Chapitre 10

Approche multi-formalismes pour la spécification des systèmes multi-agents

10.1. Introduction

L'approche multi-agents propose une nouvelle vision d'analyse de problèmes, fondée sur une critique radicale des méthodes centralisées. Elle considère les systèmes comme des sociétés composées d'entités autonomes et indépendantes, appelées agents, qui interagissent en vue de résoudre un problème ou de réaliser collectivement une tâche. Les systèmes multi-agents sont utilisés dans plusieurs domaines d'applications [FER 97], en particulier, la robotique, la résolution distribuée de problèmes, la modélisation et la simulation des systèmes complexes. Malgré les nombreuses réalisations dont ils ont fait l'objet, les systèmes multi-agents accusent un certain retard en matière de formalisation et de méthodologie de développement. Pour remédier à cet état de fait, il est nécessaire d'étudier la

problématique des systèmes multi-agents dans une perspective génie logiciel en abordant les aspects méthodes, langages et outils d'aide à leurs développements. C'est cette perspective qui a animé nos travaux de recherche sur les systèmes multi-agents que nous présentons dans ce chapitre.

Sur le plan méthodologique, nous proposons une approche de spécification fondée sur les concepts d'organisation, de rôle et d'agents. La spécification s'effectue en considérant deux niveaux d'abstraction. Le premier considère le système comme une organisation composée d'entités abstraites appelées rôles et de leur interaction. Le second niveau, qualifié d'agentification, introduit les agents comme des entités pouvant encapsuler les rôles. Les interactions entre les rôles deviennent alors des interactions entre les agents qui mettent en oeuvre ces rôles. D'un point de vue spécification formelle, nous proposons une approche multi-formalismes fondée sur la composition de spécifications exprimées en Object-Z [DUK 91] et les statecharts [HAR 87]. Afin d'inscrire notre approche de spécification dans une perspective opérationnelle, nous avons abordé aussi les problèmes liés à l'exécution des spécifications.

La section 10.2. introduit quelques principes concernant les spécifications formelles, présente un état de l'art de la spécification formelle des SMA et des spécifications multi-formalismes.

La section 10.3 introduit notre approche en présentant un exemple. La section 10.4 présente les aspects relatifs à la validation et à la vérification de spécifications produites selon notre approche. Enfin, une conclusion présente les travaux en cours et nos perspectives.

10.2. De l'intérêt des spécifications formelles

La spécification est l'étape clé du processus de développement de logiciels [GAU 96]. Partant du cahier des charges, le but de la spécification est de fournir un modèle qui décrit de manière précise les besoins fonctionnels et les contraintes qui pèsent sur le système. Le modèle ainsi conçu doit servir à la clarification du cahier des charges, à la validation du système et aux autres étapes du développement. Cependant, la définition d'une méthode de spécification des systèmes multi-agents doit prendre en considération les caractéristiques fondamentales de ces systèmes. Par rapport aux classes de systèmes identifiés dans [HAR 85], les systèmes multi-agents font partie de la classe des systèmes réactifs. Ces systèmes doivent maintenir un dialogue avec leurs environnements. Par conséquent, il n'est pas possible de les spécifier en indiquant quelles sorties finales sont produites pour quelles entrées. Il faut plutôt décrire les relations de causalité entre le système et son environnement. Il en découle que la spécification doit inclure le comportement du système et les relations avec son environnement.

10.2.1. Etat de l'art de la spécification formelle des Systèmes Multi-Agents

Dans [WOO 92], la logique temporelle est utilisée pour spécifier des agents intentionnels. La spécification distingue la représentation des états mentaux de l'agent exprimée dans le langage IAL (pour Internal Agent Logic) et le raisonnement sur les agents exprimé dans le langage AL (pour Agent Logic). Les deux langages sont fondés sur la logique temporelle. Ainsi, en plus des opérateurs temporels, les modalités Connaît, Envoie et Fait sont introduites pour raisonner sur les croyances, la communication et l'action.

D'un point de vue théorique, la spécification en logique temporelle est riche en ce sens qu'elle possède une sémantique formelle permettant d'effectuer des preuves de propriétés. D'autres formalismes allant dans ce sens ont été proposés pour la spécification d'agents intentionnels. Citons l'extension de la logique temporelle arborescente par des opérateurs de modalités [RAO 96] pour représenter les croyances, les désirs et les intentions.

Malgré l'importante contribution de ces approches à la définition d'un cadre formel pour spécifier les propriétés de systèmes multi-agents, des problèmes subsistent quant à la dérivation d'un modèle exécutable à partir de la spécification. Par ailleurs, la complexité du formalisme et le manque de règles méthodologiques pouvant assister le spécifieur dans sa tâche rend difficile l'élaboration des spécifications.

Le langage Z [SPI 92] est un formalisme qui a fait l'objet d'études assez importantes et a servi à la spécification d'applications industrielles. C'est pourquoi le langage dispose d'outils d'aide à la spécification et de documentation permettant de faciliter son utilisation. La spécification des SMA à l'aide du langage Z a été proposée dans [LUC 95]. Cette dernière proposition à l'avantage d'esquisser quelques éléments sur la démarche à suivre pour construire une spécification. En effet, la spécification s'effectue par raffinement d'une hiérarchie de schémas Z composée d'entités abstraites prédéfinies.

L'environnement constitue le sommet de la hiérarchie et encapsule les attributs pouvant être perçus par les agents. Le regroupement d'un sous ensemble de ces attributs et des actions associées forme un objet. Les actions représentent les capacités de l'objet. La troisième entité est l'agent qui possède un état mental auquel sont associés des buts à atteindre ou des tâches à réaliser. Lorsque ces buts résultent des motivations (ou désirs) de l'agent, on le considère comme étant un agent autonome. En effet, l'agent est un objet avec des buts à réaliser. L'agent autonome est un agent qui a un ensemble de motivations pouvant générer des buts à atteindre.

Les qualités et les limites de l'approche de spécification des systèmes multi-agents telle qu'elle est proposée par Luck et D'Inverno sont essentiellement liées aux possibilités offertes par le langage Z. En effet, les spécifications sont compréhensibles et se construisent par raffinement d'un squelette de spécification de très haut niveau d'abstraction. Cette approche par raffinement permet aussi d'obtenir une description détaillée du système qui est proche d'une implémentation. Dans

[LUC 97], une proposition pour l'implantation des spécifications des SMA en C++ a été esquissée. En revanche, l'inconvénient de cette approche de spécification réside essentiellement dans l'incapacité du langage Z à prendre en compte l'aspect réactif et la dynamique des systèmes multi-agents. La spécification d'un tel aspect nécessite l'utilisation d'un autre formalisme [D'I 98]. Par conséquent, l'expression en Z des propriétés liées à la dynamique du système n'est pas possible. En effet, l'absence de la spécification du comportement du système dans le temps ne permet pas de prouver des propriétés telles que la sûreté et la vivacité.

L'approche proposée dans [HER 99] consiste à déduire à partir de la spécification informelle des besoins initiaux une spécification semi-formelle puis formelle de ces mêmes besoins. C'est donc un processus de structuration et de formalisation à partir d'un énoncé qui décrit le système comme un tout. Le résultat de la structuration est une hiérarchie de composants qui définit l'architecture des agents cf. [BRA 97]. La formalisation consiste d'une part à définir les éléments de base pour le problème et d'autre part à spécifier le comportement des composants issus de la structuration. Chaque composant est défini en terme d'entrées/sorties et de contraintes temporelles. Les entrées/sorties des composants sont définies avec les éléments de base du problème. Les contraintes temporelles sont exprimées de deux façons : en tant que besoin initial du système et en tant que scénario d'exécution possible. Cette démarche conduit naturellement à vérifier, par la vérification par évaluation sur un modèle, les formules spécifiant les besoins initiaux vis à vis des scénarios d'exécution qui constituent le modèle. La démarche présentée est une approche structurée et modulaire (par composants) de spécification de SMA. L'organisation par composants de la spécification permet le raffinement d'une spécification globale en spécifications plus détaillées de sous-composants. Toutefois, le formalisme utilisé ne permet pas de déduire aisément une implémentation. De plus, la seule technique de vérification possible est la vérification par évaluation sur un modèle avec les modèles fournis par la spécification.

10.2.2. Spécification multi-formalismes

La spécification des systèmes complexes et en particulier les systèmes multi-agents soulève un certain nombre de problèmes liés à l'hétérogénéité de leurs composants et à la diversité des aspects à considérer par le processus de spécification. Malgré, les nombreuses propositions en matière d'approche de spécification, il est difficile de disposer d'un formalisme capable de prendre en compte tous ces aspects.

L'approche multi-formalismes [ZAV 93, PAI 98, COU 99] consiste à combiner plusieurs formalismes et techniques différents pour constituer une démarche de spécification cohérente et adaptée aux divers aspects du système. La spécification de

chacun de ses aspects produit une spécification partielle. La composition de ces spécifications constitue alors la spécification globale du système. Le problème essentiel posé par l'approche multi-formalismes réside dans la composition ou l'intégration des spécifications partielles. En effet, il peut y avoir des redondances entre les spécifications partielles qui introduisent une certaine incohérence dans la spécification globale. Il devient important de bien identifier les relations pouvant exister entre les spécifications partielles. Lorsque les formalismes sont utilisés pour spécifier des aspects différents du système (comme c'est le cas de notre approche), les spécifications partielles peuvent être complémentaires. Une spécification partielle peut faire référence à des éléments spécifiés dans une autre. Par exemple, une transition d'un statechart peut être à l'origine de l'activation d'une opération spécifiée dans une classe Object-Z. D'un point de vue formel et pour garantir la consistance de la spécification globale, la composition doit s'appuyer sur un cadre sémantique bien précis.

Plusieurs travaux ont abordé ce problème de composition de spécifications partielles. Dans [ZAV 93], la sémantique de la composition introduit la logique des prédicats du premier ordre pour définir un domaine sémantique commun aux différents formalismes. Cette approche repose sur la transformation des spécifications partielles en assertion de la logique des prédicats. La composition des spécifications est consistante si et seulement si la conjonction des assertions est satisfiable. La difficulté de cette approche réside dans la complexité du processus de transformation des spécifications partielles et l'absence pour certaines constructions des formalismes utilisés de correspondant en logique des prédicats du premier ordre.

La spécification multi-formalismes est aussi adoptée par le modèle ODP (Open Distributed Processes) [ITU 95] qui propose une architecture de référence pour la construction de systèmes distribués ouverts. Il s'agit de composer des spécifications partielles écrites en Z et en LOTOS [DER 96]. La composition s'effectue en deux étapes. Les spécifications LOTOS sont d'abord transformées en Z. La correction des transformations est validée par la définition d'une sémantique commune aux deux formalismes. Les techniques d'unification sont ensuite appliquées à la spécification Z, résultat du processus de transformation afin de garantir la consistance de la composition. Enfin dans [WIL 92], on propose un domaine syntaxique commun aux différents formalismes, fondé sur les grammaires. La composition est essentiellement syntaxique en ce sens que la sémantique des formalismes utilisés n'est pas prise en compte.

L'approche de composition que nous proposons prend en considération le fait que les deux langages de spécification (Object-Z et statecharts) décrivent des aspects complémentaires (structure de données et opérations, réactivité et comportement). Ainsi une entité est définie par une classe Object-Z qui fournit un moyen d'exprimer des contraintes sur les attributs, les opérations, leurs propriétés, et un statechart modélise son comportement. Bien que complémentaires, chaque description fait référence à des éléments spécifiés dans l'autre. Par exemple, les transitions d'un statechart peuvent être étiquetées par des actions spécifiées en Object-Z et l'invariant

d'une classe Object-Z peut porter sur les états du statechart correspondant. La composition nécessite alors la définition d'un cadre permettant d'établir un lien syntaxique entre les spécifications partielles. Cependant, il est aussi nécessaire de doter la composition d'une sémantique formelle permettant de raisonner sur la spécification résultat de la composition.

Ces constatations étant faites, nous pouvons maintenant expliciter notre démarche de composition en introduisant deux étapes [GRU 00b]. Tout d'abord, la première étape est qualifiée d'intégration syntaxique et consiste à faire coexister les deux notations. Nous avons défini un ensemble de types et de classes Object-Z qui spécifie les principales constructions des statecharts. Ensuite, la deuxième étape assure l'intégration sémantique qui transforme les spécifications Object-Z et les statecharts en un système de transitions [MAN 95]. Ce dernier constitue le domaine sémantique commun aux différentes spécifications partielles. C'est la définition de ce cadre sémantique qui doit nous permettre d'effectuer des preuves de propriétés telles que la vivacité, la sûreté et la satisfiabilité de l'invariant historique des classes Object-Z [GRU 00a]. Nous allons maintenant décrire notre approche de spécification formelle des systèmes multi-agents en précisant en premier le cadre méthodologique sous-jacent.

10.3.Vers une méthode de spécification

10.3.1. Motivation

Nos travaux sur la spécification formelle des systèmes multi-agents reposent sur plusieurs constatations. D'une part et comme nous l'avons signalé dans les paragraphes précédents, les systèmes multi-agents exhibent un certain nombre de caractéristiques telles que la réactivité, l'hétérogénéité qui ne peuvent être spécifiées sinon difficilement à l'aide d'un seul formalisme. C'est pourquoi, nous proposons une approche de spécification multi-formalismes fondée sur la combinaison des langages Object-Z [DUK 91] et les statecharts [HAR 87]. L'hétérogénéité dans les approches, les formalismes et par conséquent les techniques de preuve de spécifications formelles apparaît de plus en plus comme une nécessité en génie logiciel [ALP 98].

Un autre point important est la nécessité de disposer d'un guide méthodologique. En effet, l'élaboration d'une spécification formelle est une tâche ardue qui nécessite en plus du pouvoir d'expression des formalismes de spécification et de leur adéquation à la classe des systèmes considérés, des outils et des recettes pouvant aider le spécifieur dans sa démarche de construction des spécifications. Enfin, la

spécification formelle doit servir aux autres étapes de développement et en particulier à la validation et la vérification.

Dans ce cadre, nous développons une approche qui propose deux niveaux d'abstraction à considérer lors de l'analyse d'un système. Le premier niveau propose d'analyser le système d'un point de vue organisationnel : le système est vu comme une organisation composée de rôles en interaction. Le second niveau qualifié d'agentification, consiste à concevoir les agents comme étant des entités regroupant des rôles. La spécification formelle permet de donner une définition formelle aux concepts supports aux deux niveaux précédents. Nous avons aussi proposé une approche de validation et de vérification des spécifications que nous présentons dans la section 10.4.

10.3.2. Cadre général

Notre démarche de spécification s'appuie sur un cadre formel et général que l'ont peut raffiner pour spécifier un système multi-agents particulier. La spécification s'effectue en considérant deux niveaux d'abstraction :

- Le niveau organisation : ce niveau considère le système comme une société artificielle ou organisation composée d'entités abstraites, appelées rôles et de leur interaction. A ce stade, on peut utiliser une notation graphique telle que celle proposée par la méthode OOram [REE 96].
- Le niveau agentification : c'est à ce niveau que s'effectue la conception du système en termes d'agents. Les rôles sont encapsulés par des agents en s'appuyant sur des critères de conception : regroupement de rôles fortement liés par les ressources nécessaires à leurs réalisations ou pour des raisons d'efficacité.

La figure 10.1 présente ces deux niveaux, en montrant des instances d'organisation, de rôles, d'agents et leurs relations. La description de chaque niveau s'appuie sur des concepts bien identifiés. Le concept de niveau supérieur est l'organisation qui représente la projection de la mission globale du système selon un point de vue donné. Le système est lui-même vu comme étant une macro organisation. Par exemple, dans une université, un département d'enseignement ou une association d'élèves peuvent être considérés comme des organisations. La mission d'une organisation se décline en termes de rôles en interaction. Un rôle est l'abstraction d'un comportement ou d'une fonctionnalité et peut être rattaché à un statut. Les rôles interagissent pour assurer la mission de l'organisation. Le niveau agentification structure les rôles autour des agents qui les mettent en oeuvre.

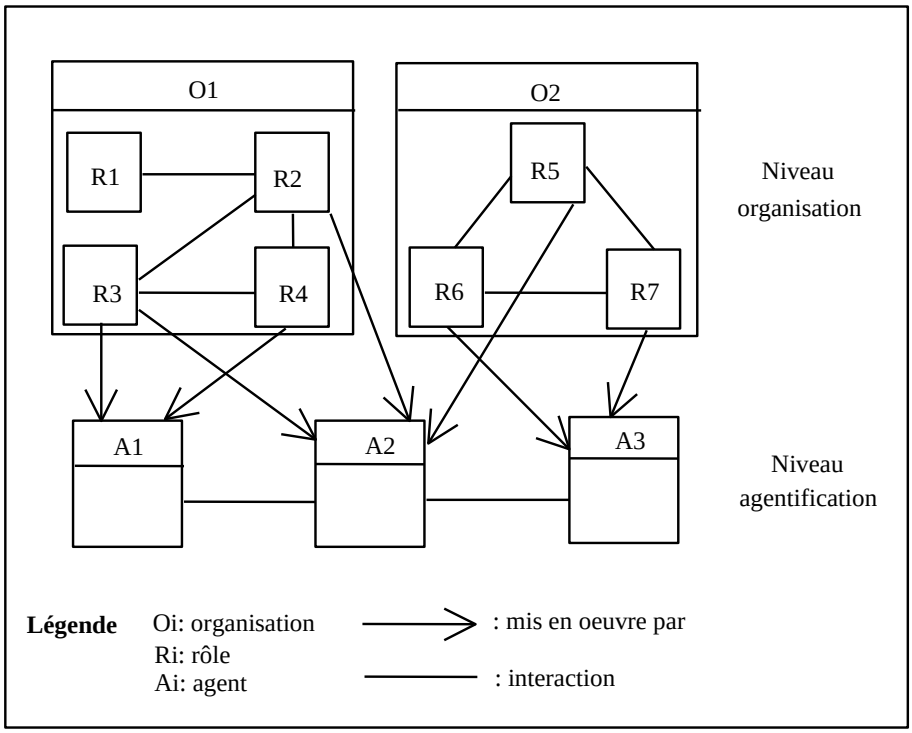


Figure 10.1. *Aperçu du modèle rôle-interaction-organisation*

Cette vision des systèmes fondée sur des concepts organisationnels se retrouve à différents niveaux de description dans certains travaux sur les systèmes multi-agents [GAS 95, FER 98, WOO 99]. Le méta-modèle Aalaadin [FER 98] et la plate-forme de développement MadKit [GUT 99] reposent sur les concepts d'agent, de groupe et de rôle. La notion d'organisation est absente du modèle ce qui rend difficile le raffinement des groupes de rôles. La notion la plus proche d'organisation telle que nous l'entenodns est celle de groupe qui est en quelque sorte une instance d'organisation. Une sémantique opérationnelle de ce méta-modèle est actuellement à l'étude [FER 99]. En fait, ce méta-modèle est applicable lors d'une phase de conception et permet grâce à MadKit d'implémenter aisément les modèles produits.

Par contre, la notion de rôle correspond intuitivement à celle que nous utilisons et qui est également utilisée dans Gaïa [WOO 99]. Le rôle pour les trois approches est un comportement abstrait de toute entité le mettant en œuvre. Gaïa ne propose pas de notion du type organisation. Le raffinement de spécifications produites selon l'approche Gaïa vers un langage d'implémentation est difficile. De plus, la spécification de la notion de rôle introduite par la méthodologie Gaïa impose certaines hypothèses (description par permissions, protocoles, activités et

responsabilités). Ces présuppositions contraignent la spécification des agents qui mettent en œuvre ces rôles.

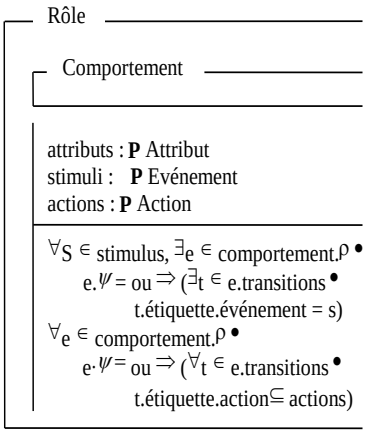
Pour illustrer ces concepts, nous considérons un exemple issu d'un projet intitulé « université virtuelle ».

10.3.3. Spécification formelle

La spécification formelle des SMA est fondée sur un ensemble de schémas de spécifications, qui peuvent être instanciés pour obtenir une spécification concrète. Cet ensemble est composé de la description formelle des concepts d'organisation introduits au paragraphe précédent.

A chaque concept est associé une classe Object-Z, pouvant faire référence à un statechart lorsque cela est nécessaire. La spécification d'un SMA particulier s'effectue grâce au mécanisme de raffinement proposé par le langage Object-Z.

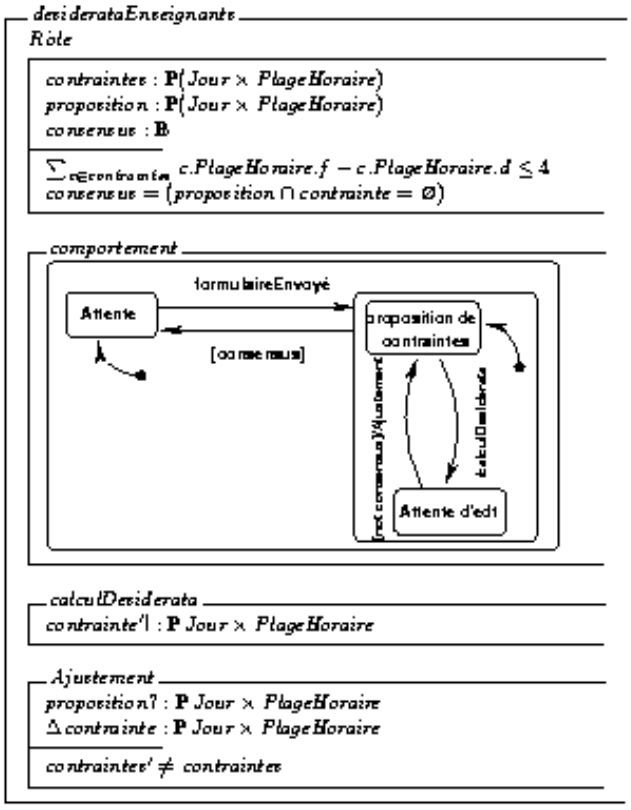
En reprenant l'exemple de la section précédente, la spécification du rôle est obtenue par raffinement du schéma *Rôle*.



Le sous-schéma comportement associe un statechart au rôle, qui spécifie ses changements d'état. Un rôle est caractérisé par un ensemble d'attributs propres, *attributs*, l'ensemble *stimulus* des événements auxquels il est sensible et les actions, *actions*, qu'il effectue en réaction à ces stimuli.

Afin d'illustrer notre approche de spécification, nous proposons un exemple. Cet exemple est extrait d'un projet d'université virtuelle pour l'automatisation de certaines tâches relatives à la vie d'une université. L'extrait présenté par la figure 10.2 est constitué de deux organisations : *Contrôle des connaissances* et *Emploi du*

temps. La première a pour objectif la gestion des contrôles et la seconde l'établissement des emplois du temps. Pour compléter ce schéma il faudrait indiquer les liens enter les rôles et les agents. C'est-à-dire quels agents vont mettre en oeuvre ces rôles.



L'établissement de l'emploi du temps est le résultat d'une négociation entre les rôles *desirataEnseignants* et *gestionPlanning*. Nous détaillons maintenant le rôle *desirataEnseignants*. Ce rôle est décrit par les attributs *contraintes* et *proposition* qui sont des ensembles de couples *([Jour],[PlageHoraire])* et par l'attribut booléen *consensus* qui est vrai si et seulement s'il n'y a pas d'intersections entre les contraintes et la proposition. Le premier ensemble constitue les horaires, au maximum quatre heures, durant lesquels l'enseignant ne souhaite pas avoir de cours. Le second est la proposition d'emploi du temps du rôle *gestionPlanning*. Le comportement du rôle est le suivant. Dès qu'une proposition d'emplois du temps est envoyé le rôle calcule des contraintes. En réponse à ces contraintes une proposition est renvoyée. Si les ensembles *contraintes* et *proposition* sont disjoints *consensus* est

vrai donc le statechart passe dans l'état *Attente*. Si les ensembles ne sont pas disjoints le rôle *desideratataEnseignants* exécute la méthode *Ajustement* pour modifier son ensemble *contraintes* et le proposer à nouveau.

La spécification donnée n'est pas complète mais elle illustre l'utilisation du framework organisationnel. Les spécifications de SMA obtenues sont structurées et abstraites ce qui facilite leur validation et leur vérification par parties. Pour spécifier des agents à partir de rôles le framework propose la classe *Agent* cf. [HIL 00a] qui est construite à partir des rôles que l'agent met en œuvre.

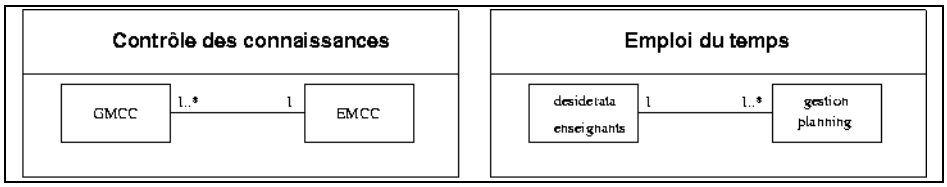


Figure 10.2. *Organisations et rôles de l'université virtuelle*

10.4. Validation et vérification des spécifications

La spécification précède les étapes de conception et de mise en oeuvre des systèmes. A ce titre, elle fournit un modèle du système qui doit servir non seulement comme point d'entrée pour ces étapes, mais aussi pour s'assurer que le modèle représente bien la réalité et possède les bonnes propriétés. Ces deux dernières activités sont liées à la validation et à la vérification de la spécification.

Une approche pragmatique de la validation consiste à faire de la spécification, un prototype du système que l'on peut simuler afin d'observer son comportement et de le confronter à la réalité spécifiée. Pour cela, il faut disposer de langages de spécification qui soient exécutables et d'environnements d'aide à la simulation. Dans notre contexte, nous avons essentiellement utilisé l'environnement Statemate [HAR 90] pour exécuter et animer les spécifications statecharts construites aussi bien au niveau organisation qu'au niveau agentification. L'exécution s'effectue soit en mode interactif, soit à l'aide d'un langage de commande et permet d'observer l'évolution temporelle du modèle. On peut ainsi observer le comportement d'un système spécifié comme cela est fait dans [HIL 00b].

Un autre rôle essentiel que l'on attribue aux spécifications formelles est de permettre la vérification de propriétés telles que la vivacité et la sûreté. La vérification peut s'effectuer en adoptant soit une approche axiomatique (méthodes déductives) soit par évaluation sur un modèle de la spécification (méthodes basées sur les modèles ou model-checking) [BID 85]. L'approche axiomatique consiste à modéliser les spécifications dans un système formel dans lequel des raisonnements déductifs permettent d'effectuer des preuves de propriétés analogues aux

démonstrations de théorèmes. Elle repose généralement sur des logiques puissantes et soulève le problème de l'automatisation des preuves. En effet, les outils logiciels supports de l'approche déductive nécessitent une forte interaction avec l'utilisateur à cause du problème de l'indécidabilité de la déduction.

L'approche par évaluation construit un modèle sémantique de la spécification qui sert de base à la vérification. C'est cette deuxième approche que nous avons étudiée en transformant les spécifications en un système de transitions et en utilisant l'outil STeP comme support à la vérification [MAN 95]. La propriété à vérifier, qui est généralement de sûreté ou de vivacité, s'exprime par une formule de la logique temporelle linéaire. L'outil de vérification essaye d'établir la validité de la formule, relativement au système de transitions. La terminaison de ce test de validité relative n'est garantie que si l'espace d'états associé au système de transitions est fini. Ainsi, pour l'exemple du système d'aide à l'établissement d'un emploi du temps, voir section 10.3., la propriété de sûreté *aucune évolution ne conduit à un état où il y a consensus et les contraintes de l'enseignant ne sont pas respectées* s'exprime par la formule :

$$\Box((\text{contraintes.Jour} = \text{proposition.Jour}) \wedge (\text{contraintes.PlageHoraire} \cap \text{proposition.PlageHoraire} \neq \varnothing) \Rightarrow \neg\, \text{consensus})$$

Suivant cette approche, nous avons spécifié et vérifié un système multi-agents pour le paramétrage de réseaux radio mobiles [HIL 99].

10.5. Conclusion

Dans ce chapitre, nous avons présenté quelques travaux sur les méthodes et les langages de spécification de systèmes multi-agents. Nous avons ensuite proposé l'approche multi-formalismes et un cadre méthodologique pour la spécification, la validation et la vérification de systèmes multi-agents. Plusieurs problèmes restent à étudier et en particulier l'approfondissement de la sémantique des spécifications et des liens entre les deux niveaux d'abstraction que nous avons identifiés (organisation, agentification). Pour approfondir la sémantique des spécifications nous travaillons actuellement sur la définition d'un ensemble de règles de transformation des spécifications Object-Z et statecharts en systèmes de transitions et la réalisation d'un environnement logiciel support à ces transformations.

Un autre aspect sur lequel nous travaillons actuellement est le développement d'un environnement composé d'un éditeur de spécification et d'un outil de transformation vers le système STeP. Nous avons aussi entamé la réflexion sur le passage d'une spécification exprimée au niveau agentification à une implantation sous MadKit [GUT 97]. Développé en Java, l'environnement MadKit est fondé sur

un méta-modèle dont les concepts (groupe, rôle, agent) sont proches des éléments à la base de notre démarche de spécification.

Pour conclure ce chapitre, nous reprenons une proposition de Ferber [FER 97] qui stipule que le génie logiciel deviendra «multi-agents » comme il est aujourd'hui « objet ». Cependant, si la technologie objet occupe une place importante dans toutes les formations et a atteint le stade de pionnier, c'est grâce au progrès accompli en matière de méthodes, de langages et d'environnements ayant accompagné son évolution. Nous pensons que cela doit être de même pour la technologie multi-agents.

Références

[ALP 98] GDR ALP. Fondements pour l'hétérogénéité des spécifications et la coopération de techniques formelles. Technical report, Groupe de travail du GDR ALP, 1998.

[BID 85] M. Bidoit et C. Choppy. ASSPEGIQUE : An integrated environment for algebraic specifications. In Maurice Nivat Hartmut Ehrig, Christiane Floyd et James Thatcher, eds, Formal methods and software development: Proceedings of the International Joint Conference on Theory and Practice of Software Development (TAPSOFT 85) : volume 2 - Colloquium on Software Engineering (CSE), volume 186 of LNCS, pages 246_260, Berlin, FRG, March 1985. Springer.

[BRA 97] F.M.T. Brazier, B. Dunin Keplicz, N. Jennings et J. Treur . Desire : Modelling multi-agent systems in a compositional formal framework. International Journal of Cooperative Information Systems , 6 :67_94, 1997.

[COU 99] S. Coudert, G. Bernot et P. Le Gall . Hierarchical heterogeneous specifications. Lecture Notes in Computer Science , 1589 :107_121, 1999.

[DER 96] J. Derrick, E. Boiten, H. Bowman et M. Steen . Supporting ODP - translating LOTOS into Z. In E. Najm et J.-B. Stefani, eds, First IFIP International Workshop on Formal Methods for Open Object-Based Distributed Systems, pages 399_406, Paris, March 1996. Chapman & Hall.

[D'I 98] M. D'Inverno, D. Kinny, M. Luck et M. Wooldridge . A formal specification of dMARS. Lecture Notes in Computer Science , 1365, 1998.

[DUK 91] Roger Duke, Paul King, Gordon Rose et Graeme Smith . The Object-Z specification language. Technical report, Software Verification Research Center, Departement of Computer Science, University of Queensland, AUSTRALIA, 1991.

[FER 97] Jacques Ferber. Les systèmes multi-agents : un aperçu général. Technique et science informatiques , 16(8) :979_1012, 1997.

[FER 98] Jacques Ferber et Olivier Gutknecht . Aalaadin : a meta-model for the analysis and design of organizations in multi-agent systems. In International Conference on Multi-Agent Systems, july 1998. IEEE Press.

[FER 99] Jacques Ferber et Olivier Gutknecht . Operational Semantics of a Role-based Agent Architecture. Proceedings of the 6th Int. Workshop on Agent Theories, Architectures and Languages. . Springer Verlag, 1999.

[GAS 95] Les Gasser. Computational organization research. In Victor Lesser, ed, Proceedings of the First International Conference on Multi_Agent Systems, pages 414_415, San Francisco, CA, 1995. MIT Press. (invited speaker talk).

[GAU 96] Marie-Claude Gaudel, Bruno Marre, Françoise Schlienger et Gilles Bernot. Précis de génie logiciel. MASSON, 1996.

[GOO 93] Richard Goodwin. Formalizing properties of agents. Technical report, Carnegie Mellon University-School of Computer Science, 1993.

[GRU 00a] Pablo Gruer, Vincent Hilaire et Abder Koukam . an Approach to the Verification of Multi-Agent Systems. In International Conference on Multi Agent Systems. UTBM/SET, IEEE Computer Society Press, 2000.

[GRU 00b] Pablo Gruer, Vincent Hilaire et Abder Koukam . Verification of Object-Z Specifications by using Transition Systems. In Fundamental Aspects of Software Engineering, number 1783 in Lecture Notes in Computer Science. Springer Verlag, 2000.

[GUT 97] Olivier Gutknecht et Jacques Ferber . Madkit : Organizing heterogeneity with groups in a platform for multiple multi-agent systems. Technical Report 97188, LIRMM, 1997.

[GUT 99] Olivier Gutknecht et Jacques Ferber . Vers une méthodologie organisationnelle pour les systèmes multi-agents. In Journées Francophones Intelligence Artificielle Distribuée & Systèmes Multi-Agents, juin 1999.

[HAR 85] D. Harel et A. Pnueli. On the development of reactive systems. In K. R. Apt, ed, Logics and Models of Concurrent Systems . Springer Verlag, 1985.

[HAR 87] David Harel. Statecharts : A visual formalism for complex systems. Science of Computer Programming, 8(3) :231_274, June 1987.

[HAR 90] David Harel, Hagi Lachover, Amnon Naamad, Amir Pnueli, Michal Politi, Rivi Sherman, Aharon Shtull-Trauring et Mark B. Trakhtenbrot. Statemate : A working environment for the development of complex reactive systems. IEEE Transactions on Software Engineering , 16(4) :403_414, April 1990.

[HER 99] D. E. Herlea, C. M. Jonker, J. Treur et N. J. E. Wijngaards. Specification of behavioural requirements within compositional multi-agent system design. Lecture Notes in Computer Science , 1647 :8_27, Springer Verlag,, 1999.

[HIL 99] V. Hilaire, T. Lissajoux et A. Koukam . Towards an executable specification of Multi-Agent Systems. In Kluwer Academic Publisher, ed, International Conference on Enterprise Information Systems'99 , 1999.

[HIL 00a] Vincent Hilaire. Vers une approche de spécification, de prototypage et de vérification de Systèmes Multi-Agents . PhD thesis, UTBM, 2000.

[HIL 00b] Vincent Hilaire, Abder Koukam, Pablo Gruer et Jean-Pierre Müller. Vers une méthodologie formelle de spécification de systèmes multi-agents. In Journées Francophones Intelligence Artificielle Distribuée & Système Multi-Agents. UTBM-IIIA, Hermes, 2000.

[ITU 95] ITU. Itu recommendation x.901-904-iso/iec 10476 1-4. open distributed processing - reference model. Technical report, ITU, 1995.

[LUC 95] M. Luck et M. D'Inverno. Structuring a Z specification to provide a formal framework for autonomous agent systems. Lecture Notes in Computer Science, 967, Springer Verlag, 1995.

[LUC 97] Michael Luck, Nathan Griffiths et Mark d'Inverno . From agent theory to agent construction : a case study. In Springer-Verlag, ed, Proceedings of the third International Workshop on Agent Theories, Architecture and Languages, Lecture Note in Artificial Intelligence, pages 49_63, 1997.

[MAN 95] Z. Manna, N. Bjoerner, A. Browne et E. Chang . STeP : The Stanford Temporal Prover. Lecture Notes in Computer Science , 915, 1995.

[PAI 98] Richard F. Paige. Heterogeneous notations for pure formal method integration. In Formal Aspects of Computing (1998) , volume 3. BCS, 1998.

[RAO 96] Anand S. Rao. Decision procedures for propositional linear-time belief-desire-intention logics. In Michael Wooldridge, Jörg P. Müller et Milind Tambe, eds, Proceedings on the IJCAI Workshop on Intelligent Agents II : Agent Theories, Architectures, and Languages , volume 1037 of LNAI, pages 33_48, Berlin, 19_20 August 1996. Springer Verlag.

[REE 96] Trygve Reenskaug. Working with Objects : The OOram Software Engineering Method. Manning Publications, 1996.

[SPI 92] J. M. Spivey. The Z Notation : A Reference Manual . Prentice Hall, 1992.

[WIL 92] D. S. Wile. Integrating syntaxes and their associated semantics. Technical Report RR-92-297, USC/Information Institute, 1992.

[WOO 92] Michael Wooldridge. The Logical Modelling of Computational Multi-Agent Systems. Phd thesis, Manchester Metropolitan University, 1992.

[WOO 99] Michael Wooldridge, Nicholas R. Jennings et David Kinny . A methodology for agent-oriented analysis and design. In Proceedings of the Third International Conference on Autonomous Agents (Agents'99) , pages 69_76, Seattle, WA, USA, 1999. ACM Press.

[ZAV 93] Pamela Zave et Michael Jackson . Conjunction as composition. ACM Transactions of Software Engineering and Methodology , 2(4) :379_411, October 1993.