ATLAS, une plate-forme pour la modélisation et la simulation de systèmes désagrégés

Cyril Ray

N° d'ordre : 2872

Thèse

présentée

devant l'Université de Rennes 1

pour obtenir

le grade de : **Docteur de l'Université de Rennes 1**

Mention *INFORMATIQUE*

par

Cyril Ray

Équipe d'accueil : *Institut de Recherche de l'École Navale (IRENav)*

École doctorale : *Mathématique, Informatique, Signal, Electronique et Télécommunications (MATISSE)*

Composante universitaire : *IFSIC*

Atlas, une plate-forme pour la modélisation et la simulation de systèmes désagrégés

soutenue le 19 septembre 2003 devant la commission d'examen :

Composition du Jury

Président :	Kadi Bouatouch, Université de Rennes 1, IRISA
Rapporteurs :	Thérèse Libourel, CNAM, LIRMM Montpellier Agnès Voisard, Fraunhofer ISST, Freie Universität Berlin
Examinateurs :	Bruno Arnaldi, INSA de Rennes, IRISA Christophe Claramunt, École Navale Yvon Kermarrec, ENST Bretagne

Remerciements

J'exprime ma profonde et sincère gratitude à Christophe Claramunt (Institut de Recherche de l'École Navale, Lanvéoc-Poulmic) et à Bruno Arnaldi (Institut National des Sciences Appliquées de Rennes et IRISA, Rennes) pour avoir accepté les charges respectives de directeur et de codirecteur de cette thèse.

Je souhaite également remercier Thérèse Libourel (CNAM et LIRMM, Montpellier) et Agnès Voisard (Fraunhofer ISST et Freie Universität, Berlin) pour avoir accepté la charge de rapporteur et pour leurs précieux conseils.

Je remercie Kadi Bouatouch (Université de Rennes 1 et IRISA, Rennes) et Yvon Kermarrec (ENST Bretagne, Brest) d'avoir bien voulu participer au jury de thèse.

Je remercie Bruno Ramstein et Jean-Yves Billard, directeurs successifs de l'Institut de Recherche de l'École Navale pour leur soutien constant durant l'évolution de ces travaux de thèse. Merci également à l'ensemble du personnel enseignant/chercheur et à l'ensemble du personnel de soutien de l'Institut de Recherche de l'École Navale pour avoir contribué, chacun à leur manière, à la bonne marche de mes recherches. Je souhaite en particulier remercier les membres du groupe SIG de l'Institut de Recherche de l'École Navale, Rodéric Bera, Yann Chevriaux, Thomas Devogele, Michel Farine, Sébastien Fournier, Éric Guilbert, Thierry Huet, Joseph Poupin, Éric Saux, Myriem Sriti et Rémy Thibaud.

Un grand merci également à Marius Thériault (Université Laval, Québec) et Taha Osman (Nottingham Trent University, Nottingham) pour leurs collaborations et leurs conseils durant cette thèse.

Je souhaite enfin adresser mes sincères remerciements à ma toute ma famille pour le soutien apporté, en particulier un grand merci à mes parents, mes grands-parents, mes sœurs ainsi qu'à Brigitte, David et Gaël.

Lanvéoc-Poulmic, le 19 septembre 2003

À la mémoire des anciens

Le proverbe est une courte maxime entrée dans l'usage courant. Du point de vue formel, il se distingue souvent par le caractère archaïque de sa construction grammaticale : par l'absence d'article, par l'absence de l'antécédent, par la non-observation de l'ordre conventionnel des mots. La structure rythmique du proverbe est souvent binaire. On y trouve l'opposition de deux propositions ou de deux groupes de mots à l'intérieur de la proposition. La rime ou l'assonance vient parfois souligner l'opposition. Cette structure est souvent renforcée par l'utilisation d'oppositions sur le plan lexical : la répétition des mots, la mise en présence syntagmatique de couples oppositionnels de mots. Les traits spécifiques du proverbe en français sont l'emploi du cas-sujet et du cas-régime dans les expressions nominales, la présence de compléments déterminatifs, l'ellipse des relatifs, les consécutives négatives, les relatives au subjonctif, l'infinitif substantivé ou servant de thème dans une phrase à prédicat, la conjonction de coordination et introduisant une principale, les phrases nominales et les constructions chiasmatiques.

Exemple : Celui qui se dit maître de lui-même est l'élève d'un imbécile.

Résumé

Cette thèse introduit une méthodologie composée d'une démarche de modélisation et d'un support informatique pour la simulation distribuée et l'analyse de systèmes complexes à larges flux de données désagrégées. Le choix méthodologique proposé est influencé à la fois par le système à simuler et par l'architecture informatique servant de support de simulation dans le but de proposer une solution où l'aspect technique soit en accord avec l'aspect conceptuel. La démarche de modélisation intègre une description hiérarchique représentant l'organisation interne d'un système complexe, la construction d'un graphe logique traduisant une décomposition structurelle du système. Le graphe logique est ensuite utilisé pour réaliser une projection qui produit un graphe physique établissant une abstraction qui lie le niveau de représentation conceptuel et le système distribué sous-jacent. La plate-forme Atlas est un support de simulation distribuée à événements discrets dirigée par le temps et dont les propriétés dynamiques reproduisent le comportement de larges flux de données désagrégées. Une des particularités de cette plate-forme réside dans le concept de migration physique par groupe qui donne une solution adaptée pour réaliser et coordonner la migration de larges flux d'objets. Cette migration par groupe possède de nombreux avantages pour la simulation de systèmes sociétaux et urbains, notamment pour les systèmes en transport. L'utilisation de notre méthodologie et de sa plate-forme est illustrée par une application en transport qui modélise et simule les flux de personnes entre les différents halls d'un terminal d'aéroport.

This thesis introduces a methodology composed of a modelling approach and a computing support for the distributed simulation and the analysis of complex systems with large disaggregated data flows. The proposed methodology is influenced by both the system to simulate and the computing architecture used as a simulation support. This ensures a solution where the technical aspect fits the conceptual one. The modelling approach integrates a hierarchical description of the internal organization of a complex system, the design of a logical graph representing a structural decomposition of the system. A physical graph then establishes a binding between the conceptual representation level and the underlying distributed system. The Atlas platform is a support for discrete events time-driven distributed simulation whose dynamic properties replicate the behaviour of large disaggregated data flows. One of particularities of this platform resides in the concept of physical migration by group. This give a suitted solution to achieve and coordinate migration of large data flows, an important aspect for the simulation of societal and urban systems, particularly for transportation systems. The method and its platform is illustrated by an application in transportation that models and simulates people flows between the different halls of an airport terminal.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

« Si parva licet componere magnis (s'il est permis de comparer les petites choses aux grandes). Virgile, poète latin (70-19 avant notre ère) »

L'étude des phénomènes complexes issus notamment des systèmes sociétaux et urbains requiert le déploiement de méthodes et de techniques adaptées. L'utilisation désormais largement acceptée des outils informatiques pour la simulation de systèmes complexes conduit l'observateur vers de nouvelles formes de compréhension dépassant les limites des méthodes analytiques. L'étude, par une approche systémique, des actions et des interactions se produisant au sein des phénomènes étudiés par le biais de la simulation permet à l'observateur de comprendre et maîtriser l'évolution dans le temps de phénomènes tenus pour complexes. Toutefois, cette philosophie consistant à déterminer les comportements de systèmes complexes au travers d'outils et de calculs numériques suppose, pour l'observateur, une connaissance approfondie à la fois des méthodes de modélisation mais aussi des supports et des techniques informatiques de simulation. La recherche présentée dans ce document de thèse introduit une méthode et un support d'exécution orienté objet pour la simulation à événements discrets reposant sur l'utilisation des systèmes informatiques distribués. Une méthodologie de conception, basée sur une organisation hiérarchique et une décomposition spatiale des composants du système reposant sur l'utilisation des graphes, guide la conception de modèles représentant les systèmes complexes devant être exécutés par la plate-forme de simulation développée *Atlas*.

Besoins en modélisation et en simulation

Cette thèse considère comme contexte la simulation des systèmes complexes à larges flux de données désagrégées. Le terme *larges flux de données désagrégées* signifie qu'un tel système a la capacité d'échanger, entre les éléments d'un niveau agrégé (meso ou macro), des éléments d'un niveau plus fin (micro) dans son organisation hiérarchique interne. Ces éléments sont alors qualifiés de données désagrégées. Leur libre circulation, décidée à titre individuel, dans la structure interne du système complexe forme un flux de données désagrégées qui représente les interactions entre les composants agrégés.

Les besoins de méthodologies adaptées pour la simulation informatique de systèmes urbains et sociétaux lorsque le champ d'étude considéré est l'analyse des flux de données entre différents composants spatiaux est toujours un problème ouvert. La complexité croissante des modèles décrivant les phénomènes complexes est souvent en opposition avec les ressources

informatiques disponibles pour gérer, traiter et simuler ces systèmes. Le besoin d'obtenir à la fois des simulations à haute résolution mais aussi des vitesses d'exécution élevées ne peut être satisfait qu'en diminuant la précision du modèle au niveau microscopique, c'est-à-dire en réduisant à un ensemble minimal les détails régissant le comportement des éléments d'un système. Le refus de compromis sur la fidélité, le niveau de résolution ou la taille des modèles entraîne une volonté de combiner la puissance d'exécution et une expressivité adaptée qui est clairement postulée par les concepteurs et les analystes notamment dans le domaine d'étude des systèmes en transport.

Démarche et objectifs de la thèse

Dans une optique de simulation qualitative de systèmes complexes à larges flux de données désagrégées, nous avons analysé les perspectives de modélisation et de simulation offertes par l'utilisation des systèmes distribués. Les systèmes distribués sont des architectures multiprocesseurs composées d'un ensemble de machines physiquement séparées et interconnectées par un réseau de communication. Les systèmes distribués, dont l'intérêt pour la simulation de système complexe a été identifié, offrent de nouvelles perspectives pour l'étude de phénomènes à larges flux de données désagrégées. Les apports potentiels des systèmes distribués pour la simulation sont multiples. L'avantage majeur réside dans leur aptitude à simuler de modèles qui nécessitent de grandes capacités de traitement. Ce type d'architecture informatique permet par ailleurs, d'un point de vue conceptuel, d'adapter la répartition des données et des calculs à un niveau de granularité approprié afin de reproduire la nature et le comportement décentralisé de nombreux systèmes désagrégés.

La démarche d'élaboration de cette thèse résulte d'une réflexion itérative des préceptes et des principes de mise en œuvre nécessaires pour réunir d'une part l'aspect conceptuel où les systèmes complexes urbains et sociétaux requièrent des mécanismes de simulation adaptés aux larges flux de données désagrégées et, d'autre part les possibilités et les contraintes des systèmes distribués. De cette convergence est issu notre concept préalable de gestion agrégée des flux de données et son implémentation par le protocole de migration *Round Trip Bus*. Le protocole de migration a pour objectif de simuler les flux induits par les déplacements d'entités modélisées à un niveau microscopique. Lorsque ces flux de données désagrégées, qui interviennent entre des niveaux d'agrégation supérieurs au sein de l'organisation hiérarchique d'un système complexe, sont également agrégés, le protocole à la capacité de refléter le comportement du système considéré. D'un point de vue du système distribué, le protocole de migration réalise de manière optimisée les migrations physiques d'objets informatiques répartis sur différentes machines. Des résultats préliminaires, est apparu que cette piste devenait un concept à explorer pour la simulation de systèmes complexes à larges flux de données désagrégées. En effet, ce développement est un des éléments clés permettant d'établir un compromis entre les besoins et les limites respectivement des systèmes à simuler et des systèmes de simulation.

Au-delà de la gestion des flux, la poursuite des développements préalables et des besoins en modélisation et en simulation des systèmes complexes a donné naissance à Atlas, un environnement générique qui intervient comme une interface de programmation entre le concepteur et matériel. Cet intermédiaire sur lequel repose l'exécution de modèles est une plate-forme pour la simulation distribuée à événements discrets dirigée par le temps. La plate-forme s'inscrit elle-même dans un contexte plus vaste dont l'objectif est de fournir un cadre plus général pour la modélisation d'un système complexe en vue de sa simulation sur

un système distribué. La plate-forme par son implémentation incarne une méthodologie de conception adaptée à ces deux extrémités (systèmes) afin d'apporter une solution au besoin de distribuer les composants et leurs traitements sur plusieurs ordinateurs. La méthodologie est influencée à la fois par le système à simuler et par l'architecture informatique servant de support de simulation dans le but de proposer une démarche où l'aspect technique soit en accord avec l'aspect conceptuel. L'élément fondamental de la méthodologie est la prise en compte de l'aspect structurel d'un système complexe afin d'en extraire une décomposition qui se traduit par la production d'un graphe. Le graphe établit un lien entre le niveau de représentation conceptuel et le système distribué sous-jacent, permettant de concevoir la projection de composants spatiaux fixes vers les ordinateurs.

La validation expérimentale et le potentiel de la plate-forme de simulation Atlas et de sa méthodologie de conception pour la modélisation et la simulation distribuée sont articulés autour d'un cas d'étude qui intègre les aspects significatifs de systèmes complexes à larges flux de données désagrégées. L'application proposée dans cette thèse modélise et simule les flux de personnes qui circulent, par l'utilisation d'une navette, entre les halls du terminal 2 de l'aéroport Paris CDG. L'objectif in fine de conception et de prototypage de cette application est de démontrer l'avantage et la pertinence des systèmes distribués pour la simulation et la compréhension de systèmes urbains et sociétaux représentables par un graphe et dans lesquels interviennent de larges flux de données désagrégées.

Plan du mémoire

Ce document est organisé en six chapitres et structuré en deux parties. La première partie, composée des trois premiers chapitres, présente les principes, les propriétés et les utilisations de la simulation à événements discrets de systèmes complexes dans lesquels interviennent de larges flux de données désagrégées. Le premier chapitre définit la notion de système complexe au travers de ses aspects structurels, fonctionnels et dynamiques. Les principes de modélisation et les mécanismes de simulation de tels systèmes sont ensuite présentés. Ce chapitre aborde également l'aspect validation et la vérification des modèles et des simulations des systèmes complexes. Le deuxième chapitre précise la nature et les propriétés des systèmes complexes à larges flux de données désagrégées et dresse un bref panorama de quelques travaux relatifs à la simulation de ces systèmes. Le troisième chapitre présente l'utilisation des systèmes distribués pour la simulation à événements discrets. Les différents mécanismes de distribution et la nature d'une simulation distribuée à événements discrets dans le modèle de communication par échanges de messages sont introduits. Ce chapitre propose par ailleurs une présentation des mécanismes et des outils de migration, techniques indispensables pour représenter des flux de données désagrégées en environnement réparti.

La seconde partie de cette thèse présente les méthodes et les outils développés pour la simulation distribuée de systèmes complexes à larges flux de données désagrégées. Le chapitre quatre introduit notre méthodologie de conception. Cette approche de modélisation guide la réalisation de modèles vers les concepts de description hiérarchique, de représentation par graphe et vers l'établissement d'une correspondance entre la représentation d'un système complexe et son support pour la simulation répartie (concept de projection). Le chapitre cinq présente l'environnement distribué agissant comme support générique pour la simulation que nous avons mis en œuvre. Le chapitre six présente une étude de cas dans laquelle est exploitée notre approche de modélisation ainsi que notre plate-forme de simulation afin d'étudier les flux de personnes entre les halls d'un terminal d'aéroport. Finalement une conclusion générale

propose un panorama des principaux points de cette recherche et aborde les perspectives de ce travail.

CHAPITRE 1

Principes de modélisation et de simulation

« Science cannot solve the ultimate mystery of Nature. And it is because in the last analysis we ourselves are part of the mystery we are trying to solve. Max Planck, Nobel de physique 1918 »

Dans notre quête de compréhension des phénomènes naturels caractérisés par la notion philosophique de l'être, ou artificiels caractérisés par la notion de l'intention d'être, nous sommes fréquemment confrontés aux problèmes de caractérisation de ces phénomènes communément qualifiés par la terminologie de *systèmes complexes*. Comprendre et maîtriser l'évolution dans le temps de phénomènes réels (ou artificiels), tel est l'objectif de la simulation. Cette méthode d'analyse n'est pas l'unique solution, toutefois, nombreux sont les dits systèmes complexes qui ne peuvent être étudiés à l'aide des outils mathématiques actuels. En effet, certains mettent en jeu un si grand nombre de variables qu'il est humainement impossible d'appréhender le système. La simulation apparaît donc comme une alternative à ces limites. Soumise à l'utilisation d'un modèle, une représentation d'un système, la simulation reproduit le comportement de ce système afin de pouvoir l'étudier dans un but de validation, de vérification, de prévision, de planification ou de compréhension de phénomènes passés, présents, ou futurs. Dans ce premier chapitre, nous présentons les propriétés essentielles des systèmes complexes, leurs mécanismes de modélisation et les principales techniques de simulation utilisées. La section 1.1 définit la notion de système complexe. La section 1.2 introduit les principes de modélisation de ces systèmes complexes et la section 1.3 présente les principaux mécanismes de simulation. La section 1.4 aborde les principes de validation et de vérification du processus de modélisation et de simulation des systèmes complexes. La section 1.5 conclut ce chapitre.

1.1 Systèmes complexes

Les systèmes complexes représentent un ensemble aux contours difficilement identifiables en raison de l'ensemble relativement vaste des phénomènes physiques ou fictifs qui peuvent êtres regroupés par ce terme. Dans cette section nous définissons et présentons les propriétés principales des systèmes complexes.

1.1.1 Système et complexité

Du grec *systema* signifiant qui tient ensemble, mais avec l'idée d'union en un tout organisé voire stable, un système est un ensemble organisé d'interactions entre des éléments. De nombreuses autres définitions d'un système ont été proposées. Elles mettent pour la plupart l'accent soit sur les éléments composant le système, soit sur les deux notions de totalité et de relation entre les éléments. La caractérisation d'un système n'a jamais été une tâche aisée, souvent sujette à simplification. Descartes en 1637, dans le discours de la méthode illustre cette problématique en tentant d'éliminer l'aléatoire ou l'incertain : *Conduire par ordre mes pensées en commençant par les objets les plus simples et les plus aisés à connaître, pour monter peu à peu comme par degrés jusqu'à la connaissance des plus composés.*

Ferdinand de Saussure (linguiste suisse, 1857-1913) définit un système par une totalité organisée, faîte d'éléments solidaires ne pouvant être définis que les uns par rapport aux autres en fonction de leur place dans cette totalité. Ludwig Von Bertalanffy (biologiste autrichien, 1901-1972) a été l'un des premiers à montrer qu'un système est un tout non réductible à ses parties. En 1950, il publie *The General System Theory* [von Bertalanffy, 1973], dans lequel il explique les nombreuses et les complexes interactions qui caractérisent les technopoles modernes et qui en influencent l'organisation technologique et sociale. Il définit alors un système par trois critères fondamentaux : le nombre d'éléments, leur espèce et les relations qui existent entre ces éléments. D'après [Sloot et al., 1997], un système est une population composée d'éléments d'une espèce unique avec des attributs bien définis. De plus, dans l'évolution spatiale et temporelle du système, si ces éléments ont des interactions non-linéaires, alors de ces interactions peuvent émerger des comportements globaux. Introduisant la notion de finalité (celle de maintenir la structure et la pérennité du système) Joël de Rosnay définit un système comme un ensemble d'éléments en interaction dynamique, organisés en fonction d'un but [de Rosnay, 1975].

La racine latine du mot complexe est *plexus* signifiant entrelacement qui engendre complexus : enchevêtrement, connexion, étreinte. Contrairement à l'idée reçue, le contraire de complexe n'est pas simple mais implexe qui caractérise une unité d'action indécomposable, irréductible pourtant à un élément unique. Edgar Morin précise que la complexité n'est pas la complication et postule que ce qui est compliqué peut se réduire à un principe simple [Morin, 1977]. Par exemple une voiture est une machine très compliquée, mais démontable en un ensemble fini de pièces. En revanche, un organisme vivant est complexe dans le sens où il ne peut être décomposé et reconstruit à partir d'éléments simples et indépendants. De la notion de complexité d'un système peut résulter trois éléments : (1) le degré élevé d'organisation, (2) l'incertitude de son environnement et (3) la difficulté, sinon l'impossibilité d'identifier tous les éléments et de comprendre toutes les relations mises en jeu. Dans des termes similaires Joël de Rosnay définit la complexité référent à un système [de Rosnay, 1975] par une grande variété des éléments, une organisation en niveaux hiérarchiques internes de ces éléments, une grande variété des liaisons entre ces éléments et la non-linéarité des interactions entre ces éléments et la difficulté voire l'impossibilité de dénombrer de façon exhaustive les éléments qui constituent ce système. La notion de complexité implique par ailleurs celle d'imprévisible possible, d'émergence plausible de nouveauté au sein du phénomène que l'on tient pour complexe.

Le Moigne [le Moigne, 1999] et Clergue [Clergue, 1997] établissent également la distinction entre les systèmes complexes et les systèmes compliqués. Les systèmes compliqués sont des systèmes que l'on peut réduire en éléments plus simples que l'on peut analyser séparément pour comprendre le système global. Dans le cas des systèmes complexes, la somme des éléments fait émerger de nouvelles propriétés qui ne sont pas dans les éléments

eux-mêmes. Construit par l'observateur qui souhaite le décrire, un système complexe est par définition un système irréductible à un modèle compliqué. L'observateur postule la complexité du phénomène sans pour autant croire qu'une telle propriété existe dans la nature ou la réalité. En fait, la complexité est relative par rapport au niveau d'observation que l'on désire d'un système. Pour son observateur, il est complexe précisément parce qu'il tient pour certain l'imprévisibilité potentielle des comportements. Dans ce contexte, une société humaine est un exemple de système complexe. La diversité des interactions entre les éléments (individus), les groupes d'éléments par exemple sociaux telle qu'une famille, les multiples natures et spécialisations des éléments en font un système sur lequel les prédictions sont difficiles à réaliser. Dans [de Rosnay, 1995] il est précisé qu'un système complexe n'est pas nécessairement compliqué. Grâce à des méthodologies appropriées et adaptées[1], on peut comprendre la complexité et agir sur elle. Un système compliqué, lui, résiste à toute forme d'analyse car hors de compréhension dans sa totalité par la somme de ses parties.

La notion d'ordre dans la complexité est une relation subjective. Le degré de complexité d'un système est souvent caractérisé par le nombre de niveaux d'organisation, d'éléments par niveaux, de relations entre les niveaux, de relations entre les éléments par niveaux et de la nature ou de la complication des relations.

Il existe de nombreuses autres définitions de la notion de système complexe. Elles ont pour la plupart en commun de définir les systèmes complexes par l'intégration d'un ordre, d'une organisation et une hiérarchie observable. Ces définitions admettent par ailleurs les caractères d'émergence, de finalité, d'interdépendance et d'interaction. De ces diverses approches d'un système, nous retenons les quatre concepts fondamentaux qui en sous-tendent la définition :

- L'organisation définie par l'agencement de relations, en niveaux hiérarchiques internes, entre les éléments (de natures diverses) qui composent le système. Les différents niveaux et les éléments individuels étant reliés par une grande variété de liaisons desquelles il résulte une haute densité d'interconnexions ;
- L'interaction définie par des relations entre les éléments d'un système complexe qui au-delà d'une simple relation causale provocant un effet sur le système peuvent, selon nous, être définies par un flux d'informations d'éléments du système ;
- La globalité (ou totalité) est définie par le fait qu'un système complexe est plus que la somme de ses éléments, et qu'il possède des propriétés que ses composants n'ont pas ;
- La complexité qui vise à articuler le tout et ses parties, le global et le particulier en un aller et retour incessant en eux (émergence et rétroaction, voir section 1.1.4) est caractéristique d'un degré élevé d'organisation et d'interactions difficiles à déterminer.

Les quelques caractéristiques et les propriétés d'un système complexe abordées au travers de ces définitions peuvent, afin d'en appréhender le comportement, être mise en lumière par la description, plus ou moins formelle, de son aspect structurel, de son aspect fonctionnel et de son aspect dynamique.

1.1.2 Aspect structurel

L'aspect structurel décrit l'organisation des composants d'un système. Les principaux traits structuraux d'un système varient d'un domaine à un autre mais la structure représente

[1] En particulier grâce à l'étude de l'émergence potentielle de propriétés nouvelles.

généralement la partie stable du système et inclut, selon [de Rosnay, 1975] et [Forrester, 1980], une limite, des éléments, des réservoirs et un réseau de communication.

Les éléments (ou entités) sont les parties constituantes d'un système. Ces éléments peuvent être dénombrés de manière plus ou moins aisée. Ils sont en général de nature diverse, hétérogènes et assemblés de façon hiérarchique par catégorie (figure 1.1). Les réservoirs sont des structures (sous-systèmes ou composants) dans lesquelles sont entreposés de l'information ou des éléments. Le réseau de communication tisse un lien entre les différents réservoirs en définissant les interconnexions possibles entre les composants du système. Les réseaux de communication agissent ainsi comme un support permettant l'échange et le transport d'informations ou d'éléments entre les composants d'un système, entre différents systèmes (existence possible de sous-systèmes) ou entre un système et son environnement. Un composant est un élément ou un ensemble d'éléments destinés à remplir une fonction particulière et un sous-système est une association de composants destinés à remplir une ou plusieurs fonctions opérationnelles au sein d'un système.

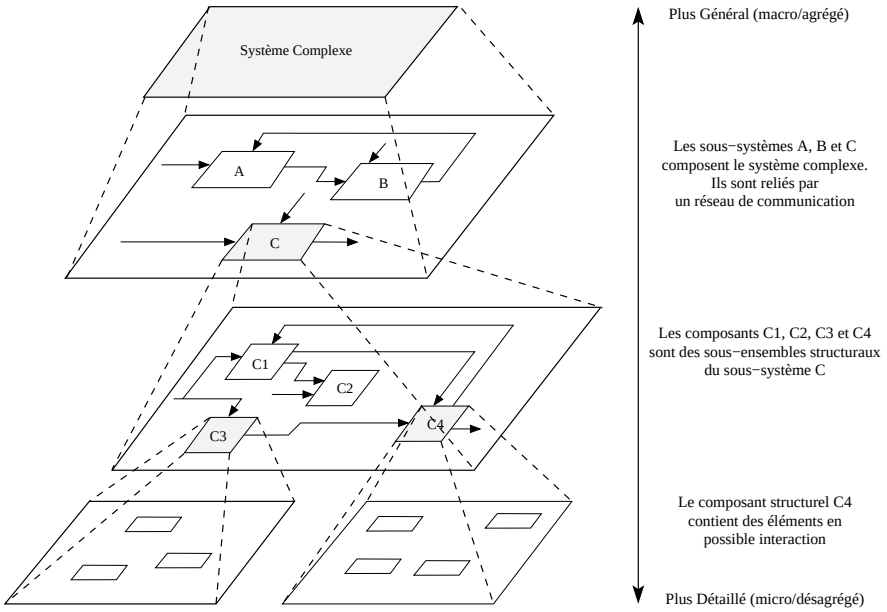


Fig. *1.1* — Structure hiérarchique d'un système complexe

La limite d'un système définit une frontière plus ou moins perméable entre la totalité des éléments et l'environnement extérieur et restreint le champ d'analyse au seul problème considéré. Cette limite peut parfois être floue ou encore mouvante par exemple lorsque l'on étudie des phénomènes sociaux de groupe. Elle peut être, selon nous, décomposée en une borne inférieure et une borne supérieure. La borne supérieure définit la limite du niveau macroscopique, c'est-à-dire les éléments organisés correspondant aux niveaux hiérarchiques supérieurs du système. La borne inférieure définit la limite du niveau microscopique du système qui correspond à la limite de description des éléments simples. Un système ouvert se définit par rapport à cette frontière, il s'agit d'un système possédant des interactions avec le monde qui l'entoure. A contrario, un système fermé n'échange aucune information avec son environnement extérieur.

1.1.3 Aspect fonctionnel

L'objectif de l'étude de l'aspect fonctionnel d'un système est de décrire les corrélations (relations et interactions) entre les éléments du système et les changements avec le temps de ces éléments. Cet aspect du système évolue généralement plus rapidement que la structure et ses principaux traits fonctionnels sont des flux, des centres de décision, des ajustements et des boucles de rétroaction [de Rosnay, 1975, Forrester, 1980].

Les flux représentent l'information ou les éléments circulant entre les réservoirs dont ils font varier le contenu. Les flux circulent dans les réseaux de communication. Les centres de décision (vannes) contrôlent et organisent le réseau de relations, c'est-à-dire coordonnent les débits des différents flux. Les ajustements (délais) résultent des différentes vitesses de circulation des divers éléments. Ces délais ont un rôle critique dans les phénomènes d'amplification ou d'inhibition qui sont typiques du comportement des systèmes complexes. Un peu comme les réservoirs, ils permettent de procéder aux ajustements nécessaires à la bonne marche du système dans le temps.

Dans un système où s'effectuent des transformations, il y a des entrées (qui résultent de l'action de l'environnement sur le système) et des sorties (action du système sur l'environnement). Les entrées et les sorties sont séparées par une durée : avant et après, passé et présent. Une rétroaction a lieu lorsque des informations sur les résultats d'une transformation sont acheminées à l'entrée du système (effet en retour). On distingue deux types de boucles de rétroaction : (1) les boucles positives dans lesquelles la réaction agit dans le même sens que l'action principale et (2) les boucles négatives grâce auxquelles la réaction agit dans le sens opposé à l'action principale. Remarquons que la boucle de rétroaction est une caractéristique typique des systèmes ouverts. Dans la section suivante, nous présentons quelques éléments de compréhension de la dynamique des systèmes qui repose pour partie sur ces boucles de rétroactions.

1.1.4 Dynamique d'un système

La dynamique des systèmes, introduite par Jay Forrester dans les années soixante, est une méthode générale conçue pour analyser certains types de problèmes complexes [Forrester, 1980]. La méthode de la dynamique des systèmes postule que les boucles de rétroaction sont responsables des changements qui se produisent dans l'évolution temporelle du système. De manière plus générale la dynamique d'un système repose sur l'action combinée des boucles de rétroaction, des flux et des réservoirs. En effet, les interactions entre les réservoirs constituent une forme de rétroaction. Les transformations internes dues aux évolutions temporelles des réservoirs et des flux conduisent le système vers son objectif. Les différentes boucles de rétroaction agissent ensuite comme des perturbateurs ou des stabilisateurs vis-à-vis de cet objectif. Le système de chauffage d'une pièce illustre parfaitement cette notion de boucle de rétroaction. Lorsque la température diminue jusqu'au-dessous d'une valeur réglée sur le thermostat, celui-ci réagit en chauffant la pièce. Lorsque la température de la pièce dépasse la température réglée sur le thermostat, celui-ci commande l'extinction du chauffage. L'action du thermostat influence le chauffage, dont l'activité influence elle-même le thermostat. Les rétroactions positives et négatives s'appliquent soit sur l'extérieur, comme illustré sur la figure 1.2, soit à l'intérieur du système.

Les boucles négatives provoquent la dynamique du maintien (régulation) et accentuent la convergence vers un but préalablement défini. Les boucles négatives favorisent le maintien de l'équilibre, la stabilité du système en fonction des stimulus ou des perturbations extérieures

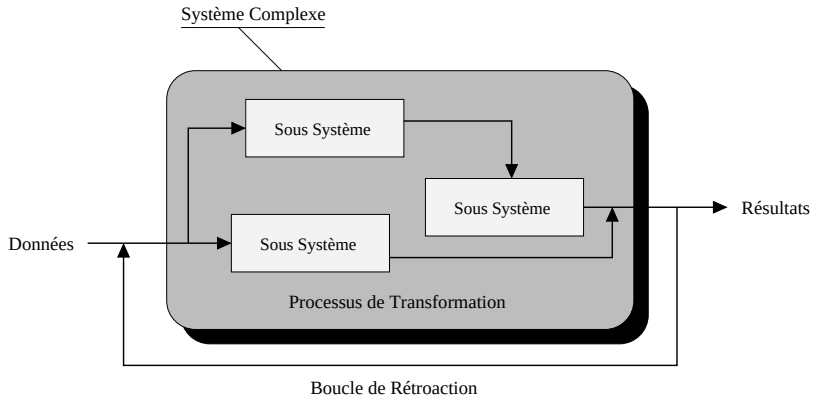


Fig. *1.2* — Dynamique d'un système complexe

et de son évolution propre. Un système homéostatique (i.e. qui résiste au changement) est un système capable de maintenir sa structure et ses fonctions par l'intermédiaire d'une multiplicité d'équilibres dynamiques. La rétroaction négative permet ainsi au système de conserver et de protéger son identité (fonctionnelle et structurelle), sa nature malgré les perturbations de l'environnement. Elle caractérise donc les systèmes ouverts qui conservent leurs structures et leurs fonctions intactes par l'intermédiaire d'équilibres dynamiques successifs.

Les boucles positives provoquent la dynamique du changement d'un système et produisent un accroissement des divergences ou des écarts entre les objectifs du système et ses sorties. Une rétroaction positive favorise la croissance et l'évolution du système. Elle peut conduire par ailleurs à des changements de finalité et la recherche de nouveaux objectifs à poursuivre. L'évolution et l'émergence d'un système sont associés à la rétroaction positive c'est-à-dire à la capacité qu'a un système d'accéder à de nouveaux points d'équilibre, et à une nouvelle identité.

La théorie des systèmes [von Bertalanffy, 1973] pose les bases d'une pensée de l'organisation des systèmes complexes dans laquelle il existe des propriétés émergentes qui naissent dans l'organisation d'un tout à partir des composants, et qui peuvent rétroagir sur les propriétés du système. Ainsi, l'organisation hiérarchique fonctionnelle et structurelle d'un système complexe et les interactions entre les éléments microscopiques qui le composent peuvent provoquer une émergence de nouvelles propriétés en provenance de différents niveaux inférieurs jusqu'aux niveaux d'organisation supérieurs conférant au système complexe la propriété de totalité. Dans [de Rosnay, 1995], Joël de Rosnay présente les phénomènes liés à l'émergence de la complexité organisée, comme ceux que l'on peut observer dans des systèmes moléculaires (e.g. origine de la vie), les sociétés d'insectes (e.g. fourmilières, ruches), les systèmes sociétaux (e.g. entreprises, marchés, économies) ou les écosystèmes. Il précise que le fonctionnement global d'un système complexe conduit à l'émergence de structures, de fonctions et de propriétés nouvelles imprédictibles.

En relation avec leurs domaines d'études spécifiques, de nombreuses définitions de la notion d'émergence ont été proposées [Ali et Zimmer, 1997]. Le problème de l'émergence est présenté dans [Ali et Zimmer, 1998, Cariani, 1989] selon trois perspectives majeures. Il s'agit de l'émergence thermodynamique, l'émergence relative à un modèle et l'émergence informationnelle. La conception thermodynamique de l'émergence (basée sur la physique) considère l'émergence comme la formation de nouvelles structures physiques. Ce phénomène est décrit en termes équations différentielles et s'applique aux univers continus. L'émergence

relative à un modèle (initiée en biologie) repose sur la définition de Rosen [Rosen, 1985] dans laquelle l'émergence est identifiée par la dérivation du comportement du système par rapport au modèle (i.e. la représentation conceptuelle du système, voir section 1.2) construit par l'observateur. En accord avec la définition de Rosen, [Heylighen, 1991] considère l'émergence comme un processus qui ne peut être décrit par un modèle fixe. Cette approche implique alors que l'émergence doit être décrite par un méta-modèle (i.e. modèle du modèle) représentant les transitions d'un modèle à un autre. La dynamique du système est ainsi en partie définie par une variation dans la structure ou la fonction qui se traduit par un changement du modèle.

La conception de l'émergence informationnelle repose sur l'idée que le comportement global d'un système provient des interactions locales entre les éléments microscopiques. Dans ce contexte Langton définit l'émergence en termes de rétroactions entre les niveaux d'organisation d'un système dynamique : la micro-dynamique locale cause une macro-dynamique globale et en retour une macro-dynamique globale contraint une micro-dynamique locale [Langton, 1989]. Cette approche, reposant sur des bases mathématiques, s'applique aux univers discrets. Les comportements émergent au travers de l'utilisation de techniques comme les automates cellulaires ou la simulation informatique (cf. section 1.3). Dans la suite de ce document, nous nous attacherons à cette approche de l'émergence qui, associée dans un va et vient avec la rétroaction, décrit les phénomènes dynamiques se produisant lors de la simulation informatique de systèmes complexes.

Le concept de facteur d'émergence dénote les entités, les relations ou les propriétés qui de par leurs caractéristiques vont permettre les conditions de manifestation de nouvelles propriétés à un niveau d'agrégation supérieur (meso ou macro). Les facteurs d'émergence peuvent avoir un impact global ou spécifique dans la hiérarchie fonctionnelle et structurelle du modèle. Les facteurs d'émergence dits généraux sont actifs dans tous les niveaux d'organisation alors que les facteurs d'émergence spécifiques sont eux propres à chaque niveau d'organisation, et dépendant des caractéristiques du niveau considéré. Ils affectent le système par transitivité, d'un niveau d'organisation à un autre. L'identification de façon systématique et exhaustive de l'ensemble des facteurs d'émergence de chaque niveau d'organisation, et la caractérisation des actions et des interactions par lesquelles ces facteurs peuvent générer les conditions de la manifestation de nouvelles propriétés reste un objectif difficile à atteindre. En effet, cette description exhaustive de l'ensemble des règles d'émergence est souvent difficile à obtenir en raison de la complexité inhérente des systèmes. C'est à ce niveau que l'intervention de la simulation est déterminante. Le découplage entre structure et fonction peut favoriser la distinction des facteurs d'émergence. Ainsi, les facteurs d'émergence de nouveaux éléments structurels influencent l'aspect structurel du modèle et de même, les facteurs d'émergence des nouvelles propriétés fonctionnelles influencent l'aspect fonctionnel du modèle. De ce fait, dans un système complexe où la structure à la faculté de déterminer partiellement ou totalement la fonction (et vice-versa) et où les propriétés émergentes peuvent avoir, en partie, des bases structurelles, il devient difficile de concevoir une description fonctionnelle indépendante d'une description structurelle.

L'évolution de nombreux systèmes ayant une composante temporelle (i.e. systèmes dynamiques) repose sur le principe de causalité. Ce principe fut en particulier expliqué par le savant français Pierre-Simon de Laplace qui recourait à une métaphore par la suite appelée le *démon de Laplace*. Il affirmait que l'état présent de l'Univers est un effet de son état précédent et la cause de son état suivant. Cela suppose qu'il existe une séquentialité des états, c'est-à-dire qu'un événement (au sens de l'initiateur d'un changement d'état) futur ne peut influencer un événement passé.

L'évolution d'un système complexe, guidé par sa dynamique propre, implique l'occurrence

et la prise d'états successifs potentiellement différents. Les événements intervenant dans le système, l'émergence, et la rétroaction entraînent le changement de l'état du système. Un système dynamique peut être qualifié de déterministe s'il existe une et une seule conséquence ou état à chaque état. Il est qualifié de stochastique ou aléatoire s'il existe une ou plusieurs conséquences ou phases possibles à partir d'une distribution de probabilités des états possibles. Un système caractérisé par une nature aléatoire est plus difficile à étudier car son comportement est non reproductible (indéterministe) et imprédictible. La notion de complexité impliquant une certaine forme d'imprévisibilité qui ne peut être tenue pour déterministe, les systèmes complexes représentent alors des environnements souvent non-réversibles et aléatoires.

L'auto-organisation est définie par la capacité pour un système de faire évoluer son état à la limite entre l'ordre et le désordre par des facteurs extérieurs et l'action combinée de croissance et de régulation. Dans ce contexte, les perturbations ou les données issues de l'environnement sont utilisées par le système pour (1) préserver son organisation et (2) accroître ses capacités organisationnelles. Dans le cas d'un tel système, il y a non seulement croissance mais également complexification du fait de l'obligation de gérer les incertitudes croissantes issues de l'élargissement des environnements dans lesquels il se déploie.

1.1.5 Typologie des systèmes

Plusieurs typologies des systèmes ont été proposées. Celles-ci procèdent souvent de façon ascendante partant du type le plus simple caractérisé par une certaine passivité (dans le sens de simple réactivité) pour aller vers des types plus sophistiqués intégrant des structures et des fonctionnalités de plus en plus riches telles que la régulation, la mémorisation ou encore la faculté d'accroître ses capacités organisationnelles (auto-organisation).

Parmi les différentes classifications proposées, Le Moigne présente dans [le Moigne, 1990] une classification en trois points : (1) les systèmes-machines qui relèvent de la mécanique et de l'ingénierie, (2) les systèmes vivants et les systèmes artificiels complexes dans lesquels apparaissent les processus de mémorisation et de décisions et (3) les systèmes humains et sociaux dans lesquels apparaissent l'intelligence et l'auto-organisation. Dans le cadre de recherches fondamentales sur le chaos déterministe a émergé, dans la seconde moitié du siècle précédent, une autre classe dans laquelle les systèmes sont dits dynamiques. Ces systèmes sont caractérisés par le fait que derrière une apparence chaotique (désordre déterministe) existe un ordre plus complexe que l'ordre apparent dans lequel apparaîtrait un certain indéterminisme et où cet ordre émergerait par auto-organisation. La typologie des systèmes proposée par J. Lesourne dans [Lesourne, 1976] distingue quatre catégories :

1. Les systèmes à états sont définis avec les seules notions d'entrée, de sortie et de transformations entre les entrées et les sorties. Cette classe comprend la plupart des machines construites par l'homme. Ce sont en général des machines commandées de l'extérieur par un mécanisme régulateur ou une intervention humaine : robot télécommandé ou un moteur de voiture par exemple ;
2. Les systèmes à buts diffèrent des précédents par le fait que le contrôle est intégré dans le système. De tels systèmes intègrent la notion de régulation interne et la capacité d'atteindre des objectifs grâce à un ou plusieurs contrôles de régulation. Un missile à tête chercheuse constitue par exemple un système de cette catégorie ;
3. Les systèmes à apprentissage disposent d'une mémoire qui enregistre les informations passées (résultats, observations, décisions) et de mécanismes de calculs et de contrôle

qui incluent la faculté de prendre des décisions en fonction de l'expérience acquise. Un tel système peut également apprendre de façon systématique par un processus de type essais-erreurs (notion d'adaptation). De tels systèmes peuvent organiser leur propre apprentissage et atteignent le stade de l'auto-organisation. A un niveau encore supérieur, le système peut se fixer lui-même ses propres objectifs. Un système expert en stratégie économique entre dans cette classe ;

4. Les systèmes à décideurs multiples possèdent une structure complexe constituée d'un certain nombre de systèmes à buts ou à apprentissage. Ces systèmes sont doués d'une auto-organisation spontanée qui conduit à des distributions hiérarchiques simples pour les organisations (e.g. armée ou entreprise) ou, complexes (hiérarchies enchevêtrées) dans le cas des sociétés où un individu peut appartenir à plusieurs hiérarchies distinctes.

1.2 Modélisation de systèmes complexes

Le modèle est au sens platonicien le paradigme, la forme idéale sur laquelle les existences sont réglées. Tout modèle est constitué d'une part de la description de la structure du système et d'autre part de la description des fonctionnements et des dynamiques qui modifient cette structure. Un modèle dont l'objectif est la formalisation et la compréhension d'un système peut être défini par trois points essentiels [Landry et Santerre, 1999] :

- La représentation : un modèle est avant tout une représentation du système étudié ;
- La ressemblance : un modèle doit ressembler au système représenté ;
- La simplification : un modèle constitue une simplification du système.

Issus d'un mécanisme de schématisation et de simplification d'un phénomène de la réalité, les modèles sont par définition des approximations et donc souvent imparfaits. George Box a eu à ce sujet une réflexion plus catégorique : *all models are wrong, some are useful*. De ce fait, il ne faut pas confondre nos représentations avec le système lui-même car ceux ci ne représentent que des visions parcellaires. Considérant que l'essentiel dans l'étape de modélisation est de ne pas confondre les représentations (modèles) avec le monde lui-même (le système), ceux-ci fournissent un instrument dans la production de la connaissance et permettent d'analyser différentes situations dynamiques (dont les résultats peuvent être confrontés à la réalité).

1.2.1 Approches analytiques et systémiques

L'approche analytique, née de la démarche cartésienne, et l'approche systémique, issue de la cybernétique [Wiener, 1948] et de la théorie des systèmes [von Bertalanffy, 1973], sont fondées sur des postulats épistémologiques différents et définissent deux façons de percevoir la réalité (le système). Ces approches utilisent des méthodologies qui leur sont propres pour construire des modèles et abordent l'étude d'ensembles possédant des niveaux de complexité divers [Lapointe, 1993].

D'après George Henry Lewes, dans Problems of Life and Mind (1875), il existe des phénomènes dits résultants, qui peuvent être abordés par une méthode analytique, et des phénomènes dits émergents, qui ne peuvent être compris à partir de la seule étude de leurs éléments constituants. L'émergence de phénomènes nouveaux, non prévisibles d'un point de vue strictement analytique ne peut être étudié correctement que par une approche holistique

considérant les interactions entre les phénomènes comme étant l'élément essentiel d'étude[2]. On parle alors de méthode systémique, seule apte à percevoir toute la dynamique des systèmes complexes par leurs rétroactions et leurs émergences.

Aristote (-384 à -322) déjà légifère sur la distinction entre le réel et le possible en établissant une différence entre phénomènes réels et évaluations virtuelles. Il se pose alors la question : toute évaluation des phénomènes est-elle déterministe à partir des conditions initiales ou bien y a-t-il des phénomènes spontanés. Cette question présuppose l'impossibilité de parvenir à comprendre les systèmes complexes sans avoir préalablement isolé les diverses parties qui les composent en laissant de coté les relations pouvant exister entre deux parties. Partant de cette approche héritée d'Aristote, Descartes posera plus tard les bases d'une modélisation analytique en énonçant (discours de la méthode) les quatre préceptes d'une approche rationnelle destinée à analyser, comprendre et contrôler la réalité :

- Ne jamais concevoir une chose comme étant vraie, sans la connaître comme telle ;
- Diviser chaque difficulté rencontrée en autant de parcelles nécessaires à leur compréhension ;
- Analyser en allant du plus simple vers le plus complexe ;
- Inventorier de façon complète afin de ne rien omettre.

L'approche analytique (cartésienne) est une démarche descendante qui vise à expliquer la nature des systèmes en descendant jusqu'aux niveaux d'organisation sous-jacents desquels une connaissance exhaustive est possible. Dans l'étude de la dynamique, l'approche analytique adopte ensuite un sens ascendant en élaborant la connaissance du système par additivité des parties. Il s'agit finalement d'une démarche sommative qui ne prend pas en compte toute la nature implicite de la complexité et des interactions entre les parties. Cette approche repose sur une démarche rigoureuse mais est néanmoins réductionniste car elle réduit la complexité des systèmes à une connaissance basée sur une agrégation des connaissances plutôt que comme un tout. L'approche analytique a par conséquent tendance à déduire de l'étude d'un seul élément appartenant à un ensemble, des lois considérées applicables à la globalité du système.

Certains doutes émis sur l'efficacité du précepte réductionniste de l'approche analytique s'amplifient avec la prise en compte du phénomène de complexification des ensembles qui fait ressortir, avec acuité, les limites de cette méthode pour l'étude des systèmes. Ashby dans [Ashby, 1956] présente l'approche analytique comme étant le plus souvent impropre à l'étude des systèmes complexes. Daniel Durand précise dans [Durand, 1993] que la prise de conscience de la complexité et de l'incertitude des systèmes réels conduit à la diffusion lente mais inéluctable du paradigme (ou modèle) systémique, seul permettant de prendre en considération et de traiter de façon adéquate, non seulement complexité et incertitude, mais aussi ambiguïté, flou ou hasard. Cette approche systémique se présente souvent comme la science du complexe de laquelle le concept moderne de système (tel que présenté dans la section 1.1) s'est peu à peu construit au cours du vingtième siècle en provenant de différents domaines d'études scientifiques.

L'approche systémique a pour but d'essayer de rendre compte de la multiplicité des facteurs permettant d'expliquer et de comprendre le phénomène étudié en prenant en compte l'imbrication et l'interaction des éléments en fonction de leur environnement. L'approche systémique cherche donc à privilégier la connaissance des relations et des interactions entre les

[2]En opposition au réductionnisme (analytique) qui préconise que le tout peut être décomposé et analysé en termes de ses composantes considérées comme fondamentales.

éléments qui constituent un système complexe. En outre, ces interactions entre les éléments sont systématiquement restituées quant à leur signification par rapport à la globalité du système considéré. Cette approche postule qu'il est impossible d'atteindre la compréhension de l'ensemble comme un tout par l'étude exclusive de ses parties et focalise donc plus l'attention sur les propriétés constitutives et les facteurs émergents d'un système.

En opposition aux préceptes cartésiens de l'approche analytique énumérés précédemment, Le Moigne définit quatre préceptes pour l'approche systémique [le Moigne, 1990] qu'il qualifie de préceptes du *nouveau discours de la méthode*. Au précepte de l'évidence, il oppose la pertinence, au réductionnisme le globalisme, au causalisme le téléologisme et à l'exhaustivité l'agrégativité :

- Le précepte de la pertinence consiste à convenir que tout élément considéré se définit par rapport aux intentions implicites ou explicites du modélisateur ;
- Le précepte du globalisme consiste à considérer toujours l'élément étudié comme une partie immergée et active au sein d'un plus grand tout ;
- Le précepte téléologique consiste à interpréter l'élément non pas en lui-même, mais par son comportement et comprendre ce comportement et les ressources qu'il mobilise au regard du projet que le modélisateur attribue à l'élément ;
- Le précepte de l'agrégativité consiste à admettre que toute représentation d'un système est volontairement simplificatrice.

Joël de Rosnay[3] dans [de Rosnay, 1975] établit une comparaison entre l'approche analytique et l'approche systémique. L'approche analytique isole et se concentre sur les éléments du système en s'appuyant sur la précision des détails. Elle fournit donc des modèles précis et détaillés mais difficilement utilisables dans l'action. Elle conduit ainsi à connaître les particularités et à programmer une action dans son détail au détriment de ses buts. L'approche analytique considère uniquement la nature des interactions et est efficace lorsque ces interactions sont linéaires et faibles. Elle ne modifie qu'une variable à la fois et affirme l'indépendance au regard de la durée ainsi que la réversibilité des phénomènes étudiés. L'approche analytique valide un fait par une preuve expérimentale à l'intérieur d'un cadre théorique et enfin conduit à un enseignement sectorisé par discipline.

L'approche systémique relie et se concentre sur les interactions entre les éléments et considère leurs effets en s'appuyant sur la perception globale. Elle fournit ainsi des modèles insuffisamment rigoureux pour servir de base de connaissances, mais est utilisable dans l'action et la décision car elle conduit à la connaissance des buts et à l'action programmée par objectifs au détriment d'une clarté des détails. Elle modifie des groupes de variables simultanément et introduit la notion de durée et d'irréversibilité. L'approche systémique est efficace lorsque les interactions sont non-linéaires et fortes. Elle valide par la comparaison du fonctionnement du modèle avec la réalité et enfin conduit à un enseignement pluridisciplinaire. La comparaison entre ces deux approches de modélisation est résumée dans la table 1.1.

Simplifiant à l'extrême ces deux façons de percevoir les systèmes complexes, l'approche analytique est l'étude des éléments et des comportements microscopiques. Joël de Rosnay précise que la démarche analytique est indispensable pour fonder la science, mais qu'elle ne suffit pas pour expliquer la dynamique et l'évolution des systèmes complexes (rétroactions, émergences, équilibres, accroissement de la diversité ou auto-organisation) [de Rosnay, 1998].

[3]Définit le macroscope pour symboliser l'outil idéal permettant de mieux comprendre les systèmes complexes. Il l'oppose au microscope utilisé pour une analyse de plus en plus fine de l'infiniment petit et au télescope utilisé pour analyser l'infiniment grand. Le macroscope permet une analyse de l'infiniment complexe.

Approche Analytique	**Approche Systémique**
Se concentre sur les éléments (isole)	Se concentre sur les interactions entre les éléments (relie)
Considère la nature des interactions	Considère les effets des interactions
S'appuie sur la précision des détails	S'appuie sur la perception globale
Modifie une variable à la fois	Modifie des groupes de variables simultanément
Indépendance de la durée (phénomènes réversibles)	Intègre la durée (phénomènes irréversibles)
Modèles précis et détaillés difficilement utilisables dans l'action	Modèles insuffisamment rigoureux pour servir de base de connaissance, mais utilisables dans la décision et l'action
Approche efficace lorsque les interactions sont faibles et linéaires	Approche efficace lorsque les interactions sont fortes et non linéaires
Conduit à un enseignement par discipline	Conduit à un enseignement pluridisciplinaire
Conduit à une action programmée dans son détail	Conduit à une action par objectifs
La validation des faits se réalise par la preuve expérimentale dans le cadre d'une théorie	La validation des faits se réalise par comparaison du fonctionnement du modèle avec la réalité
Connaissance des détails et buts mal définis	Connaissance des buts et détails flous

Tab. *1.1* — Approche analytique vs. approche systémique

L'approche systémique, que nous retiendrons pour la suite de ce document, est l'étude des éléments et des comportements macroscopiques.

1.2.2 Types de modèles

Plusieurs formes de représentation des systèmes complexes peuvent être envisagées, au travers de leurs modèles. Par exemple, pour Greenberger les systèmes sont tout d'abord répertoriés selon deux approches [Greenberger et al., 1976] représentées en figure 1.3 : les modèles mentaux et les modèles formels. Les modèles mentaux sont des représentations informelles d'un système complexe qui ne sont pas exprimés sous la forme d'un langage formel. Pour Forrester [Forrester, 1980], les modèles mentaux sont difficiles à communiquer et ne peuvent pas être manipulés de façon efficace car l'esprit humain est incapable de considérer simultanément tous les aspects d'un système complexe. Les modèles formels se caractérisent suivant leur propre mode d'expression qui peut être schématique, physique ou symbolique ou de rôle. Les modèles schématiques représentent le système réel à l'aide de dessins, de points, de lignes, de courbes ou de graphiques. Les modèles physiques représentent certains aspects des systèmes réels au moyen d'analogies physiques. Les modèles physiques sont construits en utilisant des matériaux tangibles. Les modèles de rôles (utilisés en particulier pour les simulations de jeux) représentent les systèmes réels en attribuant des rôles

à des personnes. Les modèles symboliques peuvent êtres verbaux, mathématiques ou informatiques. Les modèles verbaux représentent les systèmes réels au moyen de narrations écrites ou orales. Les modèles mathématiques représentent les systèmes réels en s'appuyant sur des équations mathématiques. Ces modèles ont souvent l'avantage d'être précis, concis et faciles à manipuler et en principe non ambigus. En dépit de ces avantages, le recours aux représentations mathématiques est limité par le degré trop restreint de maîtrise de la symbolique mathématique. Les modèles informatiques représentent les systèmes réels en recourant aux symboles des langages informatiques. Ceux-ci servent à formuler un algorithme, c'est-à-dire un ensemble de règles qui définissent une séquence d'opérations (i.e. un ensemble d'instructions données à l'ordinateur) représentant un système réel.

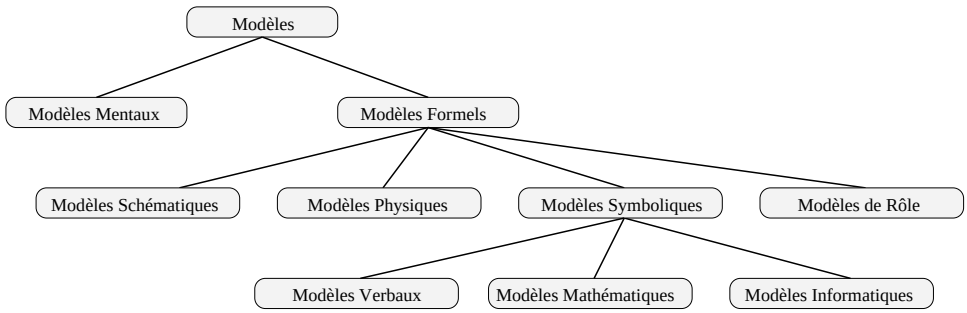


Fig. ***1.3*** — Formes d'expression des modèles

Ernest Page classe-lui, les modèles selon quatre dimensions orthogonales [Page, 1994]. La première dimension caractérise la représentation du modèle dans lequel celui-ci est abstrait ou physique. Le modèle abstrait peut être par exemple verbal ou mathématique et le modèle physique est une réplique en général à échelle réduite du système représenté. La deuxième dimension de cette classification caractérise l'objectif sous-jacent du modèle. Le modèle peut être descriptif auquel cas il décrit le comportement d'un système sans jugement de valeur sur la qualité du comportement. Un modèle normatif lui décrit le comportement d'un système en termes de qualité du comportement et dans lequel un jugement de valeur est apporté. La troisième dimension décrit la présence ou non d'une nature temporelle. Le modèle est ainsi statique ou dynamique. Finalement la quatrième dimension décrit la solution technique de résolution. Un modèle analytique fournit une solution en utilisant des méthodes formelles (e.g. déductions mathématiques). Un modèle numérique (relatif à l'approche systémique) fournit une solution par l'application de procédures informatiques.

1.2.3 Démarche de modélisation

L'étude d'un système complexe, par sa transformation en un modèle, peut être abordée de différentes manières en fonction de ses propriétés, de sa complexité, du type de modèle retenu, des besoins et des techniques disponibles. Toutefois, il est concevable de définir une démarche de modélisation dans laquelle un certain nombre d'étapes, pour la construction d'un modèle, sont incontournables. [Landry et Santerre, 1999] présente un exemple de démarche de modélisation constituée de cinq étapes comme l'illustre la figure 1.4.

Première étape : Elle consiste à identifier un problème en situant le système complexe par rapport à son environnement extérieur et ses objectifs de modélisation. Dans cette étape de spécification il convient également de préciser de manière informelle quels sont les éléments

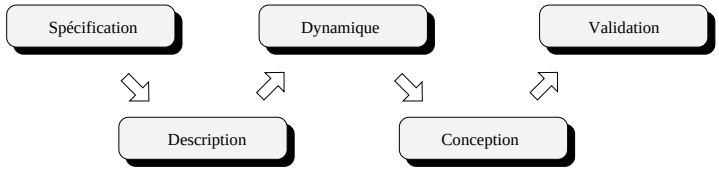


Fig. *1.4* — Démarche de modélisation

qui vont êtres modélisés et leurs fonctionnalités. Dans une approche systémique la prise en compte de la durée implique de définir la période de temps pour laquelle on désire étudier l'évolution du problème. Lorsque Le Moigne [le Moigne, 1999] postule que les projets de la modélisation d'un système complexe ne sont pas donnés, mais qu'ils se construisent, il fait référence à cette étape qu'il considère comme la plus importante. Pour cet auteur, la tâche la plus importante pour un modélisateur n'est pas de résoudre un problème présumé déjà bien posé mais de formuler le problème pertinent de résoudre : *il faut apprendre à résoudre le problème qui consiste à poser le problème*. Dans cette optique, la première tâche à accomplir dans une démarche de modélisation est donc de déterminer ce que l'on veut expliquer.

Deuxième étape : Elle consiste à mettre en relief le système étudié en dressant un inventaire formel de l'aspect fonctionnel et structurel. L'objectif d'un modèle est avant tout de promouvoir la compréhension du phénomène étudié par rapport à un objectif donné, toutefois dans la mesure où celui est soumis à une simplification de la réalité il y a donc une différence, acceptable ou non, entre le niveau de détail réel et celui décrit dans le modèle. Si trop peu de détails sont inclus, le modèle risque de ne pas fournir de comportements émergents satisfaisants pour la compréhension du système complexe. Si trop de détails sont inclus, le modèle peut devenir trop compliqué à formaliser, à implémenter et à analyser dans l'état actuel des connaissances. Nous définissons la calibration par la sélection d'un niveau de granularité approprié, mais aussi des limites macroscopiques et microscopiques. Il est donc parfois difficile de définir de façon absolue un niveau approprié de description structurelle et fonctionnelle. Le bon niveau pourrait être défini comme le niveau pertinent pour l'évaluation (i.e. en fonction des besoins de l'application), c'est à dire que le niveau de finesse approprié pour une fonction serait celui auquel on soit capable de définir un indicateur d'évaluation de son efficacité cohérent avec le niveau de préoccupation de l'évaluateur.

En modélisation on distingue essentiellement les approches *top-down* (descendante) et *bottom-up* (ascendante) comme méthodes avec lesquelles l'organisation hiérarchique interne d'un système complexe est décrite. Selon [Edmonds, 1999] une modélisation *top-down* est une approche dans laquelle on tente de définir l'organisation d'un système en formulant d'abord les principes généraux pour aller vers le détail. Inversement une modélisation *bottom-up* consiste à aborder la modélisation par les détails pour en produire les abstractions ou généralisations appropriées (typique d'une approche analytique). En fonction du domaine et de la connaissance de la structure et de la fonction d'un système (la vision est souvent parcellaire au niveau macroscopique), il peut parfois être difficile d'établir des principes généraux. Par exemple, s'il est possible de définir une fourmi, ses principaux comportements et ses interactions au sein d'une fourmilière, il n'en est pas de même de la fourmilière, l'approche *bottom-up* est alors la plus appropriée.

Troisième étape : Elle consiste à utiliser les informations recueilles lors des deux étapes précédentes pour déterminer la dynamique du système. [Landry et Santerre, 1999] choisissent dans cette étape d'élaborer un diagramme causal qui repose sur la caractérisation des boucles de rétroaction du système. Les méthodes d'élaboration des diagrammes causals en forme de boucles de rétroaction sont présentées dans [Richardson, 1991].

Quatrième étape : Cette étape porte sur le choix d'une méthodologie de modélisation. Les méthodologies fournissent des outils et proposent des procédures pour transformer, par exemple, le diagramme causal de la troisième étape en un modèle formel. Le choix d'une méthodologie influence grandement la façon de représenter le système réel. Les critères de choix d'une méthodologie pour concevoir un modèle dépendent des caractéristiques du système complexe étudié et de la perception qu'en a le modélisateur. Remarquons par ailleurs que dans un objectif de simulation informatique, l'architecture du système informatique peut influencer un choix méthodologique particulier, par exemple une approche orientée objet.

Paul Fishwick, dans [Fishwick, 1995] présente en se basant sur une méthodologie de modélisation par une approche orientée objet, plusieurs façon de concevoir les modèles. Il définit le modèle conceptuel qui représente le système complexe à un très haut niveau d'abstraction[4]. Ce modèle conceptuel peut être conçu de différentes manières : (1) le modèle déclaratif dans lequel la conception se concentre sur la forme des états courants du système et sur l'état suivant du système auquel une transition depuis ces états courants conduit. Cette conception du modèle est réalisée sans accentuer sur les fonctions qui causent le changement, (2) le modèle fonctionnel, en opposition au modèle précédent, focalise sur les fonctions qui transforment les entrées du système en ses sorties, (3) le modèle par contraintes dans lequel le système est vu comme une collection de variables d'état interconnectées entre elles par un réseau de contraintes, (4) le modèle spatial se concentre sur la géométrie du système en divisant l'espace en sous-parties plus petites et interactives. En dernier lieu Fishwick définit le multi-modèle comme une hiérarchie ou un réseau de modèles (modèle hybride basé sur les quatre modèles précédents). Nous considérerons dans la suite des méthodologies de modélisation produisant des modèles pour la simulation informatique (non nécessairement un modèle informatique qui est dépendant d'un langage particulier) et présenterons quelques-uns de ces outils en section 1.3.2.

Cinquième étape : Cette dernière étape du processus de modélisation a pour but de confronter le modèle à la réalité par vérification et validation. La validation dans une approche systémique se réalise par comparaison du fonctionnement du modèle avec la réalité. Il n'existe pas de procédure unique et formelle de validation, ainsi comme nous le verrons en section 1.4, la validation d'un modèle renvoie au processus suivant lequel on établit le degré de confiance qui est accordé au modèle lorsque utilisé dans certaines conditions et pour certains objectifs.

Un modèle d'un système complexe ne présente un intérêt que s'il est expérimenté. Un modèle représente un système réel ou fictif tandis que la simulation imite le système en reproduisant pas à pas son comportement. La simulation est donc un processus de résolution pas à pas dans lequel il est possible de définir le pas temporel suivant mais qui ne précise pas comment aller directement vers n'importe quel instant futur [Landry et Santerre, 1999]. La capacité de manipuler la complexité, c'est-à-dire un grand nombre de variables liées de façon si inextricable que les causes et les effets soient indissociables, implique le recourt aux modèles pour la simulation informatique abordant les problèmes d'un point de vue systémique. Ce

[4]En opposition au modèle informatique qui est lui une réalisation concrète au lieu d'être une idée réalisable.

processus de résolution se distingue des solutions analytiques qui décrivent l'état du système complexe un instant futur quelconque plutôt que de façon séquentielle. Dans la section suivante, nous présentons les principes de la simulation informatique.

1.3 Simulation de systèmes complexes

La simulation est une technique qui tend à reproduire le comportement d'un système complexe. Opérant à partir d'un modèle, la simulation peut être définie par la manipulation dans le temps et parfois dans l'espace des éléments du modèle afin d'en observer l'évolution, les interactions internes et externes. D'un point de vue informatique, la simulation se met en œuvre à partir d'une implémentation correspondant à l'informatisation, dans un langage quelconque, d'un modèle de simulation (opération de laquelle résulte un modèle informatique). D'après [Shannon, 1975] la simulation informatique est le processus de conception du modèle d'un système réel et la conduite d'expériences au travers de ce modèle à l'aide d'un ordinateur pour un objectif précis d'expérimentation. Lors de l'exécution, la simulation est conditionnée par l'utilisation de paramètres et de données à traiter et qui en constituent le point d'entrée, et ce, afin de produire des résultats de sortie. Lorsque cette simulation est exécutée sur une machine unique (modèle von Neumann classique), la simulation est dite séquentielle.

La simulation informatique, comme toute autre méthode de résolution et d'analyse, possède à la fois des avantages et des inconvénients. L'argumentaire classique présente fréquemment la simulation comme une bonne alternative pour l'étude de certains systèmes comportant des risques (étude du comportement d'une centrale nucléaire lorsqu'un des noyaux explose), ou dont l'étude serait trop coûteuse en temps ou en argent (test aérodynamique d'avion) ou encore humainement trop compliquée (calculs météorologiques). L'un des avantages immédiat de la simulation est que la manipulation systématique des entrées d'un programme de simulation permet l'évaluation de différentes alternatives relatives au comportement du phénomène traité. L'élément important d'une série de scénarios est alors la capacité de définir l'ensemble des alternatives désirées (en fonction des objectifs postulés lors de l'étape de modélisation) parmi l'ensemble des exécutions possibles. La technique de simulation offre ainsi la possibilité, à partir d'un modèle défini, de répéter autant que nécessaire des expérimentations avec des paramètres différents sans autre coût que le temps d'exécution et d'analyse des résultats. Notons cependant que la simulation n'est pas précise par nature (relève d'une approche systémique sous-jacente, [Ferber, 1995] parle de carence qualitative de la simulation numérique) et que par conséquent il est concevable que dans un ensemble de scénarios exécutés, aucune solution optimale ne puisse être identifiée (en admettant que cela soit un des objectifs). Dans ce contexte, des solutions générales s'obtiennent fréquemment par induction à partir des résultats numériques produits par chaque scénario.

La reproduction par la simulation informatique de phénomènes réels, lesquels intègrent une dimension temporelle, implique la prise en compte de cette dimension temporelle comme étant un élément fondamental de l'exécution. Cette composante temporelle appelée temps virtuel ou temps simulé n'est pas en règle générale lié au temps physique. Cette absence potentielle de contrainte vis-à-vis du temps réel dans lequel évolue le système complexe étudié permet au programme de simulation de faire évoluer le temps simulé avec une vitesse appropriée à la fois à l'observation du comportement, et à la fois adaptée aux capacités de la simulation numérique. Ainsi, un autre avantage de la simulation est sa capacité de paramétrer le comportement d'un système complexe dans le temps et dans l'espace par un mécanisme de compression temporelle permettant de percevoir une évolution très longue dans un intervalle

de temps très petit (éventuellement à l'inverse par une expansion temporelle, lorsque le temps réel d'évolution est très rapide, e.g. simulation des trajectoires suivies par des ions au milieu de l'électro-aimant d'un cyclotron).

1.3.1 Classes de simulation

La considération des nombreuses et des différentes caractéristiques des systèmes entraîne différents types de classification des simulations. En effet, de manière générale, les classes de simulation se réfèrent au type du système (ou modèle) simulé. Une simulation peut ainsi être statique, stochastique, dynamique, déterministe, etc. Toutefois, il existe une façon de classer les types de simulation de manière exclusive en considérant l'évolution du système représenté par rapport au temps. Ainsi comme le précise [Aiello, 1997], il existe deux principaux types de systèmes :

- Les systèmes statiques dont l'état ne dépend que des paramètres courants. De tels systèmes réagissent instantanément. Ils ne possèdent pas de mémoire puisque le passé n'influence pas l'état présent, ils sont atemporels ;
- Les systèmes dynamiques dont l'état à un instant donné dépend non seulement des paramètres courants mais également des états passés. Les évolutions temporelles de ces systèmes dynamiques peuvent s'opérer de deux manières : (1) de façon continue, auquel cas les variations d'état interviennent sans interruption dans le temps, ou (2) de façon discrète impliquant ainsi des changements d'états se produisant de manière discontinue ou ponctuelle.

Adhérant à cette dualité, comportement statique ou comportement dynamique, Ernest Page postule que la simulation informatique peut être divisée en trois catégories en fonction de l'évolution des états simulés [Page, 1994]. Notons que dans cette évolution, définie par une transition des états, on appelle événement le changement d'état du système provoqué par l'apparition de données, de messages externes ou, résultant des traitements internes :

- La simulation de type Monte Carlo est une méthode dans laquelle un problème est résolu par un processus stochastique et dans laquelle une représentation explicite du temps n'est pas nécessaire (simulation statique) ;
- La simulation continue reproduit le comportement de systèmes dynamiques dans lesquels les variables d'états évoluent sans interruption dans le temps. La simulation continue est une technique qui en théorie ne peut être exécutée que par une machine analogique [Fishwick, 1995]. L'utilisation d'un ordinateur numérique tel que nous le connaissons ne peut qu'approcher une simulation continue en considérant des intervalles de temps entre deux événements suffisamment petits pour permettre de ne pas prendre en compte la notion de transition existante entre deux événements. On parle alors de simulation pseudo-continue ou quasi-continue ;
- La simulation à événements discrets (*Discrete Event Simulation : DES*) reproduit le comportement de systèmes dynamiques dans lesquels les variables d'états changent à des instants précis dans le temps. Le comportement du système, dans cette approche de simulation, est ainsi décrit par une suite de changements d'état où les événements dont on simule l'évolution sont représentés sous la forme d'ensembles dénombrables, modélisés de façon discrète à des unités de temps régulières ou irrégulières dictées par la nature du système. Dans ce type de simulation le temps utilisé pour dater les événements produits évolue continûment, toutefois l'état du système ne change que par sauts.

La figure 1.5 issue de [Kim, 2001] représente une taxonomie des simulations de systèmes dynamiques en fonction de l'aspect continu ou discret des états et du temps. Le cas A de cette figure correspond aux systèmes et à la simulation continue telle que présentée ci-dessus. Cette classe est caractérisée par l'utilisation d'équations différentielles et d'un point de vue calculatoire est traditionnellement exécutée par des circuits ou des calculateurs analogiques (exécutée par un ordinateur, elle est rappelons le pseudo-continue). Le cas B représente des systèmes d'estampillage de données caractérisés par l'utilisation d'équations différentielles et dans lesquels le changement ou l'analyse des informations est mesuré par des séquences temporelles discrètes. Un exemple de cette classe de système est le traitement de signaux numériques (*Digital Signal processing* : *DSP*) où les signaux de nature analogique des systèmes simulés sont discrétisés. Le cas C implique des systèmes et des simulations numériques caractéristiques de celles opérées par des machines à état finis et mise en œuvre par des circuits numériques spécialisés. Finalement le cas D correspond aux systèmes à événements discrets tels que décrit dans le paragraphe précédent et détaillés dans la section 1.3.3. Ce type de simulation repose sur l'utilisation à la base de files d'attente et peut être exécuté par un ordinateur.

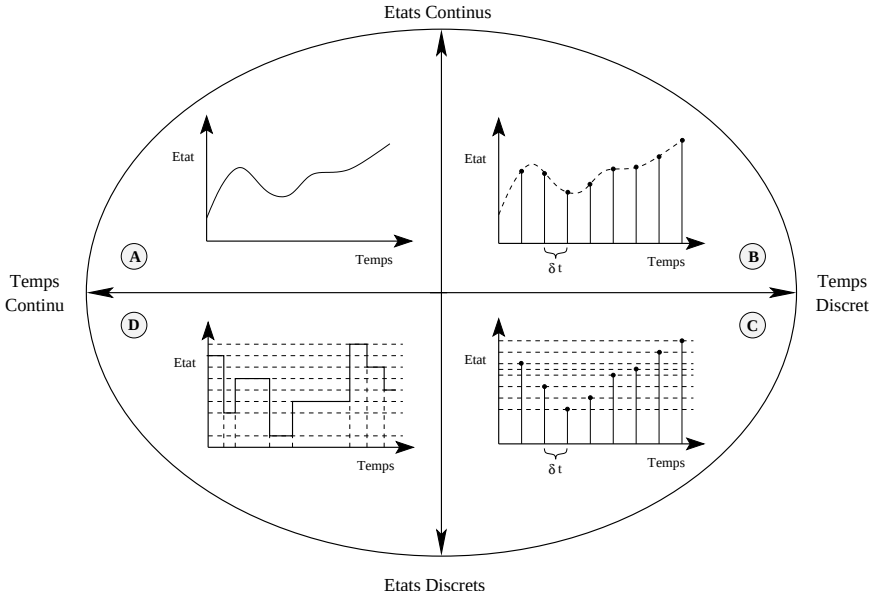


Fig. *1.5* — États et représentations continus vs. discret

L'élément essentiel caractérisant les simulations de systèmes complexes dynamiques, quel qu'en soit le type, est la conservation lors de l'exécution d'un ordre entre les événements tel qu'il se produit dans le système considéré. Les systèmes que nous considérons dans ce document sont dynamiques et modélisables par une description discrète des états. Ils respectent le principe de causalité vu en section 1.1.4 qui postule que un événement futur ne peut influencer un événement passé. Cette considération décrit en particulier les systèmes de notre monde réel dont Gottfried Wilhelm Leibniz (philosophe allemand, 1646-1716) disait à ce propos, *Natura non facit saltus*, la nature ne fait pas de sauts et qui sous-entend la nature continue des systèmes réels (impliquant de fait la nécessité d'une discrétisation pour une simulation informatique) et en particulier l'impossibilité pour ces systèmes de réaliser des sauts vers le passé.

1.3.2 Méthodologies pour la simulation informatique

De nombreuses méthodologies (quatrième étape de la démarche de modélisation, cf. section 1.2.3) peuvent être utilisées pour la conception d'un modèle destiné à la simulation informatique. Gilbert et Troitzsch présentent dans [Gilbert et Troitzsch, 1999] plusieurs de ces méthodes pour la simulation de sociétés humaines, dont, la dynamique des systèmes, les modèles à files d'attente, les automates cellulaires, ou encore l'approche agent.

La dynamique des systèmes [Forrester, 1980] constitue une méthodologie, développée à partir d'équations différentielles, qui est conçue pour la simulation des systèmes continus. La dynamique des systèmes tente d'expliquer le comportement des systèmes complexes à partir des actions conjuguées des parties du système (telles que décrite en section 1.1) en se basant sur un réseau de relations de cause à effet toutes caractérisées par des équations mathématiques (constituées de structures en boucles de rétroaction). Cette méthodologie fournit un ensemble de procédures de modélisation, des outils, et des langages informatiques tels que *DYNAMO* (développé au MIT dans les années soixante, ce fût l'un des premiers langages de simulation), *Stella* ou *Vensim*, qui sont maintenant utilisés pour modéliser et simuler un très large répertoire de problèmes.

En opposition à la dynamique des systèmes qui définit une méthodologie adaptée la simulation de systèmes continus, d'autres méthodologies forment une catégorie de modèles dont le dessein et de représenter des systèmes entièrement discrets ou discrétisés[5]. Cette approche de simulation par événements discrets est celle que nous retiendrons par la suite, car celle-ci peut être considérée comme le dénominateur commun à toute simulation sur un support informatique numérique. Ce type de simulation, qui repose à l'origine sur la théorie des files d'attente, peut ensuite être combiné avec différentes méthodologies de modélisation (agents ou objets, graphes, automates cellulaires, etc.) afin de satisfaire aux besoins et désirs de représentation d'un système complexe.

Les modèles de files d'attente (*queuing model*) souvent appelés modèles à événements discrets représentent un système en fonction de ses entités, propriétés, ensembles d'événements, activités et délais [Kheir, 1996]. Cette méthodologie procède, dans son utilisation originelle, par événements successifs. Les événements du modèle sont programmés dans une liste qui contient tous les événements futurs à simuler. Les événements passés, une fois simulés, sont retirés de cette liste mais peuvent produire de nouveaux éléments dans la liste. Dans une approche classique on utilise une liste triée par le temps croissant et on consomme en tête de liste. Par ailleurs, une source génère des événements qui sont distribués par un serveur (alimente la file) et consommés par un client. Remarquons que dans ce type de méthodologie, aucune action ne se produit entre deux événements.

Bernard Zeigler a développé, dans les années soixante dix, une méthodologie portant sur la modélisation et la simulation de systèmes à événements discrets [Zeigler, 1976, Zeigler et al., 2000]. *Discrete Event System Specification* (*DEVS*) a été introduit comme un formalisme abstrait et universel pour la modélisation à événements discrets. La théorie de modélisation et de simulation présentée par cet auteur est une méthodologie formelle pour construire des modèles en utilisant une approche hiérarchique et modulaire (approche alors similaire aux concepts de programmation orientée objet). Le développeur, par l'utilisation de ce paradigme, construit un modèle de base (modèle atomique) permettant la réutilisation des modèles ayant été validés pour former des modèles couplés. Un modèle couplé définit comment est composé un ensemble de modèles entre eux (atomiques ou couplés) pour former

[5]Lorsqu'il est possible de remplacer un continuum espace-temps par un échantillonnage de points discrets répartis, à priori, régulièrement dans le temps.

un nouveau modèle validé. Le formalisme *DEVS* intègre ainsi une construction hiérarchique par étapes successives des modèles et où chaque étape correspond à un niveau spécifique de la hiérarchie de description. La partie du formalisme *DEVS* qui traite de la simulation, expose comment générer un simulateur abstrait quel que soit le modèle auquel il se rapporte. Les transitions des états sont représentées par les transitions d'un automate d'états finis. Remarquons que cet auteur aborde également la simulation continue au travers de formalismes similaires (*DESS* et *DTSS* respectivement cas *A* et *B* de la figure 1.5).

Les automates cellulaires [von Neumann, 1951, von Neumann, 1966] introduits par John von Neumann (mathématicien hongrois, 1903-1957) décrivent l'espace sous la forme d'un réseau pouvant être représenté par un ensemble de sites reliés entre eux par un graphe de voisinage de portée unitaire. Dans leur forme classique, les automates cellulaires sont représentés par des grilles carrées où chaque case (site) est constituée de N cellules. Lorsque le nombre N de cellules est égal à un, alors l'automate est bidimensionnel. Chaque cellule peut être libre ou occupée et correspond à un emplacement possible pour un élément constitutif du système complexe simulé. Un des plus célèbres automate cellulaire est sans aucun doute *le jeu de la vie* de Conway mis en avant par [Gardner, 1970], qui semble avoir, pour partie, initié l'ère des simulations informatiques. L'évolution d'un automate cellulaire est traditionnellement fondée sur l'application de transformations locales, à partir d'un état initial. Ces transformations, appelées les règles de transition d'un automate, sont appliquées séquentiellement (par pas temporels successifs) dans un processus itératif. Tous les sites du réseau sont traités simultanément en appliquant le même ensemble de règles d'évolution (implique une uniformité de l'automate). Les automates cellulaires sont en général rapides d'exécution et l'accès direct aux sites voisins permet d'appliquer efficacement les règles locales.

Les systèmes multi-agents [Ferber, 1995] constituent une méthodologie de modélisation qui convient particulièrement à la simulation numérique de systèmes complexes. Les systèmes multi-agents sont caractérisés par un environnement comprenant un ensemble d'entités passives et un ensemble d'entités actives (agents) dont la définition et la décomposition en sous-ensembles organisés permet de reproduire les organisations hiérarchiques des systèmes complexes. Cette approche est basée sur la compréhension des entités du système qui, situées dans leur environnement dont elles ont vision partielle, sont dotées de comportements autonomes leur permettant d'atteindre des objectifs[6]. L'approche agent intègre la notion d'émergence et considère que l'action et l'interaction sont les éléments moteurs de la structuration d'un système dans son ensemble (rejoint en cela l'approche systémique).

La décentralisation des processus ou des connaissances au niveau d'éléments actifs est, pour [Gutknecht, 2001], une des caractéristiques fondamentales des modèles à agents. Cet auteur précise par ailleurs que la plupart des systèmes réels sont naturellement distribués, par décentralisation, réduction du couplage entre les éléments ou les composants, ou par le besoin de prise de décision locale. Remarquons alors que la méthodologie agent convient particulièrement à une exécution distribuée de la simulation. En effet, elle permet de reproduire les interactions, dont la complexité est intégrée dans chacun des composants, entre les différents niveaux d'abstraction en termes de perception et de communication. A ce propos [Ferber, 1995] précise que les systèmes multi-agents tentent d'appréhender le problème de l'interaction entre entités individualisées de manière générale et abstraite, considérant que les systèmes distribués ne sont qu'une application ou qu'une réalisation possible et particulièrement importante de ce concept général.

[6]Jacques Ferber présente l'autonomie pour un agent par un ensemble de tendances dont la forme principale est définie par des buts individuels à satisfaire ; le moteur de l'agent est lui-même. Cette autonomie le différencie, en partie, des concepts similaires tels que les processus ou les objets (du paradigme du même nom).

1.3.3 Simulation à événements discrets

La simulation à événements discrets [Ingalls, 2002, Banks, 2000] a pour objectif principal de calculer le plus rapidement possible l'ensemble des événements décrivant les actions d'un système complexe. Dans la mesure où les états discrets changent à des instants précis sans autre transition entre deux événements, il devient possible de traiter ces événements sans attendre les délais qui interviennent entre ceux-ci dans le temps réel. De nombreux travaux ont abordé la simulation selon cet angle pour des contextes d'étude variés. En particulier, différents travaux de thèse tel que par exemple [Page, 1994] ou [Aiello, 1997] présentent une méthodologie, voire un environnement général, de modélisation pour la simulation à événements discrets. [Maziero, 1994] présente Floria, noyau de système distribué utilisé pour la simulation à événements discrets et, [Beaumont, 1998] présente la simulation à événements discrets par l'utilisation de l'architecture matérielle ArMen. Par ailleurs, certains langages dédiés à ce type de simulation comme SimScript [Markowitz et al., 1962], GPSS [Greenberg, 1972] ou Simula [Birtwistle et al., 1973] ont été proposés. Ces langages font partie d'environnements de simulation qui proposent, en plus d'un langage, des mécanismes pour la gestion du temps simulé, de files et de listes (d'événements), et des facilités pour l'analyse statistique des résultats.

La simulation à événements discrets considère les systèmes (a priori complexes, mais non nécessairement) pour lesquels une modélisation discrète est possible. Dans ce schéma de simulation, la dynamique du modèle est représentée à un instant donné par un état. Le comportement au cours du temps est décrit par une séquence finie des états dont les transitions sont dues à l'occurrence d'événements, se produisant de façon discrète dans le temps et, mis en œuvre par la consommation d'événements contenus dans une ou plusieurs files d'attente. Deux formes d'expression peuvent décrire la simulation à événements discrets [Kreutzer, 1986, Ingels et Raynal, 1989] :

- Par une méthode basée sur la notion d'*événements* : chaque événement du système est décrit par une condition d'occurrence et par les actions à réaliser lors de son occurrence. Ces actions matérialisent le changement d'état provoqué par l'événement. Dans ce schéma de simulation, utilisé en particulier par *SimScript*, une liste globale des événements est parcourue.

- Par une méthode basée sur la notion de *processus* : chaque entité ou composant du modèle est représenté par un processus, qui exécute les actions matérialisant les changements d'états, l'évolution du temps simulé et les interactions avec les autres entités. Les interactions entre les processus peuvent s'effectuer soit par ressources partagées accédées par tous les processus (c'est le cas du langage *Simone* [Kaubisch et al., 1976]), soit par activation et désactivation directe des processus entre eux (langage *Simula* [Birtwistle et al., 1973]), soit par échange de messages (méthode utilisée par le langage *May* [Bagrodia et al., 1987]). Dans le mode de fonctionnement classique, chaque processus utilise une liste locale des événements qu'il a à traiter. La synchronisation des événements (i.e. le respect de la causalité) est alors assurée par un ordonnancement vis à vis de l'horloge physique de l'ordinateur. Lorsque le mode de fonctionnement exploite une liste globale, celle-ci est représentée par une variable partagée dont l'accès par les différents processus est synchronisé.

Remarquons que Chandy et Misra [Chandy et Misra, 1979] ou encore Peacock [Peacock et al., 1979] ont proposé relativement tôt de concevoir les modèles de simulation à

événements discrets par un ensemble de processus logiques[7]. En effet, de l'emploi du modèle original basé sur les files d'attente, au travers de processus, résulte une bonne capacité d'expression des systèmes grâce au morphisme des fonctions ou des structures inhérentes à celui-ci vers ces processus. L'utilisation de processus permet par ailleurs d'envisager une approche hybride combinant à la fois ce type de simulation et une simulation qui repose sur un modèle agent pouvant de manière naturelle représenter les hiérarchies de composants. Par ailleurs, cette approche peut être directement implémentée par les mécanismes connus de processus physiques (i.e. informatiques). Cette forme d'expression de la simulation à événements discrets permet ainsi une transposition aisée vers un contexte distribué (cf. chapitre 3). Cela fut la motivation principale des auteurs précédemment cités pour préconiser cette approche.

Les événements produits lors d'une simulation à événements discrets sont datés dans le temps de simulation. Ces événements sont par conséquent traités dans l'ordre de leur occurrence dans le temps simulé afin de garantir la causalité. Deux méthodes de traitement des événements sont traditionnellement employées : la simulation dirigée par le temps et la simulation dirigée par les événements.

La simulation dirigée par le temps (*time-driven*) est discrétisée par intervalles de temps unitaires et entretient une horloge globale représentant le temps simulé. Cette horloge avance par incréments fixes δ appelés pas qui sont de taille arbitraire mais constante comme l'illustre la figure 1.6. A chaque pas de l'horloge, le simulateur consulte la liste d'événements à la recherche d'événements ayant pour date d'occurrence la date courante de simulation T_{sim}. S'il possède un événement e_i à simuler à cette période, il exécute l'événement, autrement le simulateur passe au pas temporel suivant. Remarquons que dans T_{sim} les événements sont souvent considérés comme instantanés (à la date d'occurrence) alors que dans T_{phys} ceux-ci nécessitent un intervalle de temps pour être exécutés.

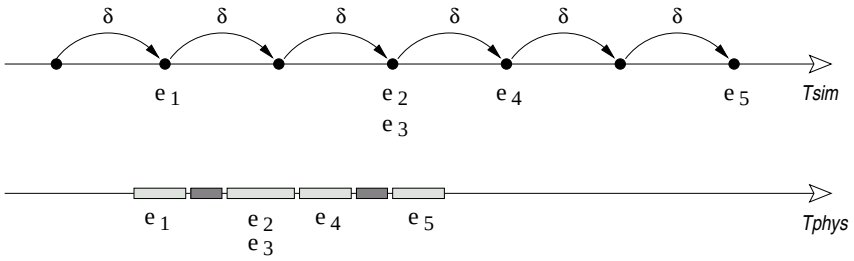


Fig. *1.6* — Simulation dirigée par le temps

La granularité temporelle (i.e. le choix du pas temporel δ) influence la précision de la simulation dirigée par le temps, notamment par la durée globale requise pour la simulation. Comme le montre la figure 1.6, tous les pas de la simulation sont exécutés, même si aucun événement n'est programmé. Le pas doit être faible pour garantir une bonne précision, toutefois, des pas courts impliquent une simulation plus longue car cela accroît la possibilité de présence d'instants où il ne se passe rien. A contrario, plus le pas est grand, plus il y a d'événements à simuler à chaque pas temporel, et moins la simulation est précise. Remarquons que le pas temporel peut être rendu variable au cours de la simulation afin de dilater ou de compresser plus encore le temps d'exécution.

[7]Chandy et Misra parlent de processus physiques représentant un système physique et de processus logiques représentant les processus informatiques. Dans la mesure où un système complexe n'est pas forcément physique et dans la mesure où le modèle n'est qu'une vision conceptuelle et logique du système complexe sans aspect physique tangible, il nous semble plus pertinent d'inverser la terminologie.

La deuxième méthode est la simulation dirigée par les événements (*event-driven*) qui dans le principe discrétise l'observation du système simulé au moment de l'occurrence des événements (cf. figure 1.7), c'est-à-dire que le temps simulé est incrémenté d'un événement à l'autre. Dans ce cas, le moteur de la simulation traite en continu les événements contenus dans la liste en choisissant à chaque fois le premier événement de la liste. Cette approche a l'avantage de simuler une évolution temporelle fixée plus rapidement que la méthode précédente car il n'existe pas de temps improductifs.

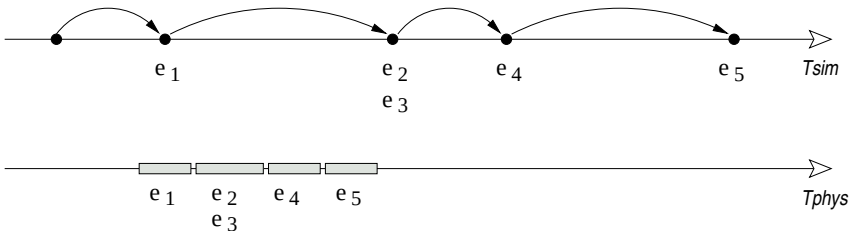


Fig. *1.7* — Simulation dirigée par les événements

La simulation dirigée par le temps possède un fort potentiel lorsqu'il y a beaucoup d'événements à simuler par incrément de temps. La simulation dirigée par les événements est par contre plus souple car toute action du simulateur fait en effet progresser la simulation, alors que dans l'approche dirigée par le temps ce n'est pas systématiquement le cas.

1.4 Vérification et validation

Une des étapes importantes de la simulation informatique des systèmes complexes est la capacité d'établir la vérification, la validité et la crédibilité de la modélisation et de la simulation. Lors de toute analyse du fonctionnement d'un système au travers de l'étude de la dynamique de son modèle, la calibration permet de mesurer les écarts entre la prévision et les résultats mesurés du phénomène étudié. Les ajustements issus de cette confrontation entre les comportements émergents identifiés après une simulation et ceux mesurés ou estimés ont pour objectif de conforter le degré de confiance dans les résultats et in-fine de valider le modèle et l'implémentation qui en dérive, ou a contrario de redimensionner et améliorer le modèle.

1.4.1 Validité des modèles

La validité est sémantiquement définie par la conformité d'un élément réel avec sa représentation. D'un point de vue de la modélisation, elle est caractérisée par le degré de correspondance entre le système étudié et le modèle conceptuel résultant. La conformité ou similitude entre l'original et sa représentation est dépendante du degré de calibration du modèle, lui-même fonction du niveau de finesse que l'on vise [Landry et Santerre, 1999]. Une des contraintes minimum à respecter pour assurer une correspondance valide est de vérifier que chaque élément du modèle a effectivement une contrepartie dans le système réel. Du fait des simplifications impliquées dans le modèle, notons que la réciproque n'est pas certaine. Si la validité d'un modèle est soumise aux omissions effectuées (souvent dans le but d'une meilleure compréhension du phénomène étudié ou car reflète une connaissance partielle du phénomène), la validité d'un modèle, peut également être conditionnée par l'intégration

volontaire *d'impuretés* telles qu'erreurs, défauts et imprécisions que l'on souhaite mettre en évidence lors de la simulation.

Pour [Banks, 2000] la validation consiste à déterminer si un modèle conceptuel est une représentation suffisamment précise du système réel. Une représentation suffisamment précise signifiant que le modèle peut être utilisé comme substitut du système réel pour les objectifs d'expérimentation et d'analyse préalablement définis. Similairement [Sargent, 2000] postule que la validité du modèle est définie par la détermination que les théories et les hypothèses sous-jacentes du modèle sont correctes et que la représentation du problème initial est raisonnable pour l'objectif de modélisation attendu. Remarquons que dans ces définitions, l'accent est mis sur la notion d'objectif du modèle. Un modèle est ainsi développé pour un objectif précis, sa validité étant alors déterminée en fonction de cet objectif.

Présentant la modélisation des systèmes d'importation de drogues aux États-Unis, [Shreckengost, 1985] aborde la validité par un aspect subjectif de confiance. La question est alors, compte tenu de la calibration effectuée, le modèle correspond t'il aux buts attendus ? est-il utile ? La notion validité (ou de confiance) d'un modèle dépend donc énormément des hypothèses simplificatrices émises lors de sa construction, de la compréhension du système représenté et surtout des objectifs poursuivis. Le concept de validité dans ce contexte constitue une échappatoire pour le concepteur puisque relevant plus de l'accroissement de la confiance de l'utilisateur plus que de la validité. [Carson, 2002] présente également la notion de confiance envers un modèle, à laquelle se réfère la crédibilité du modèle. De ce fait la validité proprement dite ne peut, elle, que reposer sur les spécifications faites pour concevoir le modèle.

1.4.2 Mesure de la validité

Le fait que le concept de validation soit lié aux objectifs de développement d'un modèle conduit forcément ce concept à une décision le plus souvent subjective. Dans ce schéma de pensée, trois approches sont utilisées pour définir la validité ou l'invalidité d'un modèle conceptuel. L'approche la plus fréquemment utilisée est soumise à l'appréciation du concepteur qui décide de la validité du modèle. Cette décision est basée sur un ensemble de tests et d'évaluations diverses du modèle selon différentes conditions. La deuxième approche consiste à faire appel à une tierce partie indépendante pour décider de la validité [Wood, 1986]. La troisième approche pour déterminer la validité consiste à établir un score [Gas et Joel, 1987]. Établi par différents processus de validation, le modèle par score reste subjectif pour établir la validité d'un modèle de simulation notamment à cause de la méthode définissant le modèle de score.

La validité d'un modèle conceptuel ou informatique, quelle soit subjective ou objective, est le facteur clé de la crédibilité du processus de modélisation, de simulation et d'interprétation des résultats. [Sargent, 2000] replace cette validité du modèle conceptuel dans un contexte plus général de validation et de vérification du processus de modélisation et de simulation en se reposant sur le schéma représenté en figure 1.8. Les données sont nécessaires pour la construction du modèle conceptuel, sa validation et pour conduire des expérimentations sur l'implémentation du modèle par le biais de la simulation. La validité des données assure que ces paramètres nécessaires sont adéquats ou du moins cohérents. La vérification du modèle informatique assure que la programmation informatique du modèle conceptuel produit un modèle informatique correct. La validité opérationnelle détermine si le comportement visible résultant de l'analyse dynamique du modèle informatique possède une précision suffisante en fonction des objectifs initiaux. La validité opérationnelle est difficile à certifier dans la mesure où l'objectif de la simulation est d'obtenir des informations supplémentaires (comportements

émergents) à partir des différents scénarios indéterministes. Elle consiste à étudier les résultats afin de les confronter aux objectifs postulés avant la modélisation.

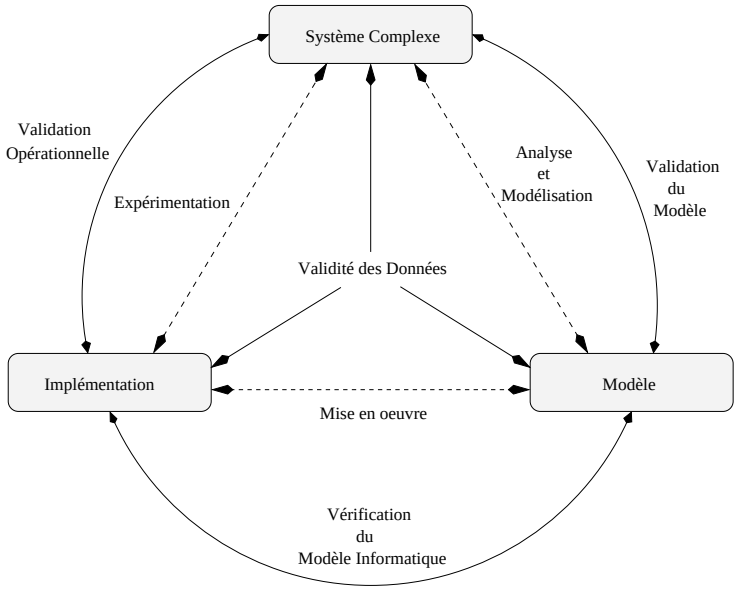


Fig. *1.8* — Validation du processus de modélisation et simulation

Dans ce schéma de conception et de mesure, la vérification intervient lorsque le concepteur expérimente un modèle apparemment correct pour les objectifs fixés. La validation intervient lorsque le concepteur ou une tierce personne, maîtrisant tout ou partie du phénomène représenté, révise et évalue le fonctionnement du modèle. Ces phases de vérification et de validation ont pour but de détecter des erreurs ou des hypothèses mal fondées nécessitant de fait un nouveau cycle de conception, validation, implémentation et vérification. Au cours de cette phase de débogage, différentes techniques de validation plus ou moins objectives peuvent être employées afin de vérifier ou valider chacune des étapes :

- L'animation consiste à étudier, à l'aide d'affichages graphiques, le comportement du modèle opérationnel dans le temps ;
- La comparaison de modèles à pour but de mettre en confrontation les résultats obtenus avec ceux produits par d'autres modèles ;
- La validité des événements consiste à comparer les événements générés avec ceux produits par le système réel afin de déterminer leur similarité ;
- La validité extrême est une façon de tester le modèle à l'aide de paramétrages extrêmes voire improbables afin vérifier que le comportement reste cohérent ;
- La validité visuelle est effectuée par une tierce personne connaissant et maîtrisant les tenants et les aboutissants du système étudié qui confirme alors un comportement correct ;
- La validité par valeurs fixes consiste à comparer des variantes du modèle avec un ensemble de paramètres (internes et externes) fixes ;
- La validité historique se vérifie par l'utilisation d'un sous-ensemble de données réelles collectées afin de construire le modèle. Le reste des données (en général le reste par ordre chronologique) est comparé avec les résultats produits par l'exécution du modèle ;

- La validation par graphiques opérationnels consiste à utiliser un ensemble de capteurs et de mesures de performances et à les reporter graphiquement afin de connaître précisément la valeur des états ;
- La sensibilité des paramètres est utilisée pour faire varier les paramètres du modèle afin d'en évaluer l'effet sur le modèle ;
- La validation prédictive consiste à réaliser une prédiction sur l'évolution du système à l'aide du modèle. Cette prédiction est ensuite comparée et vérifiée et potentiellement validée.

L'objectif pour lequel un modèle conceptuel puis sa représentation informatique sont construits est l'analyse en général prédictive du comportement du système. Au final la mesure et le jugement du cycle de conception et de développement sont réalisés sur les résultats concrets produits lors de l'étape d'expérimentation. La validation opérationnelle consiste comme nous l'avons vu à étudier et bien évidement à valider ces résultats opérationnels (obtenus par l'exécution du modèle informatique représentant le système considéré). La plupart des techniques de validation et de mesures (e.g. validation historique) peuvent être appliquées pour déterminer la validité opérationnelle. Toutefois, dans cette étape l'attribut majeur affectant la prise de décision concernant la validité est conditionné par le caractère observable ou non du système. Ceci signifie qu'il est possible, ou non, de collecter dans le temps des données et des informations sur le comportement opérationnel. La table 1.2 présente une classification de la validité opérationnelle en fonction de ce paramètre.

	Système Observable	Système Non Observable
Approche Subjective	• Comparaison entre les résultats du modèle informatique et ceux du système à l'aide d'interfaces graphiques • Exploration du comportement du modèle informatique en utilisant les techniques de validation adaptées	• Exploration du comportement du modèle informatique en utilisant les techniques de validation adaptées • Comparaison des résultats du modèle informatique à ceux produits par d'autres modèles informatiques
Approche Objective	• Comparaison entre les résultats du modèle informatique et ceux du système à l'aide de tests statistiques	• Comparaison des résultats du modèle informatique à ceux produits par d'autres modèles en utilisant des tests statistiques

Tab. *1.2* — Classification de la validité opérationnelle

1.5 Conclusion

Ce chapitre propose, en section 1.1, une description des systèmes complexes et identifie en particulier quatre concepts fondamentaux : l'organisation, l'interaction, la globalité et la complexité. Nous considérons un système complexe comme définissant un environnement réel ou fictif composé d'un ensemble d'éléments en relation, potentiellement distincts en structure, en comportement et dans le temps. La composition (ou union) des éléments qualifiés de microscopiques ou de désagrégés, de leurs comportements et surtout des interactions inter-

venant entre eux permettent de faire émerger de nouvelles propriétés à un niveau global dit agrégé (macro).

La modélisation, présentée en section 1.2, est l'étape nécessaire dans laquelle sont décrites les entités structurelles et fonctionnelles d'un système et leur dynamique. De nombreux types de modèles peuvent être envisagés et les méthodologies de conception de modèles sont très diverses. L'identification des étapes élémentaires de la modélisation et de certaines méthodologies clairement adaptées pour la simulation informatique permet d'effectuer des choix de représentation judicieux en fonction des objectifs de modélisation et de simulation d'un système complexe.

La simulation informatique reproduit le comportement des systèmes complexes dont la dynamique temporelle est difficile à décrire par une résolution analytique. Dans la section 1.3 nous avons remarqué que le principal type de simulation numérique est la simulation à événement discret, qui peut dans le cas de pas temporels fins être assimilée à une simulation continue. La nature discontinue de l'occurrence des événements dans une simulation discrète permet d'en accélérer l'exécution en supprimant les délais entre deux transitions d'état.

En section 1.4, nous avons présenté les principes de validation et de vérification appliquée dans un contexte de simulation informatique de systèmes complexes. La nature encore informelle (ou du moins subjective) des mécanismes de validité du processus de modélisation et de simulation en fait une des directions stratégiques pour l'avenir des techniques de simulation [Page et al., 1999].

Dans le chapitre suivant, nous présentons un ensemble de systèmes pouvant être considérés comme des systèmes complexes et dans lesquels interviennent *de large flux de données désagrégées*. L'exemple majeur de cette classe de système, que nous considérerons en fil conducteur de notre discours sont les systèmes sociétaux ou urbains constitués de composants spatiaux distincts au sein desquels se déplacent des entités (e.g. personnes ou véhicules), à titre individuel ou guidé par un phénomène de groupe.

CHAPITRE 2

Systèmes urbains à flux de données désagrégées

« Une ville ressemble à un animal. Elle possède un système nerveux, une tête, des épaules et des pieds. Chaque ville diffère de toutes les autres : il n'y en a pas deux semblables. Et une ville a des émotions d'ensemble. John Steinbeck, Nobel de littérature 1964 »

La définition d'un système présentée par Ludwig von Bertalanffy dans *The General System Theory*, et dans lequel il explique les nombreuses et les complexes interactions qui caractérisent les technopoles modernes et qui en influencent l'organisation technologique et sociale [von Bertalanffy, 1973], fournit un cadre de réflexion pour l'analyse de systèmes sociétaux. Cet auteur précise[1] qu'il existe des interrelations entre tous les éléments et les constituants des sociétés. De ce fait, les facteurs essentiels des problèmes, politiques et programmes publics doivent toujours être pris en compte et évalués en tant que composants interdépendants d'un système total. L'approche systémique, selon les propres termes de Ludwig von Bertalanffy, constitue une philosophie de la nature dans laquelle l'identification et l'analyse des éléments ne suffisent pas pour comprendre une totalité. Il est alors nécessaire d'étudier les relations existantes entre les composants du système. Dans les systèmes sociétaux et urbains, une des relations importantes pour en comprendre le fonctionnement correspond aux flux migratoires de personnes entre des entités spatialement distinctes qui représentent des portions du système urbain considéré. Ce chapitre aborde ce type de relation, dont la notion de système à larges flux de données désagrégés en est une généralisation. La section 2.1 présente un ensemble de définitions et de propriétés des systèmes à larges flux de données désagrégés. La section 2.2 présente un bref état de l'art de cette classe de systèmes. Cette section présente notamment les systèmes en planification en transport et en gestion de trafic qui sont des applications importantes du point de vue de la gestion de la cité, et significatives du point de vue de la modélisation en raison des variabilités des échelles spatiales et des granularités temporelles mises en œuvre. Ces domaines d'applications donnent respectivement de longues périodes temporelles et des échelles spatiales macroscopiques pour les études de transport (à l'échelle de la ville), et quasiment du temps réel et de grandes échelles spatiales pour de la gestion ou de la simulation de trafic en milieu urbain. La section 2.3 présente un ensemble de directions stratégiques, tant sur le plan conceptuel que stratégique, pour la simulation de systèmes complexes dans lesquels interviennent des flux de données désagrégées. Finalement, la section 2.4 conclut ce chapitre.

[1] Se référant à un discours tenu par E.C. Manning, premier ministre de l'Alberta, Canada.

2.1 Systèmes à flux de données désagrégées

La notion de relation dans un système complexe est caractérisée par l'échange d'informations entre les éléments du système (i.e. des interactions), et ce, quel que soit leur niveau dans l'organisation hiérarchique interne. Nous abordons dans cette section l'analyse de systèmes urbains et sociétaux, comme le préconise une approche systémique, par l'étude des interactions. Cette démarche adaptée pour l'analyse de systèmes complexes prend tout son sens lorsque le centre d'intérêt est l'étude des flux de données au sein d'un système. Dans ce cas l'étude des interactions, représentant l'échange possible d'éléments eux-mêmes du modèle, est un élément fondateur de la compréhension des comportements.

Un objectif important de la modélisation et de la simulation est la caractérisation des entités du monde réel par un ensemble d'objets abstraits pouvant être utilisés pour simuler le comportement de systèmes complexes [Zeigler et al., 2000]. L'aspect significatif de cette modélisation par un ensemble d'objets est la création de représentations artificielles à différents niveaux de simplification du monde réel. Ces modélisations définissent des cadres conceptuels pour développer des représentations dynamiques à un niveau de simulation agrégée ou désagrégée (macro- vs. micro-simulation) en fonction des objectifs établis par le concepteur pour l'étude d'un système complexe.

La modélisation des aspects fonctionnels et structurels d'un système complexe implique une caractérisation hiérarchique fonctionnelle et/ou structurelle des éléments composant son environnement. Dans cette modélisation hiérarchique, les propriétés de chaque niveau de modélisation sont dérivées de celles des niveaux de granularité plus fins [Minsky, 1986]. Selon [Miller, 2002] cette modélisation hiérarchique représente une abstraction puissante et programmable pour traiter des problèmes complexes et interconnectés. Chaque niveau d'une hiérarchie donnée possède sa propre et sa spécifique granularité temporelle et échelle spatiale. Dans le mécanisme d'analyse, plus haut est le niveau de représentation, plus petite est l'échelle spatiale et plus lente est la vitesse d'évolution.

C'est dans ce contexte de description d'une organisation hiérarchique interne des objets qu'intervient la distinction entre les approches agrégées et désagrégées. Nous constatons au travers de différentes thématiques de recherche que la terminologie micro/macro correspond à celle de désagrégée/agrégée. Le modèle désagrégé se concentre sur les éléments microscopiques alors qu'un modèle agrégé est basé sur des entités et des propriétés identifiées à un plus haut niveau d'abstraction (éléments macroscopiques). Pour qualifier le degré maximum de désagrégation (i.e. le plus bas niveau désagrégé) d'un élément, [Frihida, 2002] utilise la terminologie d'élément atomique, c'est-à-dire d'élément qui ne peut être décomposé. [Treuil et al., 2002] utilise la terminologie micro-analytique et postule que cette approche consiste à appréhender les entités élémentaires d'un certain niveau d'organisation, pour en décrire les interactions, et recomposer à partir d'elles les dynamiques de niveaux d'organisation supérieurs.

[McGregor et al., 1998] note qu'aucun de ces deux paradigmes ne fournit le meilleur concept d'étude dans la mesure où aucune de ces deux approches n'est appropriée à toutes les classes d'applications. Elles doivent simplement être en accord avec les objectifs postulés par le concepteur du modèle et de la simulation. Parmi les différents avantages de l'approche agrégée, l'un des plus importants repose sur le fait que la calibration des modèles macroscopiques est plus aisée que lors d'une étude plus fine des organisations hiérarchiques. En effet, selon [Page et al., 1999], un des problèmes fondamentaux est que les modélisateurs ne sont capables de gérer qu'un degré de complexité limité lors du développement ou de l'étude d'un

modèle. Par ailleurs, d'un point de vue calculatoire, les paramètres d'entrée de la simulation sont réduits, plus simples, et permettent l'étude de systèmes à plus grande échelle tout en impliquant des temps de calcul plus faibles. D'un autre point de vue, l'avantage principal des approches désagrégées par rapport aux approches agrégées repose sur le fait qu'elles reproduisent le comportement du système représenté à un plus fin niveau de granularité, permettant en général (bien que non systématiquement) une meilleure modélisation et étude de ce système [Thériault et al., 2002, Clarke et Holm, 1987, Goodchild, 1998, Miller, 1998]. La contrepartie est la difficulté méthodologique de modéliser des systèmes désagrégés et l'augmentation nécessaire des ressources de calculs rendues indispensables à l'exécution de simulations plus complexes. Plusieurs systèmes naturels et artificiels ont été analysés par l'utilisation des principes de modélisation désagrégée, citons par exemple les domaines de l'économie, de la biologie ou de la psychologie cognitive. Concernant les systèmes urbains, de nombreuses études récentes privilégient également cette approche pour mieux caractériser et comprendre les phénomènes sous-jacents des sociétés humaines [Miller, 2002, Buliung et al., 2002, Khan et al., 2002].

Nous définissons dans ce document un système à larges flux de données désagrégées par un système complexe dans lequel le niveau de désagrégation est relativement fin au regard de l'application considérée. Le terme *larges flux de données désagrégées* signifie qu'un tel système a la capacité d'échanger, entre les éléments d'un niveau agrégé (meso ou macro), des éléments d'un niveau plus fin dans la hiérarchie (micro). Ces éléments sont alors qualifiés de données désagrégées. Leur libre circulation dans la structure interne du système complexe forme un flux de données désagrégées qui représente les interactions entre les composants.

2.2 Simulation de systèmes urbains

Les systèmes complexes qui modélisent des applications à larges flux de données désagrégées sont de natures très diverses. Parmi les différents domaines qui tendent vers une approche désagrégée, la modélisation et la simulation de systèmes urbains et sociétaux est une classe d'application importante pour le développement harmonieux et la planification des villes, et dont le développement a été récemment accéléré par les progrès constants des systèmes informatiques. Une approche de modélisation et de simulation continue de tels systèmes implique l'utilisation d'équations différentielles pour caractériser les flux (ces équations sont ensuite discrétisées pour la simulation). En opposition, une approche de modélisation et de simulation discrète apporte des éléments d'analyse pour deux importants types de phénomènes de flux se produisant dans les systèmes urbains. Ceux-ci sont les mobilités de personnes et les déplacements de véhicules (d'autres types d'échanges se produisent bien évidement dans les villes mais nous limiterons la portée de notre étude à ces deux cas). En effet, ces flux sont difficilement caractérisables par des équations de débits, celles-ci ne prenant pas en compte à la fois les aspects sociaux, aléatoires, et les mouvements parfois contradictoires d'entités mobiles.

Les systèmes urbains sont parmi les systèmes dont la complexité, la structure, l'évolution et les comportements sont les plus difficiles à analyser et à comprendre [Claramunt et Thériault, 2001]. La dynamique de ces systèmes urbains et des comportements internes résulte de l'interaction de multiples processus se développant sur des échelles spatio-temporelles complémentaires [Vanbergue et Drogoul, 2002]. Les processus en question participent au maintien du système urbain en renforçant les tendances à la stabilité au niveau macroscopique, tout en stimulant ou en favorisant l'émergence des fluctuations et des pertur-

bations à partir des niveaux microscopiques. Ces processus sont en particulier :

- Démographiques tels que les croissances naturelles de la population ;
- Migratoires ou économiques, relatifs par exemple à la création et à la localisation de zones d'attraction ;
- Liés à l'expansion territoriale, à l'urbanisation ou à l'aménagement ;
- Liés aux évolutions culturelles et aspirations sociales des individus ;
- Contraints par les mobilités, quotidiennes (résidence-lieux de travail, d'études, de loisirs), annuelles (vacances), ou encore résidentielles (déménagements).

Le développement autour de cette thématique urbaine, de solutions informatiques comme les systèmes d'informations géographiques en particulier orientés objets [Claramunt et Thériault, 2001, Frihida, 2002], la modélisation multi-agents [Ferber, 1995], ou encore les automates cellulaires [Torrens et O'Sullivan, 2000] amènent de nouveaux horizons pour la modélisation de systèmes désagrégés. Ces solutions sont appliquées pour modéliser notamment l'évolution de systèmes urbains et environnementaux à différents niveaux d'échelle et de granularité temporelle [Batty et al., 1999, Pursula, 1999, Stillwell et al., 1999].

2.2.1 Comportements et activités sociétales

La distinction entre l'approche agrégée et l'approche désagrégée est présentée dans les travaux de thèse d'Ali Frihida pour le contexte de mobilité des personnes dans les systèmes urbains [Frihida, 2002]. La modélisation par approche agrégée est dans ce contexte définie par la prédiction des flux de déplacements entre les différentes zones décrivant le partage de l'espace. La prédiction est réalisée par une enquête origine-destination qui recueille et assigne les déplacements entre les zones agrégées de l'espace. Le désavantage majeur de cette approche pour l'étude des déplacements est que les comportements individuels sont confondus dans l'agrégation. A contrario l'approche désagrégée, selon cet auteur, explique les mobilités urbaines par l'étude des processus décisionnels et cognitifs individuels. Les modèles désagrégés ont été largement utilisés pour le développement de modèles pour la micro-simulation [Algers et al., 1997]. Ceux-ci incluent des représentations origine-destination et des analyses dans lesquelles les propriétés et les comportements émergents sont dérivés à partir des choix de déplacements locaux, et des interactions [Frihida et al., 2002, Buliung et al., 2002, Roorda et al., 2002].

Les systèmes urbains tels que les villes sont définies par [Vanbergue, 2000] comme des systèmes complexes ouverts dotés d'une organisation spatiale dans lesquels des éléments d'études importants sont les personnes, et où les migrations intra-urbaines peuvent être représentées par des flux de données désagrégées. Dans les travaux de cet auteur sont considérés les migrations des ménages. Celles-ci représentent des flux qui interviennent entre les différents quartiers de la ville de Bogota. Les quartiers sont modélisés par des agents statiques dans les cellules d'un automate cellulaire. Dans [Vanbergue et Drogoul, 2002], ces mêmes auteurs présentent une généralisation de cette approche avec *MMINUS* signifiant *Modélisation des Migrations d'INtra-Urbaines et Simulation* qui est un outil mis en œuvre pour la simulation urbaine de mobilités résidentielles des ménages.

La décennie précédente a été guidée par l'émergence d'une approche de modélisation basée sur l'activité des individus et des ménages pour l'étude des comportements dans leurs déplacements urbains [Buliung et al., 2002]. Ce paradigme basé sur l'activité représente une alternative à l'approche basée sur le déplacement fréquemment exploitée dans de nombreux

modèles. Le décalage de l'approche basée sur le déplacement en direction de l'approche orientée vers l'activité est par certains aspects un décalage dans l'échelle de représentation : agrégé vers désagrégé, zone urbaine vers individu. Le modèle urbain *Integrated Model of Urban LAnd-Use and Transportation for Environmental Analysis* (*IMULATE*) développé pour l'étude des comportements de la ville de Hamilton, Ontario, Canada, utilise cette approche. [Frihida, 2002] ajoute que l'approche désagrégée, qu'elle soit basée sur le déplacement ou sur l'activité des entités désagrégées, requiert une logistique plus sophistiquée que celle de l'approche agrégée car elle implique la connaissance d'information sur les comportements individuels dans le temps et dans l'espace.

Dans [Khan et al., 2002], la micro-simulation basée sur une approche agent étudie l'évolution de différents établissements commerciaux dans un système économique spatial dont l'espace est décomposé en un ensemble de zones commerciales interconnectées par un réseau de transport. Cette étude aborde les transitions démographiques d'établissements commerciaux, leur localisation dans le système urbain et, les développements commerciaux induits par les choix de situations géographiques. Ces travaux utilisent des techniques de programmation objet pour construire un ensemble de représentations de comportements désagrégés, au niveau des établissements de commerce individuels. Cette modélisation par le paradigme objet permet à ces représentations d'agir sur les conditions de la manifestation de comportements émergents qui soit cohérents avec les comportements agrégés théoriques ou observés.

Christophe Claramunt et Marius Thériault proposent une modélisation des comportements de mobilité des ménages et des individus au sein d'une ville (zone métropolitaine de Québec, Canada) [Claramunt et Thériault, 2001]. Cette modélisation combine l'utilisation du langage *UML* (*Unified Modelling Language*), un système d'information géographique et une base de données orientée objet. L'objectif in-fine de cette modélisation est la micro-simulation par une approche multi-agents de larges systèmes urbains dynamiques dont l'étude requiert la compréhension et la modélisation des comportements individuels et collectifs. Le but de cette micro-simulation est la prédiction des futures utilisations des sols dont les conséquences, comme dans la plupart des systèmes agrégés/désagrégés, à un niveau macroscopique héritent des décisions effectuées à un niveau microscopique (i.e. les individus constituent l'unité de modélisation et de décision). C'est pourquoi ces auteurs considèrent que la modélisation d'un système urbain dans le but de prévoir son évolution implique l'intégration des événements, des processus, et des éléments, à différents niveaux d'agrégation par l'utilisation de différentes échelles temporelles et spatiales, qui ensembles reflètent les composants, les fonctions et l'évolution du système représenté dans sa globalité.

Michel Phipps et André Langlois donnent un autre exemple de simulation d'une ville réelle avec Ottawa, Canada [Phipps et Langlois, 1997]. Dans cette approche, les cellules d'un automate cellulaire représentent des zones de la ville. L'état des cellules est défini par des variables socio-économiques telles que la population, le logement ou l'emploi. Les règles de transition de l'automate reposent sur la caractérisation des évolutions du système par des boucles de rétroaction (cf. section 1.1.3).

Dans le domaine des sciences économiques, Cathal O'Donoghue définit un modèle de micro-simulation qui considère des entités de niveau microscopique comme les unités de base d'analyse lors de l'étude des effets des politiques sociales et économiques [O'Donoghue, 2001]. Cette approche appréhende l'évolution des éléments des niveaux microscopiques et simule les entités au niveau desquelles sont prises les décisions et les actions dans le système économique et social. L'étude présente plusieurs exemples de micro-simulation dans le domaine des systèmes sociétaux et économiques, les résultats au niveau micro permettent de générer des analyses plus détaillées que lors de l'utilisation de modèles agrégés.

Un autre champ d'application de la micro-simulation de systèmes urbains est l'étude des comportements des piétons en fonction de la configuration des zones (e.g. blocs, rues) et des parties attractives du système (e.g. rue commerçante). Dans un contexte légèrement éloigné de la simulation de systèmes désagrégés, [Cutini, 1999] précise dans quelles limites les méthodes d'analyse spatiale permettent de prédire, et avec quelle validité, les déplacements des piétons en zone urbaine. L'étude comportementale des villes italiennes Grosseto et Orbetello conduit à la conclusion principale qu'il existe une corrélation forte entre le plan de la ville, les déplacements piétonniers et l'utilisation du sol. Toutefois il apparaît clairement que même si une corrélation forte existe, la configuration en grille de l'espace n'est pas un paramètre suffisant pour prédire les mouvements urbains de piétons. Bien que l'auteur n'aborde pas l'utilisation de la simulation, son application pour l'étude des flux de piétons en ville est adaptée.

2.2.2 Systèmes en transport

Les Systèmes de Transport Intelligents (*STI*) couvrent un très large éventail de technologies et d'applications dont l'un des objectifs est l'amélioration et le développement de systèmes étudiant le fonctionnement des trafics. Ceci dans le but d'améliorer leurs performances et apporter ainsi des bénéfices aux usagers, aux concepteurs et à la société en général. Pour le groupe de recherche *MADITUC* de l'École Polytechnique de Montréal, Canada, ces systèmes sont décomposés en six grandes tendances illustrées en figure 2.1. Ces types de systèmes intègrent en particulier les systèmes de planification et de gestion des transports collectifs et les systèmes de gestion de la circulation de véhicules.

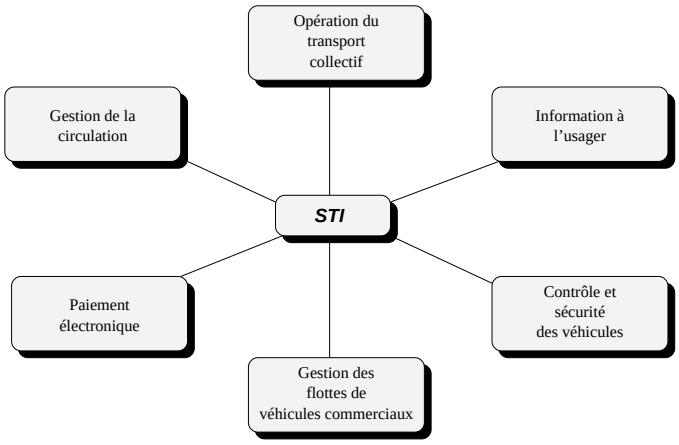


Fig. ***2.1*** — Systèmes de transport intelligents

Les systèmes en transport représentent une partie de l'ensemble des systèmes urbains. De par la nature des entités considérées (i.e. véhicules et parfois personnes) et de leurs actions (déplacements) ceux-ci sont parmi les constituants principaux des systèmes larges à flux de données désagrégées. L'un des premiers exemples précurseurs de simulation informatique du trafic de véhicules est présenté dans [Beilby, 1972]. L'auteur précise que la simulation d'un point de vue informatique peut être abordée soit par la formation et l'étude de files de véhicules aux jonctions des routes (flux agrégés basés sur un déplacement global, macro-simulation), soit par l'étude des mécaniques et des comportements des conducteurs (micro-simulation).

Les simulations microscopiques de trafic deviennent des outils indispensables pour la gestion fine et en temps réel des flux de déplacements dans un large espace urbain. Le potentiel de ces outils est multiple car n'intervient plus alors la seule étude du trafic routier mais aussi la nécessité de simuler, les voyages des véhicules dans un réseau, la mesure de la pollution atmosphérique, les conséquences sur le développement urbain, et autres paramètres affectant les comportements et les usages dans la ville. De tels modèles encore expérimentaux sont déjà utilisés pour de nombreuses villes, cependant certaines fonctionnalités de ces simulations sont encore gérées difficilement. Le problème essentiel étant le facteur d'échelle et la validité des modèles réduits de larges réseaux ainsi que la difficulté de réaliser les calculs afférents comme le montre la difficulté d'intégrer les outils de contrôles externes comme les feux de signalisation. Il est de ce fait difficile de réaliser des simulations à larges échelles réellement significatives.

Selon [Simon et Nagel, 1998], il est évident que face au problème de la dynamicité des systèmes urbains, une approche microscopique, c'est-à-dire orientée vers la description des plus petites entités, est en termes de méthodologie la plus appropriée. Cela signifie considérer les voyageurs individuellement plutôt que par des flux agrégés où il est difficile d'intégrer la gestion des politiques de déplacements autonomes et individuels, et où une simulation ne représenterait pas les comportements à un niveau adapté (i.e. véhicules ou personnes). Ces auteurs présentent un modèle de simulation de trafic urbain en utilisant un automate cellulaire et des règles de transitions stochastiques pour mettre en œuvre des déplacements de véhicules. Cette approche est utilisée pour la modélisation et la simulation de la région de Dallas-Fort Worth (Texas, USA). Elle repose sur le projet *TRANSIMS* (*TRansportation ANalysis and SIMulation System*), outil qui vise la planification des trajets au niveau des véhicules et sur les travaux de thèse de Kai Nagel où celui-ci propose un modèle de circulation en file, basé sur la théorie des automates cellulaires [Nagel, 1995].

L'élément important d'analyse dans la gestion de trafic des systèmes urbains est le contrôle des congestions sur les voies de circulation. [Simon et Nagel, 1998] mentionnent que l'un des points essentiels pour le contrôle des congestions est la gestion des signalisations lumineuses. La congestion provoque, dans un système urbain en pleine activité, un *effet de bord* dans lequel les relations causales entre les entités deviennent de plus en plus diffuses dans le temps et dans l'espace. Par exemple, lorsqu'une route est encombrée, une personne peut changer d'itinéraire ou différer son voyage. C'est le paradoxe induit par les conseils fournis par le Ministère de l'Équipement et des Transports en France (i.e. bison futé). Le respect inconditionnel des consignes par les usagers, par exemple lors de départs en vacances, implique alors un ensemble des départs différés qui conduit à reporter une congestion éventuelle à une heure ultérieure. D'où, en plus de la prédiction, la nécessité d'outils de simulation proactifs permettant une réaction immédiate au changement.

Un exemple de micro-simulation réactive, car réalisée en temps réel avec la prise en compte de facteurs externes, est présenté dans [Kosonen et Bargiela, 1999]. Le modèle orienté objet proposé pour la simulation microscopique de trafic urbain *HUTSIM* est interfacé avec *SCOOT*, un système de contrôle de trafic temps réel utilisé dans de nombreuses villes dont celle de Nottingham en Angleterre [Hunt et al., 1981]. Dans cette architecture, les mesures télémétriques macroscopiques obtenues en temps réel par le système *SCOOT* constituent des paramètres d'entrée pour les micro-simulations mises en œuvre à partir du système *HUTSIM*. L'avantage de cette approche repose sur le fait que la micro-simulation intègre des données en temps réel. La conversion du macro vers le micro implique cependant qu'une partie des informations concernant le trafic soit dérivée à partir d'une estimation statistique de l'organisation et du comportement du système urbain étudié (et inversement quand les simulations

micro réalisées avec *HUTSIM* se doublent de généralisation de ces dernières à un niveau macro). Citons en termes d'ouverture, l'utilisation de ces outils afin de fournir aux usagers de la ville de Nottingham des informations (une des thématiques des *STI* qui est encore à affiner) sur les trafics, en particulier, sur les temps de passage des bus [Peytchev et Claramunt, 2001]. Ce support est mis en œuvre par l'envoi de message *SMS* (*Short Message Service*) sur les téléphones portables. Les applications de ce type sont représentatives des perspectives d'utilisation et des services que peuvent offrir les *STI*.

Dans le domaine de la compréhension des phénomènes de trafic urbains, un autre exemple d'approche est le système multi-agents *Cybele* utilisé pour la simulation de véhicules se déplaçant entre différentes partitions de l'espace [Erol et al., 1999]. Les véhicules sont ici modélisés par des agents mobiles. Dans [Chopard et al., 1997], la dynamique des véhicules est simplifiée afin de ne conserver que les caractéristiques essentielles. Elle est modélisée par l'utilisation d'automates cellulaires. Dans cette micro-simulation, les croisements dans le réseau sont modélisés par un rond point giratoire (i.e. un automate cellulaire pour le cercle central, et un pour chaque entrée/sortie au croisement). Ce modèle est appliqué à la ville de Genève (Suisse) dans lequel l'utilisation de représentations origine-destination permet de simuler et d'évaluer les temps de déplacement d'un point à l'autre dans le réseau de la ville.

Matti Pursula dresse un panorama des différentes approches utilisées pour la simulation de trafic en précisant que cette discipline est l'une des rares à avoir été étudiée à différents niveaux de granularités [Pursula, 1999]. Par exemple, une approche macroscopique de la simulation de trafic est présentée dans [Byrne et al., 1982] et, à un niveau mesoscopique, une simulation de trafic avec comme élément atomique des groupes de véhicules est présentée dans [Leonard et al., 1978]. Pour Pursula, la plupart des simulations de trafic actuelles sont basées sur des micro-simulations dans lesquelles les composants microscopiques du système sont les véhicules et où les interactions majeures sont de type véhicule-véhicule. De ce fait, pour cet auteur, les grandes tendances conceptuelles et technologiques sont relatives à la micro-simulation. Celles-ci incluent notamment l'utilisation des préceptes et outils informatiques tels que les approches de modélisation discrète pour représenter les flux de véhicules, de programmation orientée objet ou agent, parallèle, ou encore d'outils de visualisation interactifs et de réalité virtuelle.

[Magne et al., 2000] introduisent une approche qui consiste à utiliser conjointement une modélisation micro et une modélisation macro pour la simulation de flux de trafic afin de bénéficier du niveau d'étude adéquat au moment voulu. L'objectif de cette approche hybride est de réaliser une simulation macroscopique dans laquelle les flux du réseau de transport sont caractérisés par des équations discrétisées et où il est possible de focaliser à un niveau microscopique certaines parties du réseau. Les transitions entre l'étude agrégée et désagrégée du trafic des véhicules sont établies par alternance dans le temps.

2.3 Besoins conceptuels et technologiques

Malgré des possibilités prometteuses, les techniques pour l'étude de systèmes à large flux de données désagrégées souffrent encore d'un manque de capacité de modélisation adaptée et de puissance de calcul, notamment pour la simulation de systèmes complexes. Ainsi, bien que la simulation apparaisse comme un outil de prédiction et de compréhension adapté pour les systèmes à larges flux de données, les limites de cette approche sont malheureusement nombreuses. En effet, la qualité de la micro-simulation est limitée en grande partie par les capacités de calcul qui sont confrontées au degré de complexité (cf. section 1.1.1) souvent élevé

de l'approche désagrégée. Dans une telle approche, la forte décomposition organisationnelle du système implique un nombre conséquent d'entités qui doivent être modélisées et mises en œuvre sur le support de simulation. De plus, un haut degré de complexité présente en particulier au niveau des interactions une grande non-linéarité qui est fonction du nombre d'éléments microscopiques. Il convient donc de parvenir à un compromis entre les différentes caractéristiques de la simulation [Nagel, 1995] :

- Résolution : décrit le niveau de détail du modèle, qui doit être suffisant pour identifier les phénomènes jugés intéressants voire pertinents ;
- Fidélité : le degré de réalisme du modèle (correspondance avec le phénomène réel) ;
- Taille du système : correspond à la grandeur des phénomènes que l'on désire observer ;
- Vitesse de la simulation : comparée au temps réel, elle peut être accélérée, ralentie, ou correspondre au temps physique ;
- Ressources : correspond aux ressources en temps et en matériel informatique dont on dispose.

Il est possible de classer les modèles existants en fonction des caractéristiques sur lesquelles l'accent est mis. En particulier à ressources égales on oppose, (1) les modèles macroscopiques qui privilégient la fidélité et la taille du système, au détriment de la résolution (i.e. travaillent en effet sur des valeurs agrégées) aux (2) modèles microscopiques privilégiant la résolution et la fidélité, au détriment de la vitesse et de la taille du système et où les entités sont traitées de manière individuelle.

Un des objectifs des systèmes à larges flux de données désagrégées est d'obtenir un ensemble de comportements à un niveau macroscopique[2] à partir de la simulation des interactions et des comportements à un niveau microscopique. Une correspondance entre le niveau des résultats escomptés et celui auquel les décisions de l'action sont effectuées est ainsi nécessaire. En d'autre terme, un enjeu important lors d'une simulation est de caractériser les phénomènes d'émergence au niveau macroscopique et en retour de caractériser des phénomènes de rétroaction vers le niveau microscopique (cf. section 1.1.4). Hans Baekgaard cherche à établir un tel lien entre les modèles de micro-simulation décrivant le marché du travail en Australie et les modèles de prévision et de projection macro-économique [Baekgaard, 1995]. La relation (ou liaison) entre le niveau micro et le niveau macro est présentée comme la capacité d'identifier la portée de l'émergence dans la hiérarchie du modèle et son impact en retour sur les éléments ayant produits de nouvelles caractéristiques. Cet auteur présente une taxonomie, mise en avant par H.P. Galler, qui distingue quatre méthodes pour décrire la liaison entre le niveau micro et le niveau macro :

- L'approche *top-down* consiste en un ajustement en sens unique des éléments du niveau micro afin de les faire correspondre à un comportement macro ;
- De manière similaire et inverse, l'approche *bottom-up* est une rétroaction correspondant à un ajustement du niveau macro à partir des comportements du niveau micro ;
- La combinaison d'une liaison *bottom-up* et *top-down* permet d'enrichir la dynamique interne. Dans ce cas l'émergence produit des nouvelles propriétés au niveau global qui peuvent être analysées et réinjectées. L'approche récursive est une basée sur une combinaison *bottom-up* et *top-down* dans laquelle les modèles macro et micro sont calibrés (ajustés) par périodes afin d'assurer une consistance entre les deux niveaux ;

[2] Éventuellement des tendances générales utilisables dans un processus de décision.

- L'approche itérative est une extension de l'approche récursive dans laquelle les émergences du niveau macro et micro sont résolues simultanément et où les deux niveaux convergent vers un état de stabilité.

La capacité d'établir un lien circulaire, entre le niveau macroscopique et le niveau microscopique, inférée par la définition d'une méthodologie adaptée pour la modélisation et la simulation des flux de données désagrégées, est encore à approfondir. En effet, le choix d'une méthodologie de modélisation et de simulation (cf. section 1.3.2) influence et guide la caractérisation et l'identification des liens entre le niveau micro et le niveau macro.

Les méthodes de modélisation et de résolution de systèmes complexes sont très variées. Toutefois, les différentes approches semblent converger vers une technique dans laquelle le modèle est conçu par une description hiérarchique du système avec une agrégation spatiale des composants [Page et al., 1999]. Cependant, malgré le fait que certains principes mathématiques et de modélisation communs émergent par leur application dans différentes disciplines, il est souvent difficile d'appliquer et réutiliser certains des ces principes de modélisation et de simulation d'une discipline à un autre. Il existe également une distance conceptuelle et technique entre les modèles et les paradigmes fréquemment utilisés dans les modèles désagrégés et les possibilités informatiques. Ceci est dû en partie à la difficulté de dériver les concepts de modélisation utilisés dans les approches désagrégées vers des supports de calcul appropriés, tant au niveau statique (propriétés de données) qu'au niveau dynamique (comportement de données). Selon Andrzej Bargiela, les efforts majeurs à opérer pour la modélisation et la simulation sont les suivants [Bargiela, 2000] :

- Amélioration de la compréhension du processus de formation d'abstraction d'un système en divers modèles de résolution pour faciliter la mise en œuvre d'outils de simulation multi-échelles relatifs à ces représentations ;
- Élaboration de méthodologies facilitant la dérivation de niveaux élevés d'abstraction pour la simulation qualitative ;
- Généralisation de la modélisation hiérarchique par l'intégration des abstractions structurales et comportementales ;
- Recherche et étude des abstractions (homomorphismes) qui préserve les dynamiques entre les représentations continues et discrètes.

Un enjeu important de la recherche à caractère pluridisciplinaire est l'intégration de concepts issus du calcul informatique pour conceptualiser, comprendre et modéliser les modèles désagrégés [Torrens, 2001]. Un ensemble d'initiatives de recherche et de travaux abordent aujourd'hui ces directions stratégiques. En particulier, les Systèmes d'Information Géographiques (SIG) qui intègrent toutes les caractéristiques spatiales et temporelles, lesquelles reflètent mieux la complexité intrinsèque des comportements, des activités et de déplacements fournissent une opportunité pour modéliser les composants et leurs relations d'une manière fine [Buliung et al., 2002].

Le problème essentiel de compréhension des systèmes complexes par l'utilisation des concepts et outils informatiques est donné par [Bousquet et Gautier, 1999] qui précisent que la modélisation des dynamiques spatiales de systèmes complexes pose la question de l'intégration des méthodes permettant de passer de l'analyse thématique des objets abstraits à la simulation de leurs interactions. Ces auteurs considèrent que les méthodologies pour la simulation reposant sur l'utilisation d'automates cellulaires et de systèmes multi-agents permettent de rendre compte de ces dynamiques spatiales. Ils présentent et comparent ainsi deux modèles de simulation pour décrire un processus d'expansion des terres arables. Le

premier de ces modèles se base sur la définition d'agents au niveau des individus, ceux-ci étant les acteurs du changement de l'environnement rural. Dans une deuxième approche, les agents représentent des agrégats spatiaux pour lesquels sont définis des mécanismes agrégés de changements.

Diane Vanbergue affirme également (en référence aux travaux de Bousquet et Gautier) que la simulation multi-agents résout certaines difficultés auxquelles d'autres types de modélisation se heurtent, comme l'intégration des interactions, la prise en compte de plusieurs niveaux d'abstraction[3], la prise en compte de paramètres qualitatifs et la représentation de différentes dynamiques du système considéré [Vanbergue, 2000]. Conjointement avec Alexis Drogoul, elle postule que deux approches cœxistent pour décrire un système complexe urbain et les flux (interactions) de données désagrégées sous-jacents : soit l'espace est perçu comme le support des interactions entre des sous-systèmes qui produisent la différentiation spatiale (l'espace est un agent), soit l'organisation spatiale est perçue comme le produit des interactions entre les entités spatiales (l'espace est un support) [Vanbergue et Drogoul, 2002]. Selon ces auteurs, l'intérêt des automates cellulaires et des systèmes multi-agents est certain et admis par les modélisateurs des phénomènes urbains. En effet, de telles méthodologies mettent en avant les interactions, et permettent de mieux comprendre les relations locales et globales, et de faire apparaître le caractère émergent et auto-organisé des phénomènes urbains.

Un des grands challenges encore ouvert pour la modélisation et la simulation à un niveau d'étude microscopique est de fournir aux concepteurs un ensemble de solutions et de supports informatiques à la fois adaptés à leurs objectifs de modélisation, et suffisamment flexibles pour favoriser une souplesse dans l'exécution de simulations. Ceci notamment par la calibration des paramètres d'entrée des simulations en accord avec les variabilités de scénarios relatifs aux objectifs de simulation. En effet, les simulations de systèmes complexes à larges flux de données telles que celles présentées dans ce chapitre requièrent très souvent de puissantes ressources informatiques. La solution courante consiste à diminuer la fidélité du modèle pour accélérer les calculs. Néanmoins des objectifs tels que, par exemple, les prédictions de trafic pour les gestionnaires de réseaux routiers nécessitent aujourd'hui une étude à la fois des progrès méthodologiques dans les démarches de modélisation et de simulation, et le développement de solutions informatiques adaptées, flexibles et performantes.

Notons enfin que l'histoire de l'informatique a montré que chaque fois qu'une étape était franchie dans les performances des machines, les applications faisaient de même en demandant encore plus de ressources pour pouvoir intégrer de nouvelles fonctionnalités, des méta-informations, ou autres facilités complémentaires. Cette contrainte nous amène à étudier la distribution des calculs et des données comme une perspective offrant autant de ressources que nécessaire par la multiplication des ordinateurs utilisés, ce qui permettrait de réduire voire supprimer le besoin de simplification des représentations des systèmes complexes.

2.4 Conclusion

Ce chapitre propose, en section 2.1 une définition et une description des propriétés principales des systèmes complexes où interviennent de larges flux de données désagrégées. La deuxième section aborde, parmi les différents domaines de modélisations et de simulations à une échelle microscopique, le cas particulier des systèmes complexes urbains et sociétaux qui constituent une classe d'application importante des systèmes à larges flux de données

[3]La mise en œuvre fréquente du paradigme agent par un langage objet favorise par ailleurs la description d'une organisation hiérarchique interne.

désagrégées. La section 2.3 aborde les avantages, les problèmes et les différentes directions exploitables pour la modélisation et la simulation de systèmes complexes.

Les besoins de méthodologies de modélisation adaptées de systèmes urbains lorsque le champ d'étude considéré est l'analyse des flux de données entre différents composants spatiaux est toujours un problème ouvert. La complexité croissante des systèmes d'informations résultant des modèles est en constante opposition avec les ressources informatiques disponibles pour gérer, traiter, simuler ces systèmes. En effet, les besoins pour la simulation de phénomènes complexes augmentent beaucoup plus vite que les ressources informatiques centralisées nécessaires et traditionnellement utilisées pour la simulation séquentielle. Nous constatons que de nombreux travaux sont basés, de manière distincte ou combinée, soit sur les automates cellulaires afin de représenter l'espace par une grille généralement uniforme, soit sur une approche multi-agents qui rend parfaitement compte des interactions entre les éléments (indépendamment de leur niveau d'agrégation) des systèmes urbains. Toutefois, bien que réalisant des simulations discrètes la plupart de ces travaux n'exploitent pas les concepts théoriques et outils hérités des préceptes de la simulation à événements discrets.

Le besoin d'obtenir à la fois des simulations à haute résolution mais aussi des vitesses d'exécution élevées ne peut être satisfait qu'en diminuant la précision du modèle au niveau microscopique, c'est-à-dire en réduisant à un ensemble minimal les détails régissant le comportement des éléments. Une autre approche consiste à adapter le support d'exécution pour obtenir un gain de puissance en accord avec de telles simulations. Dans cette optique de simulation qualitative de systèmes complexes à larges flux de données désagrégées, nous souhaitons analyser les perspectives de modélisation et de simulation offertes par l'utilisation des systèmes distribués. Dans le chapitre suivant nous présentons les principes et l'utilisation d'architectures multiprocesseurs pour l'exécution distribuée de simulations à événements discrets.

CHAPITRE 3

Simulation distribuée

« Divide et Impera (diviser pour régner). Maxime Romaine »

La mise en œuvre conjointe de modèles désagrégés et de simulations séquentielles à événements discrets pour l'étude de phénomènes complexes à larges flux de données dépasse fréquemment les limites des ordinateurs exploités notamment en termes de gestion du nombre d'événements. Le besoin, voire la nécessité dans ces conditions, de distribuer le traitement des événements sur plusieurs ordinateurs, implique le respect de certaines conditions et le développement de mécanismes adaptés. Cette approche multiprocesseurs présente tout son potentiel lorsque les simulations possèdent beaucoup d'indépendance, c'est-à-dire lorsque de nombreux événements peuvent être exécutés simultanément, sans remettre en défaut la précédence causale. Toutefois, il existe toujours un fossé entre les modèles et les paradigmes souvent utilisés en simulation de systèmes complexes et, les possibilités et les contraintes du calcul distribué, en particulier lorsque de larges flux de données désagrégées interviennent au sein du système. Nous présentons dans ce chapitre les principes généraux de la simulation basée sur l'utilisation d'architectures multiprocesseurs ainsi que leur potentiel et leur limites pour la simulation de systèmes complexes à larges flux de données désagrégées. La section 3.1 introduit les architectures multiprocesseurs sur lesquelles peuvent reposer l'exécution répartie de simulations séquentielles. La section 3.2 présente les mécanismes de distribution permettant de répartir l'exécution d'une simulation. La section 3.3 définit la notion d'exécution répartie d'événements dans un modèle par échange de messages. La section 3.4 présente les principes et les techniques de la simulation distribuée à événements discrets. La section 3.5 décrit le mécanisme de migration, technique employée pour modéliser et simuler les flux de données. La section 3.6 présente différentes utilisations des systèmes distribués pour la simulation, et la section 3.7 conclut ce chapitre.

3.1 Architectures multiprocesseurs pour la simulation

L'évolution des besoins et la maîtrise des architectures multiprocesseurs déjà établie dans les années soixante dix ont dirigé la simulation vers la simulation parallèle à événements discrets (*Parallel Discrete Event Simulation* : *PDES*) [Chandy et Misra, 1979, Peacock et al., 1979]. L'utilisation des architectures multiprocesseurs pour la simulation de systèmes complexes et plus généralement pour des calculs intensifs est souvent justifiée par trois argumentaires. Premièrement, les besoins de puissance de calcul sont de plus en plus importants, et cela aussi bien du point de vue des concepteurs de logiciels que de celui des uti-

lisateurs. Une machine monoprocesseur ne peut, à elle seule, répondre aux besoins de tout un chacun sur le plan de la rapidité de calcul. C'est pourquoi une solution logique est de rechercher à répartir la puissance de calcul dans des architectures multiprocesseurs. Deuxièmement, à l'heure des réseaux de communication en général et de l'Internet en particulier, les utilisateurs de l'informatique souhaitent de plus en plus utiliser leur matériel non plus comme un simple poste isolé, mais aussi comme un intermédiaire faisant partie d'un tout qui leur permet notamment de coopérer avec d'autres utilisateurs. Enfin, notons les besoins spécifiques d'applications de nature parallèle qui nécessitent un support d'exécution adapté.

Il existe, dans le principe, deux types d'architectures multiprocesseurs qui peuvent être exploitées pour la simulation. Ces architectures se distinguent essentiellement par la répartition physique des processeurs. La première classe est constituée par des machines multiprocesseurs souvent appelées calculateurs parallèles. Ce type de machine est caractérisé par un nombre limité de processeurs souvent regroupés dans une seule machine. A la différence des calculateurs parallèles, les systèmes distribués sont composés d'un ensemble de machines physiquement séparées et reliées par des réseaux de communications externes. Toutefois, cette approche de classification des architectures parallèles reste sommaire tant les distinctions entre les architectures parallèles deviennent de plus en plus subtiles. Plusieurs tentatives de classification ont été réalisées. Michael Flynn par exemple considère les architectures traditionnelles dites de von Neumann, où toutes les opérations sont effectuées de manière séquentielle [Flynn, 1972]. Cet auteur propose une classification qui repose sur la connaissance des flots d'instruction (*I*) et des flots de données (*D*) et distingue ainsi les architectures *SISD*, *SIMD*, *MISD* et *MIMD* :

- Les architectures à flot d'instructions et de données uniques (*Single Instruction stream, Single Data stream* : *SISD*) correspondent au modèle de von Neumann classique. Bien qu'aucune architecture moderne ne corresponde strictement à cette définition, celles-ci se rapprochent des architectures monoprocesseurs actuelles (e.g. PC) ;
- Les architectures à flot d'instructions unique et à flots de données multiples (*Single Instruction stream, Multiple Data stream* : *SIMD*) correspondent à une classe où plusieurs processeurs exécutent simultanément la même instruction sur des flots de données différents. Dans ce type d'architecture, chaque processeur possède en général une mémoire locale lui permettant de stocker ses résultats (e.g. machine CM-2) ;
- Les architectures à flot d'instructions multiples et à flots de données unique (*Multiple Instruction stream, Single Data stream* : *MISD*) correspondent à une classe d'architectures systoliques (réseau linéaire de processeurs de calcul) qui consiste, pour chaque processeur, à exécuter des instructions sur un ensemble de données avant de transmettre le résultat au processeur suivant ;
- Les architectures à flot d'instructions multiples et à flots de données multiples (*Multiple Instruction stream, Multiple Data stream* : *MIMD*) correspondent aux architectures les plus aptes pour la simulation à événements discrets. Ces architectures exécutent simultanément des instructions différentes sur des données différentes. Notons, que bien que restrictif, ce sont ces architectures que l'on appelle communément par abus de langage, les architectures multiprocesseurs (e.g. machine Cray-1).

Dans les architectures *MIMD*, les mécanismes de communications reposent sur des mécanismes physiques. Les principaux supports utilisés sont l'échange de messages basés sur des liens de communications physiques dédiés à la communication entre les processeurs et la mémoire partagée. Les architectures *MIMD* sont ainsi souvent classées en deux catégories en fonction du support de communication : les multiprocesseurs à couplage fort où tous les processeurs se partagent l'accès à une mémoire commune et les multiprocesseurs à couplage

faible dans lesquels les mémoires sont locales (i.e. propres) à chaque processeur. Dans le cas des machines parallèles, dont les processeurs sont géographiquement proches, il est possible de mettre en œuvre une mémoire partagée répartie câblée. Cette approche possède un avantage majeur. En effet, la programmation par mémoire partagée est plus simple à appréhender que la programmation par échange de messages car plus proche de la programmation séquentielle. La taxonomie de Flynn [Flynn, 1972] ne permet pas de classer certaines architectures à processeurs multiples notamment compte tenu des évolutions récentes et des distinctions de plus en plus difficiles à réaliser. Duncan propose lui une taxonomie plus simple basée sur le caractère synchrone ou non des calculateurs [Duncan, 1990] :

- Les architectures synchrones qui possèdent une horloge globale permettant une exécution synchronisée des instructions lorsque cela est nécessaire ;
- Les architectures asynchrones dans lesquelles chaque machine ou processeur possède sa propre horloge ;
- Les architectures mixtes qui intègrent une combinaison hybride et non définie d'architectures synchrones et asynchrones.

De ces classifications nous retenons deux caractéristiques essentielles qui vont conditionner la structure, l'emploi et la mise en œuvre de la simulation parallèle à événements discrets de systèmes complexes :

- Le caractère fortement couplé ou faiblement couplé de l'architecture utilisée ;
- La gestion synchrone ou asynchrone du temps entre les nœuds de l'architecture.

Les systèmes distribués sont composés d'un ensemble de machines physiquement séparées qui ne possèdent pas d'horloge physique commune et qui sont faiblement couplées[1]. Deux caractéristiques essentielles définissent un système distribué : un tel système possède un nombre variable d'entités de calculs indépendantes, autonomes et communicantes ; et ces entités réparties apparaissent à l'utilisateur comme une machine unique [Tanenbaum, 1993]. Remarquons qu'Andrew Tanenbaum insiste sur le fait que la propriété qui définit principalement le système distribué, est l'indépendance complète entre les processeurs. Du point de vue de la modélisation, [Guerraoui et Vinoski, 1997] considèrent qu'un système distribué peut être défini comme une collection d'objets interactifs qui communiquent en utilisant des interfaces systèmes. Dans ce type d'architecture, le réseau de communication permet d'établir un lien parmi l'ensemble des stations de travail qui sont généralement assemblées dans un contexte géographique proche afin de former un réseau dit local dédié au calcul (à la simulation dans notre contexte). La terminologie employée est alors celle de grappe (*cluster*), mais notons que l'extension des réseaux de communication permet à ces machines d'être les constituants d'un réseau plus étendu afin de s'intégrer dans un réseau global de type Internet.

La capacité d'échanger des informations entre les nœuds (i.e. ordinateurs) composants un système distribué est la propriété essentielle de telles architectures. Le mécanisme de base pour assurer cette interconnexion est l'échange de messages. Il s'opère traditionnellement entre différents processus répartis sur les différentes machines. Certains paradigmes de plus haut niveau mais reposant tous sur le principe d'échange de messages ont été proposés. Citons les deux principaux qui sont l'appel de procédures à distance (*Remote Procedure Call* : *RPC*) introduit par [Birrell et Nelson, 1984] et la mémoire virtuelle partagée (*Shared Virtual Memory* : *SVM*) introduite par [Li, 1986]. Un point commun aux différentes architectures

[1] Les systèmes distribués appartiennent à la classe *MIMD* de la taxonomie de Flynn et à la classe asynchrone dans la taxonomie de Duncan.

qui mettent en œuvre une telle mémoire partagée logicielle est la nécessité d'introduire des critères de cohérence qui définissent la sémantique de la mémoire partagée. La nécessité de cohérence est due au fait que les entités de stockage considérées sont des ensembles d'éléments locaux aux sites de l'architecture, c'est-à-dire physiquement répartis, mais dont la vision et les accès sont globaux. Pour ce type de structure, le critère de cohérence exprime le degré de synchronisation imposé sur l'évolution des données partagées. Cette cohérence imposée à la mémoire a un coût qui est lié au critère choisi, c'est-à-dire suivant le degré de synchronisation voulue sur les accès à la mémoire. [Righter et Walrand, 1989] remarquent que la mémoire partagée peut être un goulot d'étranglement lors de la simulation, et ce, particulièrement lorsque le nombre de processeurs est élevé. L'échange de message qui est le mode communication principal des systèmes distribués peut, comme la mémoire partagée, être un facteur limitant pour la performance des simulations, d'où la nécessité de définir des mécanismes de communications adaptés, et ce d'autant plus pour la simulation de larges flux de données désagrégées. Une étude comparative de ces deux approches de communication dans les systèmes distribués est présentée dans [Lu et al., 1995]. Ces auteurs comparent la communication par échange de message réalisée avec la librairie de communication *PVM* (*Parallel Virtual Machine*) [Sunderam, 1990, Geist et al., 1994], avec la communication réalisée par la mémoire partagée *TreadMarks* [Keleher et al., 1994].

L'utilisation des systèmes distribués pour la simulation de systèmes complexes présente les quelques avantages présentés en début de cette section. Compte tenu des besoins informatiques potentiels, une approche fonctionnant sur une grappe de stations de travail constitue un rapport performance/prix extrêmement valable notamment en comparaison des calculateurs parallèles[2]. Dans certains cas de simulations spécialisées, pour une certaine classe d'application, une architecture de simulation peut nécessiter d'être composée d'éléments calculatoires non symétriques et/ou séparées géographiquement. Dans cette hypothèse, la flexibilité d'une telle architecture permet une configuration spatiale des nœuds afin de répartir, au mieux pour les besoins de la simulation, les données et les calculs et autorise des ajouts ou des suppressions de nœuds de calculs à la volée en fonction des besoins.

3.2 Mécanismes de distribution

La distribution est le mécanisme de répartition de l'exécution d'une simulation, de ses données et de ses contrôles sur plusieurs machines. Rondha Righter et Jean Walrand ont proposé cinq méthodes de décomposition pour la simulation multiprocesseurs : la parallélisation automatique, la distribution des expériences, la distribution des fonctions du simulateur, la distribution des événements, et la distribution des éléments du modèle [Righter et Walrand, 1989]. Alois Ferscha propose un schéma de décomposition similaire dans [Ferscha, 1995]. Dans ses schémas de décomposition, ce dernier ne présente pas la parallélisation automatique, mais ajoute une séparation pertinente de la distribution des événements en une distribution centralisée et une distribution décentralisée.

3.2.1 Parallélisation automatique

Le mécanisme de distribution par parallélisation automatique repose sur l'utilisation d'un compilateur parallélisant. Appliqué à un code séquentiel, l'objectif de ce compilateur est

[2]Par exemple, le supercalculateur japonais *Earth simulator* mis en fonctionnement en mars 2002, et dédié à la simulation des phénomènes qui animent et agitent la terre a coûté près de 400 millions de dollars.

d'identifier les séquences de code indépendantes pouvant être exécutées en parallèle. Cette approche à l'avantage de pouvoir être appliquée, de manière transparente pour l'utilisateur, sur de larges programmes de simulation existants. Notons toutefois que la transparence est relative car en pratique l'utilisateur doit aider le compilateur en lui précisant qu'elles sont les parties du code pouvant être parallélisées. Le but de la parallélisation automatique est finalement de distribuer les calculs pour exploiter la présence de plusieurs processeurs. Bien qu'il s'agisse de la méthode privilégiée pour le calcul scientifique intense (e.g. calcul matriciel), cette méthode a le désavantage d'être difficilement applicable à l'intégralité du code. Par ailleurs, cette approche ne permet pas de détecter, ni d'exploiter le parallélisme intrinsèque du modèle à simuler.

3.2.2 Distribution des expériences

La distribution des expériences consiste en l'exécution indépendante du même code séquentiel de la simulation par chacun des processeurs. Dans le cas de simulations stochastiques, où les paramètres initiaux de chacune des simulations sont identiques mais où les évolutions des simulations sont différentes, l'objectif est d'obtenir un calcul moyen résultant de la fusion des différents résultats afin de réduire les erreurs. Dans un cas plus général où les exécutions simultanées diffèrent par l'ensemble des paramètres initiaux, les simulations permettent d'explorer parallèlement différents résultats. Si on dispose de N processeurs, cette approche permet donc d'obtenir N résultats pour le coût d'une simulation séquentielle (accélération par N). Cette approche n'est, toutefois, pas toujours applicable dans la mesure où elle suppose que chaque processeur ou machine puisse exécuter l'intégralité de la simulation séquentielle. Ceci contredit de plus l'approche de simulation distribuée dont l'objectif est de pouvoir exécuter des simulations plus grandes et plus réalistes. Notons par ailleurs que cette approche rend difficile l'interaction humaine avec les simulations.

3.2.3 Distribution des fonctions du simulateur

L'objectif de la distribution des fonctions du simulateur est de définir des machines spécialisées par fonctions. Dans cette approche les différentes tâches du simulateur sont identifiées et réparties sur chacun des nœuds de calcul. Par exemple un processeur peut être affecté à la génération des paramètres d'entrées du simulateur, un deuxième peut être en charge de l'évaluation du modèle et un troisième en charge de la collecte des informations et de l'affichage des résultats via une interface graphique. En fonction de la décomposition effectuée, cette méthode peut permettre, contrairement à la première méthode de distribution (parallélisation automatique), la prévention des interblocages entre les machines car chacune d'entre elles travaille en coopération et non en concurrence. Toutefois, cette approche n'exploite pas l'aspect structurel et fonctionnel du système modélisé.

3.2.4 Distribution des événements

La simulation par distribution centralisée des événements, similairement à la simulation séquentielle, repose sur la maintenance d'une liste des événements. Dans cette approche où la liste est une variable globale, l'exécution des événements est distribuée à chaque processeur, à partir de la tête de liste, lorsque celui en fait la demande. Ce principe est particulièrement bien adapté aux architectures possédant une mémoire partagée pouvant contenir la liste globale. Cette approche permet une bonne distribution des calculs prenant en compte la performance

de chaque nœud. Lorsque la distribution des événements est utilisée pour la simulation à l'aide d'un système distribué, elle peut provoquer un fort ralentissement de l'exécution des événements car les requêtes pour obtenir un événement à simuler se font potentiellement à d'autres machines. De manière générale cette approche n'est applicable qu'à un petit nombre de processeurs en raison de l'accès concurrentiel à la liste globale et aux mécanismes de cohérence qui doivent être employés pour en assurer une synchronisation. En référence à la section 3.3, elle nécessite de mettre en œuvre des mécanismes de cohérence pour une exécution répartie dans le modèle (ou paradigme) à mémoire partagée. Remarquons que ceux-ci sont d'autant plus inefficaces et difficiles à mettre en œuvre que la mémoire partagée est virtuelle (i.e. logicielle).

La distribution décentralisée des événements consiste à assigner les événements à différents processeurs de manière décentralisée, c'est-à-dire en accomplissant un découpage de la liste globale des événements. Dans ce schéma chaque processeur se voit associé une liste locale d'événements à simuler. Certains événements peuvent causer de nouveaux événements à rajouter dans la liste locale et certains nécessitent d'être envoyés vers d'autres processeurs. Dans ce dernier cas où il y a coopération, le système doit être en mesure d'assurer l'ordonnancement correct des événements répartis. Cette stratégie de simulation concurrente des événements avec différents estampillages (en général temporel) nécessite des protocoles de synchronisation non triviaux qui ont selon [Ferscha, 1995] été l'objet de toutes les attentions de la recherche en simulation parallèle et distribuée car cette approche de la simulation permet d'espérer le plus fort degré de parallélisme.

3.2.5 Distribution des éléments du modèle

La distribution des éléments du modèle consiste en une décomposition en plusieurs composants si possible faiblement couplés (peu d'interactions entre eux) d'un système complexe afin de les assigner à différents processus. Dans une telle décomposition chaque nœud de calcul exécute un ou plusieurs processus. Le système à simuler est généralement modélisé par un graphe où chaque nœud représente un composant et les arcs représentent les interactions possibles. L'objectif d'interactions faibles de ce modèle et le fait que celles-ci soient modélisées par un arc dans un graphe sous-entend que, bien que non exclusif, le mode de communication privilégié entre les processus est la communication par messages. De ce fait les architectures à échanges de messages telles que les systèmes distribués se prêtent bien à ce type de décomposition. Notons qu'en fonction du système à simuler, il n'est pas certain que la répartition des composants du modèle (i.e. des processus) soit homogène d'où des dissymétries de charge entre les processeurs. Dans ce cas une affectation manuelle ou des mécanismes de répartition de charge (cf. section 3.5) peuvent être nécessaires pour une performance optimale.

Cette approche de décomposition, similaire à la distribution décentralisée des événements, suscite également un grand intérêt car elle possède la capacité de tirer parti du parallélisme inhérent du modèle. Cette conception de répartition peut donc avoir un effet très bénéfique sur les performances de la simulation. De plus, ce mécanisme de distribution permet une structuration claire des données et des calculs associés qui sont ainsi répartis sur chaque machine. En fonction du système simulé et de sa topologie (statique ou dynamique), le modèle peut être décomposé selon son aspect structurel ou fonctionnel. Les nœuds du graphe représentant le modèle correspondent alors respectivement à un élément structurel ou à un élément fonctionnel. Les arcs caractérisent l'existence d'interactions entre respectivement des régions de l'espace ou des éléments inclus dans cet espace. Une des approches naturelle et appropriée pour

ce type de décomposition consiste à mettre en œuvre un système distribué orienté objet dans lequel les instances de classe représentent les composants du modèle. Cette approche a toutefois un désavantage majeur du à la répartition des événements qui sous-entend l'intégration de mécanismes de synchronisation afin d'assurer une cohérence dans l'avancement de chacun des processus.

3.3 Exécution répartie d'événements discrets

Les six approches de décomposition et de répartition présentées dans la section précédente ont chacune des avantages et des inconvénients relatifs au choix d'exécution et aux objectifs de simulation (e.g. non prise en compte de la structure du modèle ou encore nombre de nœuds limités). De ce fait, en fonction des objectifs de simulation, une combinaison entre différentes approches peut être envisagée. Dans la suite de ce document, nous considérons principalement la distribution des éléments du modèle, mais incluons également la distribution décentralisée des événements, comme méthodes de décomposition pour la simulation distribuée à événements discrets. En effet, les principes et problèmes d'exécution notamment de synchronisation inhérents à ces deux méthodes sont similaires. Par ailleurs, ces deux types de répartition sont les plus aptes à exploiter la nature et les objectifs des simulations de systèmes complexes à larges flux de données, et ce notamment parce que les flux sont d'un point de vue du système distribué des échanges de messages entre des composants spatialement distincts.

3.3.1 Exécution répartie

Dans les schémas de distribution décentralisée des événements et de distribution des éléments du modèle, la simulation peut être vue comme une collection de processus logiques [Misra, 1986b] avec en général aucun accès à des variables partagées. Chaque processus exécute une suite séquentielle d'événements qui sont en coopération ou en concurrence avec les événements des autres processus. Ce type d'exécution a été introduit par Leslie Lamport dans [Lamport, 1978] où il définit une exécution répartie dans le modèle à communication par messages. Tout au long de leur cycle de vie, les processus produisent de façon discrète trois types d'événements :

- L'envoi d'un message M produit un événement d'émission ;
- La réception d'un message M produit un événement de réception ;
- L'exécution d'une instruction relative à l'application n'induisant ni envoi ni réception de message produit un événement interne.

L'expression histoire locale définit la séquence de tous les événements produit par un processus P_i. L'union de toutes les histoires locales (ensemble des événements) munie d'une relation d'ordre partiel sur les événements constitue une exécution répartie. L'introduction nécessaire d'une relation d'ordre sur les événements régule l'exécution en parallèle de ces événements en les liant par un principe de causalité[3]. En effet, ces stratégies de distribution et ce mode d'exécution de la simulation conduisent en général à des violations de la contrainte de causalité vue en section 1.1.4.

[3]Si les processus sont totalement indépendants (i.e. aucune interaction), il n'y a pas besoin de définir d'ordre.

3.3.2 Ordre causal sur les événements

Un des challenges de la simulation distribuée à événements discrets est de maintenir la relation causale entre les événements tout en exploitant le parallélisme inhérent pour exécuter plus rapidement les tâches. Maintenir la causalité signifie garantir un certain ordre séquentiel entre les événements s'exécutant en parallèle dans plusieurs processus différents. Rappelons que, informellement, le principe de causalité signifie qu'un événement futur ne peut influencer un événement passé ou encore que les causes doivent précéder les effets. Cette causalité implique ainsi la définition d'un ordre partiel des événements. Plus formellement un ordre partiel noté $\widehat{E} = (E, \rightarrow_E)$ est constitué d'un ensemble E appelé le domaine de l'ordre partiel et d'une relation binaire transitive et non-réflexive ($\rightarrow_E$) sur les éléments de ce domaine. Le fait de noter $e_1 \rightarrow_E e_2$ signifie que e_1 précède e_2 dans l'ordre partiel.

- Une relation binaire est non-réflexive si $\forall e \in E : e \rightarrow_E e$ est faux ;
- Une relation binaire est transitive si $\forall e_1, e_2, e_3 \in E : e_1 \rightarrow_E e_2 \wedge e_2 \rightarrow_E e_3 \Rightarrow e_1 \rightarrow_E e_3$.

Une exécution répartie est un ordre partiel noté $\widehat{H} = (H, \rightarrow_H)$. H étant l'ensemble des événements produits par les processus et $\rightarrow_H$ la relation de précédence causale [Lamport, 1978]. La précédente causale notée $\rightarrow_i$ est définie par les trois relations suivantes :

i. Relation *program-order* : deux événements produits par un même nœud sont totalement ordonnés ;

ii. Relation $receive\text{-}from$: $send(m)$ précède $receive(m)$;

iii. Relation de transitivité : (e_1 précède e_2) $\wedge$ (e_2 précède e_3) $\Rightarrow$ (e_1 précède e_3).

La figure 3.1 illustre ces trois règles présentant une exécution dans laquelle, par exemple, e_{31} précède e_{25} causalement. En effet, de l'utilisation de la règle (i) il est possible de déduire que e_{12} précède e_{13} qui précède e_{14} (les e_{1i} sont totalement ordonnés). La règle (ii) nous dit que e_{31} précède e_{12} et de même nous dit que e_{14} précède e_{25}. Il est par conséquent déductible par la règle de transitivité (règle iii) que e_{31} précède e_{25}.

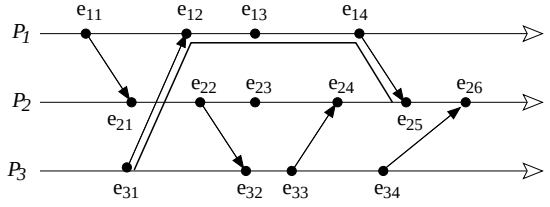


Fig. *3.1* — Précédence causale

Une exécution répartie est constituée d'un ensemble d'événements ordonnés par la précédence causale. Les synchronisations entre les processus pour assurer cet ordre sont modélisées par des messages estampillés entre ces processus. Les erreurs causales peuvent être évitées si chaque processus logique obéit à ses contraintes de causalité locale et interagit suffisamment par échange de messages.

3.3.3 Ordre sur les livraisons de messages

L'ordonnancement des événements selon les règles causales ne précise nullement comment se comportent les échanges d'information par envoi de messages. Par exemple un message peut

fort bien en doubler un autre sur le réseau de communication sans pourtant mettre en défaut le principe de l'exécution causalement cohérente. L'introduction de règles sur les livraisons des messages peut alors être nécessaire afin de conserver une simulation cohérente.

L'ordre *First In First Out* (*FIFO*) assure que si un processus P_1 émet deux messages à destination du même processus P_2 alors ces messages sont reçus dans l'ordre d'émission (relation d'ordre sur un canal de communication) [Charron-Bost et al., 1996]. Cet ordre est un ordre partiel sur les événements dans lequel est ajouté aux trois règles vues précédemment, la règle suivante : soit m_1 et m_2 deux messages destinés au même processus P_2 et émis par un même processus P_1. Cette règle nous assure qu'il n'existe pas de messages partis après $m1$ sur le même processus et qui arrivent avant lui chez le même destinataire.

iv. $send(m_1) \rightarrow_i send(m_2) \Rightarrow receive(m_1) \rightarrow_i receive(m_2)$

L'ordre causal [Birman et Joseph, 1987] est une extension de la précédence causale et une généralisation de l'ordre *FIFO* dans la mesure où il prend en compte la transitivité. Comme pour l'ordre *FIFO*, la règle *iv* est ajoutée afin d'assurer qu'il n'existe pas de messages partis causalement après un message m_i et qui arrivent avant celui-ci chez le même destinataire. La différence réside dans le fait que pour l'ordre causal aucune hypothèse n'est réalisée sur l'identité de l'émetteur. L'ordre causal est également un ordre partiel.

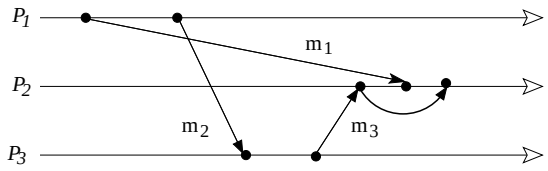


Fig. *3.2* — Exécution ne respectant pas l'ordre causal

La figure 3.2 présente une exécution qui respecte l'ordre *FIFO*, mais qui par contre ne respecte pas l'ordre causal. Pour que cette exécution respecte l'ordre causal il faut que m_3 soit délivré après m_1. En effet send(m_1) $\rightarrow_i$ send(m_3) (règle *iii*) implique que pour respecter la généralisation de la règle *iv* il faut avoir receive(m_1) $\rightarrow_i$ receive(m_3).

3.3.4 Ordre total

La relation de précédence causale est une relation d'ordre partiel. En conséquence tous les événements d'une simulation distribuée ne sont pas ordonnés, par exemple sur la figure 3.1 les événements e_{11} et e_{31} sont indépendants. Certaines simulations peuvent nécessiter un ordre total sur les événements qui est alors réalisé en définissant une extension linéaire [Charron-Bost et al., 1996] de l'ordre partiel établit. Une extension linéaire $\widehat{S} = (\mathrm{S}, \rightarrow_S)$ d'un ordre partiel $\widehat{H} = (\mathrm{H}, \rightarrow_H)$ est un tri topologique de cet ordre partiel, tel que :

- $S = H$;
- L'extension linéaire préserve l'ordre partiel précédemment établi : $\forall a, b \in H,\ a \rightarrow_H b \Rightarrow a \rightarrow_S b$;
- La relation $\rightarrow_S$ définit un ordre total.

La notion d'extension linéaire permet de définir un ordre total sur tous les événements de livraison des messages. Définir un ordre total sur les événements assure que lorsque deux processus P_1 et P_2 reçoivent deux messages m_1 et m_2, ceux-ci délivreront les deux messages dans

le même ordre quel que soit l'ordre d'émission. Ainsi les deux processus ont en permanence la même vue des livraisons de messages. Le recours a un ordre total à été souvent utilisé dans le cas d'exécutions réparties dans le modèle à communication par mémoire partagée. Dans ce schéma d'exécution les opérations d'envoi et de réception des messages sont remplacées par des opérations de lectures et d'écritures sur la mémoire partagée. La cohérence atomique est connue pour être le plus ancien critère ainsi que le plus fort. Ce critère a été introduit par Lamport [Lamport, 1986] dans le cas d'un seul écrivain (*single-writer*) puis étendu par Misra dans [Misra, 1986a] à plusieurs écrivains (*multiple-writer*) et consiste à ordonner totalement les événements.

La distribution des éléments du modèle nécessite des mécanismes de synchronisation explicites. Ces mécanismes dépendent de la nature, dirigée par le temps ou par les événements, de la simulation (cf. section 1.3). Ils dépendent également de la nature synchrone ou asynchrone de la simulation. Le choix entre simulation synchrone et simulation asynchrone consiste à choisir la manière dont va être résolu le problème d'ordonnancement des événements : ordre partiel causal, *FIFO* ou encore ordre total.

3.4 Simulation parallèle à événements discrets

La simulation parallèle à événements discrets (*Parallel Discrete Event Simulation* : *PDES*), souvent appelée simulation distribuée, consiste en l'exécution d'une simulation à événements discrets (cf. section 1.3) sur un environnement multiprocesseurs [Fujimoto, 1990]. Les méthodes de décomposition qui définissent cette approche sont celles que nous avons retenues à savoir :

- La distribution décentralisée des événements et ;
- La distribution des éléments du modèle.

Un simulateur, c'est-à-dire le programme de simulation, est un interpréteur de temps virtuel. La conception d'un simulateur distribué consiste donc à définir un schéma d'exécution répartie qui assure la progression du temps virtuel (propriété de vivacité) en respectant les relations de causalité du système simulé (propriété de sûreté) [Ingels et Raynal, 1989]. Les travaux initiés sur la simulation distribuée, notamment par Chandy et Misra, modélisent les systèmes simulés par un graphe de processus communiquant par échanges de messages (cf. section 1.3.3) pour décrire le modèle d'exécution répartie [Chandy et Misra, 1979]. Dans ce contexte, la difficulté majeure de la simulation distribuée consiste en la capacité qu'a le moteur de simulation à synchroniser les évolutions temporelles des événements générés et, par héritage à synchroniser les processus englobant. Cette synchronisation intervient quel que soit le choix de modélisation[4] et de répartition des processus. La simulation distribuée est conçue par la mise en œuvre des versions distribuées de la simulation dirigée par le temps et de la simulation dirigée par les événements de façon synchrone ou asynchrone. Dans cette section, nous présentons quelques éléments de compréhension des approches synchrones et asynchrones ainsi qu'un exemple de taxonomie des différentes techniques en fonction de leur synchronisme.

[4]Une liste locale d'événements sans nature particulière ou un composant qui modélise une entité du système complexe simulé.

3.4.1 Simulation synchrone

Une simulation synchrone est une simulation dans laquelle une horloge globale définit un temps simulé commun. La caractéristique fondamentale de cette approche est qu'à chaque instant le temps de simulation T_{sim} est a priori unique et identique pour tous les processus quel que soit leur nœud hôte. L'unicité du temps de simulation peut toutefois varier en fonction du degré de synchronisme voulu. Dans le cas d'un synchronisme fort, la différence entre deux pulsations des temps de simulation T_{sim-i} et T_{sim-j} guidant l'évolution de deux processus (ou composants) respectivement P_i et P_j est inférieure ou égale à un. A contrario, un synchronisme faible autorise une différence entre deux temps de simulation distincts pouvant être supérieure à une pulsation δ (tout en restant bornée par un nombre faible). Ce type de simulation est facilité par l'utilisation d'une architecture synchrone (dans la taxonomie de Duncan) qui possède une horloge globale physique sur laquelle peut s'appuyer le temps virtuel. Cependant les systèmes distribués (i.e. architecture retenue) sont des architectures asynchrones dans lesquelles chaque machine ou chaque processeur possède sa propre horloge. Pour fournir un temps global, il faut alors être en mesure de faire progresser les temps simulés locaux de concert par l'introduction de mécanismes et de techniques ad hoc.

La simulation synchrone dirigée par le temps est une approche de simulation dans laquelle l'horloge globale est utilisée pour guider l'avancement de chaque composant de la simulation. [Gilmer et Hong, 1986] présente un exemple de ce type de simulation sur une architecture multiprocesseurs fortement couplée (simulation sur un système à mémoire partagée) où sont simulés des scénarios de combat.

La simulation synchrone dirigée par les événements est une approche dans laquelle chaque processus (ou composant), constituant de la simulation, progresse événement par événement. L'utilisation d'une horloge globale dont la mise en œuvre est centralisée ou répartie permet d'exécuter de manière synchrone les événements distribués. Ce type de simulation par une mise en œuvre centralisée de la simulation est présenté par exemple dans [Venkatesh et al., 1986]. Ces auteurs proposent une horloge globale dans laquelle le temps courant est défini par le temps minimum du prochain événement à simuler qui implique une interaction entre des processus. Une version de la simulation distribuée synchrone impliquant une mise en œuvre décentralisée de l'horloge globale est présentée dans [Peacock et al., 1979]. Ces auteurs introduisent une approche parfois appelée *liste multiple d'événements synchronisés* dans laquelle le synchroniseur distribué attend, à chaque pulsation, que toutes les activités réparties soient terminées pour avancer l'horloge globale (synchronisme fort).

3.4.2 Simulation asynchrone

Dans le cas d'une simulation asynchrone, les différents processus possèdent leur propre horloge locale pour guider leur avancement conduisant chacun de ces processus à utiliser un temps de simulation différent des autres. Dans ce type de simulation, il faut noter qu'il y a une différence entre l'ordre chronologique d'apparition des événements et les relations de causalité entre les événements (au moins pour des événements produits par des processus différents).

Le mode de fonctionnement de la simulation asynchrone dirigée par les événements repose sur l'utilisation d'un temps local T_{sim-i} dont est doté chaque processus P_i. Chaque événement exécuté fait progresser les horloges locales de manière indépendante car la simulation est asynchrone. L'envoi d'un message m par P_i à P_j à l'instant t se traduit par l'envoi d'un doublet (m, t) qui provoque la mise à jour de T_{sim-j} sur P_j. Le problème central de ce

type de simulation est le respect du principe de causalité. Afin de garantir le respect de ces contraintes par le simulateur, différentes stratégies ont été proposées :

- La stratégie pessimiste (ou cohérente à priori) consiste à prévenir l'occurrence de toute erreur causale en déterminant par avance s'il est viable d'exécuter un événement ;
- La stratégie optimiste (ou cohérente à posteriori) consiste à autoriser des erreurs causales dans l'exécution distribuée et à utiliser des mécanismes de détection et de recouvrement tels que le retour arrière (*rollback*) pour rétablir la simulation dans un état stable.

L'un des problèmes majeurs de la simulation pessimiste est le risque d'interblocage. Cette situation d'interblocage ne correspond pas à un blocage du modèle ou de sa mise en œuvre distribuée, mais résulte de la mise en œuvre du schéma d'exécution destiné à garantir le principe de causalité[5]. Dans ce type de simulation pessimiste, deux tendances se dégagent pour faire face à ce problème :

- La prévention : un algorithme de contrôle est intégré au simulateur afin d'éviter l'apparition de blocage. [Chandy et Misra, 1979] présentent une solution basée sur l'émission de messages supplémentaires (messages *null*).
- La détection et la guérison : le simulateur exécute la simulation tant que cela est possible, en respectant les contraintes de causalité. Un algorithme distribué détecte l'interblocage une fois que celui-ci est survenu, le traite et relance un des processus [Chandy et Misra, 1981].

Le temps virtuel (*time warp*) est une stratégie cohérente à posteriori introduite par [Jefferson, 1985]. Cette technique n'impose aucune contrainte sur l'évolution asynchrone des processus de la simulation : chacun avance à son propre rythme et l'horloge locale progresse en conséquence. Lors de l'arrivée d'un message (m, t), le processus récepteur P_j fait progresser son horloge jusque t et traite le message m, ce qui peut provoquer des envois de messages estampillés avec des valeurs supérieures ou égale à t. Lorsque le temps local t' est supérieur à t il faut alors revenir en arrière car les événements exécutés dans l'intervalle de temps compris entre t' et t n'auraient pas dû l'être. Cette stratégie nécessite donc de conserver les états successifs des variables locales de chaque processus, la liste de tous les messages émis et reçus. Le temps virtuel global (GVT) introduit dans l'approche de Jefferson représente une borne inférieure sur les dates de retour arrière possibles : toutes les actions antérieures au GVT ne seront jamais défaites (ce mécanisme permet de limiter l'emploi exagéré de ressources matérielles, e.g. mémoire).

3.4.3 Exemple de taxonomie

La complexité des mécanismes mis en œuvre pour la simulation distribuée tant du point de vue logiciel qu'architectural, et la difficulté d'établir des nuances entre les techniques de résolution notamment à cause de leurs subtiles différences, entraînent quelques difficultés pour classer les différents travaux existants. Dans [Peacock et al., 1979], la simulation distribuée est présentée par une taxonomie basée sur le couplage fort ou faible de l'architecture sous-jacente. En nous basant sur les mécanismes et les terminologies couramment employés dans le domaine de la simulation, en particulier tels que présentés dans [Righter et Walrand, 1989]

[5]Le principe de précaution implique que certains processus risquent d'être bloqués, en attente de messages confirmant qu'ils peuvent avancer.

et [Ferscha, 1995] nous proposons une taxonomie de la simulation à événements discrets. La figure 3.3 illustre cette classification. Nous y retrouvons la simulation dite séquentielle (mono-machine) présentée en section 1.3 et la simulation distribuée (multiprocesseurs).

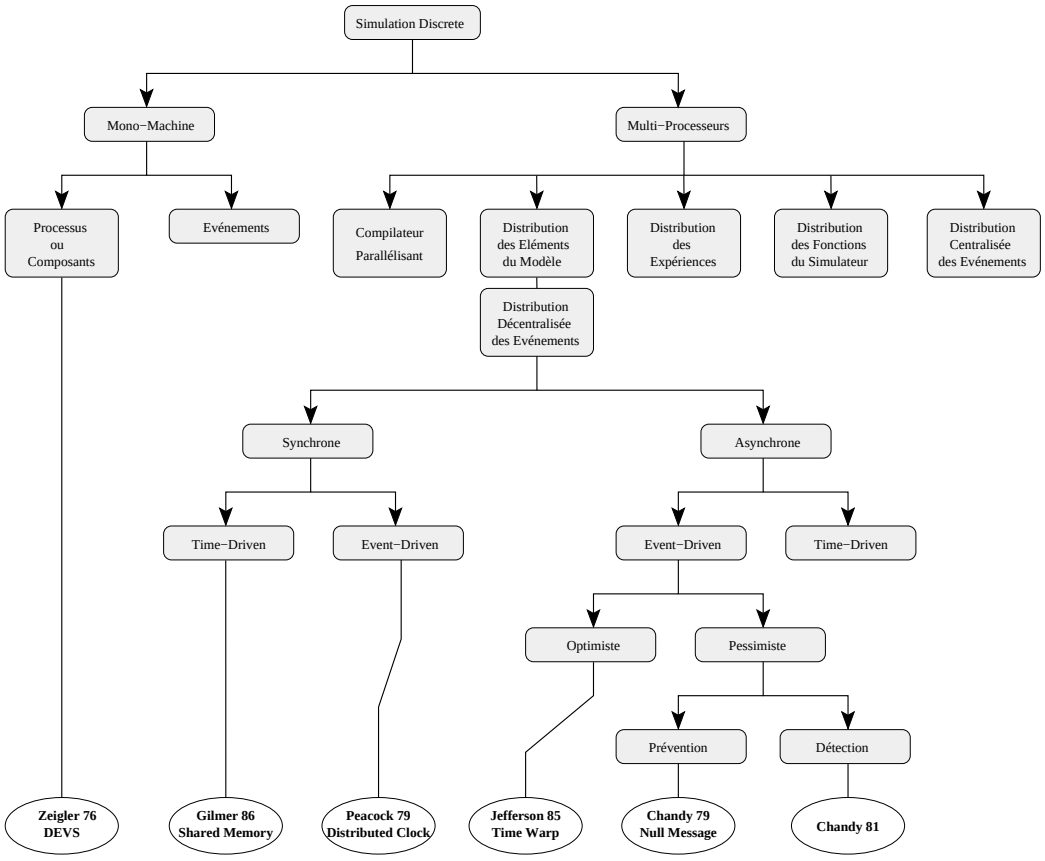


Fig. *3.3* — Taxonomie de la simulation à événements discrets

Remarquons que dans la littérature la terminologie *Parallel Discrete Event Simulation : PDES* fait en règle générale référence à la deuxième branche de la simulation multiprocesseurs (celle qui est la plus développée). Notons par ailleurs que parfois un amalgame est effectué entre, d'une part les simulations synchrones et les simulations dirigées par le temps et d'autre part entre les simulations asynchrones et les simulations dirigées par les événements. A contrario [Righter et Walrand, 1989] considèrent quatre cas différents tels qu'illustrés en figure 3.3. Nous retrouvons dans cette figure les différents cas abordés dans les sections précédentes accompagnés d'une référence représentative. De nombreuses autres études ont été proposées pour chacun de ces cas dont certaines sont présentées dans [Ferscha, 1995, Righter et Walrand, 1989, Beaumont, 1998, Maziero, 1994]. Par exemple Bernard Zeigler présente une approche de la simulation distribuée par l'utilisation du formalisme *DEVS* dans [Zeigler et Sarjoughian, 2000]. Dans [Zeigler et al., 2000] ce même auteur présente *Parallel Hierarchically coupled DEVS* qui définit une stratégie optimiste et *Parallel Modular DEVS* qui est une stratégie pessimiste.

La simulation distribuée implique un surcoût de calcul pour assurer la coordination. Elle nécessite une grande quantité de messages de contrôle qui ne correspondent à aucune action pertinente pour la simulation, qui n'existent pas dans une simulation séquentielle, qui nécessitent du temps processeur et impliquent des ralentissements dans l'exécution. Une so-

lution asynchrone ne peut pas s'appliquer à tous les cas de simulations car elle repose sur un postulat fort qui est que chaque processus peut avancer dans le temps de simulation de manière indépendante des autres. Cette solution est envisageable dans le cas où les processus ont peu, voire aucun, échange d'information (relations causales). Cette technique, quand elle s'applique, impose une gestion des synchronisations et des erreurs pour éviter les interblocages ou réduire les retours arrière. La gestion de tels mécanismes est en règle générale transparente pour l'utilisateur, mais nécessite de nombreuses tâches lourdes à réaliser pour le système informatique. La stratégie optimiste, en particulier, ne favorise pas la simulation en temps réel et l'interaction avec une interface graphique potentielle. Les retours arrière répétitifs dans la simulation constituent par ailleurs un recul dans le temps de calcul et de ce fait une charge supplémentaire qui in-fine relève d'une mauvaise utilisation du système distribué supportant la simulation.

Dans la simulation asynchrone, les ressources informatiques sont utilisées au maximum sans à priori de difficultés concernant l'asymétrie potentielle de performance des ordinateurs. Remarquons cependant qu'indépendamment du synchronisme de simulation choisie, un problème issu d'une machine plus lente que les autres peut survenir. En effet dans le cas de dissymétries (processeurs et/ou réseau de communication) dans l'architecture, les différents processus peuvent avancer (de manière synchrone ou asynchrone) mais sont régulièrement bloqués par des barrières de synchronisation (ou contraint au retour arrière) limitant ainsi l'utilisation de la pleine puissance du système distribuée. L'approche asynchrone tend à minimiser ce problème voire, tente de le résoudre en autorisant (de par sa nature) l'évolution asynchrone des processus répartis sur les différents nœuds du système distribué.

L'approche que nous retiendrons par la suite est la simulation distribuée synchrone dirigée par le temps. Cette solution est, comme sa version séquentielle, particulièrement appropriée pour les systèmes dans lesquels il se produit beaucoup d'événements. Cette approche favorise un suivi graphique car tous les processus avancent en même temps. La perte de performance souvent supposée par rapport à une solution asynchrone qui minimiserait les problèmes de causalité impliquant des retours arrières peut être améliorée par la variation du pas de simulation (pas plus grand) lorsque l'activité (i.e. le nombre d'événements) est faible (en considérant que le délai de transfert d'un message dans cette approche ne devrait pas excéder le temps d'une pulsation δ).

3.5 Mécanisme de migration

Le mécanisme de migration peut être amené à jouer un rôle important dans la simulation distribuée. Dans son principe, la migration consiste à déplacer des entités informatiques d'un ordinateur vers un autre. Ces entités dites unités de migration peuvent être des processus lourds ou légers (*thread*), des agents ou des objets simples. La migration de processus ou d'agents opère traditionnellement en trois étapes [Muller et Sens, 1997] : (1) le gel du processus ou de l'agent sur le nœud source (incluant la capture de ses paramètres d'exécution, e.g. pile système), (2) l'envoi des pages mémoires ou d'une image vers le nœud de destination, et (3) le réveil sur le nœud distant. Lorsque les paramètres d'exécution ont pu être sauvegardés, le processus ou l'agent peut reprendre son exécution à l'endroit où il s'était arrêté, autrement il reprend son exécution au début de son code. Dans le cas de migration d'objets simples, le mécanisme est moins complexe car contrairement aux processus et aux agents, l'objet est passif (i.e. il ne possède pas de flot de contrôle). Le mécanisme classique de migration consiste à (1) capturer l'état de l'objet afin d'en produire une image (on parle de sérialisation), à (2)

envoyer l'image vers le nœud distant, et (3) à désérialiser l'objet et à lui associer une référence (un nom). Le nom de l'objet peut être défini à l'aide d'un mécanisme de nommage global qui permet de retrouver cet objet quel que soit son emplacement dans le système distribué. Si un tel mécanisme n'existe pas, alors on lui attribue une nouvelle référence. Remarquons que le problème de nommage se présente également lors du déplacement de processus ou d'agent. Dans les sections suivantes nous présentons les apports de la migration pour la simulation ainsi qu'un bref état de l'art des systèmes fournissant un mécanisme de migration.

3.5.1 Optimisations

La migration est le mécanisme utilisé pour le contrôle de charge dynamique qui est d'un grand intérêt pour la simulation distribuée [Zeigler et al., 2000]. Considérons une simulation dans laquelle les événements répartis sont traités par tranche de temps Δt. Dans ce schéma, chaque processus ou composant attend les autres pour exécuter la tranche de temps Δt_i. Lorsqu'un nœud a une charge de calcul plus importante que les autres, celui-ci peut prendre du retard et ainsi bloquer tous les processus sur Δt_i pendant que ce dernier exécute toujours Δt_{i-1}. Ce problème peut par ailleurs être accentué par une dissymétrie dans la puissance des nœuds de calcul utilisés pour la simulation. De ce fait, la distribution initiale des processus ou des composants de la simulation répartie nécessite parfois des changements pendant l'exécution. Le contrôle de charge dynamique (*dynamic load balancing*) consiste à migrer les processus, agents ou objets d'une application distribuée en tentant de répartir équitablement les charges de calculs sur les différents nœuds disponibles et en tentant de réduire (d'optimiser) le nombre de communications entre les machines [Nuttall, 1997]. Cette technique repose sur l'utilisation du mécanisme de migration et également sur l'utilisation nécessaire d'une politique de migration dynamique généralement basée sur la capture de l'état des processeurs et des communications inter-processus qui permet de déterminer la répartition adéquate des processus.

Une approche similaire au contrôle de charge consiste à améliorer la performance du système distribué et ainsi la vitesse d'exécution des applications distribuées en partageant (*load sharing*) au mieux les ressources processeurs [Bernard et Simatic, 1991, Kremien et Kramer, 1992]. L'objectif du partage de charge est d'assurer qu'aucune machine est en attente alors que d'autres sont surchargées. Cela est réalisé en exécutant les applications sur les nœuds où elles disposeront d'un maximum de ressources grâce à un mécanisme d'exécution à distance des processus (*remote execution*). Cette approche convient particulièrement aux simulations utilisant une décomposition par distribution des événements à partir d'une liste globale.

De nombreux systèmes, librairies ou environnements fournissent un support pour la migration de processus. Certains pourraient constituer un support acceptable pour implémenter un simulateur distribué, d'autres permettent une réflexion sur les mécanismes et techniques exploitables pour la construction d'un simulateur ad hoc souhaitant exploiter la migration de processus ou d'agents.

Parmi ces supports de migration, quelques-uns fonctionnent sous forme de processus démons sur le système d'exploitation UNIX. Freedman par exemple propose dans [Freedman, 1991] un mécanisme simple de migration des processus UNIX en déplaçant une copie de leur image. L'utilisation de ce système est soumise à l'insertion de code supplémentaire dans le programme afin de déterminer les moments où la migration est possible. Condor [Bricker et al., 1991] contourne ce problème en réalisant une sauvegarde régulière des processus afin de définir des points de reprise (*checkpoint*). Un gestionnaire central contrôle l'état

des machines disponibles afin de déplacer les processus en fonction des charges de chacune. L'unité de migration est l'image sauvegardée du processus ce qui implique qu'après une migration un processus peut être réactivé dans un état antérieur puisque le processus continue d'évoluer après sa capture.

D'autres mécanismes de migration reposent sur l'utilisation de systèmes prototypes ayant des fonctionnalités diverses. Par exemple Zayas présente dans [Zayas, 1987] l'intégration d'un support de migration de processus pour le système Accent [Rashid et Robertson, 1981] développé à l'université Carnegie Mellon. Ce greffon réalise une migration par une technique de copie sur référence dans laquelle les pages physiques du processus ne sont réellement déplacées que lorsque utilisées sur la machine cible.

Le système d'exploitation distribué à objets Amoeba développé à l'université d'Amsterdam fonctionne sur un ensemble de stations de travail hétérogène [Mullender et al., 1990, Steketee et al., 1994]. La migration de processus démarre par le blocage du processus puis par l'envoi de son descripteur vers le site distant. Le contenu des pages mémoires du processus est ensuite transféré après quoi le processus migrant est réactivé alors que la copie initiale est supprimée. Remarquons que la migration ne laisse aucune dépendance résiduelle (i.e. de pointeur vers la nouvelle affectation). Pour retrouver un processus migrant une diffusion globale (*broadcast*) est alors réalisée.

Charlotte est une plate-forme développée pour l'expérimentation d'algorithmes distribués [Artsy et Finkel, 1989, Artsy et al., 1986]. Cet environnement fournit un mécanisme de migration en trois phases : une négociation, un transfert, et un rétablissement. Certains processus de chaque noyau de Charlotte (les *starters*) assurent la collecte des informations en vue d'un contrôle de charge dynamique. A partir des informations collectées ceux-ci entament les phases de négociations pour contrôler les migrations de processus qui sont effectuées par copie de pages mémoires. Ce système permet par ailleurs une localisation des processus indépendamment de leur affectation par un nommage global et, l'autorisation pour un site de refuser la migration.

Le système d'exploitation distribué DEMOS/MP, de l'université de Berkeley, fût l'un des premiers à offrir un mécanisme de migration transparent pour l'utilisateur [Powell et Miller, 1983]. Ce système n'est pas conçu pour le contrôle de charge mais utilise toutefois un mécanisme de gestionnaire de migration qui maintient des informations sur l'activité du système afin de déterminer, en fonction des besoins de l'utilisateur, quel processus se déplace et vers quel site. Notons que la migration de processus est implémentée à l'aide de dépendances résiduelles laissées sur chaque nœud visité par un processus.

Locus est un système d'exploitation distribué, résultant d'une modification du système UNIX et incluant la migration et le contrôle de charge par exécution distante. Une identité unique de processus (UPID) est utilisée et couplée avec des dépendances résiduelles laissées sur les sites visités.

Mosix (Multicomputer OS) [Barak, 1989, Barak et Litman, 1985] est un système compatible avec UNIX [Walker et al., 1992]. Dans ce système un processus peut migrer soit par décision du mécanisme de contrôle de charge, soit par un appel système qui peut être réalisé par un autre processus. La migration qui est réalisée par un transfert des pages mémoires fournit un mécanisme rapide. Ce système a par contre le désavantage de perdre les messages émis vers les processus lors de leur migration.

Dans le système réseau Sprite, lorsqu'une migration est décidée, un nouveau processus est créé sur le site distant avec aucune page mémoire [Douglis et Ousterhout, 1987, Douglis et Ousterhout, 1989]. Les pages mémoires sont ensuite envoyées depuis le proces-

sus source afin d'être chargées dans l'espace vide. Chaque machine Sprite utilise un démon qui calcule la charge du processeur local. Lorsqu'une machine en attente de travail informe un serveur central de migration, elle reçoit un processus.

Le système distribué V développé à Stanford fonctionne sur un ensemble de stations de travail sans disques durs et utilise à la place un serveur de fichier centralisé [Theimer et al., 1985]. Ce système permet un contrôle de charge automatique par la migration de processus et par exécution distante. L'unité de migration est la page mémoire et le système ne laisse aucune dépendance résiduelle après son déplacement.

D'autres types de systèmes tels que les systèmes micro-noyau ont également pris en compte le mécanisme de migration de processus. Mach, par exemple, est un système micro-noyau qui fournit un mécanisme de migration et de contrôle de charge [Milojicic et al., 1993]. Dans ce système, l'unité de migration est la tâche qui est déplacée par migration de son espace d'adressage (code inclut). Le système Mach, contrairement à Mosix par exemple, conserve les messages d'un processus en mémoire tampon lorsque celui-ci est en migration. Les processus sont retrouvés par un démon maintenant une liste des processus présents sur chaque machine. Dans Chorus, un autre système micro-noyau, l'unité de migration est l'acteur (équivalent d'un processus UNIX) [O'Connor, 1994]. Un module de politique de migration permet un contrôle de charge dynamique. Lorsque cela est nécessaire, ce module choisit parmi les processus qui sont dans un état stable pour les faire migrer. Une encapsulation de l'acteur est ensuite envoyée au nœud distant, puis la copie d'origine est supprimée sur demande du processus migrant. L'accès distant à un processus est assuré par un gestionnaire réseau qui réalise des diffusions (*broadcast*) afin de retrouver les processus.

Tous ces systèmes, énumérés par ailleurs dans [Nuttall, 1994], fournissent un mécanisme de migration standard et éventuellement un mécanisme supplémentaire pour le contrôle de charge. Chacun possède ses caractéristiques propres, mais ces systèmes ont en commun l'objectif de déplacer des processus individuellement.

3.5.2 Modélisation de flux

Dans le cas de simulation de systèmes à large flux de données désagrégées, la migration peut également être d'un grand intérêt. Les travaux théoriques sur la simulation abordent peu la nature des messages échangés. La raison étant que ces travaux focalisent sur la préservation d'une exécution distribuée causalement cohérente et sur l'implication qu'a le message dans la mise en défaut potentielle de la causalité. Dans ce contexte, le respect de la causalité peut être réalisé indépendamment du type de l'information véhiculée. Ces messages ont généralement une forme de doublet (t, m) où t représente l'estampille temporelle et où m représente l'information véhiculée. Le type de m peut être très varié : une chaîne de caractères représentant une interaction entre deux processus, un marqueur représentant l'ordre d'exécution d'un événement, une matrice sur laquelle un calcul est à effectuer, etc. Dans le schéma de distribution des éléments du modèle, la simulation est décomposée en un ensemble de composants du système traditionnellement mis en œuvre par des objets ou des agents répartis sur les différents processeurs. En fonction du système simulé, les composants peuvent se déplacer d'un processeur à l'autre, non pas pour améliorer la performance mais, pour reproduire le mouvement naturel des composants de l'application. Dans ce cas le message n'est pas le simple envoi d'une information de type objet mais bien la migration d'un composant du système qui va poursuivre son cycle de vie sur une autre machine emmenant avec lui son expérience passée, son état actuel et ses objectifs. Dans ce contexte, la migration est avant tout le mécanisme indispensable pour déplacer des objets représentant les entités de granu-

larité fine (désagrégée) entre les différents niveaux de hiérarchie du modèle qui sont répartis sur les différents nœuds du système distribué. La migration répétitive des objets au sein du système distribué permet ainsi de représenter et mettre en œuvre les échanges intervenant dans la simulation de systèmes complexes à large flux de données désagrégées.

Certains systèmes fournissent un mécanisme de migration d'objet à différents niveaux de granularité. Par exemple Zenith [Davies et al., 1993] est un système distribué multimédia, implémenté en tant qu'extension du système ANSAware [Herbert, 1994]. Dans ce système qui ne fournit pas de mécanisme de contrôle de charge, les objets sont encapsulés dans des capsules équivalentes à des processus UNIX. L'unité de migration est la capsule. Une capsule peut contenir plusieurs objets qui peuvent migrer entre les capsules.

Le système *Persistent Object Infrastructure* est également basé sur ANSAware [Olsen, 1992]. Dans ce système, qui ne fournit pas de contrôle de charge, l'unité de migration est l'objet simple (passif). Exploitable uniquement par un administrateur système, la migration est utilisée uniquement comme un mécanisme de reconfiguration du système. Lorsqu'un objet est invoqué, il est gelé puis une image de l'objet est réalisée. Un nouvel objet vide est ensuite créé sur un site distant dans lequel l'image est réinstallée. Finalement, l'objet source est détruit et le nouvel objet est mis à disposition du système. Remarquons l'utilisation d'un service de localisation permettant de retrouver des objets ayant migrés.

Le système COOL est la couche logicielle orientée objet du système Chorus [Habert et al., 1990, Chandra et al., 1993]. Ce système implémente un mécanisme d'objets persistants pouvant migrer. Le contrôle de charge ne prend pas en compte la gestion des charges des processeurs, mais consiste principalement à réduire les communications entre les objets en les migrant. Le système COOL supporte trois types de migration : les objets peuvent migrer d'un nœud à l'autre au sein d'un espace d'adressage distribué. Dans ce cas, le partage de l'espace d'adressage permet aux objets d'apparaître constamment au même endroit de la mémoire. Les objets peuvent également migrer dans des espaces d'adressage différents et enfin les objets en mémoire secondaire (e.g. stockage disque) peuvent migrer vers un espace d'adressage distribué sur n'importe quel nœud.

Le système Emerald est composé d'un langage orienté objet et d'un noyau d'exécution qui fournit un mécanisme de migration d'objet [Jul et al., 1988]. Dans ce système, les objets passifs et actifs (i.e. agent) peuvent migrer. Ce système fait l'hypothèse qu'une copie du code de l'objet existe sur le nœud de destination. Ainsi, la migration consiste à déplacer l'état des données sur la machine distante. Lorsque le noyau de la machine cible reçoit un message, il alloue de l'espace mémoire dans lequel il insère la copie de la zone de données copiée. Notons que Emerald intègre un mécanisme de groupe dans lequel une opération peut être appliquée à tous les objets du groupe. L'action sur un ensemble d'objets peut, en particulier, être une demande de migration pour les membres. Toutefois ceux-ci migrent individuellement et non par groupe.

Le système DC++ est un environnement orienté objet dans lequel l'unité de migration est un objet C++ [Dilley, 1993]. Il ne supporte pas la migration du code ce qui implique la connaissance par le nœud de destination de la classe de l'objet. Un objet devant se déplacer est d'abord gelé, puis un appel de procédure à distance est réalisé sur la machine distante avec en paramètre l'état de l'objet. Un nouvel objet est alors créé avec ces données de configuration. Le succès de l'opération renvoie une valeur permettant à la machine source de détruire son instance de l'objet.

Le système *Secure Open Multimedia Workstation Operating System* (en abrégé SOMIW ou SOS) définit un environnement de gestion d'objets pour le système d'exploitation UNIX

[Shapiro, 1986]. Ce système fournit un mécanisme de migration d'objets réalisé soit à la suite d'un appel distant soit par décision opérée localement. La migration consiste tout d'abord à bloquer l'objet, puis à envoyer une copie de son espace d'adressage (données, mais également le code si nécessaire) vers un nouvel espace mémoire distant. L'objet est ensuite réactivé et sa copie originelle est supprimée. L'utilisation, dans ce système, d'un identifiant unique contrôlé par un gestionnaire de noms permet de retrouver l'objet après sa migration.

Le langage objet Java offre également des mécanismes permettant la migration d'objets [Arnold et Gosling, 1996]. Le mécanisme de migration en Java possède quelques grands avantages : (1) tout d'abord, il repose sur un mécanisme simple d'utilisation : la sérialisation qui consiste à réaliser une image des objets et de leur dépendance hiérarchique (permet de mettre en œuvre le déplacement d'un groupe d'objets). Par ailleurs (2) l'environnement d'exécution du langage Java (la *Java Virtual Machine* : *JVM*) est portable, ce qui permet la migration des objets sur presque n'importe quel type d'architecture [Gosling et McGilton, 1995]. Contrairement à la plupart des systèmes présentés ci-dessus, (3) le langage Java fournit un système standard très largement utilisé. Enfin, (4) sa programmation totalement objet fait référence et convient parfaitement pour la mise en œuvre de simulation dans lesquelles le modèle est décomposé en fonction de ses éléments. Le langage Java n'est pourtant pas exempt de désavantages. En effet, à cause de la sérialisation/désérialisation, le déplacement des objets est aisé mais lent. Par ailleurs il est impossible, dans l'implémentation actuelle de la *JVM*, d'accéder aux registres système afin de capturer l'état des processus ou des agents. Certains travaux comme [Bouchenak et Hagimont, 2002] ou [Courtrai et al., 2001] visent à améliorer la performance de la migration et l'intégration de la possibilité de déplacer du code.

La contribution du mécanisme de migration est donc de deux ordres pour la simulation distribuée. Elle permet dans le cas du déplacement des entités ayant un flot de contrôle (processus ou agent) d'optimiser les performances de la simulation. Appliquée au déplacement d'objets ou d'agents du modèle, la migration permet de reproduire les mouvements naturels internes au système. Dans notre contexte de modélisation et de simulation, l'intérêt de la migration réside moins dans sa capacité à améliorer les performances de la simulation par un déplacement de charge, qu'en tant qu'outil permettant de déplacer des objets afin de reproduire le comportement des flux de données. Les mécanismes de migration précédemment cités reposent sur le principe de migration individuelle. A notre connaissance il n'existe pas de mécanisme de migration prenant en compte (et permettant de modéliser) les phénomènes de groupes, phénomènes que l'on peut observer lors de l'étude de système à large flux de données et en particulier dans les systèmes urbains et les systèmes en transport. Un mécanisme de migration en groupe pourrait représenter ces flux et par ailleurs optimiser les réseaux de communications.

3.6 Systèmes distribués et simulation

Les mécanismes présentés dans ce chapitre décrivent les bases sur lesquelles reposent les principes des simulations distribuées à événements discrets. Ces techniques ont acquis une grande maturité et sont utilisées pour de nombreuses simulations. L'intérêt des systèmes distribués pour la simulation de système complexe a été identifié et, par ailleurs selon Andrzej Bargiela, les systèmes distribués offrent de nouvelles perspectives pour la simulation de systèmes dynamiques à grande échelle [Bargiela, 2000]. Les apports potentiels des systèmes distribués pour la simulation de systèmes désagrégés, en particulier en transport où le besoin de puissance et d'expressivité adaptée est clairement postulé, sont multiples. Ils constituent

des systèmes qui adaptent la répartition des données et des calculs à un niveau de granularité approprié afin de reproduire la nature sous-jacente et le comportement d'un système désagrégé à grande échelle. Ils favorisent une répartition d'environnements de simulation à différents niveaux de granularité et supportent l'extension de réseaux ainsi constitués (locaux ou étendus comme l'Internet). Ils permettent une forme de calcul collaboratif dans l'exécution d'une simulation complexe et assurent la répartition et l'interopérabilité dans des contextes spatialement distribués [Voisard et Schweppe, 1998].

Dans [Merchant et al., 1995] un système distribué reproduit des données spatiales où chaque nœud du système distribué modélise une partie de l'espace de ces données décomposées à partir d'un index spatial. Les objets simulés supportent la migration à titre individuel, et ces migrations sont contrôlées en fonctions des lignes de segmentation partitionnant l'espace. Dans [Maniatty et al., 1993], un modèle décomposé spatialement représentant des épidémies se comporte comme un automate cellulaire probabiliste dans lequel des cellules partitionnent l'espace afin de rendre la simulation sur un environnement distribué possible. Dans un travail comparable, un système pour la simulation parallèle a été développé pour la modélisation de terrains [Maxwell, 1997]. A chaque processeur est assigné une partition différente de l'espace, chaque région de l'espace étant structurée par un ensemble de cellules. Les communications interprocesseurs sont basées sur des relations de voisinage. Ceci permet à l'automate cellulaire modélisant les terrains d'être exécuté efficacement.

Les systèmes distribués partagent plusieurs caractéristiques, en particulier ils reposent sur des propriétés issues de la théorie des graphes pour les modéliser. L'utilisation de ces propriétés convient à la simulation sur des environnements distribués, des phénomènes complexes réels dans lesquels interviennent des flux de données. En effet, la simulation de systèmes complexes à large flux de données désagrégées implique la propagation d'objets souvent asynchrone, décidée à un niveau local (i.e. sur chaque nœud), et où les objets sont autonomes dans le sens où leurs comportements sont indépendants. Il est possible d'établir un parallèle entre les migrations des objets et les propagations de messages initialement mises en œuvres dans les systèmes distribués. La propagation dynamique de messages pour résoudre des problèmes basés sur les graphes ont été parmi les premiers développements réussis des systèmes distribués. Par exemple, l'algorithme *Echo* propage une vague de messages se répandant à partir d'un nœud initial vers tous les voisins, de proche en proche, jusqu'à une couverture totale des nœuds par vagues successives [Chang, 1982]. Dans cet algorithme des messages en retour (*backward message*) sont émis lorsqu'un message atteint un nœud déjà visité par la vague. L'algorithme Echo est utilisé pour répliquer les problèmes tels que celui des plus courts chemins.

Basé sur des principes de similaires, plusieurs systèmes ont été implémentés pour propager des objets autonomes et asynchrones au travers de systèmes distribués sous-jacents [Bic et Lee, 1987, Sapaty et Borst, 1994]. Récemment, l'intégration d'agents autonomes dans les systèmes distribués a entraîné l'émergence de nouveaux systèmes académiques ou industriels dans lesquels les agents sont capables de se déplacer seuls au sein de réseaux locaux et étendus afin d'accomplir des taches spécialisées [Kotz, 1997, Baumann et al., 1998, Wong et al., 1997, Milojicic et al., 1998]. Une nouvelle classe d'application des systèmes distribués simule des comportements fortement décentralisés d'agents dans (et représentant) un monde réel. Le concept d'objets autonomes dans les systèmes distribués a été notamment utilisé par le paradigme de programmation *messengers* [Fukuda et al., 1999] afin de simuler la diffusion du métabolisme de toxines variées dans différents organes d'un organisme vivant. Chaque nœud du système distribué correspond à un organisme humain et un langage de contrôle coordonne les opérations et les interactions de fonctions spécialisées. Cette

même approche est également utilisée pour reproduire les déplacements de poissons (vitesse et orientation) dans leur environnement [Bic et al., 1996].

Les systèmes multi-agents apparaissent comme des solutions de modélisation informatique prometteuses des comportements fortement décentralisés [Ferber, 1995]. Les applications considérant l'étude d'entités désagrégées constituent une forme de système réparti et non linéaire fait d'agents autonomes qui se comportent, agissent l'un sur l'autre, évoluent et produisent des modèles globaux dont l'analyse, la prévision et la planification sont des objectifs importants de beaucoup d'études urbaines. Dans le domaine de la compréhension des phénomènes urbains les utilisations d'architectures multiprocesseurs sont encore peu nombreuses. [Erol et al., 1999] propose une approche d'agents distribués (système multi-agents Cybele) pour la simulation de trafic. Dans [Jha et al., 1995] est présenté un environnement urbain simulé et structuré en régions distribuées dans lesquelles se déplacent des véhicules entre des zones réparties sur différents processeurs. [Rickert, 1997] et [Nagel, 2002] utilisent également une répartition spatiale afin de simuler des trafics routiers à l'aide d'architectures distribuées. Le simulateur mis en œuvre par Rickert intègre par ailleurs des mécanismes de migrations et de contrôle de charge, basés sur les charges de processeurs. Notons enfin la proposition d'une combinaison explicite des techniques de simulations distribuées à événements discrets et de l'utilisation de systèmes multi-agents dans [Logan et Theodoropolous, 2001].

3.7 Conclusion

Ce chapitre présente les principes et les propriétés de la simulation distribuée à événements discrets. La première section de ce chapitre illustre les différents types d'architectures multiprocesseurs. Parmi celles-ci, les systèmes distribués constituent une classe très importante pour la simulation. La section 3.2 présente plusieurs méthodes de décomposition des systèmes complexes en vue de leur exécution sur une architecture multiprocesseurs. Les deux méthodes de décomposition les plus exploitées pour la simulation à événements discrets, à savoir (1) la distribution décentralisée des événements et (2) la distribution des éléments du modèle ont pour objectif de support les systèmes distribués. La section 3.3 définit la notion d'exécution répartie d'événements dans un modèle par échange de messages. La section 3.4 présente les principes de la simulation distribuée des systèmes complexes dans lesquelles le respect du principe de causalité est un aspect important mais qui implique un surcoût non négligeable dans l'exécution répartie des événements. La section 3.5 présente les principes de migration d'entités au sein d'un système distribué. Deux aspects importants de cette technique sont identifiés pour la simulation distribuée. Tout d'abord la migration peut être utilisée pour le déplacement d'entités actives tels que des agents afin d'équilibrer au mieux les ressources processeur et réseau. Par ailleurs, dans le cas où l'étude de flux de données est un aspect qualitatif de l'étude d'un système complexe, la migration permet de représenter et mettre en œuvre les flux migratoires entre différents processus répartis sur différentes machines. Finalement la section 3.6 présente un bref panorama de quelques techniques et simulations distribuées.

Malgré son potentiel, la distribution des données et des calculs associés nécessitent encore le développement de solutions à la fois performantes et flexibles qui intègrent un modèle de communication, des mécanismes de synchronisation et une gestion transparente des données réparties. En plus de ce contrôle, qualifié de bas niveau, un support pour le développement d'interfaces et d'outils de visualisations adaptés doit être intégré afin de permettre une communication interactive entre une simulation distribuée et son utilisateur. Certains travaux,

comme [Erol et al., 1999] dont l'objectif est d'offrir les services nécessaires et l'encapsulation des fonctionnalités distribuées afin que l'environnement logiciel fournisse un support facile d'utilisation, ont abondé dans ce sens. Cependant, des supports de simulation indépendants d'une application unique et où les principes des simulations distribuées à événements discrets sont gérés nécessitent encore de nouvelles explorations. La modélisation de larges systèmes est encore soumise à des limites de coordination, de nombre et de représentation de la complexité des interactions du système étudié (notamment pour une approche agent). Notons finalement que la fragmentation des compétences en différentes disciplines des méthodes et des techniques de modélisation, de simulation, et de gestion des systèmes distribués rendent difficiles l'application et la réutilisation des concepts développés. Selon [Page et al., 1999] deux éléments clés sont nécessaires à la simulation distribuée : la réutilisation des modèles et des logiciels (i.e. simulateurs) et la transparence des techniques de simulation, c'est-à-dire la capacité pour un concepteur d'utiliser les mécanismes de la simulation distribuée sans connaissance a priori des techniques sous-jacentes.

Le chapitre suivant introduit une méthodologie adaptée à la modélisation de systèmes à larges flux de données désagrégées sur un environnement distribué. Cette approche vise à guider la conception de modèle pour la simulation distribuée.

CHAPITRE 4

Méthodologie pour la simulation distribuée

« Pour comprendre un système, il faut... s'en extraire. Bernard Werber, Écrivain français »

Les flux, quel qu'en soit leur nature, sont au sein des systèmes complexes urbains et sociétaux des phénomènes présents à tous les niveaux. La tendance actuelle de considérer non plus les flux à un niveau global mais par une approche de gestion décentralisée permet de représenter l'enchevêtrement, la localisation et le traitement de ces flux à un niveau de représentation plus adéquat. Cette approche, vis-à-vis de la simulation, nécessite de définir des méthodologies de représentation adaptées. Nous proposons dans ce chapitre une démarche pour la modélisation des systèmes complexes à larges flux de données désagrégées qui guide le processus de modélisation vers la simulation à l'aide de systèmes distribués. Cette méthodologie de conception est décomposée en plusieurs étapes séquentielles allant de la modélisation hiérarchique par graphe à l'exécution sur une architecture (matérielle et logicielle) distribuée. La section 4.1 introduit les objectifs de ce chapitre. La section 4.2 introduit notre démarche de modélisation. Les sections 4.3 et 4.4 présentent une approche de décomposition du modèle par une approche hiérarchique basée sur les graphes. La section 4.5 présente le mécanisme de projection établissant un lien entre le modèle et le système distribué. La section 4.6 conclut ce chapitre.

4.1 Besoins méthodologiques

La simulation des systèmes complexes à larges flux de données désagrégées requiert un environnement informatique permettant l'exécution, le suivi et l'analyse de son comportement. De tels systèmes sont fréquemment distribués par nature et se composent d'éléments spatiaux agrégés entre lesquels interviennent les flux de données désagrégées (cf. section 2.1). Cette façon de percevoir le système correspond, dans la méthodologie de Paul Fishwick, au modèle spatial qui se concentre sur la géométrie du système en divisant l'espace en sous-parties plus petites et interactives [Fishwick, 1995]. L'approche de simulation séquentielle couramment utilisée présente deux problèmes pour l'aboutissement de simulations de ces systèmes. En effet, certains systèmes composés de sous-parties indépendantes et autonomes sont contraints par leur structure géographiquement et techniquement distribuée. Dans ce contexte, même si une simulation mono-machine reste possible, elle contredit la répartition naturelle, interdisant ainsi la mise en place de l'outil de simulation dans son environnement réel. Dans d'autres cas la structure répartie du système apparaît par la volonté de décentraliser les fonctions ou

les actions à un niveau de décision local. C'est le cas par exemple de la gestion des feux de circulation dans les grands systèmes urbains qui tend à être orchestrée à un niveau structurel local plutôt qu'à un niveau global. Le deuxième aspect à prendre en compte est la limite physique de l'ordinateur utilisé pour exécuter la simulation. L'expérience a montré que malgré les évolutions constantes des machines, celles-ci restent toujours insuffisamment puissantes par rapport aux objectifs fixés. Dans le cas des applications relevant de notre champ d'étude, une solution mono-machine aura forcément une limite de performance que nombre de simulations ne demanderont qu'à franchir.

Les systèmes distribués (cf. section 3.1) possèdent un grand potentiel d'un point de vue de la puissance de calcul disponible. Ces environnements offrent par ailleurs un intérêt essentiel pour la modélisation et la simulation des systèmes complexes répartis ou décentralisés, structurellement ou fonctionnellement, dans lesquels apparaissent des gestions locales. Dans le cas, par exemple, d'un système de surveillance maritime [Huet et al., 2003], l'architecture répartie reproduit les structures de gestion décentralisées du modèle. Bien que l'utilisation d'une approche distribuée semble séduisante et que certaines applications existent, en particulier dans le cas des systèmes de simulation de trafic [Nagel, 2002], il existe toujours un fossé conceptuel et technique entre les modèles et les paradigmes utilisés pour la modélisation de systèmes désagrégés et les possibilités informatiques. Ce type de simulation nécessite de fait toujours l'établissement d'un lien entre la modélisation d'un système complexe et sa simulation sur le support informatique distribué.

Les définitions d'un système complexe et du modèle résultant sont présentées dans les sections 1.1 et 1.2 par leurs caractéristiques fonctionnelles, structurelles et dynamiques. Toutefois, comme le précise Landry et Santerre, peu d'écrits existent sur la théorie et les méthodes d'élaboration des modèles [Landry et Santerre, 1999]. Similairement à [Shreckengost, 1985] dont la vision de la validité est introduite en section 1.4, ils postulent que la modélisation demeure un art dont seul les artistes maîtrisent bien les règles. D'un aspect plus général, une méthodologie de conception pour la simulation de systèmes complexes est un ensemble d'étapes séquentielles permettant à partir d'un problème initial de dériver (1) les structures de données et leurs comportements spatiaux et temporels, (2) le mécanisme de mise en œuvre (projection) sur un système informatique et (3) un environnement informatique fournissant un support pour la simulation. Notre objectif dans ce chapitre n'est pas de présenter un formalisme pour la modélisation de systèmes complexes à larges flux de données désagrégées, mais de fournir une méthodologie de conception afin de démontrer le potentiel des systèmes distribués pour la simulation de tels systèmes complexes. Cette méthodologie correspond à la quatrième étape de la démarche de modélisation présentée en section 1.2.3 et dont les files d'attente, le formalisme *DEVS* ou encore le paradigme multi-agents sont quelques exemples de méthodologies pour la simulation informatique (cf. section 1.3.2). Dans un contexte proche, [Aiello, 1997] propose de définir une méthodologie de modélisation et de simulation séquentielle générale afin de prendre en compte des systèmes issus de différents domaines. Cette étude repose sur la conception générique de modèles hiérarchiques (incluant une hiérarchie temporelle) basé pour partie sur le formalisme *DEVS* [Zeigler et al., 2000].

Dans les sections suivantes nous présentons une approche de modélisation dont l'objectif est de montrer qu'il est possible d'établir un lien entre un système complexe à larges flux de données désagrégées et les systèmes distribués par une démarche de modélisation adaptée à l'architecture matérielle sous-jacente. Cette méthodologie consiste à décrire et à guider une méthode de décomposition du modèle telle que présentée en section 3.2.5. Elle est basée sur l'établissement d'une correspondance (une projection) entre le système à simuler et son support de simulation, tous deux modélisés par un graphe.

4.2 Approche de modélisation

Un système distribué est composé de plusieurs nœuds (machines) et de connexions entre ces nœuds (réseau de communication). De par sa nature, un système distribué peut être modélisé par un graphe dans lequel les nœuds du graphe constituent les nœuds où sont effectués les calculs (i.e. ordinateurs) et les arcs, les connexions définies entre ces ordinateurs [Chandy et Misra, 1979]. La topologie d'un tel environnement peut être mise en correspondance avec celle d'un système complexe à larges flux de données désagrégées car, en effet, ces deux systèmes peuvent se réduire à une forme similaire lorsque modélisé par un graphe. La flexibilité des systèmes distribués (dans leur topologie) autorise à penser que de nombreux systèmes désagrégés modélisables par un graphe peuvent être mis en œuvre et simulés par l'utilisation d'un support informatique réparti.

Nous proposons d'utiliser les propriétés de graphe d'un système distribué afin de reproduire un phénomène réel ou artificiel en établissant une correspondance entre le graphe représentant le système désagrégé et le graphe représentant le système distribué. Cette approche illustrée en figure 4.1 est décrite par (1) la modélisation conceptuelle du monde réel et (2) la mise en œuvre des structures de données et de leurs propriétés en termes de représentation informatique.

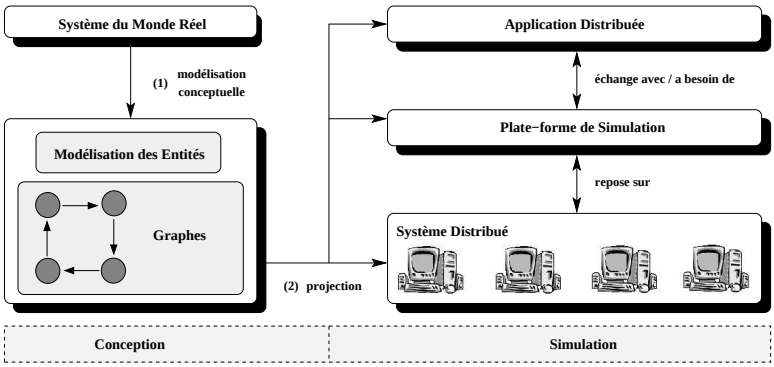


Fig. *4.1* — Principe de modélisation

Dans notre méthodologie pour la simulation distribuée à événements discrets, un système complexe est décrit à un niveau logique par une modélisation conceptuelle hiérarchique et par une décomposition structurelle de l'espace. Le niveau physique est encapsulé par une plateforme distribuée (un logiciel) qui (1) fournit un contrôle de l'architecture multiprocesseurs sous-jacente, (2) agit comme une interface de programmation entre le modélisateur et le matériel composé de plusieurs d'ordinateurs interconnectés par un réseau de communication, (3) offre un support pour simuler et analyser le comportement des systèmes à larges flux de données désagrégées et (4) reproduit la structure décentralisée du système représenté, ou l'optimise selon la meilleure répartition des performances et ce au niveau de granularité choisi en fonction des objectifs postulés par le concepteur. L'application distribuée, définie par l'utilisateur, implémente le système réel devant être simulé (notre but dans ce chapitre est de guider la conception de cette application). Cette application interagit avec la plateforme de simulation en faisant appel à l'ensemble des services offerts par celle-ci. Un des objectifs est ainsi que les propriétés du système à simuler soient reproduites à un niveau physique par le système distribué. En particulier les flux de données qui existent entre les nœuds du graphe représentant le système complexe peuvent être reproduits au niveau des communications physiques.

Notre méthodologie de conception illustrée en figure 4.2 est composée de trois étapes séquentielles pour établir un lien entre un système réel ou virtuel et sa simulation potentielle à l'aide d'un système distribué.

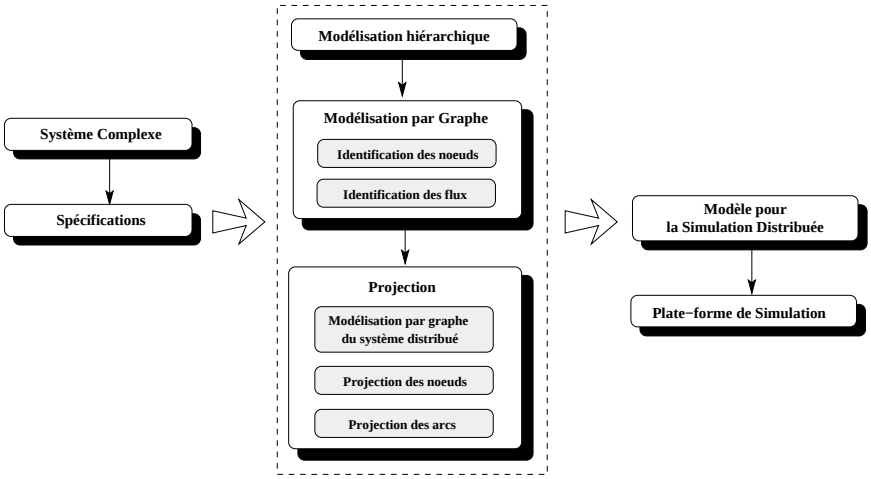


Fig. *4.2* — Méthodologie de conception

Cette méthodologie consiste tout d'abord à modéliser un système complexe par une description hiérarchique de l'organisation interne puis dans une deuxième étape à le modéliser sous forme de graphe. Une fois l'environnement original modélisé par le concepteur, la troisième étape consiste à établir une projection (i.e. correspondance) entre le modèle désagrégé et le système distribué. Le mécanisme de correspondance entre la modélisation hiérarchique et l'architecture distribuée repose sur les graphes représentant les deux systèmes. Un graphe résultant de cette projection est alors le paramètre essentiel de la plate-forme de simulation distribuée qui intervient comme support pour l'exécution de simulations et qui réalise le lien et la gestion de l'architecture répartie de manière appropriée. Les sections suivantes décrivent les trois étapes citées et le chapitre suivant présentera notre mise en œuvre d'un environnement de simulation.

4.3 Modélisation hiérarchique

La première étape de la méthode de modélisation est basée sur une description hiérarchique de l'organisation interne qui consiste à identifier de manière exhaustive les éléments intervenant dans le modèle du système étudié. Le premier point dans cette démarche de construction hiérarchique du modèle est l'identification de la hiérarchie-même du système par raffinements successifs. Cette phase revient à construire de manière ascendante ou descendante (en fonction des besoins et des connaissances du système) un arbre représentant l'organisation structurelle interne du système complexe par un ensemble de niveaux comme illustré en figure 4.3. Une relation d'inclusion structurelle (éventuellement mélangeant une relation d'inclusion fonctionnelle) existe entre les différents niveaux : une entité E d'un niveau $i+1$ est incluse dans un composant C du niveau i si le composant C est au moins pour partie un agrégat d'éléments du type de E (éventuellement un type englobant de E dans le cas d'inclusion fonctionnelle).

L'objectif principal étant la simulation de système complexes à larges flux de données désagrégées par l'utilisation des systèmes distribués, un aspect fondamental doit d'ores et déjà

apparaître dans une telle décomposition, à savoir un niveau structurel qui est fixe dans l'espace et qui potentiellement constitue, par l'union des éléments de ce niveau, le partitionnement de l'espace délimité par la borne supérieure de la limite du système étudié. Chacun des éléments de ce niveau est par ailleurs caractérisé par sa capacité d'interagir avec ses homologues. Ces interactions forment un réseau de communication entre des réservoirs qui échangent des informations (cf. section 1.1.2). Ces informations sont dans notre contexte des éléments du système, mobiles dans celui-ci. La borne inférieure définit la limite de description des niveaux microscopiques du système, c'est-à-dire du niveau de désagrégation maximal N.

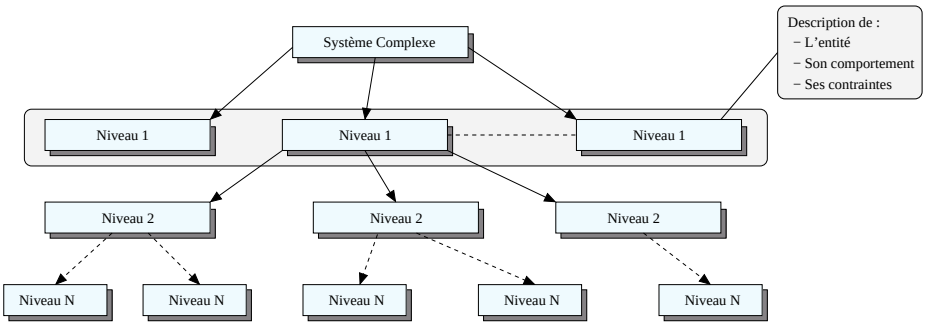


Fig. *4.3* — Modélisation hiérarchique

L'identification des comportements et des contraintes des entités de chaque niveau consiste à définir l'aspect fonctionnel et dynamique du système (cf. sections 1.1.3 et 1.1.4). En particulier il convient de préciser la nature des flux et comment ceux-ci interviennent entre les éléments du niveau structurel fixe. L'ensemble des actions de chaque élément du système quel que soit son niveau d'agrégation produira d'un point de vue de l'environnement de simulation des ensembles d'événements gérés soit par l'application distribuée, soit par la plate-forme de simulation (cf. figure 4.1). Nous nous placerons dans l'hypothèse où de ce point de vue informatique les éléments sont modélisés par des objets (du paradigme du même nom) pouvant en fonction des besoins du concepteur être doués d'autonomie (agents).

La modélisation hiérarchique produit une structuration claire, non ambiguë et exhaustive du système en fonction du postulat de complexité choisi et des spécifications émises par le concepteur. Remarquons que cette approche de modélisation est proposée dans [Bargiela, 2000] comme une des directions stratégiques à employer pour la modélisation et la simulation. En effet, elle permet d'identifier la répartition d'un système complexe à différents niveaux de granularité correspondant ainsi à une approche de modélisation de système désagrégée (plus l'arbre est profond, plus il correspond à une approche désagrégée). Elle permet aussi d'exhiber pour chaque niveau, la granularité temporelle et spatiale propre à chacune des entités le constituant. Elle favorise une approche *bottom-up* dans laquelle les propriétés de chaque niveau émergent de celle du niveau plus fin dans l'abstraction (on remonte dans l'arbre) et où en retour des rétroactions sont possibles (un niveau i influence ou contraint les comportements des éléments de ses sous-niveaux). Enfin, cette modélisation est adaptée pour la simulation à l'aide de système distribué car il existe une relation forte entre la structure d'arbre (production de la modélisation hiérarchique) et la structure de graphe qui est l'outil de modélisation le plus adapté pour établir la projection vers un système distribué.

4.4 Modélisation par graphe

La deuxième étape de la méthodologie a pour finalité de modéliser un des niveaux de l'arbre résultant de la modélisation hiérarchique par un graphe qualifié dans notre terminologie de *graphe logique*. La fonction du graphe logique est de représenter les dynamiques de flux de données au sein de la décomposition structurelle hiérarchique identifiée à la première étape. La nature de ce graphe, dont la figure 4.4 illustre un exemple, est variable et fortement dépendante du système complexe considéré.

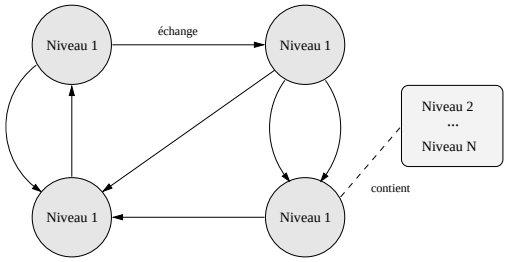


Fig. *4.4* — Modélisation par graphe

Les nœuds du graphe logique correspondent aux entités structurelles fixes qui représentent un partitionnement de l'espace, c'est-à-dire les composants de haut niveau dans l'arbre de modélisation hiérarchique (i.e. le niveau 1 encadré dans la figure 4.3). Ces composants agrégés correspondent aux origines/destinations des flux de données. Chacun des nœuds possède une descendance hiérarchique (i.e. niveau 2 à niveau N dans l'exemple) qui lui est propre. Les entités de cette descendance hiérarchique sont associées à un des nœuds en fonction de leur localisation dans la décomposition structurelle. Les interactions entre les composants agrégés permettent de définir des arcs entre les nœuds du graphe logique. Ces interactions, dépendantes du système complexe considéré, correspondent à la mobilité des entités réparties dans cette décomposition de l'espace (indépendamment de leur niveau d'agrégation). La libre circulation de ces entités forme un flux de données désagrégées. Chacun des arcs représente l'existence d'une relation bilatérale entre deux nœuds et leur concaténation illustre l'existence potentielle de parcours dans le système représenté.

Dans les terminologies employées dans le premier chapitre, les principales parties structurelles constitutives d'un système complexe (cf. section 1.1.2) sont les réservoirs et le réseau. Appliqué à cette méthodologie, les nœuds du graphe sont les réservoirs et le réseau de communication sont les arcs du graphe. Les flux de données circulent entre les réservoirs dont ils font varier le contenu. Les ajustements (ou délais) correspondent à des pondérations éventuelles des arcs. Le comportement du système complexe est inféré du comportement de chacun des nœuds du graphe logique. Eux-mêmes résultent des comportements des sous-niveaux de la hiérarchie qu'ils contiennent, de leurs enchevêtrements et des interactions qu'ils opèrent (émergence, cf. section 1.1.3).

Ces premières étapes de la démarche de modélisation (définition d'une organisation hiérarchique interne et du graphe correspondant) permettent d'identifier clairement si le système considéré est *projetable* sur un système distribué et quels sont alors les composants structurels fixes et les entités mobiles qu'ils échangent. L'étape suivante consiste à établir la correspondance entre le graphe logique et un graphe physique qui abstrait le système distribué.

4.5 Projection par homomorphisme

La projection est l'étape de modélisation dans laquelle une correspondance est réalisée entre le système désagrégé modélisé par un graphe logique et le système distribué. Cette correspondance consiste à établir un lien (1) entre les composants du modèle identifiés par des nœuds dans le graphe logique et les ordinateurs du système informatique réparti et (2) entre les interactions qui relient les composants et les flux de données qui transitent sur le réseau de communication.

4.5.1 Modélisation d'un système distribué

Un système distribué peut être modélisé par un graphe dans lequel chaque nœud est un ordinateur et chaque arc bidirectionnel est une connexion possible entre deux nœuds. La topologie des systèmes distribués utilisés pour la simulation peut varier et prendre par exemple une forme d'anneau ou d'étoile. La forme du graphe représentant le système distribué dérive alors de l'organisation des interconnections physiques entre les ordinateurs. La figure 4.5 illustre l'exemple d'un graphe fortement connexe représentant un système distribué dans lequel il existe une connexion point à point entre tous les ordinateurs. Dans ce type d'architecture répartie chaque machine peut joindre toutes les autres sans intermédiaire.

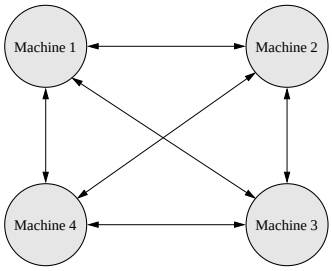


Fig. *4.5* — Modélisation d'un système distribué

Nous retiendrons dans la suite de ce document le type d'architecture présentée par la figure 4.5 car il s'agit de la plus générale. Elle permet de concevoir tous les graphes physiques possibles (cf. section suivante). Remarquons que des topologies différentes peuvent être utilisées sans aucun inconvénient car dans un réseau connexe toute machine peut être atteinte. Toutefois, le cheminement des messages est alors contraint par la topologie particulière. La prise en compte (ou non) de cet aspect structurel peut induire un impact non négligeable sur les performances de migration d'objets dans le réseau et par conséquent dans la gestion des flux de données.

4.5.2 Principe de projection

L'un des objectifs du mécanisme de projection est de déterminer quelle est la répartition des nœuds logiques sur les ordinateurs. Différents types de projection sont possibles entre le système complexe et l'architecture distribuée (cf. figure 4.6). La projection est soit (1) un homomorphisme (même forme) : un nœud du graphe représentant le système complexe (le graphe logique) équivaut à un nœud du système distribué, soit (2) la projection est un endomorphisme : plusieurs nœuds du graphe logique sont hébergés par un nœud du système distribué.

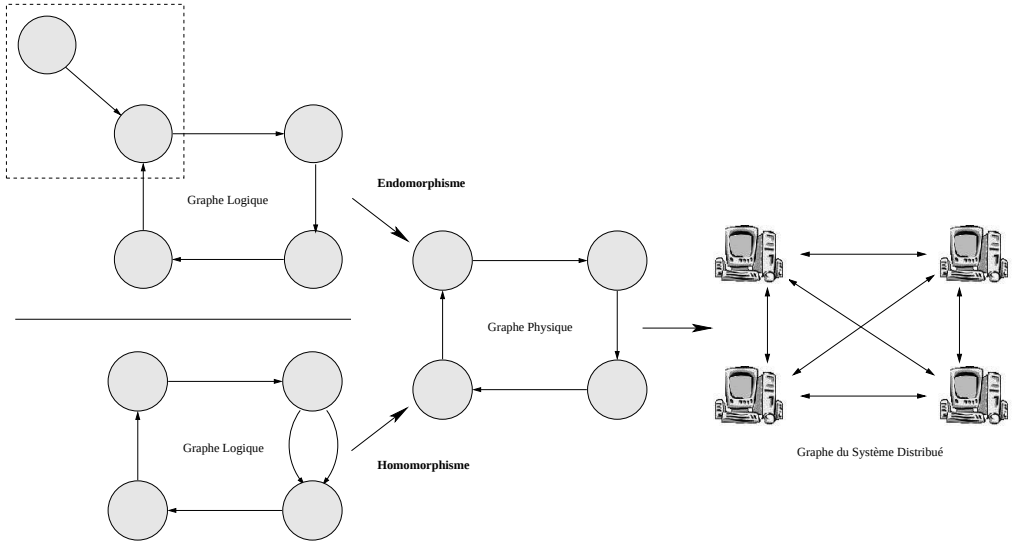


Fig. *4.6* — Principe de projection

Nous utilisons la terminologie de *graphe physique* pour définir l'abstraction du matériel informatique réparti. Le graphe physique est une représentation intermédiaire entre le graphe du système distribué et le graphe logique. Dans tous les cas de projection le graphe physique est une abstraction dans laquelle le nombre nœuds est égal au nombre de machines de l'architecture répartie (un nœud correspond à un ordinateur). Une projection issue du graphe logique en définit la structure, c'est-à-dire la correspondance existante entre les nœuds et les arcs des deux graphes. La projection définit notamment comment les arcs modélisant les flux de données identifiés au niveau logique utilisent les connexions physiques. Dans notre méthodologie, ce graphe constitue le paramètre principal de la plate-forme de simulation. La *projection des nœuds* consiste à définir comment les nœuds logiques sont associés aux nœuds du graphe physique. Pour établir une correspondance entre le système à modéliser et le support réparti on peut :

- Adapter le nombre d'ordinateurs, c'est-à-dire le nombre de nœuds du graphe physique au nombre de nœuds logiques : homomorphisme ;
- Adapter une modélisation hiérarchique inhérente du système complexe en considérant un (nouveau) niveau d'agrégation où le nombre de nœuds logiques est en accord avec le nombre de machines disponibles : préservation d'un homomorphisme ;
- Conserver la décomposition structurelle naturelle du modèle et héberger plusieurs nœuds logiques sur un même nœud physique (i.e. ordinateur) : endomorphisme.

La méthode de décomposition du modèle par endomorphisme, où tous les cas de projection d'un système logique vers un environnement de simulation distribuée sont envisageables est un objectif à atteindre. Toutefois, cet objectif nécessite la conception et la mise en œuvre de plusieurs étapes incrémentales. La première étape à résoudre est identifiée par la capacité à réaliser une projection par homomorphisme (finalement un cas particulier de l'endomorphisme). Un des aspects important dans le choix d'étude d'une projection par homomorphisme résulte de l'intérêt porté à la gestion des flux. La gestion des flux de données désagrégées intra-machine, qui est une caractéristique propre de l'endomorphisme, ne pose pas d'enjeu conceptuel fondamental mais relève plutôt de choix techniques. A contrario,

la modélisation et la mise en œuvre de flux de données désagrégées entre machines pose des problèmes ouverts de conception et de gestion au sein d'un réseau. Par ailleurs, cette capacité à gérer les flux inter-machine (i.e. migration, cf. section 3.5) est la base indispensable pour permette une simulation distribuée composée de plusieurs machines communicantes. Nous choisissons dans notre méthodologie de conception de privilégier cet aspect et de l'aborder à travers une projection par homomorphisme qui aura l'avantage de cerner cet aspect pour mieux le résoudre. Nous proposons dans la suite du document l'étude des principes conceptuels et la mise en œuvre des techniques distribuées afférentes à cette première étape. Notons qu'à titre d'illustration des perspectives envisagées, nous présenterons quelques éléments de compréhension du mécanisme de projection par endomorphisme en section 4.5.4.

4.5.3 Projection homomorphique

Le mécanisme de projection par homomorphisme considère une correspondance directe entre les nœuds du graphe logique et les nœuds du graphe physique et in fine les ordinateurs. Ce type de projection possède plusieurs propriétés qui en font un premier palier intéressant pour la simulation distribuée : (1) l'approche est optimale en termes de répartition de charge lorsque le nombre de nœud est peu élevé et que les composants (les nœuds) sont homogènes et, (2) elle permet la reproduction de l'aspect structurel du système désagrégé à un niveau physique par le système distribué. La descendance de chaque nœud logique dans l'organisation hiérarchique est elle-même associée à l'ordinateur correspondant. Dans une telle répartition, le système informatique affiche son état global en utilisant une approche *bottom-up* en agrégeant les états locaux des différents éléments de calcul. Cette configuration permet à chaque ordinateur de fonctionner au plus fin niveau de granularité et les émergences éventuelles du niveau logique sont alors issues du niveau physique.

Dans le cas d'une projection par homomorphisme, la structuration des arcs permet, à nombre égal de nœuds, de distinguer un graphe physique d'un autre. Les arcs utilisent un sous-ensemble des connexions physiques existantes pour reproduire les flux de données désagrégées identifiés dans le graphe logique. L'étape de *projection des arcs* consiste alors à définir la structure finale du graphe physique en précisant l'utilisation des interconnexions valides entre les ordinateurs et en définissant des choix méthodologiques et technologiques de gestion des flux de données logiques.

D'un point de vue conceptuel, les flux sont constitués d'entités de différents niveaux d'agrégation dans l'organisation hiérarchique. Ces entités sont dynamiques dans l'espace formé par l'union des composants modélisés par les nœuds du graphe logique. D'un point de vue de la simulation informatique distribuée, les mobilités des éléments du système complexe se traduisent par des déplacements physiques d'objets entre différents ordinateurs. Le mécanisme de migration présenté en section 3.5 est la technique qui permet de réaliser cette opération. Toutefois, les techniques et les principes abordés reposent sur des migrations individuelles qui ne permettent pas de mettre en œuvre de manière adéquate les flux de données désagrégées au sein du système complexe. Ceci notamment en considération des larges volumes d'information échangés entre les ordinateurs du système distribué.

Le principe de modélisation et de gestion des flux de données désagrégées proposé dans notre méthodologie se base sur le concept de déplacement organisé par groupe. Dans cette approche, la mobilité des entités désagrégées est conçue et coordonnée de manière agrégée par leur intégration dans un groupe qui se déplace entre les nœuds du graphe physique (cf. figure 4.7). Cette gestion des flux se traduit pour le système informatique par des groupes de données qui migrent entre les ordinateurs, elle permet de fusionner :

- D'une part, les besoins nécessaires à la compréhension d'un système complexe à larges flux de données désagrégées identifiés par, la nécessité de considérer des déplacements où les prises de décision et les actions de mobilité (en l'occurrence l'intégration d'un groupe dans notre approche) sont décidés à un niveau individuel (micro) ;
- Et d'autre part, les limites technologiques de gestion de larges flux de données désagrégées par la conception d'un mécanisme de migration d'objets approprié à la simulation distribuée qualitative. Le protocole RTB introduit en section 5.4 présente une mise en œuvre de cette gestion des flux agrégés par l'utilisation d'une métaphore de bus circulant entre les nœuds du graphe physique.

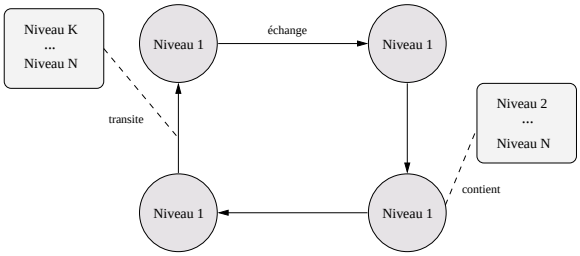


Fig. *4.7* — Modélisation des flux par groupe

Chaque groupe formé par le concepteur est un agglomérat de composants de différents niveaux d'agrégation compris entre un niveau K et le niveau N avec $2 \leq K < N$ (cf. figure 4.7). La mise en œuvre informatique d'un déplacement groupé requiert de définir une taille bornée pour le groupe (le système informatique est un environnement à variables finies). La taille (dynamique) d'un groupe est donc définie par $1 < card(groupe) < T$ où T est la taille maximale fixée en fonction des contraintes de flux identifiées au niveau logique. Cette taille est éventuellement influencée par une taille nominale pour la migration d'objet dans le système réparti (cf. section 5.1). Dans cette démarche, les entités désagrégées se déplacent de manière coordonnée entre les nœuds en fonction de leur objectif de destination dans la décomposition spatiale, c'est-à-dire en fonction du nœud du graphe physique vers lequel elles souhaitent se déplacer. Le comportement d'un nœud du graphe physique vis-à-vis des flux consiste à les organiser tel que :

- L'arrivée sur un nœud d'un flux agrégé (un groupe) implique l'extraction d'entités désagrégées lorsque celles-ci ont atteint leur destination structurelle et l'intégration de ces nouvelles entités au sein des structures de données de ce nœud ;
- Les entités demandeuses d'un déplacement (à titre individuel) sont intégrées à un groupe en partance. D'un point de vue informatique, la volonté de déplacement est opérée par l'entité elle-même (douée d'une certaine autonomie), ou par un composant supérieur dans la hiérarchie du modèle qui gère les nombreuses entités mobiles (de façon individuelle) ;
- Faire transiter de proche en proche, sur un chemin établi, les flux agrégés (joue le rôle de centre décision, cf. section 1.1.3).

Pour chaque groupe du graphe physique est associé un parcours (cyclique ou pas) qui décrit la succession des nœuds à parcourir. Un concepteur doit faire face à deux questions qui vont de pair lors de la conception des groupes : combien de groupes sont à créer et quels sont les chemins de chacun ? En règle générale, les différents parcours entre les nœuds

du graphe physique sont à définir en fonction des contraintes du système informatique et d'une corrélation possible avec les mobilités individuelles ou agrégées identifiées au niveau logique. Deux façons de concevoir un groupe (et son parcours) sont possibles en fonction de l'existence de parcours ou d'interactions bilatérales entre les nœuds du système complexe étudié. La figure 4.8 présente trois graphes logiques illustrant l'existence de parcours logiques ou d'interactions bilatérales entre les nœuds. Considérons, à titre d'exemple, les quatre halls (les nœuds des graphes) d'un terminal d'aéroport entre lesquels se déplacent des personnes :

- Graphe logique 1 : les personnes se déplacent en utilisant la navette qui parcourt un chemin cyclique (ABCDA) ;
- Graphe logique 2 : les personnes se déplacent par la navette (chemin cyclique ABCDA) ou par le passage piétons qui existe entre deux halls (une interaction bilatérale entre B et D qui n'est pas un parcours) ;
- Graphe logique 3 : les personnes se déplacent seulement à pied (uniquement des interactions bilatérales).

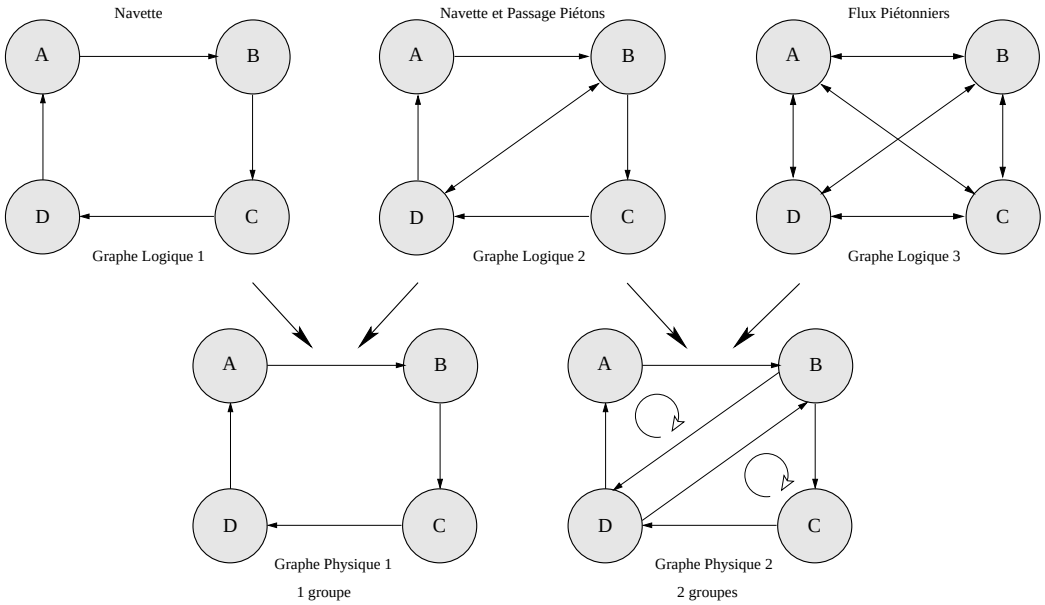


Fig. *4.8* — Conception du graphe physique

Lorsque des parcours existent au niveau logique (graphes logiques 1 et 2), la conception d'un groupe est aisée car il suffit d'associer un groupe à chaque parcours logique. Le concept de déplacement par groupe permet, dans ce contexte, de reproduire les mouvements des entités du système complexe. Lorsque les flux logiques sont eux-mêmes agrégés (cas de la navette), le déplacement groupé reproduit également la nature des flux du système complexe. Ce principe permet de simuler différents systèmes urbains dans lesquels sont considérées, par exemple, les mobilités collectives (e.g. bus, métro, mouvement de foule). Lorsque les flux sont désagrégés, un groupe intègre sans distinction toutes les entités mobiles qui se déplacent sur le chemin parcouru par ce groupe.

Le deuxième cas de conception d'un groupe est illustré par le graphe logique 3 qui est composé d'interactions bilatérales dont les concaténations ne représentent aucun parcours logique (le graphe logique 2 intègre également une telle relation). Dans ce cas, le concept de groupe a pour objectif d'optimiser les circulations des éléments intégrant un groupe de

migration, indépendamment de la nature des flux (agrégée ou désagrégée). Le concepteur définit les groupes afin de contrôler et de coordonner les migrations au sein du système distribué en limitant les migrations individuelles qui à haute fréquence sont nuisibles pour la stabilité et la performance de l'environnement de simulation. La projection du graphe logique 3 vers le graphe physique 2 présente la création de deux groupes (parcours ABDA et BCDB) pour reproduire les flux logiques. Remarquons que les graphes physiques qui possèdent plusieurs groupes nécessitent potentiellement des mécanismes gérant une correspondance entre ces groupes (le protocole RTB multigraphe présenté en section 5.4 intègre une telle gestion).

Le graphe logique 2 présente la combinaison d'un parcours cyclique et d'une interaction bilatérale entre deux nœuds. Le concepteur peut envisager pour ce cas deux projections (d'autres sont possibles) dont l'une utilise un groupe et l'autre deux. La projection du graphe logique 2 vers le graphe physique 1 illustre notamment l'intégration de différentes entités dans un groupe. Dans cet exemple, l'unique groupe assure les déplacements de la navette et des piétons. Remarquons que lorsque des parcours logiques existent, la création d'un groupe a pour vocation principale de reproduire les flux logiques. Dans ce cas, le groupe a également la faculté de mettre en œuvre de manière optimisée en coordonnant les déplacements et en limitant de fait les mobilités individuelles.

En l'absence de parcours logiques, la création par le concepteur de parcours dans le graphe physique sur lequel peuvent se mouvoir des groupes de mobilité est généralement dépendante de deux aspects. Ces deux aspects qui conditionnent la création des groupes en termes de nombres et de cheminement sont :

- Les interconnexions réseaux entre les ordinateurs et ;
- Les valeurs estimées des flux qui transitent entre chacun des nœuds du graphe logique.

L'utilisation d'une valuation des arcs du graphe logique et d'une valuation des arcs du graphe du système distribué permet de réaliser des parcours adéquats pour la mobilité des données désagrégées. Ces valuations permettent d'estimer d'une part les quantités de flux au niveau conceptuel et d'autre part, les capacités du réseau à faire transiter ces volumes de données. La définition des arcs du graphe physique à partir de ces deux sources de valuations permet de réaliser un compromis respectant au mieux les aspirations du niveau conceptuel et les contraintes matérielles. Remarquons cependant que l'optimisation du graphe physique (par le choix de ses groupes) est souvent effectuée par une approximation qui garantit une exécution performante de la simulation à défaut de pouvoir identifier la solution optimale. En effet, par certains aspects l'objectif de la simulation est d'identifier une solution optimale des flux opérés à un niveau logique. La connaissance d'une mise en œuvre optimale des migrations par groupe dans le système distribué est alors une variable à estimer grâce à un jeu d'exécutions successives qui vont définir la solution optimale des flux du niveau logique. Dans tous les cas, le choix des groupes et de leurs parcours doit être réalisé de telle sorte qu'ils n'induisent pas d'effets de bord dans le comportement du système simulé (notre plate-forme intègre des détecteurs contre cela).

De manière générale, la dépendance du graphe physique vis-à-vis de l'architecture matérielle et du système à simuler rend difficile l'identification de grandes tendances structurelles qui permettraient, pour une classe d'application donnée, de connaître par avance la constitution des groupes de migration. D'un point de vue matériel, l'utilisation d'un système distribué dans lequel chaque machine est connectée point à point permet de définir tous les chemins possibles. A contrario, un système distribué où toutes les machines sont connectées par une topologie en anneau réduit les possibilités. Dans ce cas le choix principal (non unique) se réduit à décrire un anneau virtuel calqué sur la structure de l'architecture matérielle et

sur lequel se déplace un (ou plusieurs) groupe de migration. Remarquons par ailleurs, qu'une arrête du graphe physique peut représenter plusieurs arcs du graphe logique et que de manière similaire plusieurs groupes peuvent transiter entre deux ordinateurs. Un chemin peut passer par tous les nœuds ou simplement définir un aller-retour entre deux nœuds.

Finalement, un aspect fondamental de la conception d'un graphe physique est lié au support de simulation pour lequel ce graphe est un paramètre privilégié. Ce support doit être capable de simuler toutes les configurations homomorphiques de graphe physique indépendamment de l'architecture sous-jacente. Notons que dans une solution idéale, la plate-forme prendrait en paramètre le graphe logique et le graphe du système distribué. Elle calculerait ensuite elle-même le graphe physique en fonction des recommandations de l'utilisateur et contraintes imposées (e.g. connexions existantes entre les ordinateurs).

4.5.4 Perspectives de projection endomorphique

Le mécanisme de projection par homomorphisme présenté ci-dessus est limité à la simulation de systèmes complexes dans lesquels intervient un nombre restreint de nœuds. En effet, l'approche précédente sous-entend de posséder au moins autant de processeurs que de nœuds dans le graphe logique. Lorsque le graphe logique possède beaucoup de nœuds, le nombre limité et surtout inférieur de machines dans un système distribué implique d'établir un mécanisme de projection adéquat dans lequel plusieurs nœuds du graphe logique sont modélisés par un seul nœud du graphe physique (endomorphisme).

La simulation de systèmes à très grande échelle requiert ainsi une projection différente n'impliquant plus de correspondance directe entre le graphe logique et le graphe physique. L'intérêt d'un tel découplage est d'offrir une plus grande flexibilité et un fort potentiel de montée en charge en termes de systèmes représentables et en termes de complexité simulable. Les contraintes de cette approche sont alors de déterminer la forme appropriée du mécanisme de projection, quelles sont les adaptations nécessaires à apporter à l'approche de modélisation hiérarchique et surtout comment gérer convenablement les flux de données et leurs correspondances entre le niveau logique et le niveau physique.

La première difficulté de l'endomorphisme consiste à établir des agrégats de nœuds logiques pouvant être projetés chacun sur un nœud du graphe physique. Plusieurs concepts et outils de groupement (*clustering*) ont été abordés dans différents documents tels que [Hartigan, 1975] et [Jain et al., 1999]. Les méthodes de calcul voisinage (*K-neighbouring analysis*) [Greco et al., 2002] permettent d'établir des associations entre les nœuds du graphe logique afin de former l'ensemble des nœuds du graphe physique. Les méthodes de partitionnement considèrent généralement un ensemble de n objets et un nombre K définissant la quantité de groupes (*cluster*) à former. Un algorithme de partitionnement organise les n objets dans K partitions ($K \leq n$) où chaque partition représente un groupe. Les groupes sont formés afin d'optimiser un critère partitionnement (fonction de similarité) de telle sorte que les objets au sein d'un groupe soient *similaires* alors que les objets dans des groupes différents sont différents (dans les termes du système considéré). Un exemple d'algorithme de partitionnement est le *K-means algorithm* qui fut introduit par Mac Queen dans [McQueen, 1967].

La deuxième difficulté de l'endomorphisme consiste à définir, en fonction de la répartition des nœuds logiques, la structuration des arcs du graphe physique. Dans ce contexte une partie des flux est intra-nœud et nécessite un mécanisme optimisé différent de celui proposé dans le cas de l'homomorphisme. En particulier il convient de définir si les flux internes sont agrégés ou désagrégés et quels sont alors les mécanismes à implémenter pour simuler

ces mobilités (e.g. copie mémoire). Dans ce contexte, la gestion des flux inter-nœuds par des communications (migrations) par groupe conserve la possibilité de reproduire un sous-ensemble des parcours éventuels identifiés à un niveau logique. Toutefois, ce type de gestion s'inscrit d'avantage dans le cadre d'une utilisation adaptée pour de larges flux de données et, optimisée pour l'architecture répartie sous-jacente. De ce fait la conception et la mise en œuvre de mécanismes de gestion des flux capables d'optimiser de manière automatique les interactions bilatérales en fonction des estimations d'interactions dans le système complexe et des capacités de migration du système distribué est nécessaire pour l'endomorphisme.

4.6 Conclusion

Ce chapitre présente une méthodologie de conception dont l'objectif est de guider la modélisation de systèmes complexes à larges flux de données désagrégées vers une approche de simulation à événements discrets sur une architecture distribuée. La section 4.1 introduit le besoin d'une méthodologie appropriée. La section 4.2 décrit les différentes étapes de notre méthodologie. Les sections 4.3, 4.4 et 4.5 présentent ensuite les trois étapes de notre démarche qui propose (1) une modélisation hiérarchique représentant l'organisation interne du système complexe, (2) la construction d'un graphe logique puis, (3) une étape de projection qui consiste à concevoir un graphe dit physique qui établit un intermédiaire entre le niveau de représentation logique et le système distribué sous-jacent.

Le choix méthodologique proposé dans ce chapitre est influencé à la fois par le système à simuler et par l'architecture informatique servant de support de simulation dans le but de proposer une démarche où l'aspect technique soit en accord avec l'aspect conceptuel. Dans une première approche, la méthodologie proposée est restreinte à un champ d'investigation reposant sur un homomorphisme. L'utilisation d'une approche plus générale dans laquelle le nombre de machines disponibles serait inférieur au nombre de nœuds du graphe logique modélisé (endomorphisme) est cependant introduite. Notre démarche de modélisation s'inscrit dans le cadre de travaux connexes considérant des représentations basées sur les graphes, notamment [Chandy et Misra, 1979] dans le cadre de la simulation distribuée à événements discrets ou encore [Rickert, 1997] et [Nagel, 2002] dans le cadre de simulation de systèmes de trafic. La différence de notre approche réside en partie dans la concaténation de principes issus de domaines d'applications divers afin de fournir une méthodologie qui, par une méthode de décomposition et de gestion des flux adaptée, convient aux systèmes complexes à larges flux de données désagrégées en vue de leur simulation sur un système distribué.

Le résultat des étapes successives présentées dans ce chapitre prend la forme principale d'un graphe dont l'objectif est de fixer la répartition des composants et des flux entre les ordinateurs. La mise en œuvre d'un support de simulation offrant un ensemble de services adaptés à cette production (le graphe physique) est l'étape finale indispensable de notre méthodologie. Le chapitre suivant présente Atlas, notre plate-forme de simulation distribuée mise en œuvre pour la simulation de systèmes complexes à larges flux de données désagrégées et partie intégrante de notre méthodologie.

CHAPITRE 5

Atlas, principes et outils de simulation

« Eppur si muove (et pourtant elle tourne). Galilée, homme à tout faire (1564-1642) »

Le rôle d'une plate-forme distribuée dans le cadre de la simulation est de fournir un ensemble de services essentiels pour permettre l'exécution répartie de programmes applicatifs. Dans le contexte de la simulation de systèmes complexes à larges flux de données désagrégées, les propriétés essentielles d'un support de simulation sont la gestion des composants du système et de leurs interactions qui nécessitent un contrôle des communications et un contrôle des migrations entre différents ordinateurs. Un autre point important pour la simulation est la gestion temporelle dont l'objectif est d'offrir au concepteur un temps simulé adapté à l'évolution temporelle du système considéré. L'objectif d'une telle plate-forme de simulation distribuée est double et consiste à offrir de nouvelles perspectives de modélisation plus adaptées et à dépasser les limites induites par l'approche monoprocesseur en offrant performances de calcul accrues. Ce but peut être atteint en (1) donnant un support de modélisation et de conception du système logique représenté (système complexe) et (2) en assurant un contrôle du système physique (système distribué) qui exploite au mieux les ressources informatiques. Ce chapitre présente les principes et les différents aspects de la mise en œuvre de la plate-forme distribuée Atlas qui constitue un support flexible de simulation pour la démarche de modélisation présentée dans le chapitre précédent. La section 5.1 présente l'architecture distribuée utilisée pour le développement de la plate-forme. La section 5.2 introduit les principales caractéristiques d'Atlas, puis les sections 5.3 à 5.8 présentent les six modules qui composent la plate-forme. La section 5.9 conclut ce chapitre.

5.1 Architecture multiprocesseurs

La mise en place d'un environnement de simulation distribuée expérimental nous a conduit à identifier une solution matérielle de référence sur laquelle serait construite la plate-forme. La solution matérielle retenue présente les caractéristiques typiques d'une architecture multiprocesseurs (cf. section 3.1). A partir de cette solution matérielle, nous avons étudié les performances relatives de différentes solutions de communication pour retenir une solution optimale sur laquelle peut être construite la plate-forme Atlas. Cette section présente successivement notre solution matérielle, l'analyse comparative des performances réseaux, les différents outils de communication et leurs fiabilités.

L'architecture matérielle multiprocesseurs mise en place pour le développement de la simulation de systèmes complexes à larges flux de données est un réseau local tel qu'illustré par la figure 5.1. Ce système distribué est constitué par une grappe (*cluster*) de quatre nœuds de calcul connectés point à point via deux *switch fast Ethernet*. Un cinquième ordinateur, un serveur, contrôle l'ensemble de la grappe à distance et agit comme un serveur de fichiers pour celle-ci. Le serveur est une machine bi-processeurs Pentium II 450 avec 512 mega-octets de mémoire, un ensemble de disques *SCSI* et fonctionnant avec le système d'exploitation Linux Mandrake 9.0 (noyau 2.4.19). La grappe est constituée de quatre Celeron 900 avec 256 mega-octets de mémoire fonctionnant avec le système d'exploitation Linux Mandrake 8.2 (noyau 2.4.8). Remarquons que les développements effectués sont plus génériques et non simplement limités à cette architecture précise (la grappe peut être étendue). La mise en place du support de simulation Atlas sur ce réseau local répond à un souci de prototypage dans le contexte de cette thèse. Les principes retenus sont extensibles avec des adaptations appropriées à des réseaux plus larges.

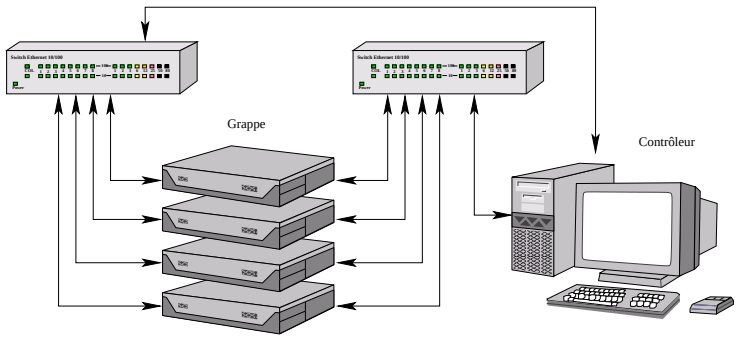


Fig. *5.1* — Architecture distribuée

Une des limitations persistante des systèmes distribués est liée aux problèmes de performance des réseaux de communication [Snell et al., 1996]. Lorsqu'un grand nombre d'entités communiquent et éventuellement migrent d'une machine à une autre les canaux de communication deviennent rapidement saturés. L'utilisation d'interfaces graphiques et de mécanismes permettant un suivi continu des états locaux de la simulation impliquent des échanges denses d'informations dans les systèmes distribués et entraînent les canaux de communication des réseaux conventionnels vers leurs derniers retranchements. Afin d'accroître les performances des communications, la technique appelée *channel bonding* [Radajewski et Eadline, 1999, Hawick et al., 1999] a été mise en place. Celle-ci consiste à multiplier la bande passante par la multiplication du matériel de communication. Dans cette architecture, chaque nœud (serveur compris) est connecté à la grappe via deux cartes *fast Ethernet* (la multiplication n'est limitée à priori que par le nombre d'emplacements libres sur la carte mère de l'ordinateur) dont la bande passante est de 100 Mbits/s permettant en théorie de dédoubler la bande passante et d'atteindre un débit de crête de 200 Mbits/s.

L'étude comparative présentée en figure 5.2 a été obtenue par une série de tests réalisés sur notre grappe à l'aide du programme *netpipe* (*Network Protocol Independent Performance Evaluator*) [Snell et al., 1996]. Les différents résultats montrent les performances en termes de bande passante pour les librairies de communications *PVM* (*Parallel Virtual Machine*, [Sunderam, 1990, Geist et al., 1994]) et *MPI* (*Message Passing Interface*, [Walker, 1993]) ainsi que les bandes passantes obtenues pour des communications de bas niveau qui combinent une programmation en langage C ou Java et les protocoles de communication *TCP/IP*

sans autre intermédiaire. Dans ce schéma en échelle logarithmique, le préfixe dual signifie que les deux cartes réseaux sont utilisées et le préfixe simple signifie qu'une seule carte à été exploitée. Lorsque cette étude est couplée avec l'étude des latences du réseau (i.e. temps nécessaire pour l'envoi du message), nous déduisons que la taille nominale de message pour une transmission optimale sur notre réseau est de l'ordre de 130 kilo-octets.

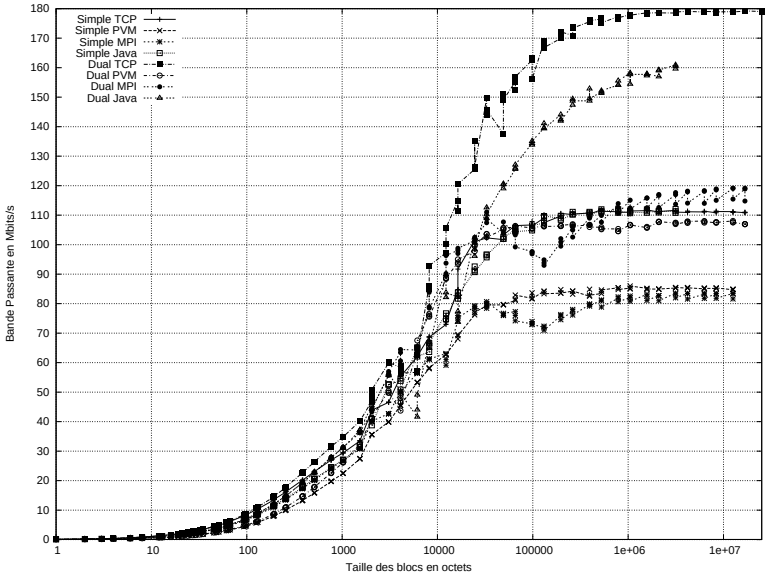


Fig. *5.2* — Bande passante

Certains logiciels tels que les librairies de communications comme *PVM* ou *MPI*, dont [Geist et al., 1996] présentent une comparaison, ou encore des systèmes plus lourds comme *CORBA* dont les performances sur un réseau *ATM* sont évaluées dans [Gokhale et Schmidt, 1996] peuvent fournir un support pour la construction d'une plate-forme et pour la gestion d'une architecture matérielle telle que celle présentée en figure 5.1. Le langage objet Java est l'un des langages les plus utilisés actuellement, il constitue également un support potentiel pour le développement des concepts et techniques de modélisation et de simulation des systèmes complexes. Ce langage intègre de nombreux outils et a par ailleurs fréquemment servi de support pour la conception de systèmes multi-agents. Citons par exemple les supports pour agents et objets mobiles tels que *JADE* (*Java agent DEvelopment Framework*) [Bellifemine et al., 1999], *Zeus* et *Leap* qui sont tous basés sur le standard *FIPA-OS* (*Foundation for Intelligent Physical Agents*) [O'Brian et Nicol, 1998] ou encore *Grassoper* et *Proactive* [Baude et al., 2002]. Par ailleurs de nombreux efforts ont été consentis par SUN microsystem sur l'aspect performance (calculs et communications). Le langage Java offre également de nombreuses fonctionnalités souvent faciles d'utilisation et relativement performantes pour le développement d'interfaces graphiques, de migrations (sérialisation, externalisation), de communications entre objets avec *Java/RMI* (*Remote Method Invocation*) ou encore de communications entre ordinateurs avec *Java/Jini*. Les différents outils existant (notamment *PVM*, *MPI* et Java) possèdent chacun des avantages et des inconvénients mais dans tous les cas ils nécessitent un effort d'intégration supplémentaire afin de fournir un service adapté à la simulation distribuée. A notre connaissance, il n'existe pas de supports informatiques à la fois intégrés et suffisamment génériques et flexibles pour réaliser différentes simulations distribuées à événements discrets de systèmes complexes à larges flux de données désagrégées.

Un système est sûr de fonctionnement s'il répond à sa spécification, c'est-à-dire que d'un point de vue de l'utilisateur le service délivré correspond effectivement au service attendu. Un système est dit défaillant quand le service délivré ne se conforme plus à sa spécification [Lapri, 1991]. La sûreté de fonctionnement d'un système peut être altérée par les fautes, les erreurs et les défaillances. La faute est la base d'une défaillance, il peut s'agir d'une erreur de conception, de perturbations magnétiques, etc. La faute conduit le système dans un état d'erreur. Une erreur peut être par exemple la corruption d'une variable par la prise d'une valeur incorrecte. La défaillance est la manifestation d'une erreur. Par exemple, l'utilisation de la variable corrompue entraîne une défaillance. Dans ce cas, le service rendu n'est pas satisfaisant. La sûreté de fonctionnement d'un système distribué est obtenue par des mécanismes de *tolérance aux défaillances* garantissant le fonctionnement correct des deux éléments essentiels, à savoir les nœuds et le réseau de communication. L'importance de la sûreté de fonctionnement pour certains types de simulation est abordée dans [Huet et al., 2003].

Un nœud fiable signifie que chaque machine reste active pendant la simulation, et que chacun des processus impliqués dans la simulation ne se termine avant sa terminaison (excepté par ordre explicite contraire). De plus, sur chaque nœud, l'ordinateur et ses processus gardent en permanence un comportement qui correspond à leur spécification, c'est-à-dire qu'ils ne tombent pas dans un comportement Byzantin [Lamport et al., 1982]. Les messages sont véhiculés sur des canaux de communication qui lient les différents sites de l'architecture entre eux. Un canal de communication est fiable si les messages ne sont :

- Ni perdus (tout message est reçu) ;
- Ni altérés ;
- Ni dupliqués ;
- Ni générés par le canal (il a été effectivement émis).

Dans les cas de perte et de duplication, l'estampillage (étiquetage) des messages permet au récepteur de savoir si un message reçu a été dupliqué, ou si un ou plusieurs messages précédents ont été perdus. Le problème d'altération peut se résoudre à l'aide de codes correcteurs chez le récepteur. Remarquons à ce sujet, que si le langage Java possède encore quelques lacunes, d'autres environnement tels que *PVM* ou *MPI* intègrent des versions tolérantes aux pannes [Fagg et Dongarra, 2000, Conan et al., 1995].

Tout en prenant conscience des perturbations temporaires ou permanentes pouvant affecter le comportement d'un système distribué, son système d'exploitation et les processus de notre simulateur Atlas, nous supposons que tous les nœuds et les canaux de communications du système distribué sont fiables.

5.2 Atlas, plate-forme de simulation distribuée

La plate-forme distribuée Atlas[1] a été développée afin de fournir un support réutilisable pour la simulation distribuée à événements discrets [Ray et Claramunt, 2003]. L'objectif de conception et de prototypage est de démontrer l'avantage et la pertinence des systèmes distribués pour la simulation et la compréhension de systèmes urbains et sociétaux représentables par un graphe et dans lesquels interviennent de larges flux de données désagrégées. La plate-forme repose sur l'utilisation d'une approche hybride de simulation qui combine une distribution des fonctions du simulateur et une distribution des éléments du modèle (cf. section 3.2).

[1]Nommée par analogie au titan de la mythologie grecque qui supportait sur ses épaules la voûte céleste.

Le serveur de l'architecture matérielle contrôle la simulation, gère un interfaçage graphique et l'interaction avec le concepteur. Les nœuds de la grappe effectuent chacun une partie des calculs liés à la simulation du modèle.

Construite au-dessus de l'architecture présentée en section 5.1, la plate-forme Atlas est une couche logicielle (un système à objets) basée sur le langage Java et ses services. Elle est constituée d'un ensemble de classes Java hébergées par le serveur et regroupées par fonctionnalités pour former les modules (composants) du système. Dans sa version courante, Atlas est composée de six modules (cf. figure 5.3) assurant chacun un service particulier : (1) un module de communication (network COMmunication : COM) assure les communications entre les nœuds de calculs et le serveur, (2) un module de migration (Round Trip Bus protocol : RTB) permet un déplacement des objets entre les ordinateurs, (3) un module de gestion d'objets génériques (object MANagement : MAN) fournit un support pour les créations, les destructions et les mises à jour des objets, (4) un module de gestion du temps de simulation (CLOCK management) définit une horloge virtuelle permettant de faire progresser la simulation et de dater les événements, (5) un module fournit un support graphique de base (INTER support) pour le contrôle des simulations, et (6) un module de surveillance (LOG) contrôle l'état de la plate-forme et conserve certaines valeurs cumulées ou instantanées décrivant l'évolution des états.

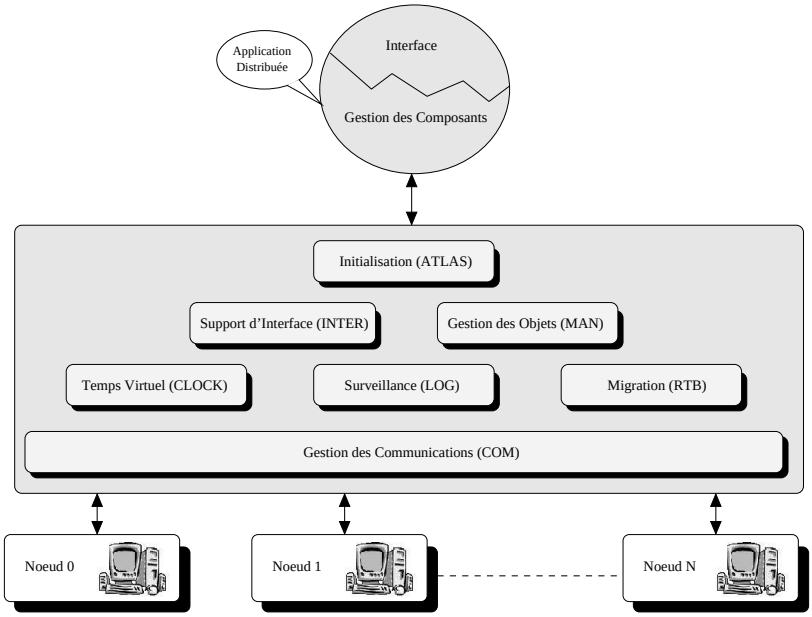


Fig. *5.3* — Architecture d'Atlas

La plate-forme Atlas se base sur la cœxistence de plusieurs machines virtuelles Java réparties. Chaque nœud du système distribué exécute une machine virtuelle et une instance de la plate-forme Atlas identifiée et différenciée des autres par son identité (un nombre entier)[2]. La plate-forme agit comme une interface de programmation et de contrôle entre l'utilisateur et le matériel. Le paramètre principal d'Atlas est la structure de graphe physique résultant de l'étape de modélisation lui permettant de réaliser l'abstraction du système distribué en tant que graphe afin de reproduire la structure logique du système complexe à simuler. La

[2]Atlas autorise et supporte l'exécution de plusieurs instances sur la même machine permettant, bien que non validé d'un point de vue conceptuel, de réaliser dans le principe des projections par endomorphisme.

flexibilité du système permet la projection de différents systèmes modélisables par un graphe, et dans lesquels interviennent de nombreuses interactions entre les nœuds. Une des particularités de la plate-forme Atlas réside dans le concept de migration physique par groupe qui donne une solution optimale et flexible pour coordonner la migration de larges flux d'objets, ce qui est un aspect important dans la simulation d'un système de transport. Le but de la plate-forme pendant l'exécution d'une simulation est alors de contrôler les flux de données à travers les chemins possibles du graphe en utilisant un sous-ensemble des connexions physiques disponibles.

L'application distribuée, définie par le concepteur selon la méthodologie introduite au chapitre 4, implémente le système complexe devant être simulé. Cette application interagit avec la plate-forme Atlas en faisant appel à l'ensemble des services offerts par celle-ci. Les entités du système réel (structure de données et comportement) sont modélisées par des objets informatiques Java (éventuellement des agents). La gestion de ces objets au niveau de l'application constitue le noyau du programme utilisateur. L'objectif de l'interface au niveau de l'application est d'offrir à l'utilisateur la possibilité d'interagir avec son application distribuée. Cette interaction se réalise à partir de l'initialisation de paramètres initiaux (configuration du graphe, propriétés des flux de données, contraintes de transport), l'utilisateur peut ensuite suivre l'évolution des états locaux successifs, des flux de données et les comportements émergents au niveau global. Nous présentons dans les sections suivantes l'essentiel des différents modules qui ont été mis en œuvre dans la plate-forme Atlas.

5.3 Module de communication

L'objectif du module de communication (COM) est de contrôler les interactions entre les nœuds du système distribué. Ce module est implémenté comme l'ensemble des autres modules en Java et utilise les protocoles de communication Internet (*TCP/IP*) fournis par le système d'exploitation. Rappelons que l'étude préalable sur les performances de cette approche Java et *TCP/IP*, dont certains éléments sont présentés en figure 5.2, a délivré de bons résultats. En effet, les performances en termes de débit et de latence sont souvent meilleures qu'une implémentation équivalente basée sur une combinaison de *C* avec *PVM* ou *MPI*.

Le module COM est constitué de quatre sous-composants assurant chacun une fonctionnalité : (1) un ensemble de fonctions bas-niveau assurent l'échange des messages entre les ordinateurs, (2) un composant maintient les paramètres des communications (en particulier la structure du graphe constituée par les machines), (3) un ensemble de fonctions haut-niveau reposant sur les précédentes structures assure le transport des messages, soit en flux continu (connexion permanente), soit en mode déconnecté (connexion à la demande), (4) enfin un gestionnaire de communication contrôle, pour chacun des ordinateurs, l'intégralité des échanges avec les autres ordinateurs et fournit un support générique (envoi, réception et traitement des messages) à l'utilisateur pour le développement de son application distribuée.

Tous les messages émis par le module COM possèdent quatre champs : le nœud d'origine, une estampille indiquant le destinataire (application utilisateur, module RTB, module MAN), une estampille libre d'utilisation (en particulier utilisée pour spécifier les commandes internes), et un champ générique (le message) qui contient un objet de type quelconque (e.g. bus RTB, texte pour l'interface graphique). L'envoi d'un tel message requiert de la part de l'émetteur la spécification en paramètre de ces quatre champs. Il spécifie par ailleurs l'ordinateur destinataire du message, puis le module COM gère le transfert. Les écouteurs (*socket handlers*) du nœud récepteur traite le message en fonction de sa nature et le stocke dans un

des tampons *FIFO* gérés par le module. L'insertion d'un message dans un tampon a pour effet de réveiller tous les processus bloqués sur le moniteur associé (selon la définition de Hoare). Le destinataire traite son message alors que les autres processus se remettent en attente passive.

5.4 Module de migration

Le module de migration de la plate-forme (RTB) a pour objectif de simuler les flux induits par les déplacements d'entités modélisées à un niveau microscopique. Ces flux de données désagrégées interviennent entre des niveaux d'agrégation supérieurs au sein d'un système complexe. D'un point de vue d'un système distribué à objets, le module de migration a pour objectif de mettre en œuvre de manière optimisée les migrations physiques d'objets informatiques répartis sur différentes machines reflétant ainsi le comportement du système considéré. Cette migration d'un objet est la suite d'actions qui correspond à l'envoi de l'objet d'une machine vers une machine distante, à sa destruction sur la machine source, puis à sa régénération sur la machine distante (cf. section 3.5). Ce module réalise, utilisant une métaphore de bus, la dynamique de flux agrégé présentée au chapitre 4 et où les décisions et les actions de déplacement sont prises et réalisées à un niveau désagrégé.

5.4.1 Protocole RTB : migration par groupe contrôlée

Le module de migration intégré dans Atlas est implémenté en tant que protocole distribué utilisé pour la distribution et le déplacement dynamique d'objets dans un environnement distribué [Ray, 2001b]. Le protocole *Round Trip Bus* : *RTB* est une technique de migration qui organise le déplacement physique des objets par groupe. Une analogie conceptuelle est réalisée entre un bus transportant des personnes et un groupe d'objets migrant entre les ordinateurs. La métaphore de bus se traduit par un objet complexe java qui transite entre tous les nœuds de l'architecture sous-jacente en effectuant un voyage circulaire. Le bus relie des structures d'arrêts de bus dans lesquelles il prend et dépose des objets qui représentent des entités (à différents niveaux de granularité) du système simulé. Les objets prennent le bus à chaque fois que celui-ci passe sur leur nœud et ce en fonction d'une politique de migration décidée par le concepteur de la simulation. Le choix du nœud de destination est également fonction des objectifs de l'entité migrante dans le système complexe.

Cette migration par groupe rend le protocole différent des approches courantes utilisées en système distribué et basées sur des migrations individuelles [Nuttall, 1994]. Ces approches, si elles ont l'avantage de migrer des objets rapidement, ne prennent pas en compte la nature du système à simuler et l'optimisation du réseau. Cela constitue un problème dans le cas d'applications nécessitant de larges transferts de données tant le coût des communications induites par les déplacements croît avec le nombre d'objets. Le protocole RTB offre une solution flexible de migration des objets par groupe qui gère des flux migratoires prenant en compte la nature contrainte des flux de données de nombreux systèmes désagrégés (en particulier en transport). Le protocole permet également de mieux exploiter les communications dans le réseau en contrôlant les déplacements afin d'éviter les congestions. Compte tenu des grands nombres d'entités microscopiques dans les systèmes complexes considérés, le déplacement de plusieurs objets ensembles permet une meilleure gestion des ressources de communications avec un nombre plus petit de message plus gros. Le protocole RTB est donc avant tout une technique de migration ayant pour avantage de mieux contrôler et coordonner les larges flux

migratoires intervenant dans le système complexe. Le deuxième apport de ce protocole est que le groupe de migration représente potentiellement les flux agrégés naturels du système complexe comme par exemple les déplacements de personnes dans le métro, en bus, en train, en avion ou encore des mouvements sociaux de groupe tel que des mouvements de foule en cas de panique ou de manifestation.

Une des propriétés de flexibilité du protocole repose sur le fait qu'aucune hypothèse n'est faite sur les objets devant migrer. Le protocole a la possibilité, par l'utilisation des mécanismes de sérialisation et de désérialisation fournis par le langage Java (cf. section 3.5), de faire migrer tous types d'objets (y compris agrégés). Toutefois le type (i.e. la classe des objets) doit être connu sur chaque machine pour que l'objet en déplacement puisse être reconstitué. La taille du groupe en migration (le nombre d'objets) varie selon les propriétés de l'application et peut être modifiée en cours de simulation. La taille maximale acceptée (i.e. la capacité de la métaphore de bus) a un impact non négligeable sur le comportement de la simulation. Cette taille, décidée par le concepteur, doit être adaptée aux objectifs de simulation. En faisant varier cette taille il est possible, entre autre, de modéliser des phénomènes d'attente (e.g. un enregistrement à l'aéroport) exprimant qu'un nombre limité de déplacement peut être réalisé à un instant donné. A contrario la taille peut être exagérément grande afin de ne pas induire de phénomènes de blocage et pour ne pas scinder un groupe en deux.

5.4.2 Fonctionnement du protocole RTB

Le protocole *Round Trip Bus* met en œuvre les flux de données désagrégés par une migration physique des objets au sein d'une structure de groupe qui effectue un parcours entre les différents nœuds d'un graphe. L'exemple présenté dans cette section explique les principes de fonctionnement de ce protocole. Considérons un système distribué représenté par un ensemble de machines (les nœuds) connectées par un réseau local privé. Nous définissons entre ces nœuds un sous-ensemble des chemins possibles modélisé par un *anneau virtuel* unidirectionnel décrivant un parcours cyclique entre les nœuds (cycle hamiltonien dans le graphe) tel qu'illustré dans la figure 5.4.

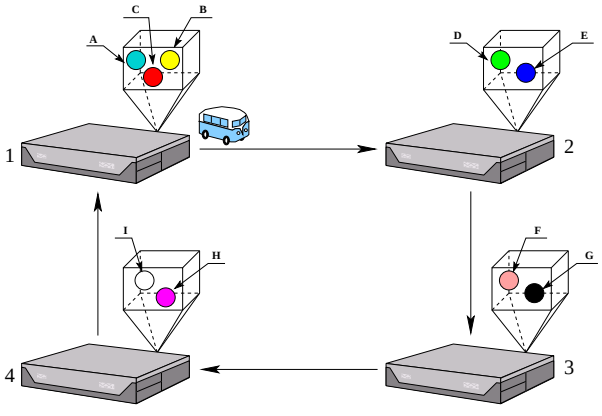


Fig. *5.4* — État initial

Chaque nœud du graphe est une machine dédiée à la simulation et à laquelle est associée un arrêt de bus. La structure d'arrêt de bus à laquelle se greffe le bus lors de son passage à un nœud est décomposée en deux zones tampons. L'une est utilisée pour stocker temporairement les objets en partance et l'autre pour stocker temporairement les objets arrivant sur le nœud

courant (i.e. descendant du bus). Dans la figure 5.4, plusieurs objets sont distribués sur les différents nœuds (numérotés pour l'exemple de *1* à *4*) et le bus est initialement arrêté sur le premier nœud (d'identité *1*). L'état initial de cet exemple est défini par les positions et les aspirations respectives de chacun des objets répartis sur les quatre machines. Les objets *B* et *C* souhaitent migrer du nœud *1* vers, respectivement les nœuds *2* et *3*. Au nœud *3*, l'objet *G* attend le bus afin de rejoindre le nœud *2* et, finalement l'objet *H* au nœud *4* tente d'atteindre le nœud *1*.

Le bus voyage circulairement en suivant le chemin établi par le concepteur de la simulation, et visitant chaque nœud. La transmission du bus d'un nœud à son successeur est effectuée très rapidement. En effet, les résultats présentés [Ray, 2001a] montrent, par exemple, que de l'envoi d'un bus de 1000 objets de 24 octets résulte l'émission d'un message de 30182 octets qui atteint sa destination en 150 ms en moyenne. Les tests réalisés montrent par ailleurs l'effet induit par la sérialisation du bus et de ses occupants. Nous avons remarqué que la taille des bus (24472 octets dans ce cas) représente environ 80% de la taille du message émis sur le réseau.

Ainsi, bien que le bus migre rapidement d'une machine à l'autre, ses déplacements sont contrôlés par le protocole. A chaque nœud du graphe est attaché un délai de livraison (impédance) avant lequel le bus ne peut être délivré. Les délais correspondent à une valuation des arcs[3] et sont fournis par le modélisateur qui en indique la valeur dans l'unité de temps virtuel (cf. section 5.6) en accord avec les contraintes et les objectifs du système simulé. Atlas effectue automatiquement la conversion du délai dans le temps simulé en termes de millisecondes d'attente. Lorsque le temps est écoulé, le bus est déclaré présent sur le nœud, les objets sont délivrés au nœud courant, puis les objets en attente du bus peuvent monter. Lorsque le nombre d'objets à l'intérieur du bus atteint la taille maximum de celui-ci, il est dans l'état *plein*. Lorsque le bus est plein ou lorsque la liste des objets en migration est vide, le bus quitte le nœud poursuivant son chemin circulaire.

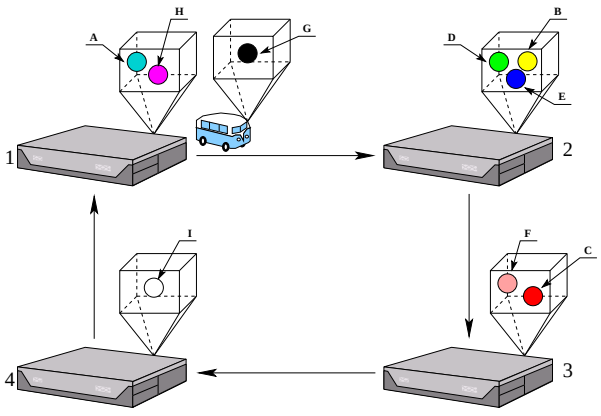


Fig. *5.5* — État après un tour

La figure 5.5 présente l'état du système après l'exécution par le bus d'un tour complet. Tous les objets ayant effectués un déplacement sont arrivés à destination, à l'exception de *G*. Ce dernier est toujours dans le bus, situé sur le nœud d'identité *1*, et n'en descendra que lors de la livraison du bus sur le second nœud.

[3]C'est-à-dire le temps simulé nécessaire pour accomplir le voyage en empruntant le lien entre le composant d'un nœud i et le composant d'un nœud j.

L'utilisation incorrecte du protocole peut conduire la simulation dans un état de blocage. En effet, le déplacement d'objets parmi les nœuds d'un graphe en utilisant le RTB vérifie la propriété de terminaison, mais pas la propriété de vivacité. Le déplacement d'un objet devant changer de nœud vérifie la propriété de vivacité s'il peut réaliser un déplacement à l'aide du bus. Le déplacement d'un objet devant changer de nœud vérifie la propriété de terminaison s'il peut atteindre son nœud de destination une fois monté dans le bus. Par définition, l'unique bus dessert la destination de l'objet souhaitant monter dans le bus car le chemin passe par tous les nœuds de l'architecture de simulation. Le fonctionnement correct du bus (supposé par l'implémentation correcte de ses spécifications) implique alors que tout objet monté dans le bus atteindra en un temps fini son nœud de destination. Par contre la vivacité ne peut-être assurée car elle dépend des contraintes de déplacement imposées par l'utilisateur comme l'illustre l'exemple suivant.

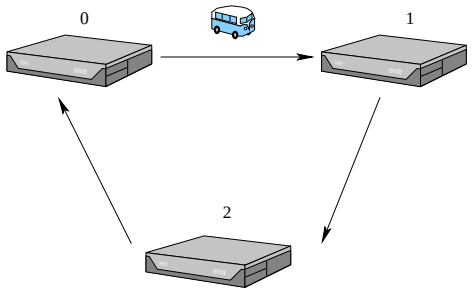


Fig. *5.6* — Problème de vivacité

Supposons un graphe G composé de trois nœuds et d'un chemin cyclique simple comme illustré sur la figure 5.6. Si le nœud estampillé 0 ne fait qu'envoyer des objets vers le nœud estampillé 2 en remplissant le bus à chaque passage, alors lors du passage sur le nœud estampillé 1 aucun objet ne descend et par conséquent aucun objet ne monte. A l'arrivée sur le nœud estampillé 2 tous les objets descendent (propriété de terminaison). Le bus repart ensuite vers le nœud estampillé 0 non chargé. La ré-exécution du même cycle à l'infini implique alors que les objets du nœud estampillé 1 sont bloqués sur ce nœud sans aucun moyen de déplacement. Dans ce contexte le recourt à un bus dont la capacité change dynamiquement peut être nécessaire.

5.4.3 Protocole RTB Multigraphe : migration par groupe assistée

Dans le protocole RTB, les objets qui souhaitent se déplacer exploitent automatiquement l'unique ligne de bus existante. L'adaptabilité de la plate-forme Atlas à certains contextes de simulation plus généraux requièrt l'évolution de ce protocole vers l'intégration de plusieurs lignes de bus capables de mettre en œuvre de nombreux flux agrégés en parallèle. Le protocole RTB multigraphe issu de cette évolution peut être assimilé à la fusion de plusieurs instances du protocole RTB enrichies de nouveaux mécanismes de contrôle[4]. Le graphe formé par cette extension est un multigraphe [Berge, 1970]. Un tel graphe est caractérisé par le fait que plusieurs arcs originaires du même nœud mènent vers la même destination. On utilise parfois la terminologie *p-graphe* qui décrit un graphe dans lequel il n'existe jamais plus de p arcs de la forme (i, j) entre deux nœuds quelconques :

[4]La mise en œuvre de l'application présentée au chapitre 6 a été réalisée avec une version de la plate-forme qui intègre la version RTB du module de migration. La version courante intègre une version opérationnelle du RTB multigraphe.

$G = (X, \Gamma, \psi)$ est un multigraphe ou X est l'ensemble des nœuds, Γ est l'ensemble des arcs et $\psi : \Gamma \rightarrow P_2(X)$ est une fonction qui à chaque arc associe un élément de l'ensemble des parties à deux éléments de X. Deux arcs $\gamma 1, \gamma 2 \in \Gamma$ sont des arcs multiples ou parallèles si $\psi(\gamma 1) = \psi(\gamma 2)$

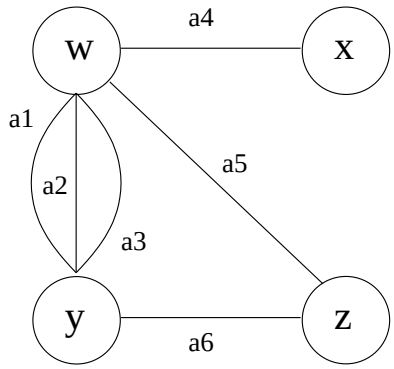


Fig. *5.7* — Multigraphe

La figure 5.7 présente un exemple de multigraphe dans lequel w, x, y, et z sont des nœuds du multigraphe et où $\psi(a1) = \psi(a2) = \psi(a3) = \{w, y\}$, $\psi(a4) = \{w, x\}$, $\psi(a5) = \{w, z\}$ et, $\psi(a6) = \{y, z\}$. Les multigraphes dans notre support de migration sont particuliers car contraints et formés par les déplacements des bus qui suivent un parcours prédéfini appelé ligne de bus. Parmi les contraintes imposées au graphe, notons par exemple, que le graphe ne peut contenir de boucle, c'est-à-dire d'arc $\{x_i, x_i\}$. Le graphe doit par ailleurs être connexe (un graphe $G = (X, \Gamma)$ est connexe si $\forall i, j \in X$ il existe un chemin de i à j) ce qui sous-entend qu'à chaque nœud passe au moins une ligne de bus (le concepteur doit prendre en compte que la réciproque est fausse). Chaque ligne de bus correspond à une instance du protocole RTB et fonctionne selon les mêmes principes. Une arrête du graphe est orientée, labellisée par le numéro de ligne de bus qu'elle représente, et pondérée par une impédance qui traduit le temps écoulé entre le départ et la livraison du bus. A chacune de ces lignes est associée une pénalité de changement de ligne (généralement utilisée pour traduire un temps d'attente). Les objets répartis sur les différents nœuds exploitent les différentes lignes de bus pour se déplacer entre les nœuds en fonction de la nature et des besoins du système complexe simulé.

Le multigraphe présenté en figure 5.8 illustre le concept de plusieurs lignes de bus. Dans cet exemple, trois lignes de bus existent entre quatre nœuds. Chaque arc du graphe est définit par un doublet *(numéro de ligne, unités de temps)* reliant les arrêts de bus entre eux. Remarquons que la ligne *1* d'impédance totale *50* correspond au chemin présenté pour le protocole de base. Le chemin *2* correspond au chemin inverse, son impédance totale est *55*. Un circuit est un chemin dont l'extrémité coïncide avec l'origine. Un chemin (respectivement, un circuit) élémentaire est défini par une suite de nœuds sans répétition (respectivement, sauf l'origine et l'extrémité). Nous qualifions les deux chemins précédents par le terme *cyclique* qui décrit des lignes de bus effectuant un parcours les menant au nœud initial sans passer deux fois par le même nœud (le cycle est hamiltonien car le circuit passe une fois et une seule par chaque nœud dans ce cas). Le chemin *3* décrit lui un parcours que nous qualifions de *linéaire*, c'est-à-dire que le bus parcourant cette ligne effectue le chemin $0 \mapsto 1 \mapsto 3 \mapsto 1 \mapsto 0$ et utilise donc un schéma aller-retour. Un tel graphe est construit en fournissant, dans le fichier de configuration de la plate-forme, une entrée décrivant chacune des lignes de bus et qui contient les paramètres suivants :

- Numéro de ligne : mis en œuvre par un entier naturel ;
- Mode de circulation : linéaire (e.g. ligne *3*) ou cyclique (e.g. ligne *1*) ;
- Chemin : la liste des identités de nœuds visités (e.g. 0-1-2-3) ;
- Impédances du chemin : chaque arc est valué avec le temps de transit du bus. Par exemple 15-25-15-20 signifie 15 unités de temps entre 0 et 1 et ainsi de suite ;
- Nœud initial : un entier naturel précisant l'identité du nœud de départ du bus ;
- Pénalité : la majoration de temps induite par la prise de cette ligne de bus lors d'un changement de ligne ;
- Taille : précise la contenance du bus.

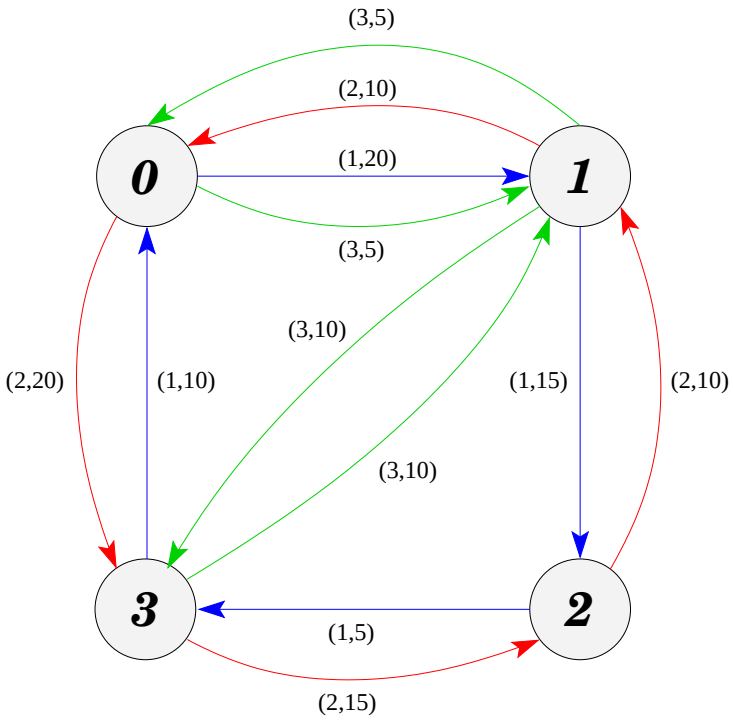


Fig. *5.8* — Plusieurs lignes de bus

Les flux d'objets au sein d'un multigraphe sont plus complexes à appréhender pour le concepteur et ce, qu'ils représentent les diverses interactions intrinsèques d'un système complexe ou, qu'ils soient des simplement groupes d'entités désagrégées sans significations particulière pour le système logique. La gestion des déplacements groupés requiert des mécanismes permettant aux objets d'atteindre leur destination. Ces mécanismes incluent par exemple, le choix d'une ligne de bus initiale (lorsque plusieurs bus transitent par le nœud d'origine d'un objet) et, les gestions des correspondances entre les lignes (certaines destinations nécessitent un changement de ligne de bus). Indépendamment de la configuration du graphe, le problème majeur pour un objet est de rejoindre sa destination si possible en employant le meilleur (optimal) chemin et éventuellement plusieurs lignes de bus. Celui-ci est traditionnellement définit par le temps minimum pour l'atteindre. L'utilisation d'un mécanisme tel qu'une table de routage est alors nécessaire. Celle-ci indique pour chaque nœud quel est le chemin à emprunter pour arriver à sa destination avec le temps de parcours le plus faible. De nombreux algorithmes existent pour le calcul des plus courts chemins d'un graphe avec chacun des

contraintes, des avantages, des méthodes d'exploration différentes, et la capacité de fonctionner sur des graphes particuliers. L'algorithme de Dijkstra [Dijkstra, 1959] fonctionne sur les graphes valués qui ne possèdent aucun circuit absorbant (i.e. un circuit de valeur négative) :

$G = (X, \Gamma, \upsilon)$ est un graphe valué ou X est l'ensemble des nœuds, Γ est l'ensemble des arcs et $\upsilon : \Gamma \rightarrow \mathbb{E}$ est une fonction de valuation des arcs tq $\forall \gamma \in \Gamma : \upsilon(\gamma) \geq 0$ (e.g. $\mathbb{E} = \mathbb{N}$)

Parmi les stratégies d'exploration classiques, existent les parcours en profondeur d'abord et en largeur d'abord. La dénomination de ces stratégies fait référence à la construction d'un développement arborescent correspondant au graphe qui est construit pendant le parcours des nœuds. L'algorithme de Dijkstra utilise une stratégie particulière d'exploration pour le calcul de la descendance à partir d'un nœud x_0 qui est qualifiée de *meilleur d'abord*. Dans le principe, cet algorithme prend en entrée le nœud x_0 et calcule le meilleur chemin pour aller à chacun des autres nœuds du graphe. Pour cela l'algorithme visite à partir du nœud initial, tous les autres en choisissant à chaque étape, le nœud qui possède la plus petite valuation cumulée (les valeurs des arcs sont ajoutées à chaque avancement dans le graphe). Les nœuds passent ainsi par les états *dehors*, le nœud n'a pas été visité, *atteint*, le nœud à été atteint par l'un des chemins d'exploration, *recouvert*, le nœud a été traité et le meilleur chemin est connu. Cet algorithme possède une particularité qui fait que certains nœuds obtiennent l'état *recouvert* alors que le calcul n'est pas fini caractérisant ainsi cet algorithme parmi la classe des algorithmes dits gloutons (des résultats partiels sont obtenus et définitifs avant le résultat final).

L'algorithme mis en œuvre dans le module *RTB Multigraphe* pour le calcul des chemins de valeur optimale est basé sur celui de Dijkstra. Bien que cet algorithme fonctionne pour les multigraphes, il ne prend en compte dans sa forme originelle que des valuations simples. Dans le type de graphe que nous déployons pour la plate-forme, une double valuation existe où chaque arc du graphe est défini par un doublet *(numéro de ligne, unités de temps)* comme illustré en figure 5.8 :

$G = (X, \Gamma, \psi, \phi)$ est un multigraphe valué, si G est un multigraphe et $\phi : \Gamma \rightarrow \mathbb{E}$ est une fonction de valuation des arcs. Nous définissons cette fonction par $\forall \gamma \in \Gamma : \phi(\gamma) = (\lambda, \nu)$ et $\nu \geq 0$

La table 5.1 illustre le fonctionnement de l'algorithme adapté de Dijkstra pour le calcul des plus courts chemins issus du nœud *0*. La gestion de l'ensemble des nœuds d'état *atteint* est effectuée à l'aide de la structure de donnée appelée *tas* qui correspond à une liste de triplets *(nœud, numéro de ligne, valeur)*, ordonnée dans l'implémentation par valeurs croissantes. La phase d'initialisation consiste à mettre dans le tas, le triplet correspondant au nœud initial. La première étape consiste à prendre le nœud qui possède la plus petite valeur dans le tas, à savoir le nœud *0* dont les successeurs sont *1* et *3*. Le tas est ensuite rempli par l'étude des différents chemins possibles allant du nœud *0* à ses successeurs. Par exemple le chemin menant au nœud *3* permet de construire le triplet *(N,L,T)* définit par :

- N = nœud de destination, c'est-à-dire 3 dans cet exemple ;
- L = dernière ligne de bus utilisée ; $null + 2 = 2$. Nous noterons, dans un souci de clarté, les utilisations successives, mais remarquons que dans la pratique notre algorithme ne conserve que le numéro de la dernière ligne et construit en parallèle une liste de doublet décrivant les nœuds parents : (nœud 0, ligne 2) dans ce cas ;
- T = somme des valuations : $0 + 20 = 20$.

Lorsque toutes les successions possibles depuis le nœud initial ont été explorées, celui passe dans l'état *recouvert*. Ainsi tout chemin menant à un nœud déjà recouvert est systématiquement ignoré car obligatoirement plus long (heuristique principale assurant la terminaison de l'algorithme), du fait du parcours par la stratégie *meilleur d'abord*. Remarquons que le tas à l'issue de la première étape contient deux chemins menant au nœud *1*. L'algorithme original de Dijkstra ne considère dans ce cas qu'une seule possibilité et aurait éliminé le triplet *(1,1,20)* alors qu'avec la double valuation il n'est pas encore possible d'écarter ce triplet (la ligne est différente). Les étapes suivantes procèdent de manière similaire jusqu'à ce que tous les nœuds passent dans l'état *recouvert*. Remarquons la présence dans ces étapes des valeurs P_1, P_2 et P_3 qui correspondent aux pénalités (un entier naturel ≥ 0) de changement de ligne auxquelles nous attribuons la valeur 20 afin de fixer un résultat pour la dernière étape. Cet algorithme utilise par ailleurs une deuxième heuristique qui correspond à un affaiblissement de celle qui aurait supprimé le triplet *(1,1,20)*. Dans la deuxième étape le triplet *(3,1+3,30+P_3)* peut être écarté du calcul car le triplet *(3,3,15)* aboutit au même nœud par la même ligne et possède une valeur inférieure à *30+P_3* quelle que soit la valeur de la pénalité.

Etape	Nouveau	Successeurs	Tas (N,L,T)	Recouverts	Chemin (N,L,T)
init			(0,null,0)		
1	0	1,3	(1,3,5) (1,1,20) (3,2,20)	0	(0,null,0)
2	1	0,2,3	(2,3+1,20+P_1) (2,1,35) (3,3,15) (3,1+3,30+P_3) (3,2,20)	0,1	(1,3,5)
3	3	0,1,2	(2,3+1,20+P_1) (2,1,35) (2,2,35) (2,3+2,30+P_2) (2,1+3+2,45+P_3+P_2)	0,1,3	(3,3,15)
4	2	1,3	null	0,1,2,3	(2,1,35) (2,2,35)

Tab. *5.1* — Algorithme de Dijkstra modifié

La table 5.2 présente un exemple de table de routage qui est obtenu avec cet algorithme. Par exemple, la migration d'un objet du nœud initial *0* vers le nœud *2* implique de prendre la ligne numéro *2* pour arriver 35 unités de temps plus tard sur celui-ci. Au passage le bus s'arrêtera au nœud *3*. La plate-forme Atlas calcule automatiquement ce type de table lors de son initialisation par l'utilisation des informations contenues dans le fichier de configuration (i.e. chaque nœud de la grappe se définit comme nœud initial, ainsi chacune des instances de la plate-forme a sa propre table distincte des autres). Cette table peut ensuite être mise à jour à n'importe quel moment pendant une simulation. Un emploi possible du calcul dynamique du routage est la prise en compte d'information de congestion des lignes qui conduirait à la mise jour de nouvelles valeurs de pénalité pour chacune des lignes de bus.

Destination	(N1,L1,T1)	(N2,L2,T2)
0	(0,null,0)	
1	(1,3,5)	
2	(3,2,20)	(2,2,35)
3	(1,3,5)	(3,3,15)

Tab. *5.2* — Table de routage

Conçu avec l'hypothèse que l'objet ne connaît que sa destination sans, a priori, aucune information sur la nature du graphe et des chemins possibles, le protocole permet non seulement de réaliser par lui-même la migration physique mais aussi d'assister l'objet dans ses choix de déplacement qui le mèneront à sa destination (incluant la gestion des correspondances). Parmi les quelques perspectives de ce protocole, notons l'extension de l'utilisation des pénalités de ligne pour la montée initiale. Remarquons cependant que l'emploi de telles variations dans les temps de parcours ne peuvent, comme la pénalité de changement de ligne, être évaluées que par le concepteur et correspondent en principe au temps moyen d'attente du passage du bus sur chaque nœud. Notons également une amélioration possible de la table de routage qui proposerait de choisir son déplacement parmi toutes les lignes possibles en fonction de paramètres établis notamment au niveau conceptuel (e.g. indice de préférence).

5.5 Module de gestion d'objets

Le module de gestion des objets (MAN) constitue l'interface privilégiée d'interaction entre l'application distribuée et la plate-forme. Ce module met en œuvre, pour chaque machine du système distribué, une mémoire locale appelée à communiquer avec les autres mémoires afin de former une mémoire globale. Les fonctions principales offertes par ce service sont la création, la destruction et la mise à jour d'objets dans le système. Chaque mémoire locale est une liste non ordonnée d'éléments appelés cellules mémoires. Chacune de ces cellules contient un objet et l'état qui lui est associé. Les objets contenus dans les cellules sont les représentations informatiques des entités définies par le concepteur et sont, comme pour le module de migration, de type quelconque (éventuellement agrégé).

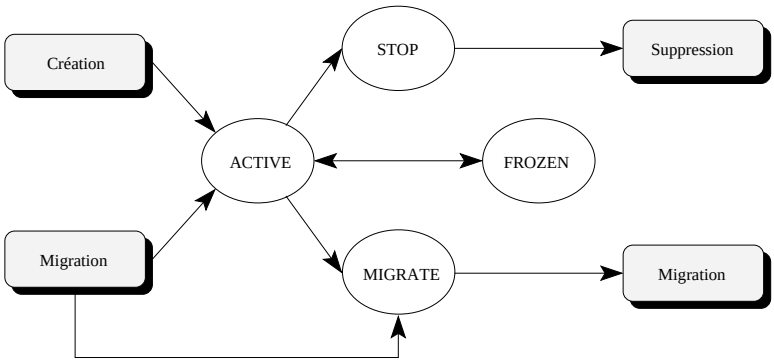


Fig. *5.9* — États possibles des objets

En cours d'exécution, chaque objet possède un attribut définissant son état à l'instant courant (cf. figure 5.9). Par défaut les objets du système sont dans l'état *active* et le choix du

changement d'état est contrôlé par l'application de l'utilisateur. L'attribut *migrate* signifie que l'objet dans cet état est en attente de déplacement vers une autre machine. Chaque passage de l'état *active* à l'état *migrate* (actuellement non réversible) induit l'ajout automatique de l'objet dans une liste de migration gérée par ordre *Fisrt In First Out : FIFO* et son déplacement vers la structure d'arrêt de bus (zone départ) de ligne bus correspondante. L'état *frozen* est utilisé pour traduire une indisponibilité temporaire pendant laquelle le module MAN bloque l'accès à un objet. Un objet positionné dans l'état *stop* est un objet qui a terminé son cycle de vie dans le système. Cet état rend l'objet invalide en attendant la collecte des objets morts par le module (Atlas force le *garbage collector* de la *JVM* en fonction de l'accélération donnée à la simulation). La connexion entre l'action de migration et l'état *migrate* dans la figure 5.9 traduit la correspondance d'un objet en déplacement (i.e. change de ligne de bus et passe d'une zone d'arrivée directement à la zone de départ d'une autre ligne).

5.6 Module de gestion du temps virtuel

La plate-forme intègre un module de gestion du temps de simulation (module CLOCK) qui fournit un support pour dater les événements et guider l'évolution d'une simulation par une approche dirigée par le temps. Cette fonctionnalité permet aux machines du système distribué d'établir une correspondance entre leur temps physique (horloge interne) et un temps de simulation par l'utilisation d'un facteur de compression donné en paramètre à la plate-forme. Une correspondance temporelle appropriée permet alors d'effectuer des simulations accélérées (ou éventuellement ralentie) dans lesquelles une évolution temporelle longue (e.g. 6 mois) peut être exécutée par les ordinateurs en un temps physique court variable selon les besoins et les performances disponibles.

La table 5.3 issue de [Zeigler et al., 2000] illustre une taxonomie temporelle en fonction de l'aspect abstrait ou réel (respectivement logique ou physique) du temps et de la perception qu'en ont les composants d'un système complexe (propre ou non à chaque composant). Les travaux en modélisation et en simulation retiennent traditionnellement le cas *A* dans lequel chaque composant utilise un temps logique et global qui est relatif aux spécifications temporelles du système simulé. Toutefois, l'exécution répartie d'une simulation (et de ses composants) sur les systèmes distribués rend difficile le maintien de cette abstraction. En effet, les simulations reposant sur les systèmes distribués exploitent des architectures offrant, pour chaque machine, un temps physique et local (cas *D*).

	Temps Logique	Temps Physique
Temps Global	(A) Global, Logique : tous les composants opèrent sur le même temps abstrait	(B) Global, Physique : tous les composants opèrent sur la même horloge système
Temps Local	(C) Local, Logique : un composant opère sur son propre temps abstrait	(D) Local, Physique : un composant opère sur sa propre horloge système

Tab. *5.3* — Taxonomie du temps

La simulation par un système distribué de phénomènes urbains et sociétaux nécessite de manière générale la capacité d'accélérer l'exécution afin de reproduire en un temps court une évolution temporelle longue. La définition d'un temps de simulation adapté à ce contexte est

réalisée en deux étapes. La première étape est la définition d'un temps logique permettant de faire avancer la simulation. Cette étape implique d'identifier un mécanisme de passage du cas D au cas C qui correspond à une approche de simulation asynchrone. Nous avons introduit en section 3.4 les mécanismes coûteux que peut nécessiter cette approche pour coordonner les évolutions temporelles des différents composants et proposions alors de considérer une approche de simulation synchrone. La deuxième étape est la définition d'un temps global permettant de préserver la relation causale des événements produits par les différents composants répartis. Cette étape implique d'identifier un mécanisme de passage du cas C au cas A qui correspond à une approche de simulation synchrone.

5.6.1 Temps logique

Notre objectif est d'offrir un temps logique qui correspond à la simulation des systèmes urbains et sociétaux. La plate-forme Atlas intègre un temps abstrait qui définit une horloge virtuelle dont le format (jour, heure, minute) correspond à ces systèmes. La mise en œuvre de ce temps simulé repose sur l'établissement d'une correspondance, que nous nommons *compression temporelle* avec le temps physique (l'horloge de l'ordinateur). Cette correspondance est réalisée par une relation qui préserve l'ordre établi dans le temps physique à un niveau logique et définissant le temps simulé de manière accélérée par rapport au temps physique.

Une base temporelle est définie par le pas minimal d'avancement de la simulation (cf. section 1.3). La base temporelle implémentée dans le module CLOCK est définie par *Base_Temporelle = 100 ms * compression*. Cette base temporelle physique est mise en correspondance avec une minute dans le temps logique. Le concepteur choisit la durée effective de la correspondance en fixant la valeur de la variable *compression*. Cette fonctionnalité paramétrable d'Atlas permet ainsi une souplesse dans l'étude et l'exécution de différentes simulations. Les durées des différents niveaux de granularité temporelle de notre horloge virtuelle sont définies par :

- $Temps_{minute} = Base\_Temporelle$
- $Temps_{heure} = Base\_Temporelle * 60$
- $Temps_{jour} = Base\_Temporelle * 60 * 24$

La figure 5.10 illustre la méthode de compression. Lorsque dans le temps logique T_{sim}, onze minutes se sont écoulées entre le temps initial de la simulation T_{log0} et l'événement produit à T_{log1}, pour la plate-forme Atlas, onze pas temporels se sont écoulés. Ceux-ci correspondent à un temps réel de simulation de 5,5 secondes lorsqu'à titre d'exemple *compression = 5*.

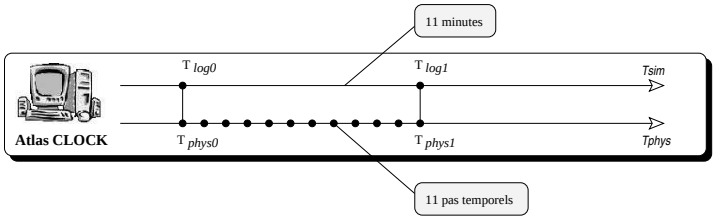


Fig. *5.10* — Compression temporelle

Le module CLOCK, dans sa mise en œuvre, compte le temps physique par le nombre de millisecondes passées entre un temps T_{physi} et le temps physique initial T_{phys0} (ne comptabilise

le nombre de pas mais leur durée totale). Par exemple lorsque *compression = 10*, une durée de 10620 ms dans le temps physique correspond à un temps logique décrit par le triplet (jour, heure, minute) égal à (0,0,10). En référence à la figure 5.10, considérons la variable $TP = T_{phys1} - T_{phys0}$. Le temps logique T_{log1} calculé par Atlas est alors défini par le triplet de valeurs entières $(TL1_{jour}, TL1_{heure}, TL1_{minute})$ tel que :

- $TL1_{jour} = TP/Temps_{jour}$
- $TL1_{heure} = (TP - (TL1_{jour} * Temps_{jour}))/Temps_{heure}$
- $TL1_{minute} = (TP - ((TL1_{jour} * Temps_{jour}) + (TL1_{heure} * Temps_{heure})))/Temps_{minute}$

Nous définissons pour le type de simulation offert par Atlas, le concept de performance par l'accélération imposée à l'horloge virtuelle. Ceci signifie que, en opposition à l'accélération communément appelée *speedup* qui caractérise une comparaison entre les vitesses d'exécution répartie et séquentielle, nous considérons la différence entre la période de temps du système simulé et son temps d'exécution par Atlas. Ce concept de performance est jugé pertinent car plus une période du temps logique peut être exécutée rapidement, plus l'environnement de simulation est considéré performant car plus la charge en termes de calcul est importante nécessitant alors des ressources informatiques adaptées. Cette conception de la performance correspond à une recherche de la compression temporelle maximale. Kai Nagel utilise également ce concept qu'il définit par le terme Real Time Ratio (RTR) et qu'il présente dans [Simon et Nagel, 1998] où lorsque simulant les trafics de véhicules pour la ville de Dallas sur une période de 24 heures à l'aide de *TRANSIM*, il obtient ratio un de 28.

Dans les approches de simulation classique, l'exécution avance selon ses propres besoins car il existe un découplage entre le temps physique et le temps logique. Ce type d'exécution va systématiquement à la vitesse maximale en fonction des différences potentielles de puissance entre les ordinateurs et leurs liens de communication. Dans l'approche proposée, la vitesse à l'avantage de pouvoir être calibrée en fonction des besoins de l'utilisateur qui pour cela fait varier la variable *compression*. Lorsque l'objectif est la rapidité d'exécution (sans considération pour le suivi en temps réel), une itération d'accélération progressive du temps de simulation permet d'identifier la limite de performance d'une configuration physique donnée. Remarquons que des capteurs de défaillances intégrés dans Atlas ont montré que l'exécution distribuée des simulations permettait d'accélérer la compression temporelle bien au-delà des limites induites par une exécution mono-machine (cf. section 6.3.6).

5.6.2 Temps global

Dans un modèle d'interaction asynchrone sans horloge globale (i.e. système distribué), il n'existe pas de temps physique partagé pouvant servir de support à la construction d'un temps logique global. La conception d'un tel temps de simulation synchrone est traditionnellement mis en œuvre par l'utilisation d'un algorithme de consensus [Guerraoui et al., 1998] entre les ordinateurs (processus ou composants répartis) validant le passage commun d'un pas temporel au suivant. Ce type d'approche permet une séparation du temps logique et du temps physique mais requiert, pendant toute la durée de la simulation, l'exécution de mécanismes d'interaction coûteux et non productifs dans le traitement du modèle simulé.

Dans notre approche, le passage d'un pas temporel à l'autre est automatiquement déclenché par l'avancement du temps physique qui s'écoule sans discontinuer (comme introduit en section précédente). Postulant que la vitesse de progression des horloges physiques est identique sur chaque machine, cette propriété (universalité du temps) permet d'envisager

une simulation synchrone où le synchronisme est alors réduit à la capacité de déterminer un instant dans le temps physique commun à toutes les machines pour définir le temps logique initial. Un consensus unique en début de simulation établit alors un accord sur l'heure de début de la simulation dans le temps réel. La simulation se déroule ensuite selon un schéma où chaque processus évolue en calculant son temps logique courant par la différence entre le temps physique courant et le temps physique de référence établit par consensus.

Le difficulté majeure dans cette approche est que l'algorithme repose sur la capacité de déterminer, dans une simulation à N ordinateurs asynchrones, N temps physiques qui permettent de définir un temps logique initial unique. Les méthodes de synchronisation des horloges physiques dans les systèmes à réseaux synchrones[5] peuvent être mis en œuvre simplement. Celles-ci reposent traditionnellement sur la connaissance des bornes dans les temps de communications. Dans un cadre plus général, où sont considérés les systèmes répartis asynchrones, il existe différentes méthodes et protocoles pour synchroniser les temps physiques. Pour des larges réseaux de type Internet, le protocole *Network Time Protocol* (NTP) peut être utilisé. Dans le cas de réseaux locaux, plusieurs algorithmes ont été mis en œuvre tel que l'algorithme de Cristian [Cristian, 1989], l'algorithme de Berkeley [Gusella et Zatti, 1985]. Plus récemment [Perumalla et Fujimoto, 2001] propose un algorithme de synchronisation de temps virtuel global (GVT, cf. section 3.4.2) pour les réseaux non fiables. L'utilisation de ces algorithmes permet de définir un temps physique identique à toutes les machines participant à la simulation. Cette synchronisation à un niveau physique permet au simulateur de déterminer une heure simultanée dans le temps réel à laquelle toutes les instances réparties de la simulation déclencheront l'initialisation du temps logique initial.

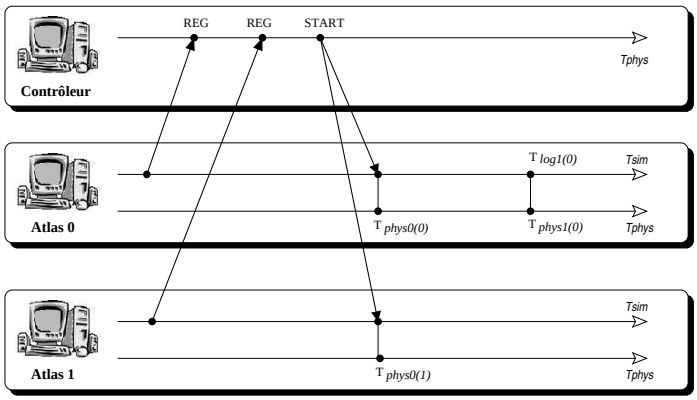


Fig. *5.11* — Synchronisation temporelle

Dans un objectif de prototypage, le module CLOCK intègre une implémentation dans laquelle la méthode d'établissement du temps logique global présentée ci-dessus est approchée. Cette approximation repose sur le fonctionnement du réseau local présenté en figure 5.1 où il est possible d'obtenir une variation des temps logiques inférieure au pas temporel (avec *compression* ≥ 2). Au stade initial de la simulation, tous les nœuds démarrant leur exécution s'enregistrent auprès du serveur par l'envoi d'un message estampillé *REG* tel qu'illustré en figure 5.11. Lorsque tous les enregistrements ont été effectués, le contrôleur envoi simultanément un ordre de démarrage à chaque instance d'Atlas. Dès la réception de la commande, chaque machine i associe son temps logique initial avec le temps physique courant $T_{phys0(i)}$.

[5]Environnements où les délais de transit des messages sont bornés. En opposition, les réseaux asynchrones ne permettent nullement d'évaluer les temps d'échanges de messages. Pire, ils ne permettent pas de faire de distinction entre message perdu ou retardé.

Le respect de la précédence causale (cf. section 3.3) dans ce temps simulé est assuré car, primo l'utilisation du temps physique induit un ordre partiel des événements dans le temps logique de chaque machine et, secundo les flux de données désagrégées sont opérés par les bus RTB et donc contraints temporellement par leurs délais de transmission et de livraison. Remarquons toutefois que l'ordre *FIFO* et l'ordre causal ne peuvent être assurés car les bus (messages) ont le droit de se doubler dans le temps simulé en fonction des impédances.

Le problème principal de notre mise en œuvre est que l'affranchissement des hypothèses (arrivée quasi-synchrone des commandes de démarrage) peut induire des variations dans les démarrages des simulations. D'un point de vue général (système asynchrone), toute la difficulté de l'approche proposée résulte des incertitudes sur les latences et les débits d'un système distribué. L'utilisation d'algorithmes adaptés permettrait d'offrir la capacité de définir sur chaque machine un temps initial synchronisé. L'avantage majeur de cette approche est de pouvoir se passer de toute synchronisation pendant la simulation limitant ainsi la charge des nœuds de calcul au seul traitement des événements productifs du point de vue de la simulation.

5.7 Support graphique

Le module INTER de la plate-forme fournit un support pour la gestion des simulations et le développement des interfaces graphiques de l'utilisateur. Ce composant est constitué de deux parties ; l'une gère les paramètres de simulation (e.g. les nœuds de calculs enregistrés) et l'autre fournit une interface graphique permettant de contrôler les simulations (figure 5.12). Le centre de contrôle d'Atlas est un panneau graphique composé de trois onglets. L'onglet *Simulation* permet notamment de gérer le démarrage des simulations et le choix d'affichage (horloge ou message d'information). L'onglet *Interaction* permet de modifier dynamiquement les paramètres de fonctionnement de la plate-forme (e.g. changement de la taille des mémoires locales) et l'onglet *About* affiche les informations sur le statut de la plate-forme (e.g. version).

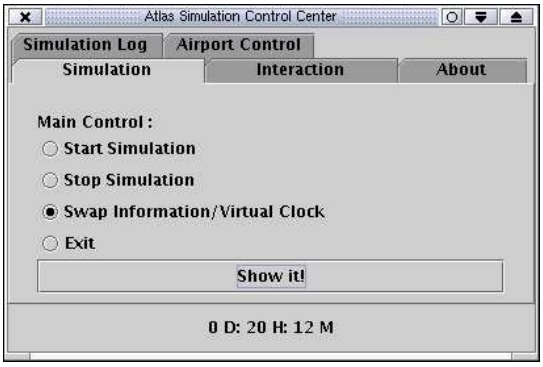
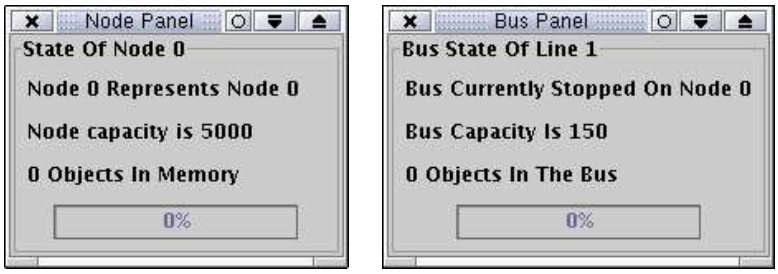


Fig. *5.12* — Centre de contrôle Atlas

Atlas autorise par ailleurs l'ajout d'onglets supplémentaires sur son interface (greffons). Les deux onglets *Simulation Log* et *Airport Control* en fournissent l'exemple. L'onglet *Simulation Log* permet une visualisation en temps réel des événements se produisant dans la plate-forme (e.g. état du bus RTB ou des mémoires locales) et capturés par le module LOG. L'onglet *Airport Control* contient le panneau de contrôle développé pour l'application aéroport présentée au chapitre 6 et contient les menus permettant d'afficher les panneaux graphiques externes de cette application (figure 6.5).

5.8 Module de surveillance

Atlas est une plate-forme qui intervient comme support de simulation pour des applications distribuées. Toutefois Atlas elle-même produit des événements datés au cours d'une simulation. Ceux-ci peuvent être interceptés par l'application en tant qu'information de simulation. Le rôle du module de surveillance (LOG) est de contrôler ces événements internes. Les données recueillies par ce composant sont ensuite reportées, selon la convenance, dans des fichiers, simplement ignorées ou encore affichées depuis le centre de contrôle à l'aide de différents panneaux graphiques dont la figure 5.13 illustre deux exemples (respectivement état des nœuds physique et des bus).



(a) État des nœuds (b) État des bus

Fig. *5.13* — Support graphique du module LOG

Le système distribué (physique) incluant les nœuds, les canaux de communication et l'horloge sont utilisés par Atlas afin de reproduire un système complexe (logique). Un élément essentiel pour ce type de simulation est que la gestion du système physique n'induise pas d'effets de bord sur les fonctionnements et contraintes du système logique, en particulier au niveau de la gestion des flux logiques. L'augmentation de la vitesse de simulation diminue le temps d'exécution mais augmente les charges processeurs et réseau. Ainsi, comme la simulation séquentielle, la solution distribuée n'est pas exempte de limites à ne pas franchir pour éviter les effets de bord indésirables. Le mécanisme de surveillance permet la calibration de la vitesse maximale de simulation. Le concepteur augmente la compression temporelle, jusqu'à ce que le module LOG indique que la simulation est trop rapide. Parmi les éléments surveillés par le module LOG, le respect des délais de transit et de livraison des bus du protocole RTB est un aspect important. En effet, un bus qui ne respecte pas les délais qui lui sont imposé conduit à des simulations dans lesquelles les entités migrantes (e.g. des personnes, des véhicules) ne peuvent respecter leurs échéances temporelles, entraînant la simulation vers des états erronés d'un point de vue du modélisateur. En d'autres termes le module LOG garantit que les contraintes de déplacement résultent de propriétés du modèle logique et non de surcharges du réseau physique. L'utilisation du module de surveillance dans ce contexte permet de certifier un déroulement correct de la simulation. Au-delà l'apparition de défaillances peut intervenir soulignant la surcharge des nœuds ou du réseau.

5.9 Conclusion

Ce chapitre présente la plate-forme de simulation Atlas. Dernière étape d'une méthodologie de conception, cet environnement établit un lien avec les architectures distribuées dont les principes d'utilisation sont présentés section 5.1. La section 5.2 introduit les

principales caractéristiques d'Atlas, puis les sections 5.3 à 5.8 présentent les six modules qui composent la plate-forme. Cette plate-forme contrôle de manière transparente les communications, fournit une interface générique paramétrable et exploitable pour différents contextes tout en permettant l'ajout de panneau définis par l'utilisateur. Elle gère le stockage local des objets et permet une migration des objets utilisateur représentant des entités de différents niveaux d'agrégation. Cette migration repose sur un concept innovant de migration par groupe qui est adapté à plusieurs systèmes complexes à larges flux de données désagrégées. En particulier ce système de migration est capable de reproduire la nature des déplacements agrégés lors de simulation de système de transport collectif incluant éventuellement plusieurs flux grâce à l'extension du mécanisme de déplacement d'objets au sein d'un multigraphe.

Un des objectifs de prototypage de la plate-forme est de vérifier dans quelle mesure peuvent être étendues les qualités et les relations d'un système complexe à larges flux de données désagrégées dans le domaine du symbole metaphorisé (i.e. domaine formant un graphe dans lequel un bus voyage entre l'ensemble de nœuds). Le chapitre suivant a pour but de confronter cette transposition, l'exploitation de notre méthodologie et de sa plate-forme distribuée, en présentant la simulation d'un terminal d'aéroport dans lequel les flux de données désagrégées représentent les déplacements de personnes entre les différents halls.

CHAPITRE 6

Application à la simulation de flux de personnes

« Life can only be understood backwards, but must be lived forwards. Søren Kierkegaard, philosophe danois (1813-1855) »

Afin d'illustrer le potentiel de notre approche de la simulation distribuée pour les systèmes à larges flux de données désagrégées nous proposons sa mise en application pour l'étude des mobilités de personnes dans un système en transport à moyenne échelle. Ce chapitre présente une étude de cas qui modélise et simule les flux de personnes entre différentes configurations des halls du terminal 2 de l'aéroport Paris Charles De Gaulle (CDG2). A travers cette application, nous présentons les résultats de différentes simulations, paramétrées avec des conditions initiales et des contraintes de flux de données particulières. Cette application, dont l'objectif est de valider la pertinence de la simulation répartie, illustre également l'utilisation des outils de contrôle et d'interaction fournis par notre plate-forme distribuée et le gain apporté tant au niveau de la reproduction du modèle qu'au niveau des performances. La section 6.1 de ce chapitre présente les spécifications du cas d'étude ainsi que la modélisation du terminal à l'aide notre approche hiérarchique basée sur l'utilisation des graphes. La section 6.2 présente quelques aspects de la mise en œuvre du terminal. La section 6.3 présente les simulations réalisées et les résultats obtenus. La section 6.4 conclut ce chapitre.

6.1 Modélisation d'un terminal d'aéroport

Un des aspects importants dans un système aéroportuaire est l'analyse de la distribution des flux de passagers à l'intérieur d'un terminal. Dans le contexte de simulation et d'analyse de flux de personnes dans un terminal d'aéroport, l'évaluation de différentes organisations du terminal, des infrastructures disponibles, et des capacités de transport est d'un grand intérêt dans l'expression de besoins futurs et la compréhension des comportements du système actuel. La validation expérimentale d'un tel système est basée sur l'hypothèse que bien que la simulation de flux de personnes ne soit pas un processus complètement déterministe, de telles simulations restent utiles, par les tendances qu'elles dégagent, pour la planification des logistiques de transport et des configurations dans un contexte d'aéroport [Gatersleben et van der Wiej, 1999].

L'application proposée dans ce chapitre modélise et simule les flux de personnes entre différentes configurations des halls du terminal 2 de l'aéroport Paris CDG [Ray et Claramunt, 2002a]. L'objectif de notre recherche, à travers ce cas d'étude, est de montrer le potentiel de la plate-forme de simulation Atlas comme outil supportant une planification préliminaire de ce terminal, et au-delà d'illustrer le potentiel de notre approche de la simulation distribuée pour les systèmes à larges flux de données désagrégées [Ray et Claramunt, 2002b].

Abordant la simulation distribuée, Kai Nagel précise dans [Nagel, 2002] que la nature exacte de la micro-simulation importe peu tant que celle-ci intègre quelques spécifications minimums (présenté dans le domaine d'étude des systèmes de trafic). Les plus importantes étant, rapportées à notre contexte, que (1) les personnes dans la micro-simulation suivent un objectif et que (2) la micro-simulation peut être exécutée rapidement par le système informatique sous-jacent même dans le cas de problèmes complexes nécessitant beaucoup de ressources. Le propos de notre mise en pratique ne pouvant ainsi être réduit à la validité d'un modèle (cf. section 1.4) qui reproduirait de manière exacte et exhaustive la nature d'un système aéroportuaire, la modélisation effectuée est essentiellement orientée vers la description des flux et des propriétés les plus significatives.

6.1.1 Simulation de systèmes aéroportuaires

Plusieurs travaux ont été proposés, dans différents contextes, pour la simulation de systèmes aéroportuaires et notamment pour l'étude des flux de passagers. Par exemple, [Gatersleben et van der Wiej, 1999] présentent une micro-simulation qui analyse les flux de passagers entre l'enregistrement et l'embarquement, à l'immigration et pendant les transferts. Cette modélisation est utilisée pour la conception interne et l'optimisation d'attente de passagers à l'aéroport de Schiphol en Hollande. Dans [Setti et Hutchinson, 1994] un modèle d'approximation des fluides, basé sur les principes de simulation utilisés en hydrologie, simule le fonctionnement des équipements de terminaux. Une mise en œuvre informatique de cette approche, de laquelle résulte le langage de simulation *TERSIM* dédié à la modélisation de passager dans un terminal, a été appliquée pour la simulation du fonctionnement d'un terminal de l'aéroport de Toronto.

Certains autres travaux abordent d'avantage la planification de terminal d'aéroport. Ceux-ci incluent des micro-simulations modélisant, en particulier, la répartition d'avions aux portes d'embarquement [Cheng, 1998], l'organisation des vols [Teretsch, 1993] et le trafic au sol [Tunasar et al., 1998]. Dans un contexte similaire, le système *SEEDS* présenté dans [Rackl et al., 1999] est un prototype de simulation dont l'objectif est de représenter les contrôles de trafics au sol dans un aéroport. Cette étude à en particulier l'intérêt de reposer sur une approche distribuée mise en œuvre par l'utilisation du système *CORBA*.

6.1.2 Description du cas d'étude

Le but de cette modélisation et des simulations qui en découlent est de représenter au niveau le plus fin de granularité les décisions de mobilités de personnes au sein du terminal 2 de CDG dont les flux sont opérés par un transport collectif. La figure 6.1 rappelle la méthode de modélisation proposée dans le chapitre 4 qui consiste pour ce système aéroportuaire à identifier et à projeter les propriétés statiques et dynamiques du système réel vers l'environnement distribué. La modélisation conceptuelle du terminal produit un modèle dans lequel se déplace une navette qui transporte les personnes entre les halls qui composent ce terminal (i.e.

flux agrégé de données désagrégées). La mise en œuvre des structures de données et de leurs propriétés en termes de représentation informatique est réalisée par (1) le paramétrage de la plate-forme Atlas à l'aide de certains des éléments de modélisation (e.g. graphe représentant le terminal) et (2) par la conception d'une application distribuée qui se base sur les services de la plate-forme (e.g. horloge virtuelle) pour coder le modèle du terminal.

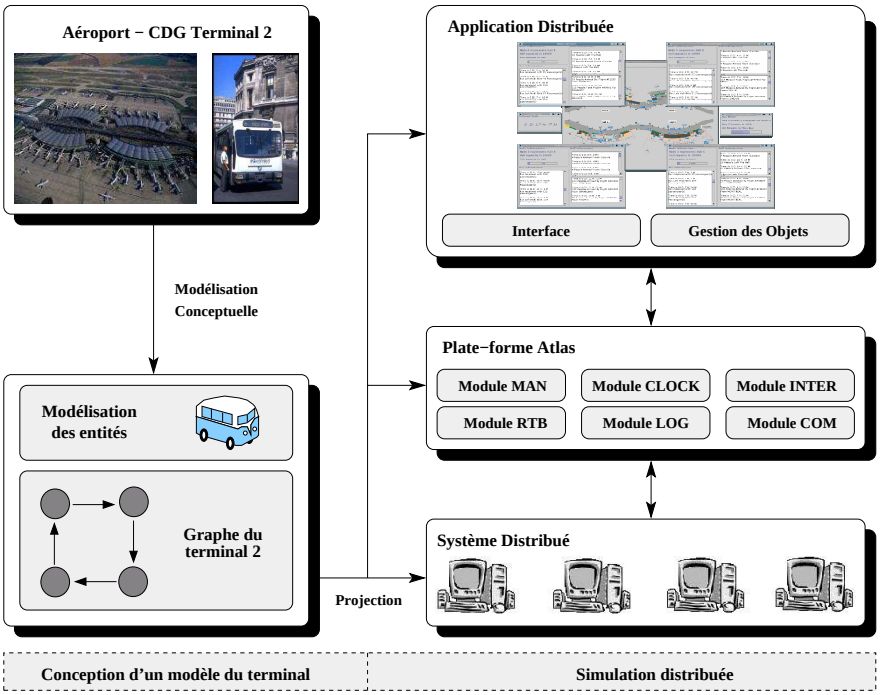


Fig. *6.1* — Principe de modélisation de l'aéroport CDG

Notre étude de cas présente, en fonction de différentes configurations et d'états initiaux, quelques propriétés intéressantes par rapport à notre objectif de simulation. Tout d'abord les activités des halls requièrent une capacité de traitement importante. Ensuite, les flux de personnes sont volumineux et contraints par la planification des vols qui permettent de dériver certains comportements émergents, des goulots d'étranglement éventuels, et des performances au niveau du terminal.

6.1.3 Modélisation hiérarchique

Le terminal 2 de CDG est modélisé à partir d'une approche hiérarchique qui organise l'information à différents niveaux de granularité (figure 6.2). Cette décomposition hiérarchique est particulièrement adaptée pour la modélisation de systèmes devant être implantés sur un système distribué notamment car elle fait clairement apparaître la structure de graphe requise pour la mise en œuvre vers l'environnement distribué (cf. section 4.3). La description sous forme d'arbre des différents éléments constituant le terminal a été conduite par raffinements successifs en partant du niveau agrégé vers les sous-niveaux désagrégés faisant ainsi apparaître trois composants essentiels. Tout d'abord ce schéma, mêlant description structurelle et description fonctionnelle, permet d'identifier les entités du modèle (les nœuds de l'arbre). Ensuite, ce schéma fait apparaître le composant qui sera représenté par un graphe (les halls encadrés en figure 6.2). Enfin, il fait apparaître le comportement dynamique principal des entités désagrégées (les feuilles de l'arbre).

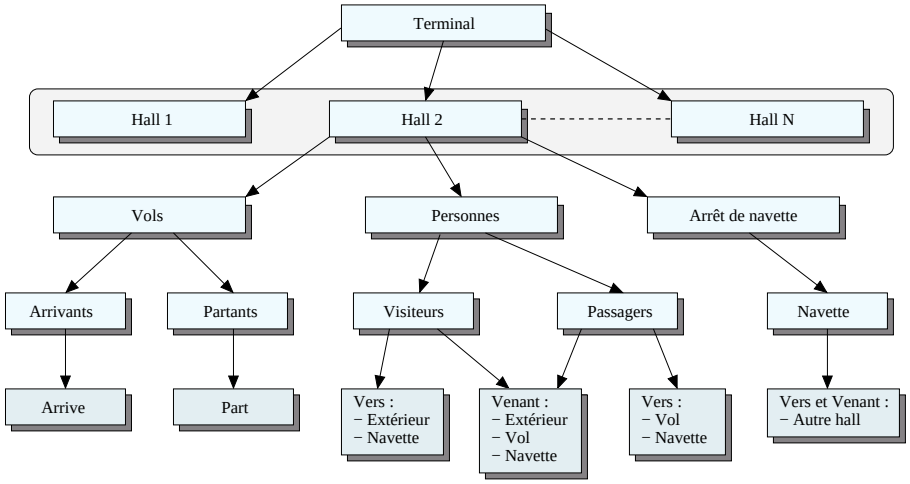


Fig. *6.2* — Modélisation hiérarchique d'un terminal de l'aéroport CDG

Les spécifications, desquelles dérive cette modélisation hiérarchique, considèrent que le terminal 2 de CDG est composé de halls entre lesquels interviennent des déplacements de personnes. Les flux de personnes sont opérés par une navette qui réalise un parcours cyclique et régulier entre ces halls prenant/déposant les personnes à leur origine/destination (les arrêts de navette associés à chaque hall). Les flux de personnes entre les différents halls sont constitués de passagers de vols en partance et à l'arrivée, et de visiteurs. Les entités modélisées dans l'arbre de cette application sont ainsi les suivantes :

- Les personnes : passagers des vols ou visiteurs (l'ensemble des objets pouvant migrer) ;
- Les halls et leurs interconnexions (modélisés par la structure de graphe) ;
- Une navette et des arrêts de navette ;
- Les vols (arrivées et départs).

6.1.4 Modélisation fonctionnelle

Dans ce système aéroportuaire, le monde *extérieur* désigne l'environnement au-delà de la borne supérieure du modèle de terminal. Au cours d'une simulation, des personnes arrivent régulièrement de l'extérieur : entrée par les portes des halls ou par l'arrivée d'un vol. De plus des personnes quittent régulièrement les halls selon les mêmes principes. Les visiteurs se déplacent entre les halls sur une base aléatoire et les passagers en fonction de l'heure et du hall de départ de leur avion. La dynamique des personnes produit des événements (lors de la simulation) qui sont schématisés selon les possibilités suivantes :

- Arrivée de l'extérieur vers un hall ;
- Arrivée d'un vol vers un hall ;
- Sortie d'un hall vers l'extérieur
- Sortie d'un hall vers un vol ;
- Passage d'un hall vers un autre hall en utilisant la navette.

Les états des personnes dans le terminal dépendent de leur origine, de leur destination et de leur nature (passager d'un vol ou visiteur). Les personnes prenant un avion sont contraintes

par l'heure de départ du vol qui leur est attribué aléatoirement. Les visiteurs sont eux répartis aléatoirement dans les halls et restent dans le terminal jusqu'à ce que leur temps de visite soit écoulé. Dans l'attente de l'heure de sortie du terminal, les comportements possibles pour les personnes dans le système sont les suivants :

- Attente d'un vol dans un hall ;
- Attente d'une navette dans un hall ;
- Attente dans un hall.

Le cycle vie des personnes au sein du terminal est illustré par le diagramme d'activité de la figure 6.3. Dans ce schéma, nous constatons que les personnes entrantes arrivent d'un vol ou par les portes des halls et qu'elles sortent de la même manière lorsque leur vie dans ce système est arrivée à terme ($Temps.Fini = vrai$). Notons que le symbole ! utilisé dans ce schéma représente la négation dans la logique booléenne. Au sein du terminal, les personnes évoluent en fonction de leur hall d'arrivée (H_i) et de leur hall de destination (qui fixe les objectifs de mobilité). Lorsque la destination H_j n'est pas le hall courant H_i, les personnes prennent alors la navette pour se déplacer.

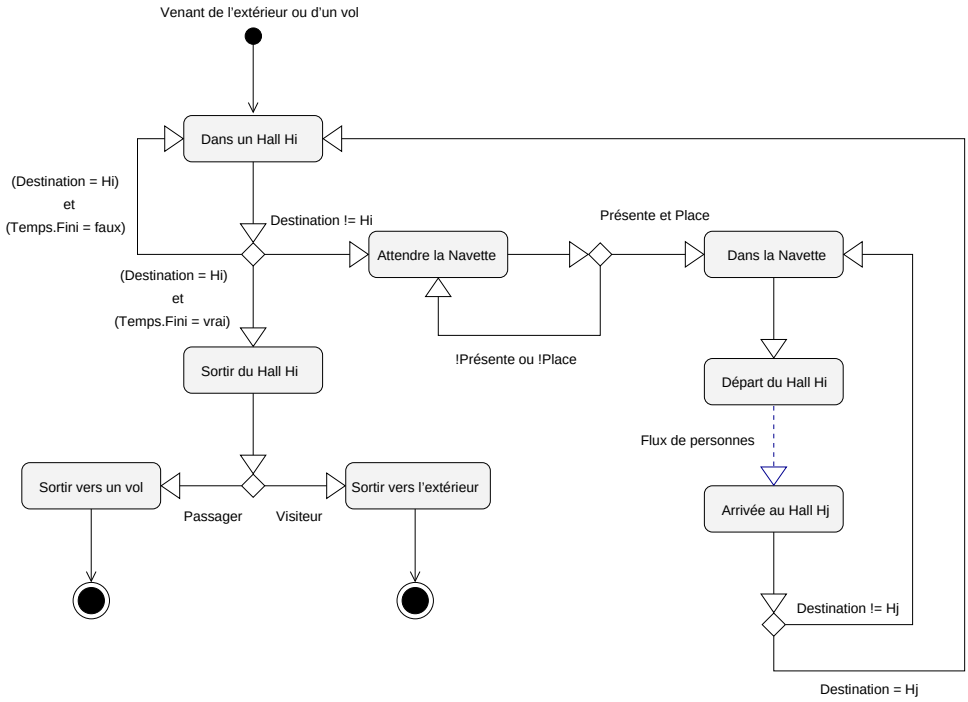


Fig. *6.3* — Diagramme d'activités des personnes

Chaque hall assure la gestion des vols qui lui sont associés en simulant leur arrivée et leur départ. Ces actions conduisent à la génération ou à la suppression de personnes dans ce hall. Les planifications des vols sont paramétrables pour chaque hall, leurs horaires permettent de faire apparaître certains comportements modélisables (e.g. vol manqué). La distribution temporelle des vols est initialisée par hall sur une base de 24 heures définissant ainsi la granularité d'une simulation. Le temps simulé dans ce modèle couvre ainsi une durée de 24 heures que nous définissons par la période minuit-minuit.

6.1.5 Modélisation des graphes

Le graphe logique définissant la structure du terminal et utilisé lors des simulations est défini par quatre nœuds qui représentent les quatre halls principaux (halls A, B, C et D) du terminal CDG 2. Les arcs représentent le chemin suivi par la navette. Celui-ci est donné par le parcours cyclique hall B, hall D, hall C, hall A et hall B tel qu'illustré en figure 6.4 (notons que la ligne principale à Roissy intègre également le terminal 1 et le hall 2F dans le trajet).

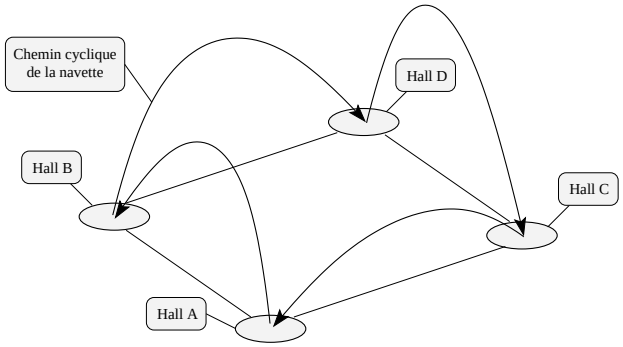


Fig. *6.4* — Chemin entre les halls

L'avantage majeur ayant guidé notre choix pour ce système à larges flux de données désagrégées est que le graphe logique représentant le terminal et le chemin réalisé par la navette principale du terminal peut être reproduit par un homomorphisme complet intégrant une bijection des nœuds et des flux agrégés. Dans ce schéma, le graphe physique est défini tel que chaque ordinateur du système distribué représente un hall (mis en œuvre par un agent). Le protocole RTB par sa métaphore de bus reproduit le comportement et les mouvements naturels de la navette et organise la migration (par groupe) des personnes en fonction des contraintes de transport liées au terminal (heures d'arrivée et de départ des avions, capacité de la navette, temps de déplacement entre les halls).

6.2 Mise en œuvre du modèle de terminal

La mise en œuvre du système aéroportuaire simulant le fonctionnement des mobilités de personnes entre les halls du terminal 2 de CDG repose sur l'application des principes présentés au cours de ce document de thèse. La conception de cette application est caractérisée par une simulation distribuée à événements discrets dans laquelle les événements sont produits par les halls et les éléments qu'ils intègrent (personnes, navette et vols) et, par la plate-forme Atlas. La modélisation du terminal par méthodologie exploite une décomposition du modèle basée sur une approche qui structure l'espace en fonction des halls. La plate-forme de simulation Atlas gère le synchronisme et la nature dirigée par le temps de la simulation selon principes et techniques présentées dans le chapitre 5.

6.2.1 Simulation distribuée dirigée par le temps

Une simulation informatique du modèle de terminal consiste en l'exécution d'instances de la plate-forme Atlas et de l'application aéroport sur chacun des ordinateurs du système distribué présenté en section 5.1. Dans cette architecture multiprocesseur, la machine contrôleur

héberge le gestionnaire de la simulation dont l'objectif est de gérer l'interface graphique qui traduit visuellement l'évolution du système simulé. Cet ordinateur exécute ainsi, par une décomposition des fonctions du simulateur (cf. section 3.2.3), une instance de la plate-forme Atlas qui soutient et permet l'exécution de la partie graphique de l'application distribuée aéroport. Chacune des autres machines du système distribué sert exclusivement à la simulation du modèle de terminal. Dans cette décomposition du modèle chaque ordinateur exécute une instance numérotée de Atlas qui offre un support pour l'exécution d'un hall.

La simulation dirigée par le temps est discrétisée par intervalles de temps unitaires et entretient une horloge globale représentant le temps simulé (cf. section 1.3.3). Cette horloge fournie par Atlas avance par incréments fixes qui sont de taille arbitraire mais constante. Dans cette application, à chaque pas de l'horloge, les composants halls consultent leur liste locale d'événements à la recherche d'événements ayant pour date d'occurrence l'heure courante (ou inférieure) de simulation T_{sim}. S'il possède un événement à simuler à cette période, le hall exécute l'action correspondante. Remarquons que pendant ce temps les divers processus d'Atlas exécutent également leurs activités de gestion et en particulier la plate-forme poursuit l'incrémentation des pas temporels. La granularité temporelle, c'est-à-dire le choix du pas de la simulation est dans cette application d'une minute (cf. section 5.6).

La navette circule avec une contrainte de temps, de telle sorte que son voyage d'un hall vers le suivant requiert un temps non nul pour effectuer le déplacement. Le temps de transit entre deux halls est fixé, dans ce modèle, à 5 minutes dans le temps simulé. Le parcours cyclique du terminal est ainsi réalisé en 20 minutes. L'implémentation de l'application a été réalisée avec une version de la plate-forme qui intègre la version RTB du module de migration (cf. section 5.4). Le bus mis en œuvre dans ce protocole reproduit le chemin de la navette défini dans le modèle. Le bus intègre une impédance pour chaque arc reliant deux nœuds qui permet de simuler une attente de 5 minutes avant la livraison du bus à son destinataire. Les passagers et les visiteurs sont les objets Java gérés par Atlas (module MAN) et dont les migrations à l'aide du bus sont déclenchées à un niveau de décision désagrégé (i.e. individuel) par l'agent hall.

6.2.2 Interface de simulation

L'interface de l'application aéroport illustrée par la figure 6.5 repose sur les fonctionnalités graphiques offertes par la plate-forme Atlas. La définition pour cette application de certains affichages complémentaires accédés depuis les greffons du centre de contrôle de Atlas (cf. figure 5.12) informe le modélisateur des événements produits par la plate-forme et par l'application aéroport et, affiche des éléments permettant le suivi et la compréhension des états successifs de la simulation.

Lors de l'exécution d'une simulation, chaque changement d'état local est transmis au contrôleur lui permettant ainsi de reporter en temps réel sur cette interface graphique les états successifs de la simulation. L'interface intègre plusieurs composants qui délivrent des informations spécifiques sur l'état de la simulation (e.g. arrivée de personnes, déplacements de la navette, départ d'avions). Le panneau central de l'application affiche une représentation graphique du terminal 2 de CDG et permet un suivi graphique des déplacements et des positions successives de la navette entre les halls du terminal (figure 6.5). D'autres panneaux complémentaires affichent l'état de la navette, l'état et les statistiques des halls ou encore le temps virtuel selon un format jour, heure, minute. Ces éléments permettent, en fonction des différentes configurations de la simulation, de rendre compte de l'évolution du terminal, des propriétés émergentes au niveau global et de tout goulot d'étranglement potentiel.

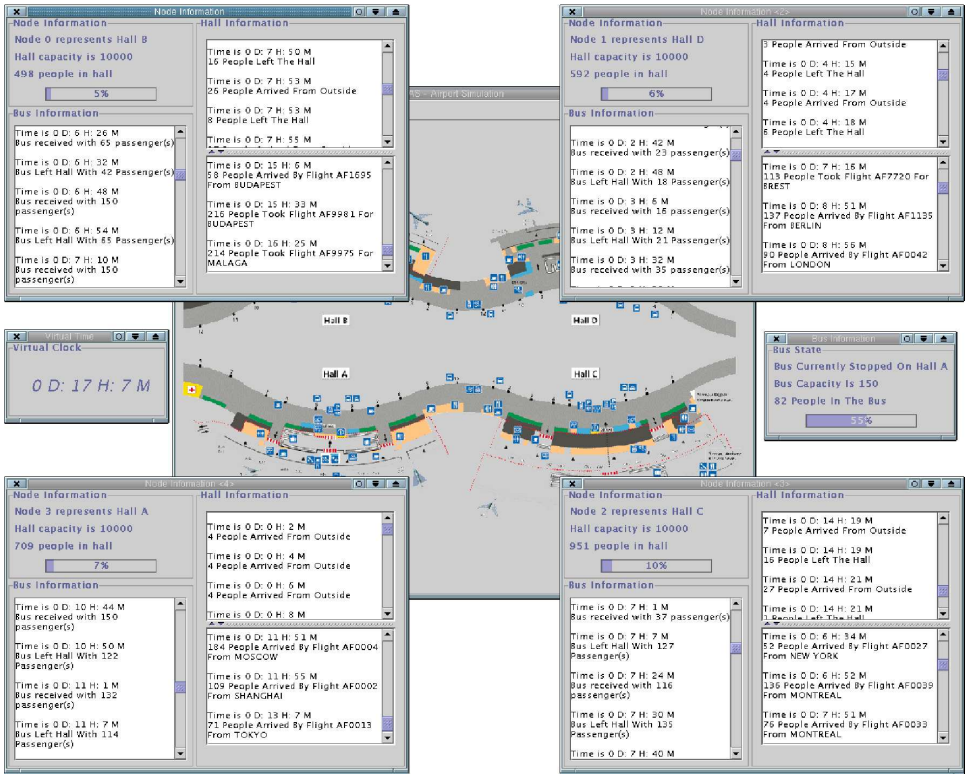


Fig. *6.5* — Interface de l'application aéroport

La figure 6.6 présente l'interface spécifique à chaque hall. Celle-ci affiche les paramètres relatifs au hall qu'elle représente (numéro de nœud physique, nom du nœud logique) et la charge du nœud en termes de nombre de personnes, les mouvements des personnes (entrées, sorties, déplacement par la navette), les événements relatifs à la navette (i.e. bus RTB) et les arrivées et les départs des avions.

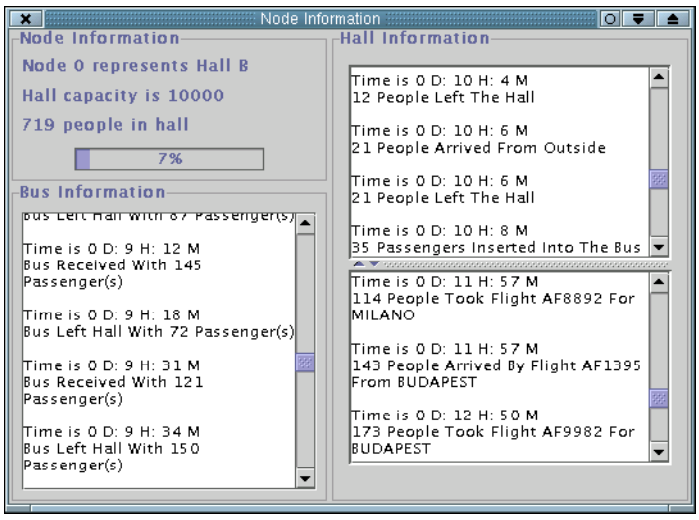


Fig. *6.6* — Événements produits par un hall

Pour un nœud donné du système distribué, la boite d'information le décrivant est initialisée dès l'enregistrement de celui-ci auprès du contrôleur. Tous les événements qui se produisent ensuite sur un nœud sont estampillés avec l'heure locale et sont envoyés au contrôleur qui réalise l'affichage dans la boite correspondante.

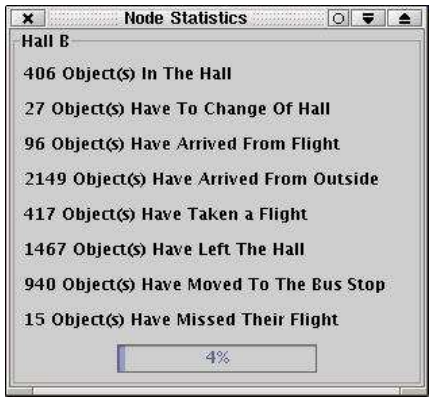


Fig. *6.7* — État de la simulation

La figure 6.7 présente le panneau graphique de l'application aéroport dans lequel sont affichées les informations relatives au fonctionnement d'un hall. Ce composant combine les informations collectées par l'application elle-même avec celles fournies par le module *LOG* de la plate-forme. C'est le cas par exemple du nombre de personnes ayant déjà pris la navette (940) ou encore du nombre de personnes devant se déplacer (27) et qui correspond au nombre d'objets ayant l'état *migrate* dans le module *MAN* (i.e. taille de la file d'attente de migration).

6.3 Simulation et résultats

Afin d'analyser le comportement de l'application distribuée de l'aéroport, différentes configurations ont été testées et analysées. Mesurer et évaluer le comportement des halls et la façon dont ce terminal réagit dans son ensemble impose l'identification et l'implémentation de compteurs et d'indices d'observation. Ces indicateurs devraient permettre de mettre en relief les dysfonctionnements et les goulots d'étranglement potentiels du système. Dans les simulations développées, nous évaluons plusieurs configurations possibles de ce terminal en choisissant une configuration identique ou non des halls, des flux de transport, et du nombre de halls. En postulant qu'un terminal fonctionnant correctement minimise le nombre de vols manqués, un indicateur pertinent en première approximation du comportement des halls est donné par l'estimation du nombre/pourcentage de personnes ayant manqué leur vol dans chacun des halls.

Dans ces simulations, les arrivées, les départs et les flux de personnes ainsi que les horaires d'avions sont dérivés de configurations plausibles (et réduites) du terminal 2 de CDG. Pour chaque nœud exécutant Atlas, les paramètres initiaux sont entre autre l'identité de la machine, le nombre total de nœuds, la configuration du graphe représentant l'application ou encore la taille initiale de chaque mémoire locale. Bien que la plate-forme de simulation Atlas soit extensible à un nombre variable de nœuds, l'application aéroport est configurée pour prendre en compte de deux à quatre halls reproduisant ainsi la structure principale du terminal 2 de l'aéroport CDG. L'application aéroport est également paramétrée à l'aide de variables communes ou propres à chaque scénario. Les paramètres constants de ces simulations sont

la capacité des halls (10 000 personnes), le nombre initial de personnes dans chaque hall (50 personnes), le nombre de vols arrivant (16 vols) et la capacité d'un avion (200 passagers). Le nombre de vols en partance est variable (entre 5 et 24 vols) et peut être défini de façon homogène ou non sur les halls en fonction des objectifs de chaque simulation. La capacité de la navette (et par conséquent du bus RTB définie au niveau de la plate-forme) est fixée à 150 personnes (excepté le scénario 2). L'événement générant l'arrivée de nouvelles personnes (un nombre aléatoire qui est fonction de l'heure) dans un hall se produit toutes les cinq minutes dans le temps simulé. Ces simulations couvrent une période journalière de 24 heures qui correspond au niveau de granularité temporelle requis pour l'analyse du comportement d'un terminal d'aéroport car son cycle de vie obéit à une rotation de 24 heures (par exemple sur les vols). Ces simulations sont réalisées en 24 minutes de temps réel ce qui représente une accélération par 60 entre le temps physique et le temps simulé.

Notons que les simulations suivantes présentées à des fins illustratives sont calibrées en utilisant essentiellement une variation des vols en partance. En effet, les vols arrivant ne donnent que des générations supplémentaires de personnes à un instant donné et n'influent pas sur les contraintes de déplacement des personnes qui sont fixées par leur heure de départ.

6.3.1 Scénario 1

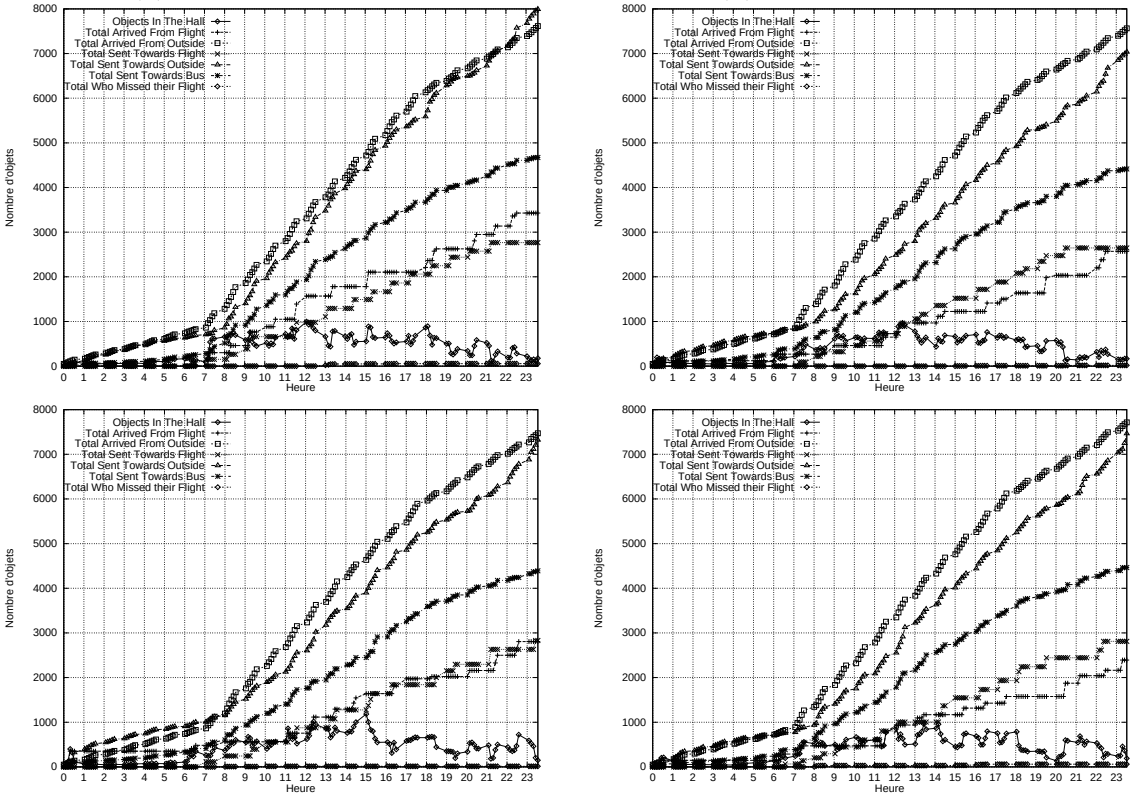
Le premier scénario correspond à une simulation paramétrée avec quatre halls et dans laquelle la configuration des halls est homogène avec une répartition égale de 18 vols en partance chacun. La figure 6.8 représente les évolutions de ces quatre halls sur une période de 24 heures. Cette figure contient deux types d'indicateurs : un compteur instantané qui présente le nombre de personnes dans un hall en fonction du temps et des compteurs cumulés qui donnent des valeurs agrégées depuis le démarrage de la simulation. Parmi ces compteurs, un indicateur de performance présente le nombre de personnes ayant manqué leur avion dans un hall.

Au regard des courbes de la figure 6.8, nous constatons qu'une configuration homogène des halls conduit à des comportements relativement similaires de l'ensemble sur une même journée. Les nombres de personnes entrantes et sortantes des halls respectivement de et vers l'extérieur sont relativement équilibrés et oscillent entre 7000 et 8000 en fin de journée. Similairement, les nombres de passagers arrivés et partis par un vol suivent la même tendance (2 500 à 3 500). La valeur cumulée des personnes arrivées par les portes des halls et par avion est, dans cet exemple, de 41567. Ceci signifie qu'au cours de cette simulation qui a duré 24 minutes, 41567 objets Java ont (1) été créés, (2) ont effectués, en fonction de la personne représentée et de ses objectifs un cycle de vie et, (3) ont été détruits. L'accroissement du nombre moyen des arrivées défini dans notre modèle entre 7 heures et 18 heures implique un accroissement du nombre de personnes dans chacun des halls durant cette période. Constatons par ailleurs que le nombre de personnes prenant la navette suit la même tendance que les arrivées dans le hall car très logiquement plus il y a de personnes dans un hall et plus il y a de personnes souhaitant se déplacer vers un autre hall.

La figure 6.9 montre une analyse comparative de la performance des halls. Les courbes donnent, pour chaque hall, l'évolution du pourcentage de personnes ayant manqué leur avion par rapport au nombre total de personnes qui auraient dû prendre un avion dans ce hall. Notons que la performance globale de cette configuration atteint un équilibre relatif dès la fin de l'après-midi. Le pourcentage des personnes ayant manqué leur avion oscille, en fin de journée, entre 0,5% et 2% en fonction du hall.

(a) évolution du hall B

(b) évolution du hall D



(c) évolution du hall C

(d) évolution du hall A

Fig. *6.8* — Scénario 1 : évolution des halls

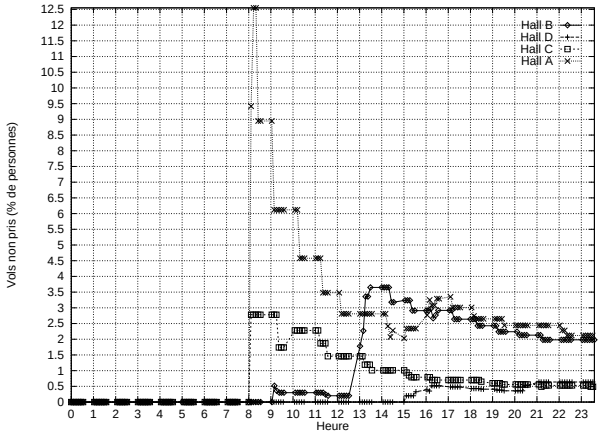


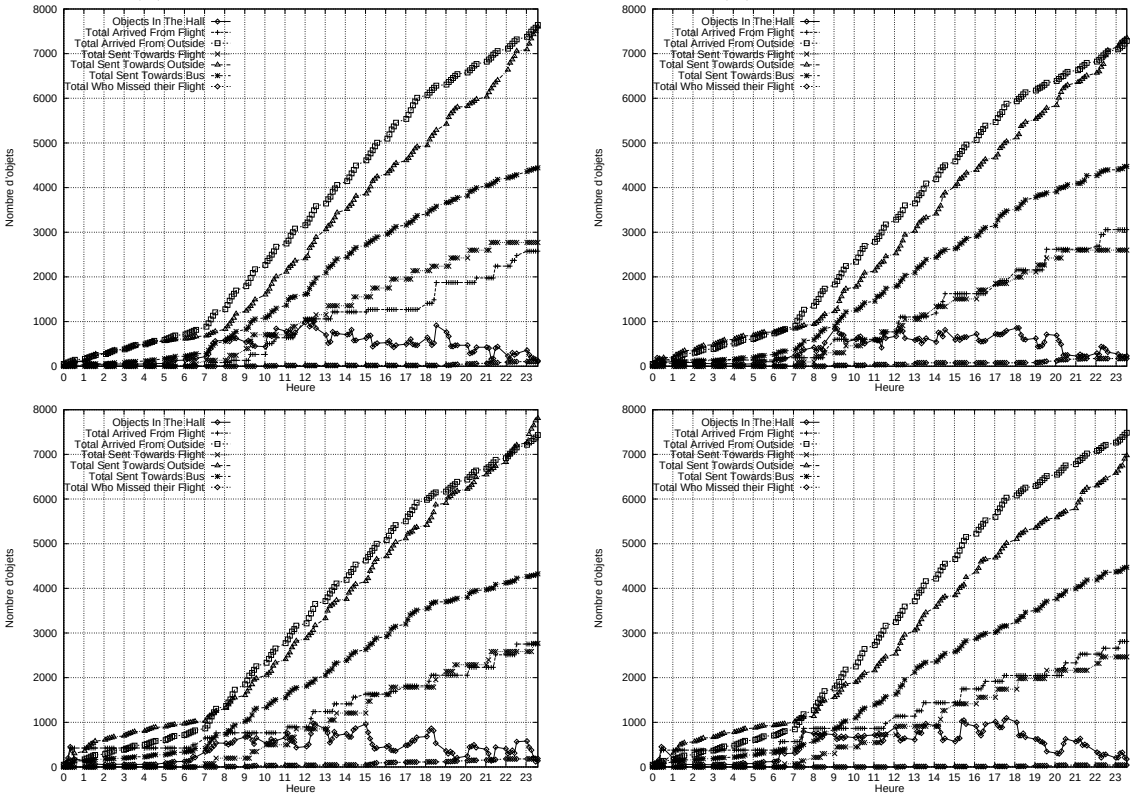
Fig. *6.9* — Scénario 1 : performance des halls

Le pic observé dans le comportement du hall A (figure 6.9) peut être expliqué par le fait qu'à 8 heures peu de passagers ont pris un avion dans ce hall (un seul vol à 7h30). En conséquence un faible nombre de personnes ayant manqué leur avion influe grandement sur l'indicateur de performance. Cet état est renforcé par le fait que cette période est relativement chargée en termes de transport causant un goulot d'étranglement dans les déplacements des passagers entre les halls entraînant ainsi des difficultés dans la réussite des connexions.

6.3.2 Scénario 2

Le second scénario présenté dans cette section reproduit les propriétés et les paramétrages du scénario précédent, mais avec une capacité de transport moindre. La figure 6.10 représente l'évolution des halls dans cette simulation à quatre nœuds où la répartition des vols en partance est équilibrée avec 18 départs d'avions pour chaque hall. La capacité maximale de la navette est dans ce scénario limitée à 75 personnes. Cette simulation illustre une des propriétés supposées du terminal : lorsque les flux de personnes sont volumineux, alors la capacité de la navette à un impact non négligeable sur les réussites de départs par avion.

(a) évolution du hall B (b) évolution du hall D



(c) évolution du hall C (d) évolution du hall A

Fig. *6.10* — Scénario 2 : évolution des halls

La figure 6.11 donne l'évolution du pourcentage des personnes qui ont manqué leur vol par rapport au nombre total de personnes qui devaient prendre un vol dans chacun des halls. Les manquements de vol sont dans cette simulation plus élevés, en comparaison avec le premier scénario, induisant une performance globale plus faible. En fin de journée, les halls ont un pourcentage d'échec dans la réussite des connexions qui oscille entre 1,5% et 6,5%. Le nombre

cumulé de personnes ayant manqué leur avion est dans ce scénario de 507 alors que le scénario 1 ne comptait que 148 échecs (augmentation de 342% pour diminution de 50% de la capacité de la navette).

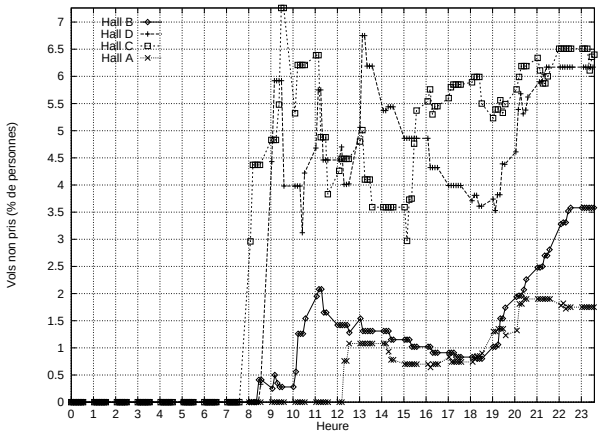


Fig. *6.11* — Scénario 2 : performance des halls

6.3.3 Scénario 3

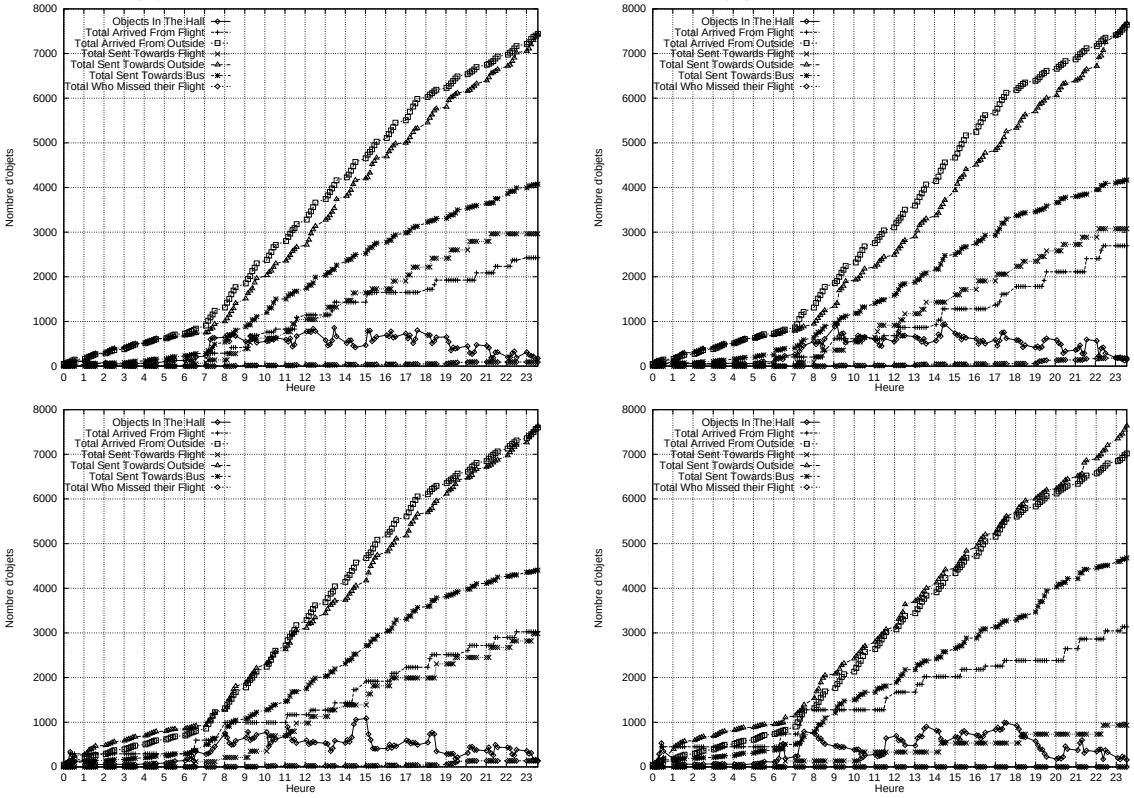
Un troisième scénario présente une dernière simulation à quatre halls mais dans laquelle le comportement des halls est hétérogène (figure 6.12). Dans ce scénario, la configuration au niveau du terminal est similaire à la première simulation mais au niveau des halls une dissymétrie est introduite. Les halls B, D et C sont configurés avec 22 vols en partance et le hall A avec 5 vols seulement. Comme la génération de personnes et le nombre de vols arrivant sont identiques dans chaque hall, les demandes de déplacement depuis le hall A vers les autres halls augmentent par rapport à la première simulation. Ce comportement, au niveau du terminal, a pour résultat d'augmenter les demandes de transport à partir de ce hall vers les autres halls, entraînant des difficultés dans les correspondances vers les vols en partance des autres halls.

La distribution non équilibrée des vols a un impact triple dans le hall A : (1) les résultats montrent que le nombre de personnes envoyées par la navette vers le hall A est plus grand que le nombre moyen des autres halls (111%), (2) le nombre de déplacement de personnes vers l'extérieur à partir de ce hall est plus grand que le nombre d'arrivées (109%), (3) le taux de remplissage moyen des vols dans le hall A est plus fort que dans les autres halls. En effet, le nombre de départs par avion dans le hall A est égal à 31% du nombre moyen de départs dans les autres halls, alors que le nombre de vols en partance dans le hall A est égal à 23% du nombre de vols des autres halls (5 contre 22).

La figure 6.13 montre l'évolution du pourcentage de personnes qui ont manqué leur avion par rapport au nombre total de personnes qui devaient prendre un avion. L'hétérogénéité de cette configuration conduit à des performances variables selon les halls et indique un mauvais fonctionnement au niveau du terminal. Les performances de ces halls se dégradent tôt dans la journée et contrairement au premier scénario ne tendent pas à s'améliorer avec le temps. Notons enfin que contrairement aux autres halls, le hall A se comporte relativement bien dans la mesure où aucun passager n'a manqué son vol dans ce hall.

(a) évolution du hall B

(b) évolution du hall D



(c) évolution du hall C

(d) évolution du hall A

Fig. *6.12* — Scénario 3 : évolution des halls

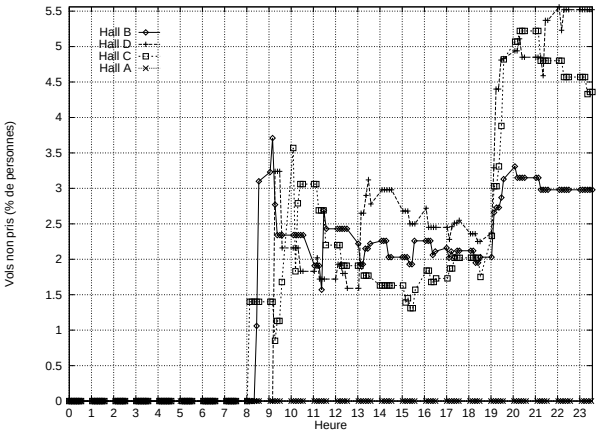


Fig. *6.13* — Scénario 3 : performance des halls

6.3.4 Scénario 4

La simulation suivante reproduit pratiquement les conditions initiales de la première simulation à quatre halls mais cette fois avec un terminal à trois halls. Cette simulation présente l'avantage de pouvoir évaluer comment le terminal réagit si un des halls devenait indisponible. Cette simulation est initialisée avec une répartition homogène des vols qui sont au nombre de 24 dans chaque hall. Tout comme les vols, et afin de conserver un comportement du terminal comparable aux simulations précédentes, le nombre total de personnes sur une journée est réparti sur les trois halls. L'effet résultant est une augmentation des mouvements dans chaque hall et entre les halls. Cet accroissement du nombre de personnes dans les halls a un impact négatif sur les flux de transport entre les halls au vu de l'augmentation des demandes de déplacement. Ce phénomène entraîne un accroissement des temps de déplacement et par conséquent du nombre de personnes qui manquent leur vol.

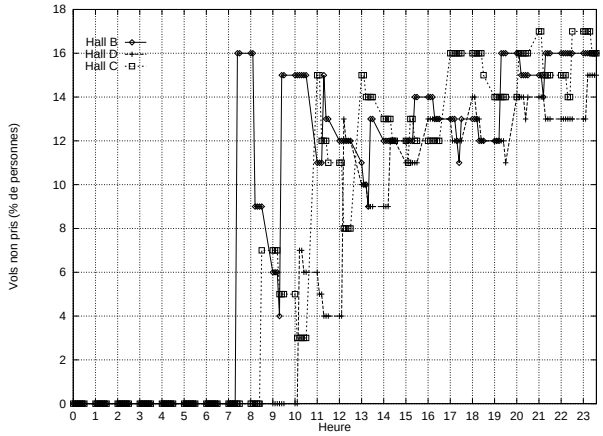


Fig. *6.14* — Scénario 4 : performance des halls

La figure 6.14 illustre le comportement peu performant des halls dans cette configuration en présentant l'évolution du pourcentage de personnes qui n'ont pas réussi à prendre leur avion. Ce pourcentage augmente fortement dès 7h30 et tout au long de la journée pour atteindre des valeurs comprises entre 15% et 20% à minuit. Ce scénario montre notamment que, à charge égale, la réduction du nombre de halls dans un terminal doit s'accompagner de l'augmentation des possibilités de transport afin de conserver un comportement global acceptable.

6.3.5 Synthèse des scénarios

La matrice de graphes présentée par la figure 6.15 illustre la distribution des passagers, des flux de transport et les performances des différentes simulations à différents instants de la journée. Chaque graphique présente pour chaque hall le pourcentage des passagers ayant pris/manqué leur avion ainsi que le volume des départs. Les flux cumulés des personnes sortant des halls sont également affichés avec des valeurs relatives pour chaque instant représenté. Les différents scénarios présentés évoluent en fonction du temps et convergent vers un état d'équilibre en fin de journée. Cette figure souligne l'homogénéité du scénario 1 et le bon comportement du terminal. Le dernier état du troisième scénario démontre comment une hétérogénéité des halls affecte les performances du terminal. Remarquons enfin dans le scénario 4 l'impact de l'accroissement de la charge des halls sur le fonctionnement du terminal

qui converge vers un état homogène au niveau des halls mais peu performant. Notons que les états des trois simulations à 8 heures, lorsque le nombre de personnes ayant pris l'avion est faible, sont relativement similaires. Cela laisse apparaître une certaine reproductibilité dans ce système non déterministe. Ces graphiques font partie des outils fournis à partir de la plate-forme Atlas. Ils sont générés à la demande pour tout instant d'une simulation. Ils produisent une interface intuitive pour l'analyse des performances des halls.

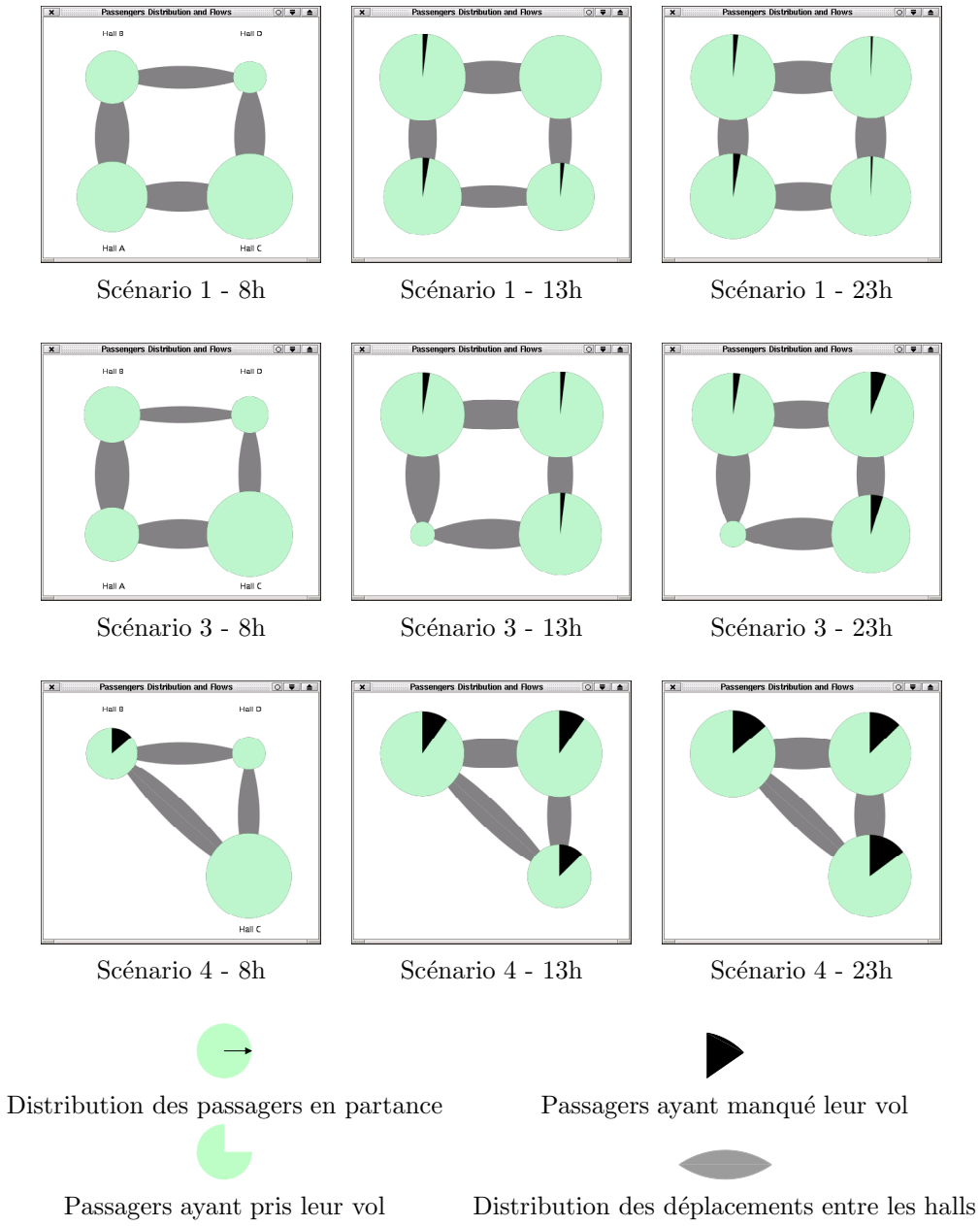


Fig. *6.15* — Simulation de la distribution des passagers et des flux de personnes entre les halls du terminal de simulation

La figure 6.16 donne une comparaison des comportements des trois simulations à quatre halls présentées dans ce chapitre. Ces graphiques affichent le nombre instantané de personnes dans chaque hall en fonction du temps de simulation. Cette illustration montre un comportement global qui est à la fois similaire et pourtant indéterminisme et, qui est fonction variations

dans les configurations de chaque hall. Le nombre initial de personnes au démarrage de la simulation était fixé à cinquante. La croissance qui peut exister après minuit est due à l'arrivée des derniers avions de la journée précédente. Durant la nuit le nombre de personnes reste stable puis croit fortement dès 7 heures et ce jusqu'à 18 heures. L'activité dans les halls diminue ensuite pour se rapprocher du nombre initial de personnes en fin de journée. Ceci délivre un comportement intéressant pour l'analyse et la planification des équipements commerciaux et administratifs au niveau de halls et du terminal.

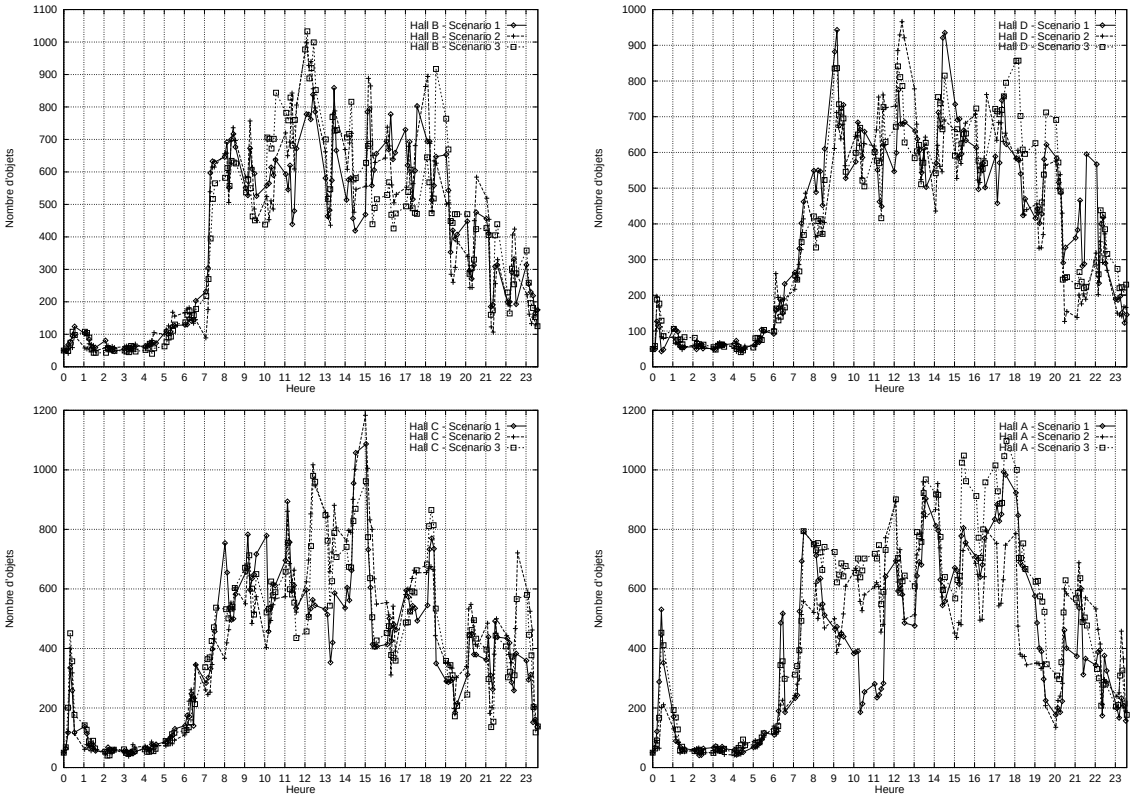


Fig. *6.16* — Comparaison entre les simulations à quatre halls

Nous constatons, parmi les quelques inconvénients de cette application, le manque de finesse dans le modèle du terminal qui limite l'émergence de nouvelles propriétés à un niveau global. Toutefois, l'émergence de comportements par ailleurs identifiés (ou attendus) tels que des goulots d'étranglement ou de manière générale des saturations apparaît effectivement au cours des simulations. L'étude des vols manqués illustre les problèmes induits par cette propriété de saturation qui se produit de manière éparse et non équitable au cours d'une journée simulée et en fonction de la configuration des halls. La détection dynamique d'un tel phénomène (attendu) pourrait rétroagir sur la mise en place au niveau du terminal d'une navette plus grande (éventuellement d'une deuxième) ou plus petite (i.e. changement dynamique de la capacité du bus RTB).

6.3.6 Évaluation des performances informatiques

Une comparaison entre une solution mono-machine classique et une solution distribuée qui utilise plusieurs machines montre l'intérêt d'une telle approche pour la simulation de larges flux de données désagrégées. En effet, l'exécution de simulations sur un unique ordinateur

conduit rapidement à la saturation de la machine en termes de mémoire et de puissance de calcul [Ray et Claramunt, 2002b]. L'avantage d'une solution distribuée est renforcé lors d'exécutions où le temps de simulation est fortement accéléré, d'une extension du nombre de nœuds et, lors de l'augmentation des flux de transport.

Les simulations effectuées sur un seul ordinateur (la machine contrôleur) ont été réalisées en utilisant les paramètres initiaux similaires aux scénarios précédents. Dans la mesure où il n'est pas concevable de réaliser une simulation séquentielle avec Atlas, puisqu'il s'agit d'une plate-forme distribuée et que cela impliquerait une complète réécriture du code, nous avons, dans le but d'étudier et d'approximer la simulation séquentielle, réalisé un endomorphisme simple qui consiste à exécuter tous les nœuds (halls) de la simulation sur une seule machine. Les flux de personnes ne sont pas, d'un point de vue informatique, optimisés et utilisent le bus RTB à l'identique de la version distribuée. Toutefois, dans cette simulation pseudo-séquentielle, le bus circule en réalisant des communications réseaux qui sont intra-machine. Les résultats montrent que la simulation de l'application aéroport utilise, tout au long de l'exécution, le maximum des ressources disponibles du processeur. Pour une simulation à deux halls, la machine contrôleur doit exécuter ces deux nœuds plus le gestionnaire de simulation (partie graphique). La charge processeur pour la partie graphique est environ de 10% et les deux halls utilisent chacun environ 40% des ressources (ces charges sont relatives à la machine utilisée et notons que 10% du processeur est alloué au système d'exploitation). Dans une simulation à trois nœuds, l'allocation du processeur pour la fonction de gestionnaire est toujours la même, mais les 80% des capacités du processeur précédemment allouées à deux halls, sont maintenant réparties pour le traitement informatique de trois halls. Ainsi chaque hall n'a plus que 27% du processeur à sa disposition. Cette utilisation permanente de la ressource processeur est due en partie à la mise en œuvre des halls qui à chaque pas temporel de la simulation (dirigée par le temps) consulte la liste des événements à exécuter (e.g. départs et arrivées d'avions, génération ou suppression de personnes) de manière répétitive[1]. Ainsi lorsque les nombres de halls, de personnes, de vols augmentent, la solution mono-machine tente de gérer chaque hall au mieux mais ne dispose plus des capacités de traitement et de stockage nécessaires. L'utilisation d'un seul ordinateur pour simuler un système tel que celui du terminal 2 conduit à une surcharge informatique. Ces évaluations sur ce cas particulier montrent que la quantité de mémoire et les besoins en capacité de traitement d'une solution mono-machine n'est pas adaptée pour la simulation de systèmes à larges flux de données désagrégées.

Un avantage important de la simulation est la capacité de définir une période physique qui accélère le temps de simulation selon les besoins d'utilisateur. La surcharge de calcul empire par un calibrage dans lequel le temps simulé est fortement accéléré car dans ce cas la simulation requiert (exige) plus de capacités de traitement. Afin de valider cette contrainte, des essais ont été réalisés en utilisant un temps virtuel plus compressé. Dans les tests précédents, un jour pour l'aéroport était simulé en 24 minutes de temps physique. Dans des conditions de simulation identiques à l'étude précédente nous avons conduit des essais supplémentaires lorsque le temps simulé est accéléré à 2 minutes et 24 secondes (dix fois plus rapidement que les scénarios précédents). Cette compression temporelle est celle maximum testée et non une limite. Elle définit un facteur d'accélération (le Real Time Ratio défini par Nagel dans [Simon et Nagel, 1998]) de 600, c'est-à-dire que la simulation est exécutée 600 fois plus rapidement que le temps ne s'écoule pour le concepteur. La quantité de traitement exigée, lors de

[1]La mise en œuvre récente d'un mécanisme de moniteur dans les termes de la définition de Hoare permet de verrouiller/activer tous les processus de la simulation, limitant ainsi le phénomène de pleine charge induit par l'attente active.

la simulation de deux halls par une solution mono-processeur (utilisation du contrôleur seul), est trop haute à cette vitesse de simulation et l'ordinateur n'est plus en mesure d'effectuer une simulation cohérente. Par exemple un fort pourcentage des arrivées bus RTB ne respecte plus les délais de livraison du modèle, et des échéances de la liste des événements ne sont plus respectées. A contrario, une simulation identique (au caractère indéterministe près) avec les quatre halls exécutée sur l'ensemble du système distribué (les 5 cinq machines) fonctionne sans défaillance. Chaque hall requérant, à cette vitesse d'exécution, environ 60% des capacités de calcul de nos machines, son affectation exclusive sur un nœud de l'architecture lui permet de disposer des ressources suffisantes afin d'être simulé correctement : la gestion de la simulation n'induit pas, contrairement à l'approche séquentielle, d'effets de bord sur le comportement du système logique.

6.4 Conclusion

Ce chapitre décrit une utilisation de notre approche de simulation distribuée orientée vers la modélisation et la simulation de systèmes à larges flux de données désagrégées. Notre développement comporte une modélisation et une mise en œuvre qui repose sur notre plateforme de simulation Atlas qui reproduit les propriétés statiques et dynamiques d'un système représenté par un graphe. La section 6.1 présente les spécifications et la modélisation d'un terminal d'aéroport dans lequel des personnes se déplacent en utilisant un service de transport par navette qui circule entre des halls. La section 6.2 présente quelques aspects de la mise en œuvre de l'aéroport Charles De Gaulle (CDG) et la section 6.3 présente quelques-unes de simulations réalisées.

Cette étude basée sur l'application aéroport CDG montre que le système Atlas fournit un environnement interactif de simulation permettant l'analyse de différentes configurations et contraintes des halls d'un terminal d'aéroport. Ces simulations montrent le potentiel et la flexibilité du système Atlas pour évaluer les performances de différentes configurations d'un terminal d'aéroport modélisé comme un graphe sous les contraintes de flux de données désagrégées. L'application distribuée aéroport utilise les propriétés de modélisation induites par le système Atlas en particulier le protocole RTB qui permet la migration physique des objets représentant des entités du système simulé.

Bien que le modèle appliqué à la simulation du terminal d'aéroport proposée soit éloigné de la complexité réelle d'un tel système, notre développement offre une application démonstrative de référence pour le développement de futures applications. Ces simulations montrent par ailleurs que notre approche de modélisation et de simulation dans laquelle les propriétés du système logique sont reproduites à un niveau physique est applicable aux systèmes désagrégés et qu'elle permet l'analyse et l'évaluation de différentes configurations.

Nous remarquons par les quelques résultats obtenus sur les performances informatiques que la simulation séquentielle de systèmes à larges flux de données désagrégées atteint rapidement ses limites et que l'approche de simulation distribuée offre des capacités de simulations et des vitesses d'exécution bien supérieures. La constatation de la simplicité du modèle mis en œuvre et présenté dans ce chapitre en termes de ressources requises nous conforte dans l'idée que la simulation distribuée est une voie à approfondir pour la simulation des systèmes considérés dans ce document.

Conclusion Générale

« La théorie, c'est quand on sait tout et que rien ne fonctionne. La pratique, c'est quand tout fonctionne et que personne ne sait pourquoi. Ici, nous avons réuni théorie et pratique : Rien ne fonctionne... et personne ne sait pourquoi ! Albert Einstein, Nobel de physique 1921 »

Cette thèse introduit l'utilisation des systèmes distribués pour modélisation et la simulation de systèmes complexes à larges flux de données désagrégées. De nombreuses applications incluant les systèmes dits à larges flux de données ont une structure distribuée inhérente ou décentralisée dans l'action qui peut être reproduite en ayant recourt à la simulation distribuée. Pour ce faire nous avons proposé une méthodologie qui, par pas successifs, guide la démarche de modélisation vers la simulation distribuée. Cette approche combine les concepts de hiérarchie structurelle, d'utilisation de graphes et de gestion groupée des flux pour représenter un système complexe. La dernière étape de cette méthodologie consiste à utiliser un support de simulation mettant en œuvre l'exécution répartie. Notre implémentation d'un tel environnement est illustrée par la conception de la plate-forme Atlas. Ce logiciel, support de simulation distribuée pour les systèmes à larges flux de données désagrégées, fournit un lien entre les modèles désagrégés et leurs implémentations vers une architecture multiprocesseurs qui surpasse les solutions mono-machine.

Méthodologie de conception Atlas

La méthodologie proposée comporte une démarche de modélisation et une plate-forme de simulation pour l'exécution répartie et l'analyse de systèmes complexes à larges flux de données désagrégées. Le choix méthodologique proposé est influencé à la fois par le système à simuler et par l'architecture informatique servant de support de simulation dans le but de proposer une démarche où l'aspect technique soit en accord avec l'aspect conceptuel. La démarche de modélisation intègre une description hiérarchique représentant l'organisation interne d'un système complexe, la construction d'un graphe logique traduisant une décomposition structurelle du système puis une projection qui consiste à concevoir un graphe physique qui établit une abstraction liant le niveau de représentation conceptuel et le système distribué sous-jacent.

La méthodologie de conception présentée guide le choix de décomposition structurelle d'un système en un ensemble d'éléments structuraux, entre lesquels interviennent des flux de données et qui sont répartis sur les ordinateurs d'un système distribué. Deux aspects

incrémentaux de la démarche décrivent le choix de répartition (homomorphisme et endomorphisme). Nous avons dans cette thèse proposé une solution pour une première étape qui est l'homomorphisme. Dans cette approche, une bijection est définie entre les composants spatiaux fixes d'un système complexe et les machines du système distribué. Cette méthodologie propose une méthode de gestion des flux de données désagrégées par des déplacements agrégés qui se traduisent par des migrations d'objets entre les ordinateurs d'un système distribué.

La plate-forme Atlas, dernière étape notre méthodologie, est un support de simulation distribuée à événements discrets dirigée par le temps et dont les propriétés dynamiques reproduisent le comportement de larges flux de données désagrégées. Une des particularités de cette plate-forme réside dans la mise en œuvre du concept de flux agrégés par une migration physique groupée qui donne une solution adaptée pour réaliser et coordonner la migration de larges flux d'objets, ce qui est un aspect important pour la simulation de systèmes sociétaux et urbains et, notamment pour les systèmes en transport. Cette solution intègre par ailleurs la capacité reproduire, à un niveau physique, la nature et le comportement de certains flux logiques lorsque ceux-ci sont eux-mêmes agrégés.

D'un point de vue architectural, la plate-forme Atlas est une couche logicielle dédiée à la simulation et qui agit comme un intermédiaire entre le concepteur et une architecture distribuée. La plate-forme inclut plusieurs fonctionnalités telles que la représentation et la gestion de structures de données, la migration d'objets, la gestion des communications entre machines et la gestion du temps de simulation. La plate-forme offre par ailleurs un support graphique permettant un contrôle et un suivi des simulations. Dans le contexte d'un homomorphisme, cette plate-forme permet une étude du comportement des flux de données à un niveau global par une agrégation des comportements locaux du système. La solution Atlas offre un support générique, car conçue en réalisant un découplage entre les différents modèles spécifiques et le noyau de simulation. Cette optique a pour objectif de permettre à Atlas d'être appliquée à différents contextes applicatifs modélisables par un graphe, et où l'on souhaite analyser des flux de données désagrégées entre les nœuds de ce graphe.

Application à la simulation de systèmes désagrégés

L'utilisation de notre méthodologie et de sa plate-forme est illustrée par une application en transport qui modélise et simule les flux de personnes entre les différents halls d'un terminal d'aéroport. Notre choix pour ce système à larges flux de données désagrégées est guidé par la possibilité de reproduire, par un homomorphisme complet intégrant une bijection des nœuds et des flux agrégés, les aspects structurels, fonctionnels et la dynamique du terminal et de sa navette qui parcourt un chemin cyclique entre les halls. L'application distribuée aéroport utilise les propriétés de modélisation induites par la plate-forme Atlas, notamment la migration physique des objets pour reproduire les flux de personnes utilisant les navettes de l'aéroport.

Le comportement du système aéroportuaire modélisé par notre méthodologie est analysé en fonction de différentes configurations des halls et des flux de transport. L'analyse du système simulé, par la mise en relief de ses dysfonctionnements et des goulots d'étranglement, est basée en partie sur le postulat qui définit un terminal performant par un fonctionnement minimisant le nombre de vols manqués. Les simulations montrent le potentiel du système Atlas pour analyser et évaluer les performances de différentes contraintes et configurations d'un terminal d'aéroport. De manière plus générale, cette étude de cas donne un aperçu du potentiel d'Atlas et des systèmes distribués pour la simulation de phénomènes complexes.

En particulier, l'approche de modélisation et de simulation convient pour les systèmes en transport dans lesquels l'objectif est l'étude de flux migratoires représentant différents types de mobilités sociétales ou urbaines.

Les résultats obtenus sur les performances informatiques d'une simulation séquentielle du système aéroportuaire ont montré que la simulation mono-processeur atteint rapidement ses limites. A contrario, l'approche de simulation distribuée offre des capacités de simulations et des vitesses d'exécution bien supérieures. Cette solution permet de réduire tout compromis éventuel sur un niveau de description microscopique lors de la modélisation d'un système donné. Dans un contexte de validation des concepts de modélisation et de simulation distribuée reposant sur notre méthodologie, la modélisation du terminal est essentiellement orientée vers la description des flux et des propriétés les plus significatives. Les besoins peu prononcés de ce cas d'étude en termes de ressources informatiques requises confirme que la simulation distribuée est une voie à approfondir pour la simulation des systèmes considérés dans ce document.

Perspectives

Ces travaux de thèse établissent une passerelle entre, d'une part, les systèmes complexes et, d'autre part, les systèmes distribués. La méthodologie Atlas est une première étape dans l'aboutissement de simulation à événements discrets qualitative appliquée à la modélisation et la simulation de systèmes sociétaux et urbains. La validité d'une approche de simulation distribuée pour les systèmes complexes à larges flux de données désagrégées illustrée par notre cas d'étude, suggère la poursuite de certains concepts et développements dans cette direction.

De nombreux développements sont envisageables concernant la plate-forme Atlas. L'extension de certains services, l'amélioration du moteur de simulation, en particulier au niveau de la synchronisation temporelle et de la gestion des files d'événements sont à considérer. Les perspectives de ce système portent également sur l'intégration d'objets autonomes (agents) pour mettre en œuvre des entités désagrégées. En effet, l'extension des concepts de modélisation utilisés vers une meilleure autonomie des objets dans le système (certains objets du niveau micro pourraient être considérés comme des agents) devrait permettre de (1) favoriser des phénomènes d'auto-organisation des agents, (2) susciter des formes de collaborations entre agents, (3) accroître les capacités d'autodétermination (vie, reproduction, action des agents). Ces outils élargiraient le champ d'utilisation de la plate-forme de modélisation à de nombreuses applications notamment dans les domaines socio-économiques (trajectoires de vie et évolution de population) et biologique (groupes et espèces concurrentes).

D'un point de vue de la modélisation et de l'implémentation applicative les perspectives portent sur la mise en œuvre de simulation de systèmes plus complexes. En reprenant notre cas d'étude cela se traduirait par un système aéroportuaire à N-Halls/N-terminaux, par l'élaboration d'un modèle comportemental plus fin des personnes incluant des flux piétonniers entre les halls qui favoriseraient l'émergence de nouvelles propriétés dans le système. La souplesse de l'approche développée devrait permettre son application à d'autres contextes applicatifs dans le domaine des systèmes sociétaux et urbains notamment lorsque sont considérés des phénomènes de groupes (e.g. mouvement de foules).

L'objectif d'étude d'une méthodologie qui intégreraient un découplage entre les topologies logiques et physiques (endomorphisme) est conditionné par le temps requis pour l'étude et la mise en œuvre d'une projection par homomorphisme, première étape de notre démarche. La section 4.5.4 présente quelques éléments de compréhension concernant de fu-

turs développements de notre méthodologie en direction d'une projection par endomorphisme. Parmi les applications futures relatives à cette perspective, nous envisageons la mise en application des préceptes et techniques présentées dans ce document pour la planification de mobilités urbaines (e.g. déplacement résidence-travail et magasinage) incluant des paramètres réels et un couplage avec les systèmes d'information géographique.

Bibliographie

[Aiello, 1997] Aiello, A. (1997). *Environnement orienté objet de modélisation et de simulation à evénements discrets de systèmes complexes.* Thèse de doctorat, Université de Corse, France. 137 pages.

[Algers et al., 1997] Algers, S., Bernauer, E., Boero, M., Breheret, L., di Taranto, C., Dougherty, M., Fox, K., et Gabard, J.-F. (1997). Review of micro-simulation models. Rapport de recherche, University of Leeds, Leeds, Angleterre. 60 pages.

[Ali et Zimmer, 1997] Ali, S. et Zimmer, R. (1997). The question concerning emergence. In *Proceedings of CASYS97*, pages 22–26.

[Ali et Zimmer, 1998] Ali, S. et Zimmer, R. (1998). Emergence : A review of the foundations. *Systems Research and Information Systems*, 8(1) :1–24.

[Arnold et Gosling, 1996] Arnold, K. et Gosling, J. (1996). *The Java Programming Language.* The Java Series. Addison-Wesley.

[Artsy et al., 1986] Artsy, Y., Chang, H.-Y., et Finkel, R. (1986). Processes migrate in Charlotte. Rapport de recherche 655, University of Wisconsin, Madison, Wisconsin, USA.

[Artsy et Finkel, 1989] Artsy, Y. et Finkel, R. (1989). Designing a process migration facility : the charlotte experience. *IEEE Computer*, 22(9) :47–56.

[Ashby, 1956] Ashby, W. (1956). *An introduction to Cybernetics.* Chapman and Hall.

[Baekgaard, 1995] Baekgaard, H. (1995). Integrating micro and macro models. Rapport de recherche CP1995-010, NATSEM. 6 pages.

[Bagrodia et al., 1987] Bagrodia, R., Chandy, K., et Misra, J. (1987). A message-based approach to discrete event simulation. *IEEE Transaction on Software Engeenering*, 13(6) :654–665.

[Banks, 2000] Banks, J. (2000). Introduction to simulation. In Joines, J., Barton, R., Kang, K., et Fishwick, P., editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 9–16.

[Barak, 1989] Barak, A. (1989). The evolution of the MOSIX multicomputer UNIX system. Rapport de recherche 89-17, The Hebrew University, Jerusalem, Israël.

[Barak et Litman, 1985] Barak, A. et Litman, A. (1985). MOS : a multicomputer distributed operating system. *Software, Practice and Experience*, 15(8) :1–20.

[Bargiela, 2000] Bargiela, A. (2000). Strategic directions in simulation and modelling. Rapport de recherche, The Nottingham Trent University, Nottingham, Angleterre. présenté à U.K. Computing Research Strategy Meeting.

[Batty et al., 1999] Batty, M., Xie, Y., et Sun, Z. (1999). Modelling urban dynamics through gis-based cellular automata. *Computers Environment and Urban Systems*, 23(1) :205–233.

[Baude et al., 2002] Baude, J.-F., Caromel, D., Huet, F., et Vayssière, J. (2002). Objets actifs mobiles et communicants. *Techniques et Sciences Informatiques*, 21(6) :1–36.

[Baumann et al., 1998] Baumann, J., Hohl, F., Rothermel, K., et Straber, M. (1998). Mole concepts of a mobile agent system. *WWW Journal, Special Issue on Applications and Techniques of Web agents*, 1 :123–127. Baltzer Science Publishers.

[Beaumont, 1998] Beaumont, C. (1998). *Simulation distribuée : de l'application vers un support système*. Thèse de doctorat, Université de Rennes 1, Rennes, France. 165 pages.

[Beilby, 1972] Beilby, M. (1972). Road traffic simulation on a small computer. *Computer Journal*, 15(2) :134–137.

[Bellifemine et al., 1999] Bellifemine, F., Poggi, A., et Rimassa, G. (1999). JADE - a FIPA-compliant agent framework. In *Proceedings of International Conference on Practical Application of Intelligent Agents and Multi-Agents (PAAM)*, pages 97–108, Londres, Angleterre.

[Berge, 1970] Berge, C. (1970). *Graphes et hypergraphes*. Dunod, Paris, France.

[Bernard et Simatic, 1991] Bernard, G. et Simatic, M. (1991). A decentralized and efficient algorithm for load sharing in networks of workstations. In *Proceedings of EurOpen 91*, Tromsø, Norvege.

[Bic et al., 1996] Bic, L., Fukuda, M., et Dillencourt, M. (1996). Distributed computing using autonomous objects. *IEEE Computer*, 29(8) :55–61.

[Bic et Lee, 1987] Bic, L. et Lee, C. (1987). A data-driven model for a subset of logic programming. *ACM TOPLAS*, 9(4) :618–645.

[Birman et Joseph, 1987] Birman, K. et Joseph, T. (1987). Reliable communication in the presence of failures. *ACM Transactions on Computer Systems*, 5(1) :47–76.

[Birrell et Nelson, 1984] Birrell, A. et Nelson, B. (1984). Implementing remote procedure call. *ACM Transactions on Computer Systems*, 2(1) :39–59.

[Birtwistle et al., 1973] Birtwistle, G., Dahl, O., Myhrhaug, B., et Nygaard, K. (1973). *Simula Begin*. Chartwell-Bralt.

[Bouchenak et Hagimont, 2002] Bouchenak, S. et Hagimont, D. (2002). Services de mobilité et de persistance des applications java. In *Les intergiciels - développements récents dans CORBA, Java RMI et les agents mobiles*. Hermès Science. chapitre 6.

[Bousquet et Gautier, 1999] Bousquet, F. et Gautier, D. (1999). Comparaison de deux approches de modelisation des dynamiques spatiales par simulation multi-agent : les approches spatiale et acteurs. *Cybergeo*, 89.

[Bricker et al., 1991] Bricker, A., Litzkow, M., et Livny, M. (1991). Condor technical summary. Rapport de recherche 1069, University of Wisconsin, Madison, Wisconsin, USA.

[Buliung et al., 2002] Buliung, R., Kanaroglou, P., et Moah, H. (2002). GIS, objects and integrated urban models. In *Geoide International Colloquium on the Behavioural Foundations of Integrated Land-Use and Transportation Models*, Québec, Canada.

[Byrne et al., 1982] Byrne, A., de Laski, A., Courage, K., et Wallace, C. (1982). *Hanbook of computer models for traffic operations analysis*. Technology Sharing Report.

[Cariani, 1989] Cariani, P. (1989). *On The Design of Devices with Emergent Semantic Functions*. Thèse de doctorat, State University of New York. 236 pages.

[Carson, 2002] Carson, J. (2002). Model verification and validation. In Yücesan, E., Chen, C.-H., Snowdon, J., et Charnes, J., editors, *Proceedings of the 2002 Winter Simulation Conference*, pages 52–58.

[Chandra et al., 1993] Chandra, R., Gupta, A., et Hennessy, J. (1993). Data locality and load balancing in COOL. In *Proceedings of the 4th ACM Symposium on Principles and Practice of Parallel Programming*, pages 239–259. ACM Special Interest Group on Programming Languages (SIGPLAN), ACM Press.

[Chandy et Misra, 1979] Chandy, K. et Misra, J. (1979). Distributed simulation : a case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, 5(5) :440–452.

[Chandy et Misra, 1981] Chandy, K. et Misra, J. (1981). Asynchronous distributed simulation via a sequence of parallel computations. *Communications of the ACM*, 24(11) :198–206.

[Chang, 1982] Chang, E. (1982). Echo algorithm : depth parallel operations on general graphs. *IEEE Transactions on Software Engineering*, 8(4) :391–401.

[Charron-Bost et al., 1996] Charron-Bost, B., Mattern, F., et Tel, G. (1996). Synchronous, asynchronous, and causally ordered communication. *Distributed Computing*, 9(4) :173–191.

[Cheng, 1998] Cheng, Y. (1998). A rule-based reactive model for the simulation of aircraft on airport gates. *Knowledge-Based Systems*, 10(4) :225–236.

[Chopard et al., 1997] Chopard, B., Dupuis, A., et Luthi, P. (1997). A cellular automata model for urban traffic and its application to the city of geneva. In *Proceedings of Traffic and Granular Flow '97*.

[Claramunt et Thériault, 2001] Claramunt, C. et Thériault, M. (2001). A UML-based modelling approach for analysing travel behaviour. In Heeminck, A., Dekker, L., de Swaan, H., Smit, I., et van Stijn, T., editors, *Proceedings of the 4th European Simulation Symposium (EUROSIM 2001)*, Delft, Hollande.

[Clarke et Holm, 1987] Clarke, M. et Holm, E. (1987). Microsimulation methods in spatial analysis and planning. *Geografiska Annaler*, 69B :145–164.

[Clergue, 1997] Clergue, G. (1997). *L'apprentissage de la Complexité*. Edition Hermès.

[Conan et al., 1995] Conan, D., Taponot, P., et Bernard, G. (1995). Rollback recovery of PVM applications. In *Proceedings of Romanian Open Systems Event (ROSE '95)*, pages 15–23, Bucarest, Roumanie.

[Courtrai et al., 2001] Courtrai, L., Mahéo, Y., et Raimbault, F. (2001). Java objects communication on a high performance network. In Werner, B., editor, *Proceedings of the 9th Euromicro Workshop on Parallel and Distributed Processing*, pages 203–210, Los Alamitos, Californie, USA. IEEE Computer Society.

[Cristian, 1989] Cristian, F. (1989). Probabilistic clock synchronization. *Journal of Distributed Computing*, 3 :146–158.

[Cutini, 1999] Cutini, V. (1999). Urban space and pedestrian movement : A study on the configurational hypothesi. *Cybergeo*, 111.

[Davies et al., 1993] Davies, N., Davy, M., Blair, G., et Mariani, J. (1993). Object invocation and management in the zenith distributed multimedia information system. *Information and Software Technology*, 35(5) :259–266.

[de Rosnay, 1975] de Rosnay, J. (1975). *Le macroscope : vers une vision globale*. Editions du Seuil.

[de Rosnay, 1995] de Rosnay, J. (1995). *L'homme Symbiotique*. Editions du Seuil.

[de Rosnay, 1998] de Rosnay, J. (1998). Une vision du futur : la coévolution entre technologie et société. *Revue des Sciences Humaines et Sociales*, 59(1). Boeck Université Éditeur.

[Dijkstra, 1959] Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Matematik*, 1 :269–271.

[Dilley, 1993] Dilley, J. (1993). Object-oriented distributed computing with C++ and OSF DCE. *Lecture Notes in Computer Science*, 731 :256–269.

[Douglis et Ousterhout, 1987] Douglis, F. et Ousterhout, J. (1987). Process migration in the sprite operating system. In *Proceedings of the 7th International Conference on Distributed Computing Systems*, pages 18–25, Berlin, Allemagne. IEEE Computer Society.

[Douglis et Ousterhout, 1989] Douglis, F. et Ousterhout, J. (1989). Transparent process migration for personal workstations. Rapport de recherche CSD-89-540, University of California, Berkeley, Californie, USA. 30 pages.

[Duncan, 1990] Duncan, R. (1990). A survey of parallel computer architectures. *Computer*, 23(2) :5–16.

[Durand, 1993] Durand, D. (1993). *La Systémique*. Collection que sais-je ?

[Edmonds, 1999] Edmonds, B. (1999). The pragmatic roots of context. In Bouquet, P., Serafini, L., Brézillon, P., Benerecetti, M., et Castellani, F., editors, *Proceedings of the 2nd International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-99)*, volume 1688 of *Lecture Notes in Artificial Intelligence*, pages 119–132, Trento, Italie. Springer-Verlag.

[Erol et al., 1999] Erol, K., Levy, R., et Wentworth, J. (1999). Application of agent technology to traffic simulation. Rapport de recherche, U.S. Department of Transportation, McLean, Virginie, USA.

[Fagg et Dongarra, 2000] Fagg, G. et Dongarra, J. (2000). FT-MPI : Fault tolerant MPI, supporting dynamic applications in a dynamic world. *Lecture Notes in Computer Science*, 1908 :346–354.

[Ferber, 1995] Ferber, J. (1995). *Les Systèmes Multi-Agents : Vers une Intelligence Collective*. InterEditions.

[Ferscha, 1995] Ferscha, A. (1995). Parallel and distributed simulation of discrete event systems. Rapport de recherche, Institut für Angewandte Informatik, University of Vienna, Vienne, Austriche. 65 pages.

[Fishwick, 1995] Fishwick, P. (1995). *Simulation Model Design and Execution*. Prentice Hall.

[Flynn, 1972] Flynn, M. (1972). Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, 21(9) :948–960.

[Forrester, 1980] Forrester, J. (1980). *Principes des Systèmes*. Presses Universitaires de Lyon.

[Freedman, 1991] Freedman, D. (1991). Experience building a process migration subsystem for UNIX. In *Proceedings of USENIX Technical Conference*, pages 349–356, Dallas, Texas, USA. USENIX.

[Frihida, 2002] Frihida, A. (2002). *Utilisation du formalisme UML dans la conception des bases de données spatio-temporelles orientées objets sur la mobilité des personnes*. Thèse de doctorat, Université de Montréal, Montréal, Canada.

[Frihida et al., 2002] Frihida, A., Marceau, D., et Thériault, M. (2002). Spatio-temporal object-oriented data model for disaggregate travel behavior. *Transactions in GIS*, 6(3) :277–294.

[Fujimoto, 1990] Fujimoto, R. (1990). Parallel discrete event simulation. *Communications of the ACM*, 22(10) :30–53.

[Fukuda et al., 1999] Fukuda, M., Bic, L., Dillencourt, M., et Cahill, J. (1999). Messages versus messengers in distributed programming. *Journal of Parallel and Distributed Computing*, 57(2) :188–211.

[Gardner, 1970] Gardner, M. (1970). The fantastic combinations of John Conway's new solitaire game life. *Scientific American*, 223(4) :120–123.

[Gas et Joel, 1987] Gas, S. et Joel, L. (1987). Concepts of model confidence. *Computers and Operations Research*, 66(2) :250–258.

[Gatersleben et van der Wiej, 1999] Gatersleben, M. et van der Wiej, S. (1999). Analysis and simulation of passenger flows in an airport terminal. In Farrington, P., Nembhand, H., Sturrock, D., et Evans, G., editors, *Proceedings of the 1999 Winter Simulation conference*, pages 1226–1231.

[Geist et al., 1994] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manachek, R., et Sunderam, V. (1994). *PVM : Parallel Virtual Machine*. MIT Press, Cambridge, Massachusetts, USA.

[Geist et al., 1996] Geist, G., Kohla, J., et Papadopoulos, P. (1996). PVM and MPI : A comparison of features. *Calculateurs Parallèles*, 8(2) :137–150.

[Gilbert et Troitzsch, 1999] Gilbert, N. et Troitzsch, K. (1999). *Simulation for the Social Scientist*. Open University Press.

[Gilmer et Hong, 1986] Gilmer, J. et Hong, J. (1986). Replicated state space approach for parallel simulation. In *Proceedings of the 1986 Winter Simulation Conference*, pages 430–433, Washington D.C., USA.

[Gokhale et Schmidt, 1996] Gokhale, A. et Schmidt, D. (1996). Measuring the performance of communication middleware on high-speed networks. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 306–317. ACM Special Interest Group on Data Communications (SIGCOMM), ACM Press.

[Goodchild, 1998] Goodchild, M. (1998). Geographical information systems and disaggregate transportation modelling. *Geographical Systems*, 5(1–2) :19–44.

[Gosling et McGilton, 1995] Gosling, J. et McGilton, H. (1995). *The Java Language Environment*. SUN Microsystems Computer Company.

[Greco et al., 2002] Greco, G., Greco, S., et Zumpano, E. (2002). A stochastic approach for modeling and computing web communities. In *Proceedings of the Third International Conference on Web Information Systems Engineering (WISE'02)*, pages 43–52, Singapour, Chine.

[Greenberg, 1972] Greenberg, S. (1972). *A GPSS Primer*. John Wiley and Sons, New York, New York, USA.

[Greenberger et al., 1976] Greenberger, M., Crenson, M., et Crissey, B. (1976). *Models in the Policy Process : Public Decision Making in the Computer Era*. Basic Books.

[Guerraoui et al., 1998] Guerraoui, R., Raynal, M., et Schiper, A. (1998). Validation atomique et consensus : une approche synthétique. *Technique et science informatiques (TSI)*, 17(3) :279–298.

[Guerraoui et Vinoski, 1997] Guerraoui, R. et Vinoski, S. (1997). Guest editorial. *Distributed Systems Engineering*, 4(3) :129–130.

[Gusella et Zatti, 1985] Gusella, R. et Zatti, S. (1985). The berkeley UNIX 4.3BSD time synchronization protocol : protocol specification. Rapport de recherche UCB/CSD 85/250, University of California, Berkeley, Californie, USA.

[Gutknecht, 2001] Gutknecht, O. (2001). *Proposition d'un modèle organisationnel générique de systèmes multi-agents*. Thèse de doctorat, Université de Montpellier II, Montpellier, France. 220 pages.

[Habert et al., 1990] Habert, S., Mosseri, L., et Abrossimov, V. (1990). Cool : Kernel support for object-oriented environments. In Meyrowitz, N., editor, *Proceedings of European Conference on Object-Oriented Programming (ECOOP'90)*, pages 269–277, Ottawa, Canada. ACM Press.

[Hartigan, 1975] Hartigan, J. (1975). *Clustering Algorithms*. Wiley, New York, New York, USA.

[Hawick et al., 1999] Hawick, K., Grove, D. A., et Vaughan, F. (1999). Beowulf - a new hope for parallel computing ? In *Proceedings of the 6th IDEA Workshop*, Rutherglen, Victoria, Australie.

[Herbert, 1994] Herbert, A. (1994). An ANSA overview. *IEEE Network*, 8(1).

[Heylighen, 1991] Heylighen, F. (1991). Modelling emergence. *World Futures : the Journal of General Evolution*. Special Issue on Creative Evolution.

[Huet et al., 2003] Huet, T., Osman, T., et Ray, C. (2003). Modelling traffic navigation network with a multi-agent platform. In Al-Dabass, D., editor, *Proceedings of 17th European Simulation Multiconference (ESM2003)*, pages 111–117, Nottingham, Angleterre.

[Hunt et al., 1981] Hunt, P., Robertson, D., Bretherton, R., et Winton, R. (1981). SCOOT : a traffic responsive method of coordinating signals. Rapport de recherche 1014, Transport and Road Research Laboratory.

[Ingalls, 2002] Ingalls, R. (2002). Introduction to simulation. In Yücesan, E., Chen, C.-H., Snowdon, J., et Charnes, J., editors, *Proceedings of the 2002 Winter Simulation Conference*, pages 7–16.

[Ingels et Raynal, 1989] Ingels, P. et Raynal, M. (1989). Simulation répartie de systèmes a evénements discrets. partie 1 : modélisation et schemas d'exécution. Rapport de recherche RR-1057, INRIA Rennes. 25 pages.

[Jain et al., 1999] Jain, A., Murty, M., et Flynn, P. (1999). Data clustering : a review. *ACM Computing Surveys*, 31(3) :264–323.

[Jefferson, 1985] Jefferson, D. (1985). Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3) :404–425.

[Jha et al., 1995] Jha, M., Joshi, A., et Sinha, K. (1995). A framework for dynamic network traffic simulation on distributed systems. In *Proceedings of applications of advanced technologies in transportation*, Capri, Italie.

[Jul et al., 1988] Jul, E., Levy, H., Hutchinson, N., et Black, A. (1988). Fine-grained mobility in the Emerald system. *ACM Transactions on Computer Systems*, 6(1) :109–133.

[Kaubisch et al., 1976] Kaubisch, W., Perrot, R., et Hoare, C. (1976). Quasiparallel programming. *Software-Practice and Experience*, 6(3) :341–356.

[Keleher et al., 1994] Keleher, P., Dwarkadas, S., Cox, A., et Zwaenepoel, W. (1994). Treadmarks : distributed shared memory on standard workstations and operating systems. In *Proceedings of the Winter 1994 USENIX Conference*, pages 115–131.

[Khan et al., 2002] Khan, A., Abraham, J., et Hunt, J. (2002). A system for microsimulating business establishments : Analysis, design and results. In *Geoide International Colloquium on the Behavioural Foundations of Integrated Land-Use and Transportation Models*, Québec, Canada.

[Kheir, 1996] Kheir, N. (1996). *Systems Modeling and Computer Simulation*. Marcel Dekker, New York, New York, USA, second edition.

[Kim, 2001] Kim, T. (2001). Introduction to DEVS formalism. In *13th European Simulation Symposium (ESS2001)*, Marseille, France.

[Kosonen et Bargiela, 1999] Kosonen, I. et Bargiela, A. (1999). A distributed traffic monitoring and information system. *Journal of Geographic Information and Decision Analysis*, 3(1) :31–40.

[Kotz, 1997] Kotz, D. (1997). Mobile agents for mobile internet computing. *IEEE Internet Computing*, 1 :58–67.

[Kremien et Kramer, 1992] Kremien, O. et Kramer, J. (1992). Methodical analysis of adaptive load sharing algorithms. *IEEE Transactions on parallel and distributed systems*, 3(6) :747–760.

[Kreutzer, 1986] Kreutzer, W. (1986). *System Simulation Programming Styles and Languages*. Addison-Wesley Publishing, Reading, Massachusetts, USA.

[Lamport, 1978] Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7) :558–565.

[Lamport, 1986] Lamport, L. (1986). On interprocess communication, part I : basic formalism. *Distributed Computing*, 1(2) :77–85.

[Lamport et al., 1982] Lamport, L., Shostak, R., et Pease, M. (1982). The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3) :382–401.

[Landry et Santerre, 1999] Landry, R. et Santerre, M. (1999). Méthodes de simulation en science politique. Rapport de recherche, Université Laval, Québec, Canada. 26 pages.

[Langton, 1989] Langton, C. (1989). Artificial life. In *Artificial Life*, pages 1–47. Addison-Wesley.

[Lapointe, 1993] Lapointe, J. (1993). L'approche systémique et la technologie de l'education. *Éducatechnologiques*, 1(1).

[Lapri, 1991] Lapri, J.-C. (1991). Surêté de fonctionnement : concepts de base et terminologie. *Dependable Computing and Fault-Tolerant Systems*, pages 47–95.

[le Moigne, 1990] le Moigne, J.-L. (1990). *La Théorie Générale du Système*. Dunod.

[le Moigne, 1999] le Moigne, J.-L. (1999). *La Modélisation des Systèmes Complexes*. Dunod.

[Leonard et al., 1978] Leonard, D., Tough, J., et Baguley, P. (1978). CONTRAM : a traffic assignement model for predicting flows and queues during peak periods. Rapport de recherche 841, Transport and Road Research Laboratory.

[Lesourne, 1976] Lesourne, J. (1976). *Les systèmes du destin*. Dalloz.

[Li, 1986] Li, K. (1986). Shared virtual memory on loosely coupled multiprocessors. In *Proceedings of the IEEE International Conference on Computer Languages*.

[Logan et Theodoropolous, 2001] Logan, B. et Theodoropolous, G. (2001). Distributed simulation of multi-agent systems. *Proceedings of the IEEE*, 89(2) :174–185.

[Lu et al., 1995] Lu, H., Dwarkadas, S., Cox, A., et Zwaenepoel, W. (1995). Message passing vs. distributed shared memory on networks of workstations. In *Proceedings of Supercomputing'95*.

[Magne et al., 2000] Magne, L., Rabut, S., et Gabard, J.-F. (2000). Towards an hybrid macro-micro traffic flow simulation model. In *Proceedings of INFORMS Spring 2000 Conference*, Salt Lake City, USA.

[Maniatty et al., 1993] Maniatty, W., Szymanski, B., et Caraco, T. (1993). Implementation and performance of parallel ecological simulations. In *Proceedings of IFIP WG10.3 International Conference on Applications in Parallel and Distributed Computing*, Amsterdam, Hollande. Elsevier Science.

[Markowitz et al., 1962] Markowitz, H., Hausner, B., et Carr, H. (1962). *Simscript : a simulation programming language*. Prentice Hall.

[Maxwell, 1997] Maxwell, T. (1997). An open geographic modeling environment. *Simulation Journal*, 68(3) :175–185.

[Maziero, 1994] Maziero, C. (1994). *Conception et Réalisation d'un Noyau de Système Réparti pour la Simulation Parallèle*. Thèse de doctorat, Université de Rennes 1, Rennes, France. 119 pages.

[McGregor et al., 1998] McGregor, M., Rola-Rubzen, F., et Murray-Prior, R. (1998). Micro and macro-level approaches to modelling decision making. In *Proceedings of the 15th International Symposium of the Association for Farming Systems Research-Extension*, Pretoria, Afrique du Sud.

[McQueen, 1967] McQueen, J. (1967). Some methods of classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium in Mathematics, Statistics and Probability*, volume 1, pages 281–296, Berkeley, Californie, USA.

[Merchant et al., 1995] Merchant, F., Bic, L., Borst, P., Corbin, M., Dillencourt, M., Fukuda, M., et Sapaty, P. (1995). Simulating autonomous objects in a spatial database using wave. In *Proceedings of the 9th European Simulation Multiconference (ESM95)*, pages 768–773, Prague, République Tchèque.

[Miller, 1998] Miller, E. (1998). Emerging themes and research frontiers in GIS and activity-based travel demand forecasting. *Geographical Systems*, 5 :189–198.

[Miller, 2002] Miller, E. (2002). Propositions for modelling household decision-making. In *Geoide International Colloquium on the Behavioural Foundations of Integrated Land-Use and Transportation Models*, Québec, Canada.

[Milojicic et al., 1998] Milojicic, D., Laforge, W., et Chauhan, D. (1998). Mobile objects and agents (MOA). *Distributed Systems Engineering*, 5 :241–227.

[Milojicic et al., 1993] Milojicic, D., Zint, W., Dangel, A., et Giese, P. (1993). Task migration on the top of the mach microkernel. In *Proceedings of the USENIX Mach III Symposium*, pages 273–289, Santa Fe, Nouveau Mexique, USA.

[Minsky, 1986] Minsky, M. (1986). *The society of mind*. Simon and Shuster.

[Misra, 1986a] Misra, J. (1986a). Axioms for memory access in asynchronous hardware systems. *ACM Transactions on Programming Languages and Systems*, 8(1) :142–153.

[Misra, 1986b] Misra, J. (1986b). Distributed discrete-event simulation. *Computing Surveys*, 18(1) :39–65.

[Morin, 1977] Morin, E. (1977). *La méthode, la nature de la nature*. Seuil.

[Mullender et al., 1990] Mullender, S., van Rossum, G., Tanenbaum, A., van Renesse, R., et van Staveren, H. (1990). Amoeba : a distributed operating system for the 1990s. *IEEE Computer*, 23(5) :44–53.

[Muller et Sens, 1997] Muller, G. et Sens, P. (1997). Migration et points de reprise dans les systèmes distribués : technique de mise en oeuvre et mécanismes communs. In *Actes de l'école d'été de placement dynamique et répartition de charge*, pages 103–114. Collection Didactique INRIA.

[Nagel, 1995] Nagel, K. (1995). *High-Speed Microsimulation of Traffic Flow*. Thèse de doctorat, Mathematisches Institut, Universität zu Köln, Köln, Allemagne.

[Nagel, 2002] Nagel, K. (2002). Distributed intelligence in large scale traffic simulations on parallel computers. In *Proceedings of Collective Cognition : Mathematical Foundations of Distributed Intelligence*, Santa Fe, Nouveau Mexique, USA.

[Nuttall, 1994] Nuttall, M. (1994). Survey of systems providing process or object migration. Rapport de recherche 94/10, Imperial College, Londres, Angleterre.

[Nuttall, 1997] Nuttall, M. (1997). *Cluster load balancing using process migration*. Thèse de doctorat, Imperial College, Londres, Angleterre.

[O'Brian et Nicol, 1998] O'Brian, P. et Nicol, R. (1998). FIPA – towards a standard for software agents. *BT Technology Journal*, 16(3).

[O'Connor, 1994] O'Connor, M. (1994). Process migration in the chorus micro-kernel. Master's thesis, Trinity College, Dublin, Irelande.

[O'Donoghue, 2001] O'Donoghue, C. (2001). Dynamic microsimulation : A methodological survey. *Brazilian Electronic Journal of Economics*, 4(2).

[Olsen, 1992] Olsen, M. (1992). A persistent object infrastructure for heterogeneous distributed systems. In *Proceedings of International Workshop on Object Orientation and Operating Systems*, pages 49–56, Dourdan, France. IEEE Computer Society.

[Page, 1994] Page, E. (1994). *Simulation Modeling Methodology : Principles and Etiology of Decision Support*. Thèse de doctorat, Department of Computer Science, Virginia Tech., Blacksburg, Virginie, USA. 293 pages.

[Page et al., 1999] Page, E., Nicol, D., Balci, O., Fujimoto, R., Fishwick, P., L'Ecuyer, P., et Smith, R. (1999). Strategic directions in simulation research. In Farrington, P., Nembhard, H., Sturrock, D., et Evans, G., editors, *Proceedings of the 1999 Winter Simulation Conference*, pages 1509–1520.

[Peacock et al., 1979] Peacock, J., Manning, E., et Wong, J. (1979). Distributed simulation using a network of processors. *Computer Network*, 3 :44–56.

[Perumalla et Fujimoto, 2001] Perumalla, K. et Fujimoto, R. (2001). Virtual time synchronization over unreliable network transport. In *Proceedings of the fifteenth workshop on Parallel and distributed simulation*, pages 129–136, Lake Arrowhead, Californie, USA.

[Peytchev et Claramunt, 2001] Peytchev, E. et Claramunt, C. (2001). Experiences in building decision support systems for traffic and transportation GIS. In Aref, W., editor, *Proceedings of the 9th ACM International Symposium on Advances in Geographic Information Systems (GIS-01)*, pages 154–159, New York, New York, USA. ACM Press.

[Phipps et Langlois, 1997] Phipps, M. et Langlois, A. (1997). *Automates cellulaires, application à la simulation urbaine*. Hermes Science.

[Powell et Miller, 1983] Powell, M. et Miller, B. (1983). Process migration in DEMOS/MP. In *Proceedings of the 9th ACM Symposium on Operating Systems Principles (SOSP82)*, pages 110–119, Bretton Woods, New Hampshire, USA. ACM Press.

[Pursula, 1999] Pursula, M. (1999). Simulations of traffic systems : an overview. *Journal of Geographic Information and Decision Analysis*, 3(1) :1–8.

[Rackl et al., 1999] Rackl, G., de Stefani, F., Heran, F., Pasquarelli, A., et Ludwig, T. (1999). Distributed airport simulation using CORBA and DIS. In *Proceedings of 4th International Symposium on Software Engineering for Parallel and Distributed Systems*, pages 166–173, Los Angeles, Californie, USA.

[Radajewski et Eadline, 1999] Radajewski, J. et Eadline, D. (1999). Beowulf installation and administration. Rapport de recherche, Beowulf Underground. 33 pages.

[Rashid et Robertson, 1981] Rashid, R. et Robertson, G. (1981). Accent : a communication oriented network operating system kernel. In *Proceedings of the 8th ACM Symposium on Operating Systems Principles (SOSP81)*, pages 64–75, Pacific Grove, Californie, USA. ACM Press.

[Ray, 2001a] Ray, C. (2001a). Round Trip Bus : A group migration protocol. Rapport de recherche 2001–05, IRENav. 28 pages.

[Ray, 2001b] Ray, C. (2001b). Round Trip Bus : A protocol for migrating groups of objects. In *11th Ph.D. Workshop on Object Oriented Systems, 16th European Conference on Object-Oriented Programming (ECOOP 2001)*, Budapest, Hongrie.

[Ray et Claramunt, 2002a] Ray, C. et Claramunt, C. (2002a). Atlas : A distributed system for the simulation of large-scale systems. In Chen, S.-C. et Voisard, A., editors, *Proceedings of 10th ACM International Symposium On Advances In Geographic Information Systems*, pages 155–162, McLean, Virginie, USA. ACM Press.

[Ray et Claramunt, 2002b] Ray, C. et Claramunt, C. (2002b). Atlas : A novel distributed computing system for the simulation of disaggregated data flows. Rapport de recherche 2002–01, IRENav. 32 pages.

[Ray et Claramunt, 2003] Ray, C. et Claramunt, C. (2003). A distributed system for the simulation of people flows in an airport terminal. *Knowledge-Based Systems*, 16(4) :191–203. Elsevier Science Publication.

[Richardson, 1991] Richardson, G. (1991). *Feedback Thought in Social Science and Systems Theory*. University of Pennsylvania Press.

[Rickert, 1997] Rickert, M. (1997). *Traffic Simulation on Distributed Memory Computers*. Thèse de doctorat, Mathematisches Institut, Universität zu Köln, Köln, Allemagne. 158 pages.

[Righter et Walrand, 1989] Righter, R. et Walrand, J. C. (1989). Distributed simulation of discrete event systems. *Proceedings of the IEEE*, 77(1) :99–113.

[Roorda et al., 2002] Roorda, M., Doherty, S., et Miller, E. (2002). Operationalizing household activity scheduling models : Addressing assumptions and the use of new sources of behavioural data. In *Geoide International Colloquium on the Behavioural Foundations of Integrated Land-Use and Transportation Models*, Québec, Canada.

[Rosen, 1985] Rosen, R. (1985). *Anticipatory Systems*. Pergamon Press.

[Sapaty et Borst, 1994] Sapaty, P. et Borst, P. (1994). An overview of the WAVE language and system for distributed processing of open networks. Rapport de recherche, University of Surrey, Guildford, Angleterre.

[Sargent, 2000] Sargent, R. (2000). Verification, validation, and accreditation of simulation models. In Joines, J., Barton, R., Kang, K., et Fishwick, P., editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 50–59.

[Setti et Hutchinson, 1994] Setti, J. et Hutchinson, B. (1994). Passenger-terminal simulation model. *Journal of Transportation Engineering*, 120(4) :517–535.

[Shannon, 1975] Shannon, R. E. (1975). *Systems simulation, the art and science*. Prentice Hall.

[Shapiro, 1986] Shapiro, M. (1986). SOS : a distributed object-oriented operating system. In *Proceedings of the 2nd ACM SIGOPS European Workshop, on Making Distributed Systems Work*, Amsterdam, Hollande. ACM Press.

[Shreckengost, 1985] Shreckengost, R. (1985). Dynamic simulation models : How valid are they ? *NIDA Research Monograph*, 57 :63–70. Self-Report Methods of Estimating Drug Use : Meeting Current Challenges to Validity.

[Simon et Nagel, 1998] Simon, P. et Nagel, K. (1998). Simplified cellular automaton model for city traffic. *Physical Review E*, 58(2) :1286–1295.

[Sloot et al., 1997] Sloot, P., Schoneveld, A., de Ronde, J., et Kaandorp, J. (1997). Large scale simulations of complex systems part i : Conceptual framework. Technical Report SFI Working Paper : 97-07-070, Santa Fe Instute, Santa Fe, Nouveau Mexique, USA. 85 pages.

[Snell et al., 1996] Snell, Q., Mikler, A., et Gustafson, J. (1996). Netpipe : a network protocol independent performance evaluator. Rapport de recherche, Ames Laboratory. 6 pages.

[Steketee et al., 1994] Steketee, C., Zhu, W., et Moseley, P. (1994). Implementation of process migration in amoeba. In *Proceedings of the 14th International Conference on Distributed Computing Systems (ICDCS'94)*, pages 194–201, Poznan, Pologne. IEEE Computer Society.

[Stillwell et al., 1999] Stillwell, J., Geertman, J., et Openshaw, S. (1999). *Geograhical information and planning*. Advances in Spatial Science. Springer-Verlag, Heidelberg, Allemagne.

[Sunderam, 1990] Sunderam, V. (1990). PVM : A framework for parallel distributed computing. *Concurrency : practice and experience*, 2(4) :315–339.

[Tanenbaum, 1993] Tanenbaum, A. (1993). Distributed systems anno 1992 : What we have learned so far ? *Distributed Systems Engineering*, 1(1) :3–10.

[Teretsch, 1993] Teretsch, D. (1993). Scheduling flights at hub airports. *Transportation Research, Part B*, 27(2) :133–150.

[Theimer et al., 1985] Theimer, M., Lantz, K., et Cheriton, D. (1985). Preemptable remote execution facilities for the V system. In *Proceedings of the 10th ACM Symposium on Operating Systems Principles (SOSP85)*, pages 2–12, Orcas Island, Washington, USA. ACM Press.

[Thériault et al., 2002] Thériault, M., Claramunt, C., Séguin, A.-M., et Villeneuve, P. (2002). A spatio-temporal model for lifelines analysis. In *Proceedings of the 10th International Symposium on Spatial Data Handling*. Springer-Verlag.

[Torrens, 2001] Torrens, P. (2001). Can geocomputation save urban simulation ? Throw some agents into the mixture, simmer, and wait... Rapport de recherche 32, University College, Londres, Angleterre. 37 pages.

[Torrens et O'Sullivan, 2000] Torrens, P. et O'Sullivan, D. (2000). Cities, cells, and complexity : developing a research agenda for urban geocomputation. In *Proceedings of the 5th International Conference on GeoComputation*, University of Greenwich, Angleterre.

[Treuil et al., 2002] Treuil, J.-P., Mullon, C., Perrier, E., et Piron, M. (2002). Simulation multi-agent de dynamiques spatialisées. In Sanders, L., editor, *Modèles en analyse spatiale*. Hermes Science.

[Tunasar et al., 1998] Tunasar, C., Bender, G., et Young, H. (1998). Modelling curbside vehicular traffic at airports. In Medeiros, D., Watson, E., et Manivannan, M., editors, *Proceedings of the 1998 Winter Simulation Conference*, pages 1113–1117, Piscataway, New Jersey, USA. IEEE Computer Society.

[Vanbergue, 2000] Vanbergue, D. (2000). Modélisation de phénomènes urbains : Simulation des migrations intra-urbaines. In *Proceedings of 8èmes Journées Francophones d'Intelligence Artificielle Distribuée (JFIAD'00)*, St Jean la Vêtre, France.

[Vanbergue et Drogoul, 2002] Vanbergue, D. et Drogoul, A. (2002). Approche multi-agent pour la simulation urbaine. In *Proceedings of 6èmes Journées du GDR CASSINI*, pages 95–112, École Navale, France.

[Venkatesh et al., 1986] Venkatesh, K., Radhakrishnan, T., et Li, H. F. (1986). Discrete event simulation in a distributed system. In *Proceedings of the International Computer Software and Applications Conference (COMPSAC)*, pages 123–129. IEEE Computer Society.

[Voisard et Schweppe, 1998] Voisard, A. et Schweppe, H. (1998). Abstraction and decomposition in interoperable GIS. *International Journal of Geographic Information Science (IJGIS)*, 12(4). Taylor and Francis Publishing.

[von Bertalanffy, 1973] von Bertalanffy, L. (1973). *Théorie Générale des Systèmes*. Dunod.

[von Neumann, 1951] von Neumann, J. (1951). The general and logical theory of automata. In Jeffress, L., editor, *Cerebral Mechanisms in Behavior*, pages 1–41. John Wiley and Sons, New York, New York, USA.

[von Neumann, 1966] von Neumann, J. (1966). *Theory of self-reproducing automata*. University of Illinois Press.

[Walker et al., 1992] Walker, B., Popek, G., English, R., Kline, C., et Thiel, G. (1992). The LOCUS distributed operating system. In Ananda, A. et Srinivasan, B., editors, *Distributed Computing Systems : Concepts and Structures*, pages 145–164. IEEE Computer Society, Los Alamos, Californie, USA.

[Walker, 1993] Walker, D. (1993). MPI : A standard message passing interface for distributed memory environments. In *Intel Supercomputer User Group 1993 Conference (ISUG)*, pages 67–75, St. Louis. Intel Scientific Computers.

[Wiener, 1948] Wiener, N. (1948). *Cybernetics*. MIT Press.

[Wong et al., 1997] Wong, D., Paciorek, N., Walsh, T., et DiCelie, J. (1997). Concordia : an infrastructure for collaborating mobile agents. In Rothermel, K. et Popescu-Zeletin, R., editors, *Proceedings of the First International Workshop on Mobile Agents*, volume 1219 of *Lecture Notes in Computer Science*, pages 86–97. Springer-Verlag.

[Wood, 1986] Wood, D. (1986). MIT model analysis program : what we have learned about policy model review. In *Proceedings of the 1986 Winter Simulation Conference*, pages 248–252, Washington D.C., USA.

[Zayas, 1987] Zayas, E. (1987). Attacking the process migration bottleneck. In *Proceedings of 11th ACM Symposium on Operating System Principles (SOSP87)*, pages 13–24, Austin, Texas, USA. ACM Press.

[Zeigler, 1976] Zeigler, B. (1976). *Theory of modeling and simulation*. John Wiley and Sons, New York, New York, USA.

[Zeigler et al., 2000] Zeigler, B., Praehofer, H., et Kim, T. (2000). *Theory of modeling and simulation*. Academic Press, second edition.

[Zeigler et Sarjoughian, 2000] Zeigler, B. et Sarjoughian, H. (2000). Creating distributed simulation using DEVS M and S environments. In Joines, J., Barton, R., Kang, K., et Fishwick, P., editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 158–160.

Revues internationales

Cyril Ray, Christophe Claramunt : A distributed system for the simulation of people flows in an airport terminal, Knowledge-Based Systems, volume 16(4), pages 191-203, juin 2003, Elsevier Science publication, ISSN 0950-7051

Cyril Ray : Un système distribué pour la modélisation de flux de données spatiales désagrégées, Revue Internationale de Géomatique, SIG transport, volume 13(2), pages 201-224, octobre 2003, édition Hermès, ISSN 1260-5875

Conférences internationales avec comité de lecture et actes

Cyril Ray, Christophe Claramunt : ATLAS : A distributed system for the simulation of large-scale systems, 10th ACM International Symposium On Advances In Geographic Information Systems, pages 155-162, novembre 2002, McLean, Virginie, USA, ISBN 1-58113-591-2

Thierry Huet, Taha Osman, Cyril Ray : Modelling traffic navigation network with a multi-agent platform, European Simulation Multiconference (ESM2003), pages 111-117, juin 2003, Nottingham, Angleterre, ISBN 3-936150-25-7

Conférences nationales avec comité de lecture et actes

Cyril Ray : Un système distribué pour la modélisation de flux de données spatiales désagrégées, $6^{èmes}$ Journées du GDR CASSINI, pages 75-93, septembre 2002, École Navale, France

Conférences internationales avec comité de lecture

Cyril Ray, Christophe Claramunt : A new distributed computing approach for the modelling and simulation of disaggregated data flows : application to an airport system, Geoide International Colloquium on Integrated Land-use and Transportation Models, 14 pages, juin 2002, Québec, Canada

Cyril Ray : Round Trip Bus protocol : a protocol for migrating groups of objects, 11th Ph.D. Workshop on Object Oriented Systems, 16th European Conference on Object-Oriented Programming (ECOOP 2001), 11 pages, juin 2001, Budapest, Hongrie

Autres conférences internationales

Cyril Ray, Thierry Huet : Active objects in distributed environment, Middleware 2000, session poster, 2 pages, avril 2000, Hudson River Valley, New York, USA

Rapports de recherche

Thierry Huet, Taha Osman, Cyril Ray : Modelling traffic navigation network with a multi-agent platform, IRENav / Nottingham Trent University, rapport de recherche TR 2003-01, 11 pages, février 2003

Cyril Ray, Christophe Claramunt : Atlas : A novel distributed computing for disaggregated data flows, IRENav, rapport de recherche TR 2002-01, 32 pages, mars 2002

Cyril Ray : Round Trip Bus protocol : a group migration protocol, IRENav, rapport de recherche TR 2001-05, 28 pages, juillet 2001

Michel Farine, Thomas Devogele, Thierry Huet, Eric Saux, Rémy Thibaud, Cyril Ray, Rodéric Bera : rapport d'activité 1997-2000, IRENav, 14 pages, juillet 2000

Cette thèse introduit une méthodologie composée d'une démarche de modélisation et d'un support informatique pour la simulation distribuée et l'analyse de systèmes complexes à larges flux de données désagrégées. Le choix méthodologique proposé est influencé à la fois par le système à simuler et par l'architecture informatique servant de support de simulation dans le but de proposer une solution où l'aspect technique soit en accord avec l'aspect conceptuel. La démarche de modélisation intègre une description hiérarchique représentant l'organisation interne d'un système complexe, la construction d'un graphe logique traduisant une décomposition structurelle du système. Le graphe logique est ensuite utilisé pour réaliser une projection qui produit un graphe physique établissant une abstraction qui lie le niveau de représentation conceptuel et le système distribué sous-jacent. La plate-forme Atlas est un support de simulation distribuée à événements discrets dirigée par le temps et dont les propriétés dynamiques reproduisent le comportement de larges flux de données désagrégées. Une des particularités de cette plate-forme réside dans le concept de migration physique par groupe qui donne une solution adaptée pour réaliser et coordonner la migration de larges flux d'objets. Cette migration par groupe possède de nombreux avantages pour la simulation de systèmes sociétaux et urbains, notamment pour les systèmes en transport. L'utilisation de notre méthodologie et de sa plate-forme est illustrée par une application en transport qui modélise et simule les flux de personnes entre les différents halls d'un terminal d'aéroport.

This thesis introduces a methodology composed of a modelling approach and a computing support for the distributed simulation and the analysis of complex systems with large disaggregated data flows. The proposed methodology is influenced by both the system to simulate and the computing architecture used as a simulation support. This ensures a solution where the technical aspect fits the conceptual one. The modelling approach integrates a hierarchical description of the internal organization of a complex system, the design of a logical graph representing a structural decomposition of the system. A physical graph then establishes a binding between the conceptual representation level and the underlying distributed system. The Atlas platform is a support for discrete events time-driven distributed simulation whose dynamic properties replicate the behaviour of large disaggregated data flows. One of particularities of this platform resides in the concept of physical migration by group. This give a suitted solution to achieve and coordinate migration of large data flows, an important aspect for the simulation of societal and urban systems, particularly for transportation systems. The method and its platform is illustrated by an application in transportation that models and simulates people flows between the different halls of an airport terminal.