, $\mathcal{P}rev_3$
	Favoriser le morphisme entre modèle et implémentation, afin d'identifier plus facilement la part du modèle à l'origine d'une erreur identifiée dans l'implémentation.	$\mathcal{P}rev_2$

problèmes sont nombreux et sont plus ou moins bien exprimés dans chaque approche de conception de simulations. Nous en résumons les principaux dans la table 1.1(b). Afin d'éviter au maximum les problèmes de simulation, le modèle doit fournir un compromis exprimant au mieux chaque principe. Il doit pour cela **définir une représentation des connaissances appropriée**.

De ce point de vue, le paradigme *« agent »* et les ***systèmes multi-agents*** **sont particulièrement adéquats à la modélisation de simulations**. En effet, la représentation des connaissances y repose sur la définition du comportement d'entités autonomes (les agents) pouvant interagir. Elle a pour particularités :

- de reposer sur un concept familier aux experts du domaine. Elle favorise donc leur intervention dans le processus de conception de simulations ;
- d'implémenter les entités sous la forme d'agents. Elle favorise donc le morphisme entre modèle et implémentation ;
- de fournir, dans certains cas, des outils logiciels favorisant modularité et constitution de librairies logicielles.

Toutefois, les approches reposant actuellement sur ce paradigme **ne sont pas forcément toutes adaptées à la conception de simulations explicatives**. Dans le chapitre qui suit, nous étudions différents modèles, méthodologies et plateformes permettant de concevoir des systèmes multi-agents et en analysons les avantages et limites. Nous identifions par ce procédé les propriétés qui, à notre sens, doivent être exprimées par une approche de conception de simulations large échelle.